SK네트웍스 **Family AI** 과정 **12**기

모델배포 개발된 LLM 연동 웹 애플리케이션

산출물 단계	모델배포
평가 산출물	LLM 연동 웹 애플리케이션
제출 일자	2025.08.09
깃허브 경로	https://github.com/SKNETWORKS-FAMILY-AICAMP/SKN12-FINAL-6TEAM
작성 팀원	이정민

1. 프로젝트 개요

- 프로젝트명: 마이 무디(MY MOODY)
- 작성일: 2025.08.09
- 작성자: 이정민
- 버전: v1

2. 시스템 개요

- 목표: 벡터 데이터베이스와 LLM 연동을 통한 HTP 기반 그림검사 결과 제공 및 성격 유형별 프롬프트 최적화 챗봇을 활용한 멘탈케어 및 상담 제공
- 주요 기능:
    - OpenSearch 벡터 데이터베이스와 LLM 연동 기반 그림검사 분석 모델 구현
    - 성격 유형별 프롬프트 최적화 설계
    - 안전한 API 키 관리 및 예외 처리 체계 구축

3. 시스템 아키텍처

- 백엔드:
    - 언어: Python (3.10.18)
    - 프레임워크: FastAPI

- LLM 연동: OpenAI GPT-4o
- 분류모델: BERT -> HuggingFace (링크 삽입)
  - 모델의 Hugging Face Hub 경로 : Bokji/HTP-personality-classifier
  - 허깅페이스 다운로드 주소 : https://huggingface.co/Bokji/HTP-personality-classifier
- 벡터 데이터베이스: OpenSearch(EC2에 업로드)
- 데이터베이스: PostgreSQL
- 기술 스택:
  - API 관리: RESTful API + 모듈화된 라우터
  - 보안: JWT + Google OAuth 2.0 하이브리드

- 프론트엔드:
  - 프레임워크: React (18.3.1) / Node.js
  - 상호작용: Axios
  - UI: TypeScript + TailwindCSS

4. LLM 연동 구현

4.1 LLM과 벡터 데이터베이스 연동

- 연동 목적: 벡터 데이터베이스(OpenSearch)에 저장된 HTP 그림검사 해석체계 정보를 기반으로 LLM이 전문성 있는 그림 분석 결과를 생성하도록 연동
- 프롬프트: 사용자가 업로드한 그림을 4단계 분석 절차를 통해 감정 및 심리 상태를 분석하도록 설계
  - 1단계: GPT-4o 기반 초기 분석 수행
  - 2단계: 심리 분석 요소 추출(`extract_psychological_elements` 모듈)
  - 3단계: 벡터 데이터베이스 OpenSearch 기반으로 관련 자료 검색(`search_rag_documents` 모듈)
  - 4단계: RAG context 기반으로 최종 분석 수행

```
from opensearch_client import OpenSearchEmbeddingClient
opensearch_client = OpenSearchEmbeddingClient(host='3.39.30.211')

load_dotenv()
OPENAI_API_KEY = os.getenv("OPENAI_API_KEY")
```

```python
def search_rag_documents(query_elements):
    """
    OpenSearch를 사용하여 관련 RAG 문서 검색
    """
    if not opensearch_client or not query_elements:
        return []

    try:
        # 모든 요소를 하나의 쿼리로 합침
        combined_query = ' '.join(query_elements)

        # 하이브리드 검색 수행
        search_results = opensearch_client.hybrid_search(
            index_name=RAG_INDEX_NAME,
            query_text=combined_query,
            k=10,
            use_reranker=True
        )

        # Reranker 기준 1번째 결과 반환
        if search_results:
            top_result = search_results[0]
            return {
                'text': top_result['text'],
                'metadata': top_result.get('metadata', {}),
                'document': top_result.get('document', ''),
                'element': top_result.get('element', ''),
                'score': top_result.get('rerank_score', top_result.get('score', 0))
            }
    except Exception as e:
        print(f"RAG 검색 실패: {e}")

    return None
```

```python
try:
    # 1차 GPT 해석 (초기 분석)
    print("1단계: 초기 심리 분석 수행 중...")
    initial_analysis = analyze_image_with_gpt(image_path, PROMPT)
    print("\n[초기 분석 결과]")
    print(initial_analysis)

    # 심리 분석 요소 추출
    print("\n2단계: 심리 분석 요소 추출 중...")
    psychological_elements = extract_psychological_elements(initial_analysis)
    print(f"추출된 요소들: {psychological_elements}")

    # OpenSearch RAG 검색
    print("\n3단계: RAG 시스템을 통한 관련 자료 검색 중...")
    rag_result = search_rag_documents(psychological_elements)

    if rag_result:
        print(f"검색된 관련 자료: {rag_result['document']} - {rag_result['element']}")
        print(f"관련도 점수: {rag_result['score']:.4f}")

        # RAG 컨텍스트를 포함한 최종 분석
        print("\n4단계: RAG 컨텍스트를 활용한 최종 분석 수행 중...")
        final_prompt = f"""
                        아래는 심리 그림 검사의 초기 분석 결과입니다:

                        {initial_analysis}

                        위 분석 결과를 바탕으로, 제공된 참고 자료를 활용하여 더욱 정확하고 전문적인 최종 심리 분석을 제공해 주세요.
                        특히 참고 자료의 전문적 해석을 반영하여 분석의 깊이를 더해주세요.
                        반드시 ~입니다 체로 작성해 주세요.
                        """
        result_text_gpt = analyze_image_with_gpt(image_path, final_prompt, rag_result)
    else:
        print("관련 RAG 자료를 찾을 수 없어 초기 분석 결과를 사용합니다.")
        result_text_gpt = initial_analysis

    print("\n[최종 분석 결과]")
    print(result_text_gpt)
```

