목차

[Travia 프로젝트 - Gemini API 연동 2](#_Toc196774538)

[Python 3.13 + 가상환경(venv) 재구축 2](#_Toc196774539)

[.env 문제 해결 2](#_Toc196774540)

[모델 404 오류 해결 2](#_Toc196774541)

[Gemini API 연동 테스트 성공 2](#_Toc196774542)

[API 호출 및 원본 응답 받기 - main.py 4](#_Toc196774543)

[Gemini API 호출 및 파싱 - gemini\_api\_client.py 5](#_Toc196774544)

[Gemini 호출 함수 - def generate\_travel\_recommendation(prompt: str) -> dict: 5](#_Toc196774545)

[파싱 로직 (원본 텍스트 → JSON 변환) - def parse\_gemini\_recommendation(text: str) -> list: 6](#_Toc196774546)

[프로젝트 파일 구조 및 흐름도 7](#_Toc196774547)

[동작 흐름 7](#_Toc196774548)

[데이터 흐름 예시 (출력 형태) 8](#_Toc196774549)

[\*\*최종 JSON 형식 응답 테스트\*\* 10](#_Toc196774550)

# Travia 프로젝트 - Gemini API 연동

Gemini API를 이용해  
→ 사용자의 여행 성향을 기반으로  
→ 여행 도시 추천 + 하루 일정 정보까지 파싱해서 JSON 형태로 제공하기

## Python 3.13 + 가상환경(venv) 재구축

새 가상환경 생성:

py -3.13 -m venv travia-env

.\travia-env\Scripts\activate

필요한 패키지 설치:

pip install google-generativeai==0.8.5

pip install python-dotenv

pip install fastapi uvicorn

## .env 문제 해결

load\_dotenv()로 환경변수 정상 불러옴 확인

## 모델 404 오류 해결

list\_models()로 사용 가능한 모델 확인

model = genai.GenerativeModel("models/gemini-1.5-flash-latest")

## Gemini API 연동 테스트 성공

테스트용 프롬프트 작성 (설문 결과 기반 예시)

generate\_content() 정상 작동 확인

리스본, 로마, 바르셀로나 추천 결과 출력

일정 추천까지 포함된 답변 정상 수신

스크린샷, 소프트웨어, 텍스트, 멀티미디어 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

## API 호출 및 원본 응답 받기 - main.py

**역할:**

* 테스트용 프롬프트(test\_prompt)를 준비해서
* generate\_travel\_recommendation() 함수로 Gemini API 호출
* 결과를 받아서 출력 (pprint 사용)

**연결되는 흐름**

* generate\_travel\_recommendation(prompt) 호출 →
* 내부에서 gemini\_api\_client.py의
  + model.generate\_content(prompt) 실행
  + Gemini 원본 응답(response.text) 출력
  + parse\_gemini\_recommendation()으로 파싱 처리
  + 최종 JSON 결과 반환

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

## Gemini API 호출 및 파싱 - gemini\_api\_client.py

### Gemini 호출 함수 - def generate\_travel\_recommendation(prompt: str) -> dict:

**역할:**

1. 환경변수(GOOGLE\_API\_KEY)로 Gemini API 연결 설정
2. 프롬프트 전달 → Gemini 모델 호출 → response.text 출력
3. parse\_gemini\_recommendation()을 통해 결과 파싱
4. 파싱된 결과를 딕셔너리 형태(status, data)로 반환

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

### 파싱 로직 (원본 텍스트 → JSON 변환) - def parse\_gemini\_recommendation(text: str) -> list:

**파싱 흐름:**

1. 도시별 블록 구분 (\\*\\*\d+\.\s+ 기준)
2. 각 도시에서:
   * 도시 이름 (도시 이름: 또는 1. 도시 이름)
   * 국가 (국가:)
   * 성향과의 매칭 이유 (성향과의 매칭 이유:)
   * 하루 일정 예시 (Day 1, Day 2, ... → time & activity) 추출
3. 리스트 형태로 정리하여 반환

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다. 텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

## 프로젝트 파일 구조 및 흐름도

📂 2025 GDGoC spring 프로젝트/

├── .env # 👉 GOOGLE\_API\_KEY 저장

├── main.py # 🔥 API 호출 및 결과 출력 (테스트)

├── gemini\_api\_client.py # ✅ API 호출 함수

├── travia-env/ # 🟢 가상환경 (venv)

│ └── Scripts/python.exe # 가상환경 파이썬 실행파일

└── requirements.txt # 📦 필요한 패키지 목록

가상환경 활성화: .\travia-env\Scripts\activate

설치: pip install -r requirements.txt

## 동작 흐름

1. main.py  
→ **generate\_travel\_recommendation(prompt)** 호출  
→ gemini\_api\_client.py로 이동

2. gemini\_api\_client.py

* generate\_travel\_recommendation(prompt)  
  → Gemini API 호출  
  → 결과 텍스트 출력  
  → parse\_gemini\_recommendation(text) 로 파싱

3️. parse\_gemini\_recommendation(text)

* 도시별로 파싱 (도시, 국가, 성향 이유, 하루 일정)
* JSON 형태 리스트 반환

4️. main.py

* 결과 pprint() 로 예쁘게 출력

## 데이터 흐름 예시 (출력 형태)

텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

* + JSON 형식 수정 (reason에서 일정 제거)

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

## \*\*최종 JSON 형식 응답 테스트\*\*

**1. 목표**

* **Gemini API**를 사용해
* 사용자 성향(예산, 여행 스타일, 선호 지역 등)에 맞는
* **여행 일정과 도시 추천**을
* **JSON 포맷으로 응답 받아서**
* 백엔드(FastAPI) → 프론트(React)로 넘기기

**2. 주요 문제**

**문제 1:**

Gemini가 응답을 Markdown 코드블록(json ... ) 형태로 주기 때문에,  
json.loads()에서 파싱 오류 발생 (Expecting value: line 1 column 1 (char 0)).

**문제 2:**

Gemini의 응답 JSON 구조가 {"data": {"data": [...]}}처럼  
**불필요하게 중첩**되어 있어서, 실사용 시 번거로움.

**3. 해결 방법**

**(1) Markdown 코드블록 제거 함수 구현**

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**(2) generate\_travel\_recommendation 함수 구현**

* 응답에서 parts[0].text 우선 사용
* 코드블록 제거 후 json.loads()로 파싱
* 중첩된 "data": {"data": [...]}에서 ["data"]만 꺼내서 반환

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**4. 지금까지 구현한 핵심 포인트**

* Gemini API 연결 완료
* 마크다운 코드블록 제거 로직 (clean\_markdown\_json)
* JSON 파싱 로직 (json.loads)
* 중첩 제거 (["data"]만 반환)
* 테스트 프롬프트로 정상 동작 확인 완료

**5. 테스트용 프롬프트**

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**6. 실행 테스트**

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**7. 응답 확인**

텍스트, 스크린샷이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.