3 차 팀 프로젝트 가이드

RAG 기반 AI 에이전트 서비스 구축과 운영

1. 문제와 시장 근거, RAG 품질, 에이전트 설계와 신뢰성, 배포와 운영
2. Python, LangGraph, RAG, FastAPI, Docker, Langfuse 필수
3. GCP Cloud Run 배포, Postgres pgvector 권고
4. 자동화(증분 수집과 정시 실행), 파인튜닝, GraphRAG, MCP 와 Skill 통합 가산

2026 년 10 월 23 일

이 프로젝트의 정체성

이 프로젝트의 본질은 "근거를 붙여 답하고, 스스로 다시 찾고, 사람의 승인 앞에서 멈추는 에이전트를 끝까지 배포하고 측정하는 것"입니다. 2 차가 LLM 을 서비스로 세우는 경험이었다면, 3 차는 그 서비스 안에 검색(RAG)과 상태를 가진 에이전트(LangGraph)를 넣고, 잘 돌아가는지 숫자로 증명하는 경험이에요.

세 프로젝트의 자리는 이렇게 이어집니다.

차수	핵심 질문	다루는 것
1 차	모델을 이해했는가	딥러닝 모델 직접 사용과 튜닝 (TTS, STT)
2 차	서비스로 세울 수 있는가	LLM API 제어, 서빙, 관측, 평가
3 차	신뢰할 수 있는 에이전트 시스템을 끝까지 배포하고 측정해 봤는가	RAG 검색 품질, LangGraph 상태 그래프와 사람 승인, 배포와 운영

2 차 가이드가 "3 차의 몫"으로 남겨 둔 RAG 와 LangGraph 를 이제 씁니다. 반대로 2 차에서 이미 요구했던 FastAPI, Docker 한 줄 기동, Langfuse, 평가셋 재측정, 신뢰 경계는 3 차에서 새로 배우는 것이 아니라 가져와서 이식하는 것입니다.

> 📌 **목표는 "경력 2~3 년처럼 보이는 증거" 네 가지입니다**
>
> 신입을 뽑는 AI 공고는 드뭅니다. 그래서 이 프로젝트는 "이런 걸 해봤습니다"가 아니라 남이 열어 볼 수 있는 증거를 남기는 데 맞춰져 있어요.
>
> 1) 동작하는 배포 주소: 평가용 테스트 계정으로 로그인해 열어 볼 수 있는 Cloud Run 주소
>
> 2) 개선 전후 점수가 있는 평가 리포트: 같은 평가셋으로 바꾸기 전과 후를 잰 숫자
>
> 3) 운영 트레이스: Langfuse 에 남은 실제 호출 기록과 비용, 응답 시간
>
> 4) 실패 사례 분석 글: 무엇이 왜 틀렸고 어떻게 다뤘는지
>
> 이 네 가지는 평가 5 축 중 ⑤ 증거 패키지에서 점수가 되고, 이력서와 면접에서 그대로 쓰입니다.

3 차 프로젝트의 한 줄 원칙

"돌아간다"는 주장이 아니라 증거다. 증거가 없으면 돌아가지 않는 것이다.

RAG 와 에이전트는 같은 질문에도 다른 문서를 가져오고 다른 경로로 답합니다. "잘 되던데요"는 근거가 되지 않아요.

어떤 질문에서 검색이 실패했는지, 그래프가 어디서 멈추는지, 한 번 처리에 얼마가 드는지를 남이 확인할 수 있는 형태로 내놓는 것이 목표입니다.

평가의 5 가지 축

축	평가 내용	배점
① 문제와 시장 근거	목표 직무의 공고 근거, 페르소나 및 페르소나 저니 맵, 대체재 대비 우위, 지불 주체와 실재성	15
② RAG 품질	코퍼스 설계와 파싱 검증, 청킹과 임베딩 선택, 하이브리드와 리랭킹 개선 전후, 평가셋, 검색 실패 분석	25
③ 에이전트 설계와 신뢰성	그래프 설계 판단, 멀티에이전트 구조, 재검색 루프, 사람 승인과 재개, 인용 검증, 신뢰 경계	25
④ 배포와 운영	compose 재현성, 클라우드 배포와 API 설계, 비용과 응답 시간 실측, CI 회귀 테스트	20
⑤ 증거 패키지와 발표	발표자료, 시연 영상, 발표력	15

이 5 축 100 점과 별도로, 필수 게이트 7 개(못 지키면 고정 점수가 깎임, 최대 -50)와 가산점(상한 +10)이 있습니다. 축별 배점은 10 장에 공개합니다.

2 차의 4 축에서 달라진 점은 이렇습니다.

- "LLM 제어력"(출력 계약, 모델 선택)은 ② RAG 품질과 ③ 에이전트 설계로 나뉘었습니다.
- 2 차에서 모두가 약했던 "대체재 대비 우위"와 "지불 주체"는 ①에서 구성합니다.
- 모델 선택 근거는 ②의 "청킹과 임베딩 선택 근거", ④의 "비용과 응답 시간 실측"에 들어갔습니다.

