생성형 AI 기반 RAG 개발자 과정

Day3 최종 프로젝트
문서 업로드형 RAG 챗봇

Session 24 / 33 — Day 3

오늘 만든 모든 조각을 하나의 서비스로

박수현 · (사)한국정보공학기술사회 · 2026

LLM 이론 · LangChain · ChromaDB CRUD · Agent — 4일 핸즈온

완성 기능 체크리스트

📄 문서 관리 (CRUD)

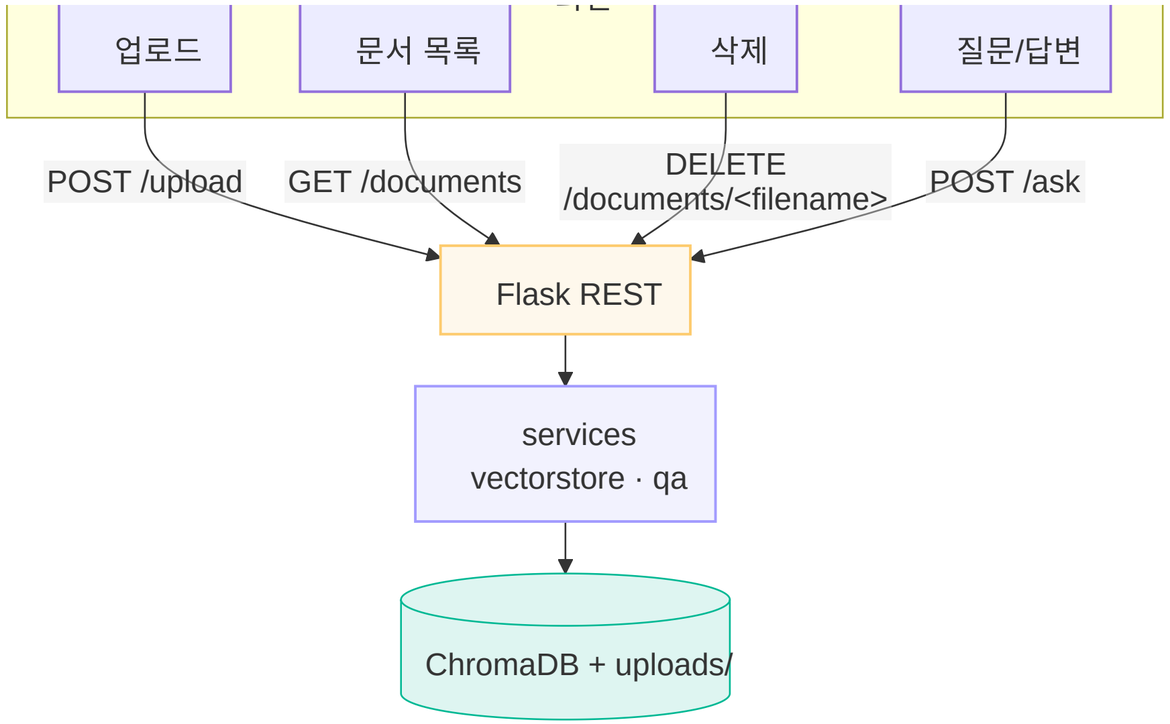
- ✅ PDF / TXT **업로드** (DOCX 확장 가능)
- ✅ 문서 **목록 조회**
- ✅ 문서 **삭제**

💬 대화

- ✅ ChatGPT 기반 **질의응답**
- ✅ **SSE 스트리밍** 답변
- ✅ **출처(Source) 표시**
- ✅ **Chat History** 유지

> 🎯 목표: 개별 구현한 API 를 **한 화면**에서 동작하는 완결된 서비스로 통합.

화면 ⇄ API ⇄ 서비스 통합



4개 화면 동작 → Flask REST(4개 API) → 서비스 계층 → ChromaDB+uploads/

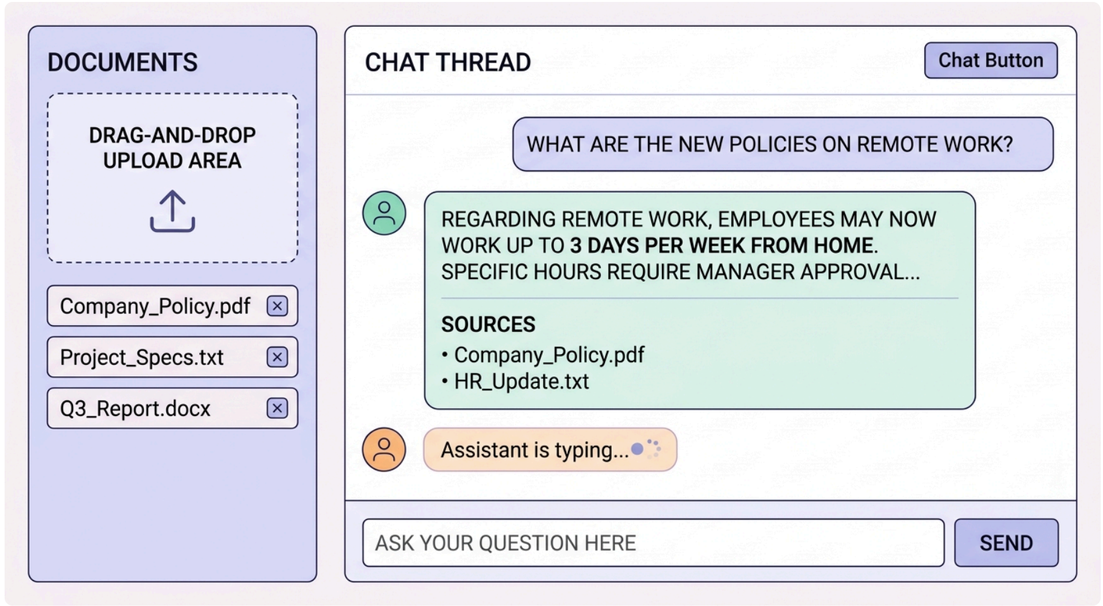
지금까지 구현한 구성요소의 총합

> 💡 4개 API + 서비스 계층 + 영속 저장 = 지금까지 구현한 구성요소의 총합.