- 프롬프트 최적화:
    - HTP 검사 체계에 맞춘 단계별 세분화 프롬프트 설계
    - LLM이 그림과 일치하는 최적 결과를 도출하도록 주기적 테스트 및 최적화 진행

```
PROMPT = '''
        당신은 HTP(House-Tree-Person) 심리검사 분석 전문가입니다. 주어진
그림을 분석해 주세요.
        분석 방법
        1단계: 관찰된 특징들
        그림에서 보이는 구체적인 특징들을 나열해 주세요:

        집: 크기, 창문, 문, 지붕, 굴뚝 등의 특징
        나무: 크기, 줄기, 가지, 잎, 뿌리 등의 특징
        사람: 크기, 자세, 얼굴, 옷차림 등의 특징
        전체: 배치, 선의 굵기, 그림 스타일 등

        2단계: 심리적 해석
        각 요소가 나타내는 심리적 의미를 설명해 주세요:

        집 → 가족관계, 안정감, 소속감
        나무 → 성장욕구, 생명력, 적응력
        사람 → 자아상, 대인관계, 정서상태

        3단계: 핵심 감정 키워드
        분석 결과를 바탕으로 주요 감정 키워드를 3-5개 제시해 주세요.
        형식: 키워드만 한 줄씩 나열 (예: 불안, 안정감, 소외감)

        **작성 규칙**
        - 모든 답변은 한글로 '~입니다' 체로 작성
        - 단정적 표현보다는 '~로 보입니다', '~한 경향을 나타냅니다' 등 완화된
표현 사용
        - 부정적 해석과 긍정적 해석을 균형있게 제시
        - 이제 주어진 HTP 그림을 분석해 주세요.
        '''
```

4.2 LLM 챗봇 프롬프트 엔지니어링

- 목적: 성격 유형별 말투와 특징을 반영한 프롬프트 엔지니어링을 통해 개별화된 LLM 챗봇 제공
- 공통 프롬프트 설계
    - 기본 답변 우선 생성 후 개별 프롬프트에 체이닝

- 윤리적 문제 대응을 절대 우선 사항으로 설정하고 기본 소통 원칙을 공통 규칙으로 제공

- _generate_common_response 함수

```
        def _generate_common_response(self, session: ChatSession, messages:
list, user_message: str, **context) -> Tuple[str, Dict[str, int]]:

        """1단계: 공통 규칙으로 기본 답변 생성"""

        try:

        # 공통 규칙 프롬프트 로드

            common_rules = self.chained_prompt_manager.load_common_rules()

            # 사용자 닉네임 가져오기

            user_nickname = context.get('user_nickname', '사용자')

            # 그림검사 분석 결과 컨텍스트 준비

            user_analysis_context = ""

            if context.get('user_analysis_result'):

                analysis_result = context['user_analysis_result']

                user_analysis_context = f"""

[사용자 그림검사 분석 정보]

이 사용자의 그림검사 분석 결과를 참고하여 더 개인화된 상담을 제공해주세요:

- 주요 심리 특성: {analysis_result.get('result_text', '분석 정보 없음')}

- 분석 요약: {analysis_result.get('raw_text', '')[:200]}...


위 정보를 바탕으로 사용자의 심리 상태와 성향을 고려한 답변을 생성해주세요."""

            # 대화 요약 컨텍스트 준비

            conversation_context = ""

            if session.conversation_summary:

                conversation_context = f"""

[과거 대화 요약]

{session.conversation_summary}

위 요약은 이전 대화의 핵심 내용입니다. 이를 참고하여 대화의 연속성을 유지해주세요."""
```