1. 프로젝트 일정

구분	일시
주제 공지	10 월 26 일 (월)
주제 선정 마감 (주제서 1 장, 필요한 조는 사전 논의 신청)	10 월 29 일 (목)
착수 (킥오프)	10 월 30 일 (금) 오후 세션
코드와 설정 동결	11 월 16 일 (월) 자정
최종 제출 (코드, 배포, 리포트, 발표 자료)	11 월 17 일 (화) 자정
발표 및 수료식	11 월 18 일 (수)

> 마감이 둘로 나뉘어 있습니다. 코드는 월요일 자정에 멈추고, 화요일은 제출물을 정리하고 발표를 준비하는 날입니다.
>
> 11/16 자정 이후의 커밋은 채점에 반영하지 않아요. 마감 시점의 레포와 배포 주소로 평가합니다. 배포 주소는 발표일까지 살려 두세요.

1-1. 시간은 100 시간입니다

10/30 오후 4 시간과 11/2(월)부터 11/17(화)까지 12 일(하루 8 시간)을 합쳐 100 시간입니다. 이 구간에는 공휴일이 없지만, 2 차처럼 "달력상 3 주니까 여유 있다"고 읽으면 안 됩니다.

필수 요건(RAG, 멀티에이전트, 평가 재측정, 배포, 증거 네 가지)의 양이 많아서 실제로는 하루도 비어 있지 않아요.

1-2. 권장 진행 흐름

구간	일	이 구간에 끝내야 할 것
10/26 ~ 10/29	-	주제 선정과 주제서 1 장 제출 (다른 산업은 사전 논의 신청도 10/29 까지)
10/30 (금) 오후	D0	Spec 과 숫자 기준 합의, 역할과 담당 축, 저장소 정리, 빈 FastAPI 를 compose 로 띄워 배포 주소 확보
11/2 ~ 11/4	D1 ~ D3	코퍼스 수집과 파싱, 11/3 평가셋 v1 30 건(관련 15, 비관련 15) 동결과 베이스라인 점수, 하이브리드 검색과 리랭킹
11/5 ~ 11/6	D4 ~ D5	에이전트 그래프, 평가셋 v2 50 건 확장, Supervisor 와 서브에이전트 분리, 사람 승인, 11/6 중간 점검
11/9 ~ 11/11	D6 ~ D8	서비스화(FastAPI, compose, Supabase, Langfuse), 신뢰 경계 시험, 11/11 비용과 응답 시간 실측, CI 회귀 테스트

11/12 ~ 11/13	D9 ~ D10	검색 실패 분석과 개선 2 회차, 대체재 비교표, 11/13 제출물 점검
11/14 ~ 11/15	-	주말. 자동 실행이나 배포를 새로 두지 않습니다. 필요하면 평가 파이프라인만 한 번
11/16 (월)	D11	전날부터 빌드를 유지하고 최소 수정만. 24:00 코드와 설정 동결
11/17 (화)	D12	실패 사례 분석 글, 포트폴리오 1 장, AI 코딩 도구 사용 기록, 발표 자료. 24:00 최종 제출
11/18 (수)	-	발표

> ⚠ **D0 에서 배포 주소를 못 얻으면 팀 전체가 멈춥니다**
>
> 첫날 오후 4 시간의 목표는 기능이 아니라
>
> "빈 FastAPI 가 compose 로 뜨고, Cloud Run 주소가 열린다" 입니다.
>
> GCP 계정, 결제 연결, 필요한 API 활성화, 저장소 권한, 조 저장소에서 GitHub Actions 가 도는지를 이날 확인하세요.

1-3. 주제 선정 일정 (10/26 ~ 10/29)

- 10/26(월)에 주제를 공지합니다.
- 10/29(목)까지 조별로 주제를 정해 주제서 1 장을 냅니다. 양식은 부록에 있습니다. 제출 방법은 공지에서 안내합니다.
- 추천 산업 6 종 중에서 고르면 별도 신청이 필요 없습니다. 주제서만 내면 됩니다.
- 그 밖의 산업을 쓰려는 조만 사전 논의를 거칩니다. 신청도 10/29 까지이고, 승인 여부도 같은 날 회신합니다.
- 프로젝트 시작 하기 전에 확정되는 것을 추천합니다.

2. 주제 및 요구 사항

(1) 필수 조건 7 가지 (게이트)

아래 7 가지는 못 지키면 해당 항목만큼 점수가 고정으로 깎입니다. 가산점은 7 가지를 모두 충족한 조만 받을 수 있습니다. 감점은 합쳐서 최대 -50 입니다.

