

**SK네트웍스 Family AI과정 14기  
 모델링 및 평가 시스템 아키텍처**

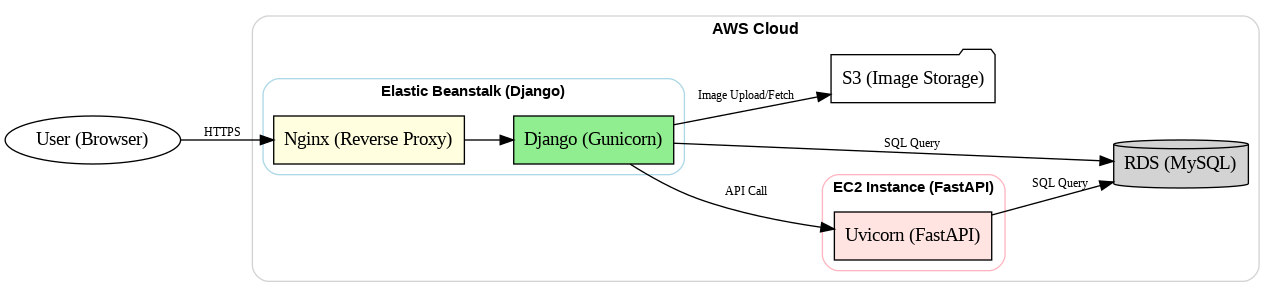


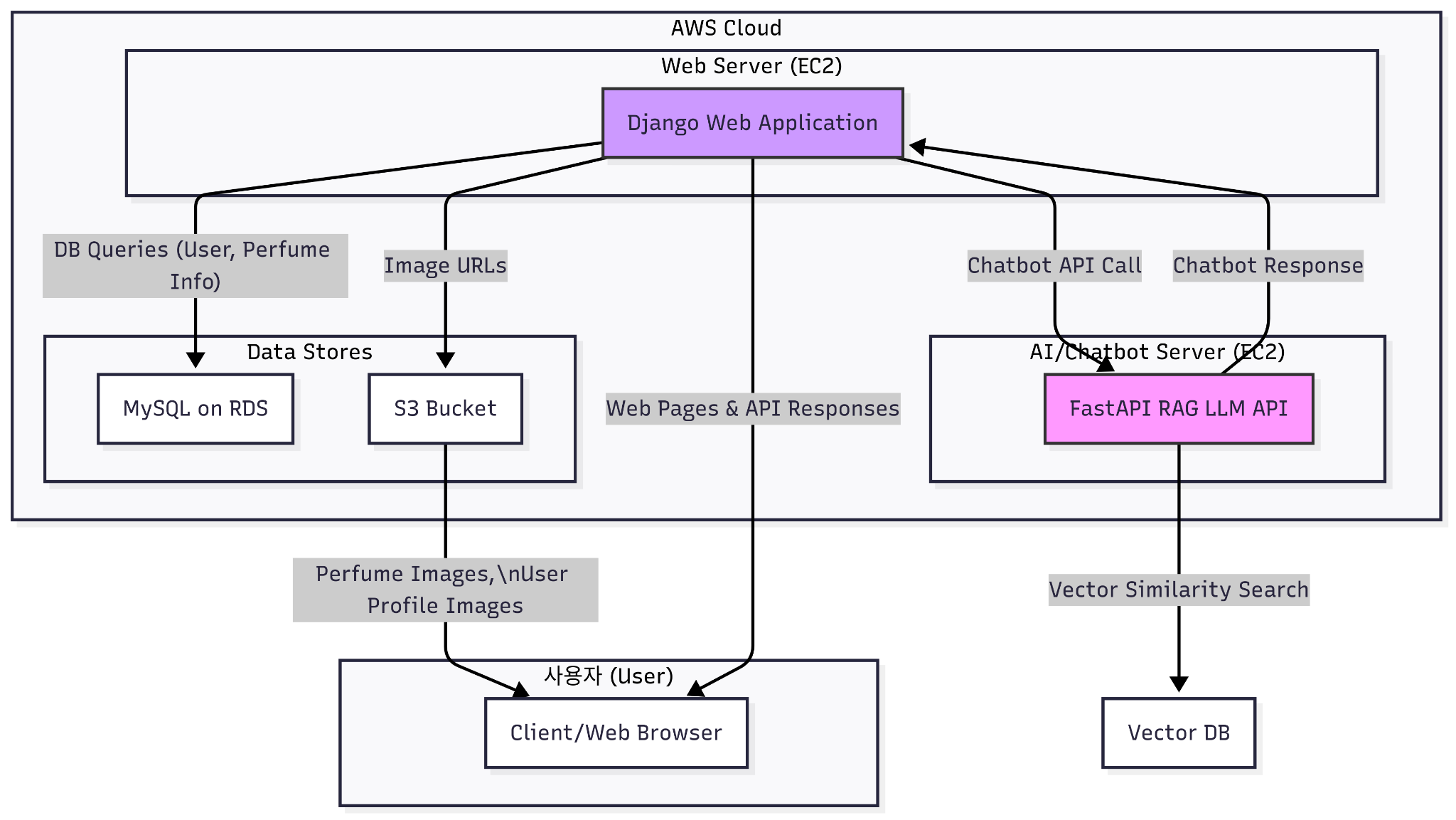
**□ 개요**

* 산출물 단계 : 모델링 및 평가
* 평가 산출물 : 시스템 아키텍처
* 제출 일자 : 2025.09.26
* 깃허브 경로 : [skn-ai14-250409/SKN14-Final-2Team: Repository for SKN14-Final-2Team](https://github.com/skn-ai14-250409/SKN14-Final-2Team)
* 작성 팀원 : 전정규, 박빛나

| **컴포넌트 다이어그램** | * 컴포넌트 다이어그램 * 설명 |
| --- | --- |
| **시퀀스 다이어그램** | * 향수 추천 챗봇 Langgraph 시퀀스 다이어그램 * 주요 흐름 |
| **액티비티 다이어그램** | * 구성 * 주요 액션 노드 |

**1. 컴포넌트 다이어그램**

****



**구성 요소**

1. 클라이언트(Client):

- 컴포넌트:

- 웹 브라우저 (Chrome 등)

- Django 웹 페이지 (HTML, CSS, JS)

- 인터페이스:

- 소셜 로그인 (Google, Naver, Kakao OAuth 2.0)

- HTTPS 기반 JSON 요청/응답

- 이미지/정적 리소스 렌더링

1. AWS Cloud:

- Web Server (EC2):

- Django Web Application

- 메인 웹 프레임워크 (템플릿 렌더링, 라우팅, REST API)

- Nginx

- 리버스 프록시 및 정적 파일 제공

- Gunicorn

- ASGI 서버로 Django 실행

- AI/Chatbot Server (EC2):

- FastAPI RAG LLM API

- 추천 및 대화형 질의 처리

- 벡터 검색(Vector DB)과 연결

- ML 모델 추론 수행

1. 데이터 스토어 (Data Stores):

- Amazon RDS (MySQL)

- 사용자 정보, 향수 메타데이터 저장

- Amazon S3 Bucket

- 향수 이미지, 사용자 프로필 이미지 저장

- Vector DB

- 임베딩 기반 유사도 검색 (향수 추천 시 활용)

1. 외부 서비스:

- OpenAI ChatGPT API

- 대화형 답변 및 언어 모델 기반 추천

- RunPod / Hugging Face Inference

- ML, DL 모델 추론 환경

**설명**

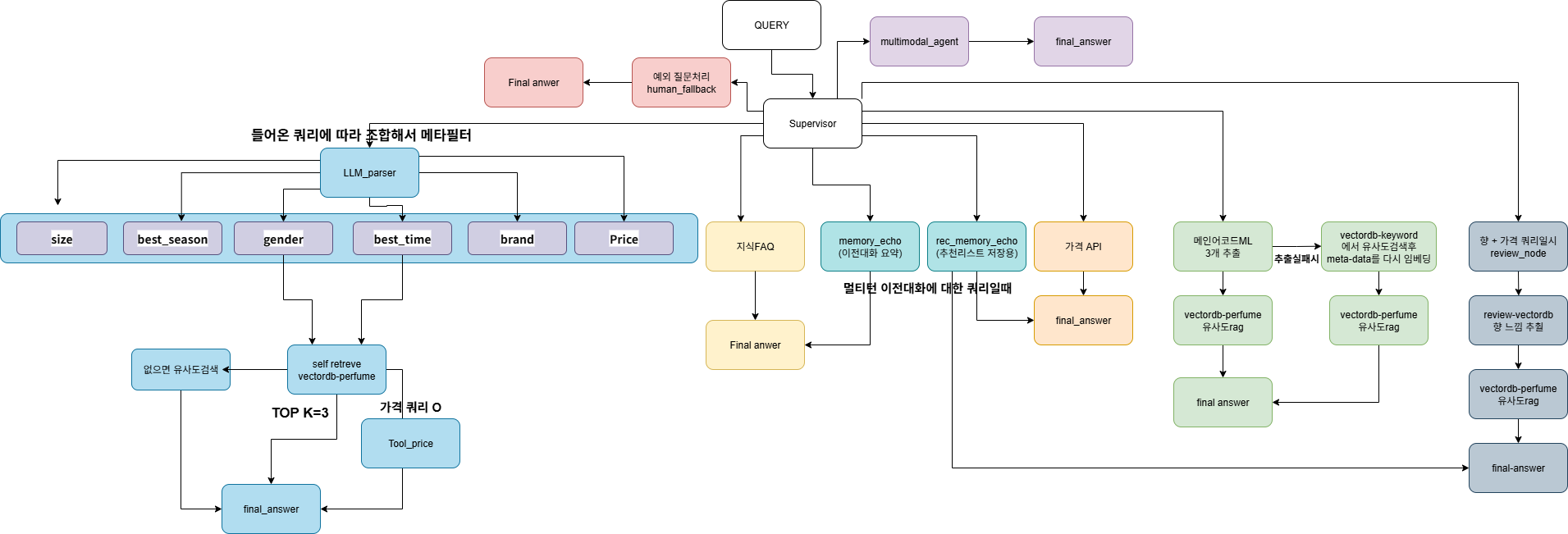
- 사용자는 웹 브라우저를 통해 Django 웹 애플리케이션에 접속하며, 소셜 로그인(OAuth 2.0)을 통해 인증을 수행

- Django Web Application은 MySQL (RDS)와 S3 Bucket에서 데이터/이미지를 불러오고, 필요한 경우 FastAPI 서버에 API 요청을 전달

- FastAPI 서버는 Vector DB를 통해 유사도 검색을 수행하고, ML 모델 및 ChatGPT API를 활용해 추천/대화 응답을 생성

- 최종적으로 처리된 데이터와 추천 결과가 다시 사용자 브라우저에 반환되어 화면에 표시됨

**2. 향수 추천 챗봇 Langgraph 시퀀스 다이어그램**

****

**구성 요소**

1. supervisor: 사용모델 gpt-4o-mini

- 구성 역할:

역할: 사용자 메시지 해석해서 정확히 하나의 에이전트로 라우팅함. 입력은 USER\_QUERY, REC\_CONTEXT, LAST\_AGENT, IMAGE\_URL 참고함. 출력은 JSON만 반환함

메타의도 1: 사용자가 방금 자기 말 묻으면 → memory\_echo 로 보냄

메타의도 2: 직전 추천을 가리키는 말(방금/1번/2번/이름/노트 등) + REC\_CONTEXT 존재 시

* 가격·딜 질문이면 → price\_agent 로 보냄 (followup=true)
* 목록 재표시·요약이면 → rec\_echo 로 보냄 (followup=true)
* 상세·비교·노트 등 다른 후속이면 → ML\_agent 로 보냄 (followup=true)
* 이미지: IMAGE\_URL 있으면 무조건 → multimodal\_agent 선택함
* 비-향수 키워드(데오드란트/섬유유연제/방향제 등) 포함 시 → human\_fallback 로 보냄
* 5대 파싯 ≥1 → LLM\_parser
* 5대 파싯 =0 → review\_agent

## 2. LLM\_parser

* 구성 역할: 복합 제품 파싯(brand/season/gender/sizes/day\_night\_score)이 섞인 질의를 구조화함.
* 동작: supervisor가 넘긴 질의를 JSON 슬롯으로 정규화 → 방어로직(허용값 매핑/오탈자 보정/NULL 정리) 적용 → vectordb에 메타데이터 필터 걸어 RAG 수행함.
* 폴백: 필터 히트가 없으면 유사도 기반으로 상위 결과를 가져와 추천 본문 생성함.
* 출력: 사용자 답변, rec\_history.items({brand,name,size,detail\_url}), 파싱 결과, 검색 매치 반환함.
* 비고: 가격 언급이 있어도 파싯이 1개 이상이면 LLM\_parser가 우선임.