📌 명칭 안내

- 파일관리 예제(`5.file_manager_restapi`)는 `/files` 계열
- 최종 프로젝트(`13.document_qa`)는 `/documents` 계열
- 이 세션은 **최종 프로젝트의** `/upload` · `/documents` · `/ask` 를 기준으로 한다.

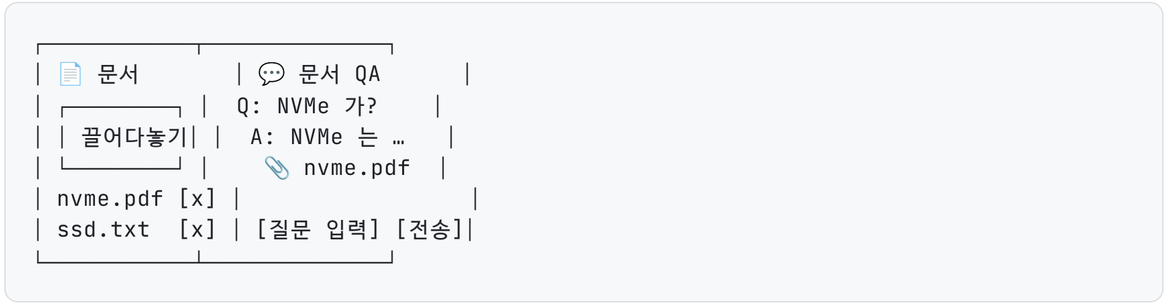
한 화면에 담기는 것 — 좌: 문서 / 우: 대화



최종 문서 QA 챗봇 UI · 좌측 업로드·문서목록(삭제), 우측 질의응답·스트리밍·출처(Source) 표시

> 🔑 프론트엔드는 JSON 을 수신해 화면에 렌더링하며, 로직은 전부 백엔드에 위치한다.

화면 영역 ⇄ 엔드포인트



영역별 연결

- **업로드 박스** → `POST /upload`
- **문서 목록 + [x]** → `GET /documents` / `DELETE`
- **질문 입력 → 답변** → `POST /ask`
- **출처 칩** ← 응답의 `sources`

> 🔑 좌측 사이드바(업로드·목록·삭제)와 우측 채팅(스트리밍·출처)으로 구성.

출처: Day 3 | 최종 프로젝트

토큰 단위로 출력되는 SSE 스트리밍 답변

검색은 LangChain, 생성은 스트리밍

- `similarity_search(question, k=3)` 로 근거 검색
- OpenAI `stream=True` 로 토큰 단위 생성
- SSE(`data: ...\n\n`)로 전송
- 종료 시 `sources` → `[DONE]` 전송

```
def generate():
    response = client.chat.completions.create(
        model="gpt-4o-mini", stream=True,
        messages=[...context + question...])
    for chunk in response:
        c = chunk.choices[0].delta.content
        if c:
            yield f"data: {json.dumps({'content': c})}\n\n"
    yield f"data: {json.dumps({'sources': sources})}\n\n"
    yield "data: [DONE]\n\n"
```

💡 동일한 답변이라도 **체감 응답성**이 크게 향상된다. 프론트엔드는 `content` 누적 → `sources` 표시 → `[DONE]` 종료 순으로 처리한다.

학습 결과물 — 직접 구현 항목

항목	구현
OpenAI API 사용	✅
Embedding 생성 · Vector Search	✅
ChromaDB 영속	✅
LangChain RAG Chain	✅
PDF/TXT 업로드 (DOCX 확장 가능)	✅
문서 목록 · 삭제 (CRUD)	✅
질의응답 + 출처 표시	✅
SSE 스트리밍 응답	✅
Chat History (멀티턴 확장)	✅
실제 RAG 웹서비스	✅

> 💡 여기까지가 사내 문서검색 챗봇 한 세트다. 문서만 교체하면 실무에 바로 투입할 수 있다.

내 서비스로 완성하기

핵심 (필수)

- [ ] 내 분야 문서 3~5개 업로드 → 목록 확인
- [ ] 5개 질문으로 **출처가 정확한지** 검증
- [ ] 문서 1개 삭제 → 그 내용이 답에서 사라지는지 확인

확장 (옵션)

- [ ] 답변이 **토큰 단위로 흐르는지**(스트리밍) 눈으로 확인
- [ ] 후속 질문("그럼 ~는?")으로 **Chat History** 동작 확인
- [ ] `style.css` 만 교체해 화면 디자인 바꾸기

핵심 5가지

1. ✅ **업로드·인덱싱·목록·삭제·질의응답**을 한 서비스로 통합
2. ✅ 4개 REST API + **서비스 계층** + **영속 저장**의 합
3. ✅ 프론트는 **정적 파일**(JSON 소비)만 담당하고, 로직은 백엔드에
4. ✅ 답변은 **SSE 스트리밍** + **출처** 로 신뢰·체감응답성 확보
5. ✅ 문서만 바꾸면 **어떤 도메인이든** 문서검색 챗봇