번호	조건	감점
G1	GCP Cloud Run(2 차에서 다른 플랫폼을 썼다면 그 플랫폼도 가능)에 배포된 주소가 있고, 제출한 평가용 테스트 계정(2 개)으로 로그인해 핵심 시나리오를 끝까지 볼 수 있다 (또는 영상)	-5
G2	`docker compose up` 한 줄로 서비스가 `10` 분 안에 뜬다 (키 없이 뜨는 데모 모드 포함, `9` 장)	-5
G3	서로 다른 청크 1 만 개 이상과 서로 다른 문서 50 건 이상, 문서 파싱 1 종 이상. 인덱스 목록(manifest)으로 확인한다. 서로 다른 청크는 본문 sha256 으로 센다(정정공시 같은 중복은 한 개)	-10
G4	LangGraph 상태 그래프가 있고, 멀티에이전트 최소형(Supervisor 와 서브에이전트 2 개 이상), 체크포인트, 사람 승인(interrupt) 지점이 있다. 로그인 기능이 있고, 로그인한 사용자 id 로 대화(thread)와 장기 기억(Store namespace)이 분리되며, Store 기반 시멘틱 메모리가 있다	-10
G5	평가셋 50 건 이상(동결 시점 기준, 관련 질문과 비관련 질문 각각 절반)으로 개선 전후를 두 번 이상 재서 회귀를 숫자로 보인다. 차이는 달라진 문항 목록과 함께 적는다	-10
G6	Langfuse(또는 동급)로 트레이스가 남고 평가 점수가 연결된다	-5
G7	README, `.env.example`, 비밀값 커밋 없음, 3 분 이내 시연 영상(정상 경로와 실패 경로)	-5

왜 G3 와 G4 가 무거운가

둘은 이 프로젝트의 정체성입니다. G3 는 "많은 문서를 실제로 다뤄 봤는가"를, G4 는 "에이전트를 구조로 설계했는가"를 묻습니다.

1 만 청크는 실제 산업의 RAG 에서는 작은 규모입니다. 그래도 하이브리드 검색과 리랭킹의 차이가 눈에 띄고, 검색이 실패하는 유형이 다양해지는 최소 규모라서 정했어요.

(2) 무엇을 만드나

그 산업의 공개 문서를 코퍼스로 삼아, 질문에 근거를 붙여 답하고, 필요하면 되묻고 다시 검색하며, 사람의 승인이 필요한 행동 앞에서 멈추는 에이전트형 RAG 서비스입니다. FastAPI 로 열고 docker compose up 한 줄로 뜨며, 배포 주소와 트레이스가 있어야 합니다.

```
사용자 → FastAPI 엔드포인트 → Supervisor (질의 분석과 라우팅)

Supervisor → 검색 에이전트: 하이브리드 검색 (BM25 + 벡터) → 리랭킹, 메타데이터 필터 → 문서 채점
      문서 채점이 "근거 부족"이면 질의 재작성 → 재검색 (루프)
      문서 채점이 "충분"이면 → 작성 에이전트 (근거를 붙인 답변) → 검증 에이전트 (인용과 안전 검증)

검증 뒤 "승인이 필요한 행동"이면 사람 승인 대기 (interrupt) → 응답, 아니면 바로 응답

벡터 DB (Supabase pgvector 등) ← 문서 파싱과 청킹이 채우고, 하이브리드 검색이 읽는다

Langfuse 트레이스 ← Supervisor 가 남긴다 / 평가셋과 회귀 테스트 → Supervisor 를 검증한다
```

멀티에이전트란 여러 에이전트로 구성해서 오케스트레이션 구조를 만드는 것을 말합니다. 이 프로젝트에서는 최소형이 필수입니다. Supervisor 하나와 서브에이전트 둘 이상(예: 검색, 작성과 검증)으로 역할과 컨텍스트를 나누고, 같은 평가셋으로 단일 에이전트 대비 개선(또는 개선이 없었다는 정직한 기록)을 보입니다.

(3) 행동 유형과 사람 승인

유형	예시	사람 승인이 필요한 곳
답하는 일	근거를 붙인 질의응답, 되묻기	근거가 부족하면 거절
만드는 일	보고서, 요약, 초안 작성	저장이나 게시 전 승인
판단을 돕는 일	규정 점검, 체크리스트	최종 판정은 사람

어떤 유형이든 "사람이 승인해야 나가는 행동"이 하나 이상 있어야 하고, 그 지점에서 `interrupt` 로 멈춘 뒤 체크포인트로 이어서 재개되어야 합니다. 외부로 실제 발송하는 일은 요구하지 않습니다. 저장 예정 보고서 승인처럼 되돌릴 수 있는 행동이면 충분해요.

(4) 코퍼스는 그 산업의 공개 문서입니다

코퍼스는 공시, 법령, 가이드라인, 기술 문서, PDF 같은 그 산업의 공개 문서입니다. 서로 다른 청크 1 만 개 이상과 서로 다른 문서 50 건 이상을 채워야 합니다(G3). 청크가 많을수록 좋지만 서로 다른 청크여야 하고, 복제나 약간의 변형은 세지 않습니다(2 만 개 이상을 권장합니다). 공고 50 건은 주제 선정 근거를 세우기 위해 집계하는 것이고, 어떤 RAG 를 만들지, 코퍼스, G3 와는 상관이 없습니다. 공고 문서는 청크 수에 세지 않습니다. 서로 다른 청크는 본문의 sha256 해시가 서로 다른 것으로 셉니다. 정정공시처럼 본문이 같은 문서가 둘이면 한 개로 칩니다.