- 개별 프롬프트 설계

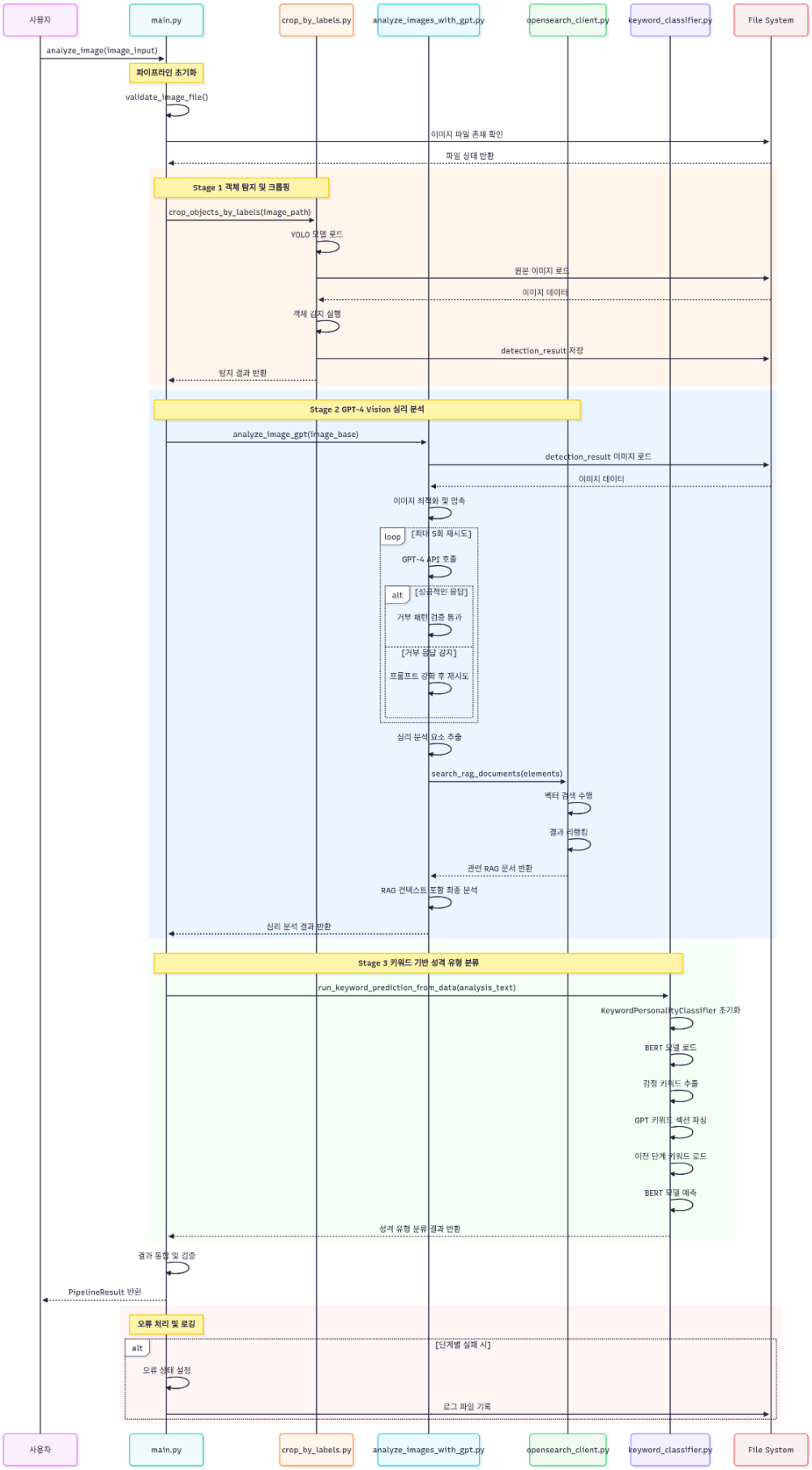
- 페르소나 성격에 맞춘 기본 답변 조정(체이닝 시스템)
- 소통 어조 중심의 캐릭터성 강조
- 응답 예시 제공
- 체이닝 모듈(_transform_to_persona)

```
# 페르소나 매핑

            persona_mapping = {

                "내면형": "nemyeon", "추진형": "chujin", "관계형":
"gwangye", "안정형": "anjeong", "쾌락형": "querock”}

            persona_key = persona_mapping.get(persona_type)

            if persona_key is None:

                raise ValueError(f"유효하지 않은 persona_type 값입니다:
{persona_type}")

            persona_prompt =
self.chained_prompt_manager.load_persona_prompt(persona_key)



            # 공통 규칙도 함께 로드

            common_rules =
self.chained_prompt_manager.load_common_rules()
```

