

SK네트웍스 Family AI 과정 19기

모델링 및 평가 LLM 활용 소프트웨어

산출물 단계	모델링 및 평가
평가 산출물	LLM 활용 소프트웨어
제출 일자	2026.1.12
깃허브 경로	https://github.com/SKN19-Final-4team
작성 팀원	김종민

1. 시스템 개요

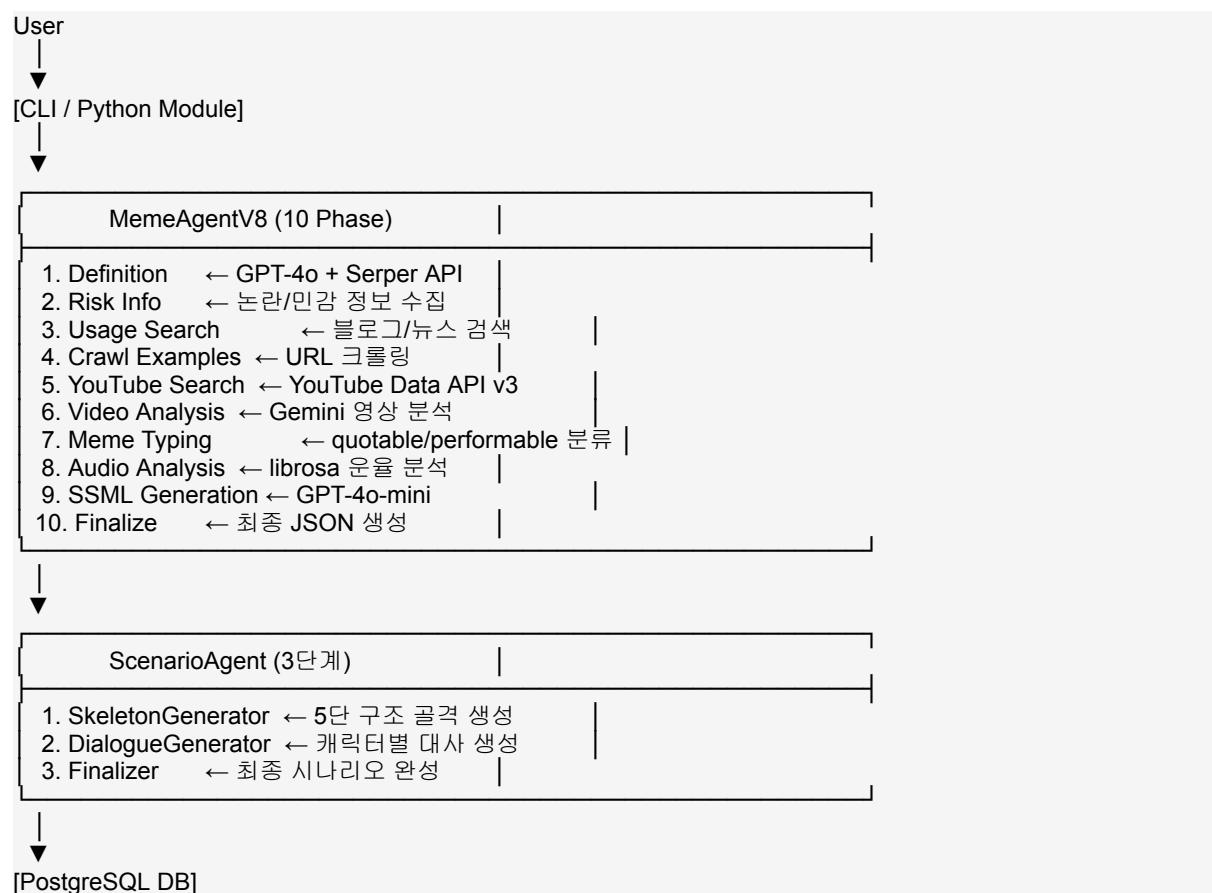
본 소프트웨어는 AI 기반 밍 콘텐츠 생성 시스템으로, LangGraph Agent와 다중 LLM(GPT-4o, Gemini)을 활용하여 밍 데이터를 수집하고, 캐릭터가 밍을 따라하는 웃음 영상 시나리오를 자동 생성합니다.

기능	설명
밈 수집	LangGraph StateGraph 기반 10단계 파이프라인으로 밍 정의, 위험 정보, YouTube 영상 분석
시나리오 생성	5단 구조(hook→meme→punchline) 코미디 시나리오 자동 생성
영상 생성 프롬프트	Gemini 영상 분석으로 Hedra용 motion_prompt 생성

 API Key 등 민감 정보는 환경 변수로 관리되며, Rate Limit 재시도 및 프롬프트 최적화를 통해 안정적인 응답을 보장합니다.