(5) 주제 고르기: 추천 산업 6 개

주제는 두 단계이고 서로 별개입니다. 먼저 목표 직무와 산업을 정하고 그 직무의 공고 50 건으로 시장 근거를 세웁니다(주제 선정 근거). 그다음 그 산업의 공개 문서로 어떤 RAG 를 만들지 정합니다(실제 주제). 아래 여섯 산업 중에서 만들 RAG 를 고르는 것을 기본으로 합니다.

산업	서비스 예 (RAG)	지식 층 (청크 후보)
금융, 핀테크	기업 공시와 사업보고서를 근거로 사업 구조와 리스크를 묻고 답하는 RAG	OpenDART 사업보고서와 공시 (Part IV 1 주 07 강이 DART 수집을 다룹니다. 공시 원문은 압축과 XML 처리가 필요합니다)
공공기관, 행정	관련 법령과 정책 문서를 근거로 제도와 절차를 묻고 답하는 RAG	국가법령정보, 정책브리핑 (법제처 API 는 GitHub 실행 서버에서 막힌 사례가 있어 시험이 필요합니다)
헬스, 제약	의약품 정보와 규제 문서를 근거로 제품과 규제를 묻고 답하는 RAG	식약처 등
물류, 무역	관세 법령과 통관 고시를 근거로 통관 업무를 묻고 답하는 RAG	관세 법령, 통관 고시
IT, 플랫폼	프레임워크 공식 문서와 오픈소스 문서를 근거로 기술 질문에 답하는 RAG	프레임워크 공식 문서, 오픈소스 문서
창업	지원사업 공고 본문과 관련 법령을 근거로 신청 준비를 돕는 RAG	기업마당, K-Startup 지원사업 공고, 법령

> ⚠ **소스는 직접 시험해서 확인하세요**
>
> 위 표는 공식 페이지에서 확인한 범위이고, 키 발급과 실제 호출은 해 보지 않았습니다. 호출 한도, 이용 조건, 본문 길이,

> 실행 서버에서 막히는지는 소스마다 다릅니다. 키 발급에 영업일이 걸리는 곳이 있으니 주제를 정한 뒤 바로 신청하세요.

주제서 1 장에 들어갈 것은 부록의 양식을 따릅니다. 누가, 어떤 업무에서, 무엇을 묻고, 지금은 어떻게 하는지, 범용 챗봇과 무엇이 다른지, 누가 쓰고 누가 지불하는지(가설이면 가설이라고 적습니다)입니다.

2 차 서비스를 이어서 만들어도 됩니다. 이어 간다면 LLM 호출부, compose 틀, Langfuse 설정은 그대로 가져오고, RAG 와 LangGraph 를 얹는 범위를 주제서에 적으세요.

다른 산업을 쓰려면 10/29 까지 사전 논의를 신청합니다. 다음 두 가지를 한 장으로 적어 승인을 받으세요: 코퍼스 소스 URL 3 개 이상과 예상 문서 수의 근거(1 만 청크가 가능한지), 그 소스의 이용 조건(약관 또는 라이선스) 확인 한 줄. 두 가지가 모두 있으면 승인합니다.

3. 프로젝트 진행 절차

3 차도 1, 2 차와 같은 디자인 씽킹 기반 문제 해결 절차를 따릅니다.

단계	수행 내용
① 주제 선정	목표 직무와 산업 선택, 해결하고 싶은 문서 업무 하나 고르기
② 사용자 중심적 사고	그 산업에서 반복되는 "문서를 읽고 판단하는 일"에 공감하고 가설 수립
③ 문제 정의	누가 무엇 때문에 막히는지, 범용 챗봇이 왜 부족한지 정의
④ 고객 여정 맵	경험 흐름과 단계별 pain point 시각화 (페르소나 포함)
⑤ 솔루션 제안	근거를 붙여 답하는 에이전트형 RAG 로 어떻게 푸는지 제안

세부 수행 방법(사용자 분석, 문제 정의 문장, 고객 여정 맵 양식)은 1 차 프로젝트 가이드 4 장과 같습니다. 3 차에서 하나 더해진 것은 주제 선정 근거입니다. 목표 직무의 공고 50 건에서 시장이 요구하는 것을 읽고 방향을 세웁니다. 이것은 어떤 RAG 를 만들지(실제 주제)와 별개입니다.

> **3 차에서 특히 조심할 것**
>
> RAG 와 LangGraph 가 화려해서, 문제 정의를 건너뛰고 구조부터 그리는 팀이 나옵니다.
>
> ✗ 안 되는 사고: "LangGraph 랑 RAG 써야 하니까 뭘 만들지?"