- 프롬프트 최적화
  - 챗봇별 성격 반영을 통한 차별성 확보
  - 팀원마다 각자 1개의 성격유형을 맡아 프롬프트 고도화 진행
    (김승학-내면형,남의헌-안정형,이정민-관계형,이지복-추진형,조성지-쾌락형)

4.3. 시퀀스 다이어그램

그림검사_시퀀스다이어그램.png


사용자
main.py
crop_by_labels.py
analyze_images_with_gpt.py
opensearch_client.py
keyword_classifier.py
File System
analyze_image(image_input)
파이프라인 초기화
validate_image_file()
이미지 파일 존재 확인
파일 상태 반환
Stage 1 객체 탐지 및 크롭핑
crop_objects_by_labels(image_path)
YOLO 모델 로드
원본 이미지 로드
이미지 데이터
객체 감지 실행
detection_result 저장
탐지 결과 반환
Stage 2 GPT-4 Vision 심리 분석
analyze_image_gpt(image_base)
detection_result 이미지 로드
이미지 데이터
이미지 최적화 및 압축
loop
[최대 5회 재시도]
GPT-4 API 호출
alt
[성공적인 응답]
거부 패턴 검증 통과
[거부 응답 감지]
프롬프트 강화 후 재시도
심리 분석 요소 추출
search_rag_documents(elements)
벡터 검색 수행
결과 리랭킹
관련 RAG 문서 반환
RAG 컨텍스트 포함 최종 분석
심리 분석 결과 반환
Stage 3 키워드 기반 성격 유형 분류
run_keyword_prediction_from_data(analysis_text)
KeywordPersonalityClassifier 초기화
BERT 모델 로드
감정 키워드 추출
GPT 키워드 섹션 파싱
이전 단계 키워드 로드
BERT 모델 예측
성격 유형 분류 결과 반환
결과 통합 및 검증
PipelineResult 반환
오류 처리 및 로깅
alt
[단계별 실패 시]
오류 상태 설정
로그 파일 기록
사용자
main.py
crop_by_labels.py
analyze_images_with_gpt.py
opensearch_client.py
keyword_classifier.py
File System

5. 예외 처리 및 보안

5.1 예외 처리

- GPT가 그림분석을 거부하는 경우에 대한 예외 처리 설계
    - ‘I’m sorry’, ‘죄송합니다’ 등 분석 거부 표현 검출 시 재검사 최대 5회까지 수행
    - 재시도 시 프롬프트 강화 후 분석 재시도

```
for attempt in range(max_retries):
        try:
            # 재시도 시 프롬프트 강화
            if attempt > 0:
                enhanced_prompt = f"""
                {prompt}

                [중요] 이전 시도에서 이미지 분석이 거부되었습니다.
                이번에는 반드시 이미지의 시각적 요소들을 관찰하여 HTP 심리검사
분석을 수행해주세요.
                이미지가 흐리거나 불분명하더라도 보이는 요소들(선, 모양, 크기,
위치 등)을 바탕으로 분석해주세요.
                완전한 거부보다는 관찰 가능한 요소라도 분석해주시기 바랍니다.
                """
            else:
                enhanced_prompt = prompt
```

    - 예외 처리 코드 예시:

```
# 거부 응답 패턴 확인

            is_rejection = False

            for pattern in rejection_patterns:

                if pattern.lower() in result_text.lower():

                    is_rejection = True

                    print(f"거부 응답 패턴 감지: '{pattern}' (시도
{attempt + 1}/{max_retries})")

                    break

            # 거부 응답이 아니거나 마지막 시도라면 결과 반환

            if not is_rejection or attempt == max_retries - 1:

                if is_rejection and attempt == max_retries - 1:

                    print(f"경고: 모든 재시도가 실패했습니다. 마지막 응답을
반환합니다.")

                return result_text
```

5.2 보안 관리

- Frontend와 Backend에서 필요한 API 키를 각 폴더 하위의 `.env` 파일로 관리
- 코드 상에서 `load_dotenv()`를 호출하여 직접 노출 방지