## 3. 예외 질문처리 노드(human\_fallback)

* 구성 역할: 향수와 무관한 질의(예: “점심 뭐 먹었어?”)를 안전하게 차단/유도 응답함.
* 동작: 비-향수 키워드/의도 감지 → “향수 관련 구체 질의로 다시 요청” 안내함.
* 수집: 추후 DB에서 예외 사례 통계/패턴 수집하려는 용도로 로그 남김.

## 4.지식 FAQ 노드(FAQ\_agent)

* 구성 역할: 정의/차이/용어 질문 처리함(예: “베이스 노트가 뭐야?” “EDT랑 EDP 차이?”).
* 동작: 현재는 LLM 내재 지식에 의존해 간결·정확한 설명 생성함.
* 로드맵: FAQ/QnA 데이터가 쌓이면 DB 기반 답변으로 점진 전환 예정임(출처/참고 링크 확장 가능)

## 5. 메인어코드 ML 노드(ML\_agent)

* 구성 역할: 단일 취향/무드 중심 추천 담당함(예: “달달한 향 추천”, “포근한 겨울향”).
* 동작: 취향 키워드(향/무드) 메인어코드 다중분류로 추출 → 임베딩 생성 →perfume-vectordb 유사도 검색 → 상위 N개 후보를 표준 스키마로 정리해 추천함.  
  추출실패시 keyword-vectordb에서 노트 추출 -> 다시 perfume-vectordb로 가서 rag후 추천천
* 팔로업 지원: “2번 노트 알려줘/비교해줘” 같은 직전 추천 후속 질의 처리함.
* 출력: 사용자 답변과 함께 rec\_history.items 저장해 rec\_echo가 바로 재표시할 수 있게 함.

## 6. 가격 API 노드(price\_agent)

* 구성 역할: 가격 전용 단일 질문 처리함(예: “샤넬 No.5 가격 알려줘”, “10만원대 향수 추천해줘”).
* 동작: 네이버 쇼핑 API용 검색 키워드 추출 → 가격/상품명/링크 수집 → 예산(budget/budget\_min~max) 규칙에 맞게 필터 후 요약함.
* 출력: 최저가 중심의 간결한 가격 섹션 또는 예산 내/외 구분 결과 반환함.
* 비고: 브랜드/제품이 없고 “만원대/이하/이상” 같은 가격만 있는 질의도 처리함.

## 7. 멀티모달 노드(multimodal\_agent)

* 구성 역할: 사용자가 업로드한 이미지 단서(무드/색감/스타일)를 분석해 향수 추천에 반영함.
* 동작:
  1. state에서 image\_url 확인함(없으면 이미지 요청 응답).
  2. 사용자 최근 질의 텍스트와 함께 비전 LLM으로 이미지 분석(JSON 강제: day\_night\_score, gender, season, free\_text 등)함.
  3. 분석 JSON에 메타필터 적용해 안전한 슬롯만 살림.
  4. free\_text를 임베딩해 perfume-vectordb RAG 검색(top-k)함(+ 메타필터).
  5. 검색 결과로 최종 추천 본문 생성함.
* 입력: image\_url, 최근 사용자 텍스트 질의.
* 출력: 사용자 답변, search\_results, final\_answer, perfume\_list({id,brand,name,score,text}), rec\_history.items 저장함.
* 비고: 이미지 단서가 있으면 슈퍼바이저가 항상 multimodal\_agent로 라우팅하도록 설계함.

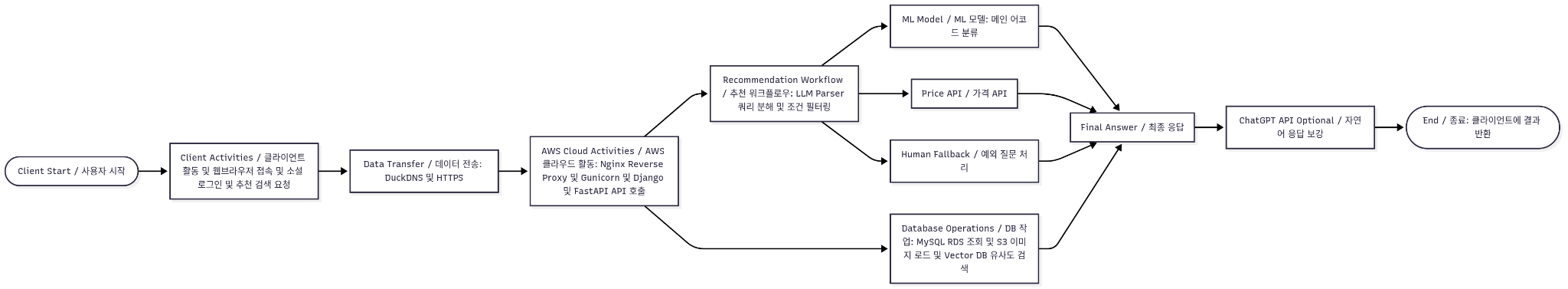
## 8. 추천 에코 노드(rec\_echo)