2. 시스템 구성 요소

전체 아키텍처



DB 테이블 구조

카테고리	테이블
밈 데이터	meme, youtube, video, audio
시나리오	script, scene_clip, generated_video
시스템	job, cost_record, prompt_version

3. 코드 구조

밈 수집 Agent (/scripts/)

```
scripts/
  └── __init__.py          # 최상위 export (MemeAgentV8)
  └── agent/               # 밤 수집 Agent
    ├── meme_agent_v8.py    # 메인 Agent (LangGraph)
    ├── state.py            # AgentV8State 정의
    ├── prompts.py          # 프롬프트 중앙화
    ├── tools_v8.py          # LangChain 도구 정의
    └── nodes/               # Phase별 노드 함수
      ├── definition.py     # 정의/기원 수집
      ├── risk_info.py       # 위험 정보 수집
      └── video_analysis.py # Gemini 영상 분석
    ...
  └── analysis/             # 분석 모듈
    ├── video/
    │   ├── analyzer.py     # Gemini 영상 분석
    │   └── search.py       # YouTube 쇼츠 검색
    ├── audio/
    │   ├── extractor.py    # 오디오 구간 추출
    │   └── voice_analyzer.py # librosa 운율 분석
    └── tts/
      └── ssml_generator.py # LLM 기반 SSML 생성
  └── utils/
    ├── retry.py            # Rate Limit 재시도
    └── json_parser.py      # JSON 추출 유ти
```

시나리오 생성 (/backend/)

```
backend/
  └── scenario/
    ├── agent.py            # ScenarioAgent (오케스트레이터)
    ├── schema.py           # Pydantic 스키마
    └── generators/
      ├── skeleton.py        # 골격 생성
      ├── dialogue.py        # 대사 생성
      └── finalizer.py       # 최종화
    └── prompts/
      ├── templates.py        # 프롬프트 템플릿
      └── manager.py          # 버전 관리
    └── evaluation/
      └── evaluator.py        # 품질 평가
  └── db/
    ├── connection.py        # PostgreSQL 연결
    └── repository.py        # Repository 패턴
```

4. 주요 코드

4.1 Rate Limit 재시도 (예외 처리)

```
# scripts/utils/retry.py
import time
from functools import wraps

def retry_on_rate_limit(max_retries: int = 3, wait_seconds: int = 20):
    """
        Rate Limit 발생 시 자동 재시도하는 데코레이터
        LLM API 호출의 안정성을 보장
    """
    def decorator(func):
        @wraps(func)
        def wrapper(*args, **kwargs):
            for attempt in range(max_retries):
                try:
                    return func(*args, **kwargs)
                except Exception as e:
                    if "rate_limit" in str(e).lower() and attempt < max_retries - 1:
                        print(f"Rate limit hit. Waiting {wait_seconds}s...")
                        time.sleep(wait_seconds)
                    else:
                        raise
            return wrapper
        return decorator
```

4.2 환경 변수 설정

.env 파일

```
OPENAI_API_KEY=sk-xxx          # GPT-4o, Whisper, SSML 생성
SERPER_API_KEY=xxx             # Google/YouTube 검색
GEMINI_API_KEY=xxx             # Gemini 영상 분석
DATABASE_URL=postgresql://...
```

로드 방식

```
from dotenv import load_dotenv
import os

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

4.3 LangGraph Agent 구조

```
# scripts/agent/meme_agent_v8.py
from langgraph.graph import StateGraph, END
from scripts.agent.state import AgentV8State

class MemeAgentV8:
    """
        멈 데이터 수집 통합 Agent (10 Phase 파이프라인)
        LangGraph StateGraph 기반으로 단계별 처리
    """
    def __init__(self):
        self.graph = self._build_graph()

    def _build_graph(self) -> StateGraph:
        workflow = StateGraph(AgentV8State)
```

```

# 노드 등록
workflow.add_node("definition", definition_node)
workflow.add_node("risk_info", risk_info_node)
workflow.add_node("video_analysis", video_analysis_node)
# ... 10개 Phase 노드

# 조건부 분기 (quotable만 오디오 분석)
workflow.add_conditional_edges(
    "meme_typer",
    lambda s: "audio_analysis" if s["needs_audio"] else "finalize"
)
return workflow.compile()

```

5. 프롬프트 최적화

5.1 임 정보 수집 프롬프트

```

# scripts/agent/prompts.py
def get_finalize_prompt(meme_name: str, crawled_data: dict) -> str:
    """
    수집된 데이터를 기반으로 최종 JSON 생성 프롬프트
    할루시네이션 방지: 크롤링된 원문만 사용하도록 제약
    """
    return f"""
당신은 임 데이터 분석 전문가입니다.