- 환경 변수 예시:

```
from dotenv import load_dotenv

load_dotenv()

OPENAI_API_KEY = os.getenv("OPENAI_API_KEY")
```

6. 코드 모듈화 및 주석

6.1 모듈화

- 각 기능을 별도의 모듈로 분리하여 main.py에서 유지보수 용이성 확보
    - 예시: `backend/llm/model` 디렉터리 내에 `crop_by_labels.py`, `analyze_images_with_gpt.py`, `keyword_classifier.py` 등 모듈화 파일 구성
    - 필요한 모듈만 호출하여 `main.py`에서 실행

6.2 주석

- 주석 예시

```
def optimize_image_for_gpt(image_path: str, max_size: tuple = (1024, 1024), quality: int = 85) -> tuple:
    """
    GPT API 호출을 위해 이미지를 최적화

    Args:
        image_path (str): 원본 이미지 경로
        max_size (tuple): 최대 크기 (width, height)
        quality (int): JPEG 압축 품질 (1-100)

    Returns:
        tuple: (optimized_base64_string, compression_info)
    """
    try:
        # 원본 파일 크기 확인
        original_size = os.path.getsize(image_path)

        # 이미지 로드
        with Image.open(image_path) as img:
            # EXIF 회전 정보 적용
            img = ImageOps.exif_transpose(img)

            # RGB로 변환
            if img.mode != 'RGB':
                img = img.convert('RGB')
```

7. 서비스 기능 소개

1) 랜딩 페이지
    - 구글 소셜 로그인(사용자 편의성 고려)
    - 닉네임 설정 기능 제공(챗봇이 사용자를 부를 호칭, 중복 방지)
2) 메인 페이지
    - 서비스 소개 제공
    - 챗봇 시연 영상 첨부(그림검사 이후에만 챗봇 사용 가능하므로 진입 장벽 완화 목적)
3) 그림검사 페이지

    a. 그림 업로드

    - 사용자가 그림 업로드 또는 직접 그리고 저장(그림판 기능)
    - 그림 분석 진행 중 스피너 표시(평균 1분 소요)
    - 그림분석 3단계 절차
        1) 이미지 처리: YOLO 기반 객체 탐지 후 라벨링 이미지 생성 및 GPT-4o에 전달
        2) 패턴 분석: GPT-4o 기반 LLM과 OpenSearch 벡터 데이터베이스 연동을 통한 심리검사 수행 및 텍스트 요약본 생성
        3) 결과물 생성: 텍스트 요약본을 BERT 분류 모델에 입력하여 5가지 성격 유형 중 최고 확률 유형 제공

    b. 그림검사 결과페이지

    - 그림검사 요약본 텍스트와 분류된 성격 유형 제공
    - 이미지 저장 기능 제공
    - 만족도 조사 기능 제공(thumbs up / thumbs down)
4) 페르소나 소개 페이지
    - 페르소나 5유형 및 각 한줄 소개 제공
    - '{페르소나 이름}과 대화하기' 버튼을 통해 새 채팅 세션 이동(그림검사 전에는 버튼 비활성화)
    - 뱃지 제공
        - 대화중: 가장 최근 대화한 페르소나
        - 매칭됨: 가장 최근 그림검사에서 매칭된(확률이 가장 높은) 페르소나
5) 챗봇 페이지

    a. 대화 화면

    - 중앙에 페르소나 캐릭터 배치 및 최근 답변 표시
    - {닉네임}님, {닉네임} 그대 등 캐릭터 성격에 맞춘 호칭 제공(프롬프트 엔지니어링 적용)

- 첫 메시지 시 최근 그림검사 텍스트 요약본 기반 대화 시작

b. 사이드탭

- 일반 채팅 UI 형태로 사용자/챗봇 실시간 채팅 내역 확인 가능
- ‘다른 캐릭터와 대화하기’ 버튼을 통해 만족도 조사 후 페르소나 소개 탭으로 이동

6) 마이페이지

a. 채팅 히스토리
   - 최근 대화한 페르소나 순서로 배치
   - ‘이어서 대화하기’ 버튼으로 동일 세션에서 채팅 지속 가능

b. 그림 검사 결과
   - 그림검사 결과 텍스트. 사용자의 그림, 매칭된 페르소나, 각 페르소나별 분류 확률 확인 가능

c. 기타 기능
   - 닉네임 수정, 프로필 이미지 등록, 회원 탈퇴