* 구성 역할: “방금 추천 다시 보여줘 / 2번 뭐였지?” 같은 재표시/요약 요청을 처리함.
* 동작:  
  1. rec\_history[-1].items가 있으면 그 목록을 사용함. 없으면 search\_results에서 recs/candidates/matches를 복원함.
  2. 직전 어시스턴트 답변과 items만으로 한 줄 요약을 시도함(요약 실패 시 깔끔한 플레인 리스트로 대체).
  3. followup\_reference.index가 있으면 해당 항목에 ⭐ 표시하여 포커스 제공함.
* 입력: rec\_history(우선), 없으면 search\_results. 선택적으로 router\_json.followup\_reference.index.
* 출력: 재표시 텍스트(final\_answer), messages, 필요 시 복원된 rec\_history 보정 저장함.
* 비고: 새 추천을 생성하지 않음. 기존 결과만 예쁘게 재정렬/요약해서 보여줌.

## 9. 메모리 에코 노드(memory\_echo)

* 구성 역할: “내가 방금 뭐라 했지?” 같은 직전 사용자 발화를 요약·되짚어주는 전용 노드임.
* 동작:
  1. 대화 스택에서 직전 사용자 질문과 그 직후 어시스턴트 답변을 찾아냄.
  2. 시스템 프롬프트 규칙에 따라 간결 요약 생성함(불필요한 추측 금지).
  3. 마지막에 원문 질문을 함께 인용해 사용자가 맥락을 바로 복구하게 함.
* 입력: 대화 messages에서 직전 Human/AI 턴.
* 출력: 요약 결과(final\_answer), messages 업데이트.
* 비고: 직전 발화를 못 찾으면 사용자에게 조금 더 대화를 이어달라고 안내함

**3. 액티비티 다이어그램**

****

1. 시작 지점
   1. 사용자가 웹 브라우저(Client) 에서 서비스에 접속
2. 클라이언트 활동
   1. Chrome 브라우저에서 웹페이지 접속
   2. Google / Naver / Kakao OAuth 2.0 로그인 (소셜 인증)
   3. 향수 검색·추천 요청 생성
3. 데이터 전송
   1. DuckDNS 도메인을 통해 요청이 AWS EC2 서버로 전달
   2. JSON 데이터가 HTTPS를 통해 안전하게 전송
4. AWS 클라우드 활동
   1. Nginx: 리버스 프록시로 요청 수신
   2. Gunicorn + Django: 웹 애플리케이션 처리 (템플릿, DB 연동)
   3. 필요한 경우 FastAPI 서버로 추천/챗봇 API 요청 전달
5. 추천/챗봇 워크플로우
   1. LLM Parser가 쿼리에서 주요 속성 추출 (season, gender, brand, price)
   2. DB 조회 (RDS, S3, Vector DB) → 없으면 유사도 검색
   3. ML 모델: 메인 어코드 분류
   4. 가격 API: 가격 정보 요청
   5. Human Fallback: 예외적인 질문 처리
6. 데이터베이스 작업
   1. MySQL (RDS): 사용자·향수 메타데이터 조회
   2. S3: 향수 이미지·프로필 이미지 로드
   3. Vector DB: 임베딩 기반 유사도 검색
7. 외부 서비스 호출
   1. 필요 시 ChatGPT(OpenAI API) 호출하여 자연어 응답 보강
8. 결과 반환
   1. Django를 통해 최종 응답(HTML/JSON/추천 결과)이 클라이언트로 반환
9. 종료 지점
   1. 사용자 브라우저에 추천 향수 및 챗봇 답변 표시

[첨부이미지 - 시스템 구성도]