✓ 맞는 사고: "이 직무 담당자가 하루에 이 문서들을 읽고 판단하는 데 시간을 쓴다. 근거를 붙여 20 분으로 줄일 수 있나?"
문제와 시장 근거 15 점이 여전히 별도로 있습니다. 기술 조건은 통과 요건이지 점수 자체가 아니에요.

4. 기술 활용 가이드

4-1. 필수 스택

스택	요구하는 것	왜 필수인가
Python	백엔드와 평가 코드	국내외 AI 공고의 기본 언어
LangGraph	상태 그래프, 조건부 라우팅, 체크포인트, 사람 승인 1 곳 이상, 멀티에이전트 최소형 (Supervisor 와 서브에이전트 2 개 이상)	조사한 공고의 33%. Part IV 2 주차에서 배운 것
로그인과 사용자별 기억	로그인 기능(구현 필수), 로그인한 사용자 id 로 대화(thread)와 장기 기억(Store namespace) 분리, Store 기반 시멘틱 메모리(무엇을 기억할지 기준, State 와 Store 의 경계, 틀린 기억을 고치는 절차, DB 기반 Store). 제출 때는 평가용 테스트 계정 2 개를 비공개로 냅니다	Part IV 2 주 06 강(thread_id 격리)과 07 강(Store)에서 배운 것. 사용자 식별자가 있어야 기억을 나눌 수 있습니다
RAG	문서 파싱 (PDF, 표, 스캔 중 1 종 이상), 청킹, 하이브리드 검색, 리랭킹, 인용 근거	공고에서 가장 자주 나오는 요건 (79%). Part IV 1 주차와 특강 1
FastAPI	질의 엔드포인트와 상태 확인, 오류 응답 규약	공고의 30%. 2 차에서도 필수였습니다
Docker	`docker compose up` 한 줄로 서비스가 뜬다 (10 분 안)	공고의 33%. 2 차에서도 필수였습니다
클라우드	GCP Cloud Run 배포 주소 (2 차에서 다른 플랫폼을 쓴 조는 그 플랫폼을 이어 써도 됩니다)	공고의 27%. 2 차 가이드의 기본 구성
코퍼스 규모	서로 다른 청크 1 만 개 이상, 서로 다른 문서 50 건 이상. 인덱스 목록(manifest)으로 확인	RAG 역량의 상징.

벡터 DB	Supabase 의 pgvector 를 권고. 다른 pgvector 나 벡터 DB(Chroma, Qdrant 등)도 허용하되 선택한 이유를 적습니다	청크 본문, 메타데이터, 벡터를 한곳에서 관리

4-2. 벡터 DB: Postgres pgvector 권고

Part IV 1 주 10 강은 FAISS 와 Chroma 를 실행했고, pgvector 는 DB 서버가 필요해서 개념으로만 다루었습니다. pgvector 는 클라우드에 올라간 서비스에서 그대로 쓸 수 있고 청크 본문, 메타데이터, 벡터를 한곳에서 관리할 수 있어서 권고합니다. 아래는 Supabase 공식 문서(2026-10-19 확인)에서 확인한 제약과 대응입니다.

확인한 제약	출처	대응
Free 프로젝트는 데이터베이스 500MB. 넘으면 읽기 전용으로 바뀝니다	supabase.com/docs/guides/platform/database-size	아래 용량표를 보고 규모를 정합니다
Free 는 활성 프로젝트 2 개 제한	supabase.com/pricing	2 차에서 쓴 프로젝트가 있으면 합산됩니다. 한도가 차면 2 차 프로젝트를 정리하세요. 혹은 다른조원의 계정으로 진행하세요.
Free 프로젝트는 1 주 동안 활동이 없으면 일시 정지	supabase.com/docs/guides/platform/free-project-pausing	하루에 몇 번 DB 를 호출하는 상태 확인(health check)을 둡니다. 11/17~18 평가 기간에는 특히 확인하세요
직접 연결은 기본이 IPv6. IPv4 만 되는 환경은 풀러(Pooler)를 씁니다	supabase.com/docs/guides/database/connecting-to-postgres	Cloud Run 에서 직접 연결이 되는지는 확인하지 못했습니다. 풀러 연결 문자열을 기본으로 쓰고, 마이그레이션은 직접 연결로 하세요

청크 수와 저장 용량 (1024 차원 float32 벡터 기준, 벡터만의 크기이고 인덱스와 본문은 별도입니다):

