Chapter 4-10: 최종 완성, 데이터 플레이팅

Prologue: 마지막 디테일, 그리고 완벽한 서빙

우리의 셰프(Gemini)는 완벽한 요리(분석 결과)를 내놓았습니다. 하지만 요리가 서빙되기 직전, 친절한 웨이터가 실수로 요리 위에 장식용 덮개(마크다운)를 씌워놓았습니다. 이제 우리가 할 일은, 손님(Notion)에게 전달하기 직전에 이 덮개를 살짝 벗겨내어, 완벽한 상태의 요리를 접시에 담아내는 것입니다.

[Mission 28] Gemini 응답 정제 및 최종 저장

Step 1: `gemini_analyzer.py`**에 '포장지 제거' 로직 추가**

`analyze_job_posting` 함수가 순수한 JSON 텍스트만 남기고, 그것을 파싱하여 **딕셔너리 객체**를 반환하도록 최종 수정합니다.

```
# analysis/gemini_analyzer.py (최종 수정)

import os
import json # json 라이브러리를 임포트합니다.
import google.generativeai as genai
from dotenv import load_dotenv

load_dotenv()
genai.configure(api_key=os.environ.get("GEMINI_API_KEY"))
model = genai.GenerativeModel('gemini-pro') # 또는 신호용님이 성공한 다른 모델명

def analyze_job_posting(job_description: str):
    # ... (prompt 내용은 동일) ...
    try:
        response = model.generate_content(prompt)

        # ★★★★★ 포장재(마크다운) 제거 로직 ★★★★★
        cleaned_text = response.text.strip().replace("```json", "").replace("```", "")

        # 정제된 텍스트를 파이썬 딕셔너리로 변환하여 반환
        return json.loads(cleaned_text)

    except (json.JSONDecodeError, Exception) as e:
        # 파싱 실패 시, 원본 텍스트와 함께 에러를 출력하여 디버깅을 돕습니다.
        print(f"  🚨 [Gemini 오류] 응답 파싱 중 문제 발생: {e}")
        if 'response' in locals():
            print(f"  -> 원본 응답: {response.text}")
        return None # 파싱에 실패하면 None을 반환
```

최종 행동 계획 (Next Action)

1. ☐ `gemini_analyzer.py` **파일만** 위 최종 코드로 업데이트합니다. (`main.py`는 이제 완벽하므로 수정할 필요가 없습니다.)
2. ☐ **마지막 최종 시운전**을 실행합니다.
3. ☐ 터미널 로그에서 더 이상 `[Gemini 오류]`가 발생하지 않는지 확인합니다.
4. ☐ **Notion 데이터베이스**를 열어, 마침내 **'AI 요약'과 '핵심 기술' 속성에 아름답게 데이터가 채워지는, 이 프로젝트의 가장 빛나는 순간을 확인합니다.**