[수집된 데이터]
{json.dumps(crawled_data, ensure_ascii=False)[:4000]}

[규칙]
1. 수집된 원문에 없는 정보는 절대 생성하지 마세요
2. usage_examples는 반드시 source_url 포함
3. original_quote는 원문 그대로 복사

임 "{meme_name}"에 대한 최종 JSON을 생성하세요.
"""

```

5.2 motion_prompt 생성 (Gemini)

```

# scripts/analysis/video/analyizer.py
def generate_motion_prompt(video_id: str, meme_name: str) -> dict:
    """
    Gemini가 영상을 직접 시청하고 영상생성용 프롬프트 생성
    50-100 단어의 상세한 영어 프롬프트 출력
    """
    prompt = """
Analyze this meme video and generate a motion prompt for AI video generation.

Output JSON:
{
    "motion_prompt": "Detailed 50-100 word description...",
    "motion_sequence": [{"time": "0-2s", "action": "..."}, ...],
    "body_parts": {"head": "...", "face": "...", "arms": "..."},
    "style_keywords": ["cute", "kawaii", "anime"],
    "duration_seconds": 5.0
}
"""

    # Gemini API 호출 (영상 + 프롬프트)
    return gemini_client.analyze_video(video_id, prompt)

```

5.3 시나리오 골격 프롬프트 (토큰 최적화)

```
# backend/scenario/prompts/templates.py
def get_skeleton_prompt(meme_data: dict) -> str:
    """
    5단 구조 시나리오 골격 생성
    코미디 패턴 포함: Rule of Three, 티키타카
    """

    # 필수 정보만 추출하여 토큰 수 최적화
    essential = {
        "name": meme_data["name"],
        "meme_type": meme_data["meme_type"],
        "key_phrase": meme_data.get("key_phrase"),
        "motion_prompt": meme_data.get("video_generation", {}).get("motion_prompt")
    }

    return f"""
5단 구조로 30초 짧은 시나리오를 생성하세요:
1. hook (3초) - 시청자 관심 유도
2. setup (5초) - 상황 설정
3. buildup (7초) - 긴장 고조
4. meme (10초) - 밈 재현 (핵심!)
5. punchline (5초) - 반전/웃음

밈 정보: {json.dumps(essential, ensure_ascii=False)}
대사는 [PLACEHOLDER: 설명] 형태로 남기세요.
    """
```

6. 보안 고려 사항

항목	적용 방식
API Key 관리	.env 파일 + python-dotenv 로드
Git 보안	.env는 반드시 .gitignore에 추가
DB 인증정보	환경 변수 DATABASE_URL로 분리
로그 마스킹	API Key가 로그에 노출되지 않도록 처리

.gitignore

```
.env
*.env.*
data/raw/audio/
data/raw/video_cache/
```

7. 테스트 시나리오

7.1 밈 데이터 수집 테스트

실행

```
uv run python -m scripts.agent.meme_agent_v8 "나니가 스키"
```

입력: 밈 이름 "나니가 스키"

출력 예시:

```
{  
    "basic_info": {  
        "name": "나니가 스키",  
        "definition": "..."  
    },  
    "meme_classification": {  
        "meme_type": "quotable",  
        "key_phrase": "何が好き"  
    },  
    "video_analysis": {  
        "time_stamp": {"start": 2.8, "end": 3.4}  
    },  
    "video_generation": {  
        "motion_prompt": "Cute character saying 'Nani ga suki'..."  
    }  
}
```

Gemini 임 구간 탐지 + motion_prompt 생성 확인

7.2 시나리오 생성 테스트

실행

```
uv run python -m backend.scenario.agent --meme-id 2 --evaluate
```

입력: **meme_id=2** (나니가 스키)

출력: **5단 구조 시나리오 + 품질 점수 (0-100)**

8. 평가요소표

평가 항목	대응 내용
LLM 연동 및 프롬프트 최적화	GPT-4o + Gemini 멀티 LLM 구조, 토큰 수 제한 및 필수 정보만 추출
예상치 못한 상황 예외 처리	retry_on_rate_limit() 데코레이터, try/except로 API 오류 처리
코드 모듈화 및 주석 작성	Phase별 노드 분리, Generator 패턴, 함수별 docstring 포함
보안 정보 노출 방지	.env 파일 + 환경 변수 활용, .gitignore 등록
빠른 응답을 위한 프롬프트 최적화	필수 정보만 추출, 중복 제거, 토큰 수 제한 (4000자)
할루시네이션 방지	크롤링된 원문만 사용하도록 프롬프트 제약
품질 평가 체계	ScenarioEvaluator (100점 만점 자동 평가)

9. 기술 스택 요약

구분	기술
Agent Framework	LangGraph (StateGraph)
LLM	GPT-4o (정확), GPT-4o-mini (SSML), Gemini (영상 분석)
검색 API	Serper (Google, YouTube), YouTube Data API v3
음성 분석	librosa (운율), OpenAI Whisper (fallback)
DB	PostgreSQL (memedb, 21개 테이블)
패키지 관리	uv
영상 생성 (예정)	Hedra Character-3 API