청크 수	벡터만	판단
1 만	약 41MB	필수 기준
2 만	약 82MB	권장

5 만	약 205MB	인덱스와 본문을 더하면 Free 500MB 에 가까워집니다
10 만	약 410MB	Free 로는 어렵습니다

알아 둘 것

- 한국어 키워드 검색은 Postgres 기본 전문 검색이 약합니다. 1 만~2 만 청크라면 BM25 는 애플리케이션에서 형태소 분석 후 계산하고 벡터 검색은 pgvector 로 하여 합쳐도 됩니다. 하이브리드는 Part IV 1 주 06 강(RRF)을 그대로 쓰세요.
- 서비스 역할 키(service role)는 서버에서만 쓰고 프런트나 저장소에 올리지 않습니다. 읽기 전용 질의에는 권한을 낮춘 접속 정보를 쓰세요.
- 공공 데이터는 수집을 로컬에서 한 번 하고 스냅샷(원문과 manifest)을 저장소에 둘 수 있습니다. 서버에서 호출이 막히는 소스를 피할 수 있어요. 다만 원문을 저장소에 올려도 되는지(재배포 허용 여부)를 소스별로 확인하고, 안 되면 수집 목록과 접근 방법만 두세요.
- 평가자용 스냅샷 파일이 수십 MB 가 될 수 있으니 저장소 크기 한도를 확인하세요.

4-3. 임베딩, 모델, 비용

- 임베딩과 리랭킹은 로컬 모델(Part IV 1 주 18 강의 bge-m3, KURE, cross-encoder)과 API 중 조가 고릅니다. 어느 쪽이든 인덱스는 오프라인에서 만들어 벡터 DB 에 적재하세요.
- 임베딩 처리 속도는 프로젝트 초반에 문서 100 건으로 직접 재 보세요. 1 만 청크가 몇 시간 걸리는지 미리 알아야 일정을 세울 수 있습니다. 참고로 DART 공시 11,022 청크는 로컬 모델 KURE-v1 이 Apple MPS 에서 361 초 걸렸고(CPU 전용은 측정하지 않았습니다), OpenAI 임베딩 API 는 1 만 청크에 약 0.15 달러였습니다. 매번 전체를 다시 임베딩하지 말고 본문 해시가 바뀐 청크만 다시 하세요.
- 로컬 모델(임베딩, 리랭커)은 각각 약 2GB 입니다. 평가자 컴퓨터에서 내려받기가 막힐 수 있으니 미리 받아 두고 HF_HUB_OFFLINE=1 로 오프라인 실행이 되는지 확인하세요.
- 조별 LLM 예산은 조당 30 달러(OpenRouter 키 한도)입니다. 키는 조당 하나이고, 받는 방법은 킥오프에서 안내합니다. 모델은 자유롭게 고르고, 모델 목록이나 단가 제한은 없습니다. 팀별 30 달러 한도만 있습니다.
- 최종 평가셋 50 건, 조건 2 개, 재측정 3 회라면 생성과 채점이 각각 약 300 회입니다(추정, v1 30 건으로 잰 베이스라인 비용은 별도). 생성은 저가 모델로 하고, 같은 조건의 반복은 응답 캐시로 줄이세요. 비용을 줄이는 것 자체가 ④의 평가 항목입니다.

- 한도를 다 써도 무료 모델은 동작합니다. 다만 무료 모델은 혼잡해서 429 오류가 자주 납니다.

4-4. 프레임워크를 쓰는 이유

LangGraph, LangChain 같은 프레임워크를 쓰는 것이 이 프로젝트의 전제입니다. 프레임워크 없이 핵심 루프를 직접 다시 구현하는 것은 점수를 주지 않습니다. 직접 짜 보는 것은 좋은 공부지만, 3 차의 평가는 "프레임워크 위에서 설계와 측정을 얼마나 잘했는가"에 있어요.

5. AI 사용 정책

이 과정에서 AI 사용은 금지가 아니라 권장입니다. 다만 어떻게 썼는지가 평가됩니다. 채용 공고에서도 AI 코딩 도구 활용이 요건으로 등장하기 시작했습니다.

구분	내용
✅ 허용, 권장	AI 에디터와 코딩 에이전트 전면 허용, 전체 파일 스캐폴딩 요청, 에러 해결 질의, 리팩터링, 네이밍, 주석 위임, 코드 리뷰 받기
⚠️ 조건부	아키텍처 결정(어떤 임베딩을 쓸지, 어떻게 청킹할지)은 AI 제안을 받되, 이유를 본인 언어로 리포트에 정리
❌ 금지	이해 없이 복붙, 타 팀 코드를 AI 에 넣어 자기 것으로 포장

3 차에서 덧붙이는 것이 있습니다.

- 전 일정에서 코딩 에이전트를 작업 방식으로 쓰세요. 그리고 어떻게 썼는지 1 장으로 기록해서 냅니다 (어떤 도구, 어떤 작업을 맡겼는지, 막혔던 순간, 직접 고친 곳).
- 코딩 에이전트가 옛 기획으로 되돌아가는 일을 막기 위해, 정본 문서(`SPEC.md`)와 에이전트용 지침 한 장(`AGENTS.md`)을 두세요. 6 장에 자세히 있습니다.

> 발표와 평가 때 강사가 여러분 코드의 임의 지점을 짚고 "이거 왜 이렇게 했어요?"라고 묻습니다. 답하지 못하면 그 부분은 점수에 반영되지 않습니다. AI 가 쓴 코드여도 상관없어요. 설명할 수 있으면 됩니다.

6. 팀 운영 원칙: 2 차에서 가져오는 것

2 차 회고와 평가에서 반복해서 나온 어려움을 3 차의 장치로 바꿨습니다. 점수가 걸린 것이 아니라 일정을 지키기 위한 장치입니다.

2 차에서 나온 어려움	3 차의 장치
기획이 두세 번 바뀌어 갈피를 잃음, 문서는 있는데 지켜지지 않음, 코딩 에이전트가 옛 기획으로 회귀	정본 문서 한 곳 (`SPEC.md`)과 에이전트용 지침 한 장 (`AGENTS.md`). 코드를 바꾸면 SPEC 도 같은 PR 에서 바꿉니다
구현보다 Spec 과 평가셋을 먼저 정한 조가 가장 안정적이었음	평가셋과 Spec 커밋이 첫 구현 커밋보다 앞서야 합니다 (커밋 순서로 확인)
동결 직전에 일이 몰려 일부 측정을 못 재고 한계로 남김	동결 전날부터 빌드 상태를 유지하고 마지막 날은 최소 수정만 합니다. 측정과 정리는 일정에 미리 둡니다
일은 나누었는데 서로의 일이 어떻게 이어지는지 모름, PR 방치와 충돌	매일 아침 15 분 스탠드업, 작은 PR 과 최신 main 충돌 확인, 매주 20 분 전원 중간 점검
맡은 일이 끝나면 다음 일을 찾지 못함	백로그 보드에 "내 일이 끝나면 가져갈 다음 일" 칸. 담당 축 외에 보조 담당 축을 하나 정합니다
도메인을 몰라 헤맴	주제서에 목표 직무의 공고 근거를 붙이고, 외부 시선 1 명(현직자, 채용 담당 등)의 검토를 받습니다
대체재와의 비교, 지불 주체가 약함	같은 질문 5 개를 범용 챗봇과 나란히 놓은 비교표가 필수 증거입니다
발표를 한 명이 맡거나 읽는 수준, 시연에 설명이 없음	발표는 조원 전원이 파트를 맡고, 시연은 화면마다 한 문장으로 해설합니다
배포 주소에 로그인 보호가 걸려 외부에서 확인하지 못함	로그인은 구현 필수입니다. 심사자가 쓸 평가용 테스트 계정 2 개를 제출하거나 영상을 함께 냅니다 (G1)
제출물 일부 누락, 응답 시간 기준을 중간에 낮춤	제출물 체크리스트를 전날 점검합니다. 응답 시간은 현재 코드로 단계별로 실측하고 기준을 중간에 낮추지 않습니다
평가자가 응답 시간을 재려 해도 키가 없어 못 재는 서비스	키 없이 뜨는 데모 모드가 필수입니다 (G2, 9 장). 평가는 실제 서비스를 돌려 보는 방식이고 평가자에게는 키가 없습니다

역할

역할은 평가 축에 맞춥니다. 3 인 조는 한 명이 두 역할을 겸하고, 모든 조원은 보조 담당 축을 하나 더 가집니다.

역할	맡는 축
검색과 데이터 담당	② RAG 품질
에이전트 담당 (Supervisor 와 서브에이전트 설계 포함)	③ 에이전트 설계와 신뢰성
서비스와 운영 담당	④ 배포와 운영
평가와 문서 담당 (Spec, 평가셋, 증거 패키지)	①과 ⑤

7. RAG 품질 (축 ②, 25 점)

항목	무엇을 보이나
코퍼스 설계와 문서 파싱 검증	어떤 문서를 왜 골랐고, 파싱한 결과를 원문과 대조했는지 (D1 에 3 건 이상). manifest 에 출처, 수집일, 문서당 청크 수를 적는 식으로 서로 다른 청크 1 만 개 이상을 확인할 수 있게 합니다
청킹과 임베딩 선택 근거	둘 이상을 비교해 고른 이유. 모델 선택 근거도 여기서 봅니다
하이브리드와 리랭킹의 개선 전후	사전 등록하고 한 번에 하나만 바꿔 재측정한 숫자
평가셋 설계와 사람 판정	v1 30 건(관련 15, 비관련 15)을 D2 에, 최종 v2 50 건 이상(관련과 비관련 각각 절반)을 D5 전에 동결 (sha256). 정답 근거를 사람이 판정
검색 실패 유형 분석	어떤 질문에서 왜 못 찾았는지 유형으로 묶기

평가셋은 가장 먼저 만듭니다

2 차에서 구현보다 평가셋을 먼저 정한 조가 가장 안정적이었습니다. 3 차는 11/3(D2)에 평가셋 v1 30 건을 사람이 판정하고 동결(sha256 으로 파일 해시를 기록)하며, 11/6 중간 점검 전까지 v2 50 건 이상으로 확장해 다시 동결합니다. 동결한 뒤에는 정답을 바꾸지 않아요. 결과가 안 나온다고 정답을 고치면 측정이 아니라 맞추기가 됩니다. 코퍼스가 작거나 바뀌는 중에 만든 평가셋은 어떤 개선도 보여 주지 못합니다(검색 방식 네 가지가 거의 같은 점수를 냈습니다). 코퍼스 범위를 확정한 뒤에 v1 을 동결하세요.

골든 데이터셋은 이렇게 구성합니다

- 관련 질문: 코퍼스 안에 정답 근거가 있는 질문입니다. 근거가 되는 문서와 위치를 사람이 판정해 적습니다.
- 비관련 질문: 주제 안의 질문이지만 코퍼스에 답이 없는 질문입니다. 정답은 "근거가 없어 답할 수 없다"는 거절입니다(Part IV 3 주 08 강의 거절해야 하는 질문). 코퍼스를 검색해서 정말 답이 없는 것만 비관련으로 씁니다.
- 비관련 질문은 한 종류로 채우지 마세요. 도메인 밖 질문, 코퍼스에 없는 대상(기업 등), 있는 대상이지만 코퍼스에 없는 항목, 같은 공시인데 기간이 다른 질문(3 분기를 물었는데 코퍼스에는 2 분기만 있음)이 모두 섞여야 쉬운 거절만으로 점수가 부풀지 않아요. 질문을 쓰는 사람은 정답 청크를 보지 않고 쓰고, 정답 청크는 다른 팀원이 나중에 찾으세요.
- v1(D2, 11/3): 30 건, 관련 15 건과 비관련 15 건입니다. 사람이 판정하고 sha256 으로 동결한 뒤 검색 베이스라인 점수를 잽니다.
- v2(D5, 11/6 중간 점검 전): 최종 골든 데이터셋입니다. 50 건 이상이고 관련과 비관련이 각각 절반입니다(50 건이면 25 건씩). v1 의 30 건은 그대로 두고 관련 10 건과 비관련 10 건 이상을 더해 다시 동결합니다. 50 건은 동결 시점 기준입니다.
- 동결한 뒤 코퍼스를 바꿔 정답이 달라진 문항이 생기면 평가셋 파일을 고치지 말고 evals/exclusions.json 에 문항 id 와 사유를 적어 지표에서만 빼세요. 사유를 공개하면 감점하지 않으며, 제외 때문에 50 건 밑으로 줄어도 괜찮아요.
- v2 를 동결한 뒤에는 D4 에 v1 으로 잰 단일 에이전트 기준 점수를 v2 로 다시 재고, 멀티에이전트와 이후 모든 개선 전후 비교도 v2 로 잽니다. 평가 리포트에는 건수와 관련, 비관련 건수, 판정 기준, 두 버전의 해시를 적습니다.

"사전 등록"과 "한 번에 하나만"

- 사전 등록: 바꾸기 전에 "무엇을 바꾸고, 어떤 지표가 얼마나 오를 것으로 기대하는지"를 기록해 두는 것입니다. 결과를 본 뒤에 기준을 정하면 어떤 결과든 성공처럼 보여요.
- 한 번에 하나만: 청킹과 리랭킹을 동시에 바꾸면 무엇이 올렸는지 모릅니다. 비교는 같은 입력, 같은 평가셋, 같은 판정 기준에서만 성립합니다.

- 기대는 숫자 범위로 적으세요(예: 0.60~0.75). 기대가 틀려도 불이익이 없고, 틀린 기대를 기록으로 남기는 것이 사전 등록의 목적입니다. 측정한 뒤에는 맞음과 틀림 칸만 채우고 기대는 고치지 않아요.
- 개선 전후 차이가 1~4 문항이면 개선이라고 쓰지 말고 달라진 문항 목록을 같이 적으세요. 평가셋이 작으면 그 정도 차이는 우연일 수 있어요. 개선이 없거나 나빠졌어도 사전 등록과 원인 분석이 있으면 불이익이 없습니다.
- 보고 지표는 하나의 정확도로 합치지 말고 다섯 개로 나누세요. 정답 근거 인용률, 오답 근거, 과잉 거절(답이 있는데 거절), 올바른 거절, 환각 위험(답이 없는데 답함)입니다.

정답 기준은 문서에 적습니다

평가셋의 정답을 팀이 정해도 됩니다. 중요한 것은 정답을 정한 기준을 적는 것입니다. 팀원끼리 판단이 갈리는 항목이 나오면 그 자체가 좋은 발견이니 리포트에 적으세요.

G4 와 G5 는 테스트가 통과했다는 것만으로는 증거가 아닙니다. 핵심 코드를 한 줄 깨뜨렸을 때 그 테스트가 실제로 실패하는지 확인하고 docs/GATES.md 에 3 건 이상 적으세요. 예를 들어 재작성 횟수 상한 테스트가 상수끼리 비교하면 상한을 99 로 올려도 통과합니다. 숫자 2 같은 값과 비교해야 상한 변경을 잡아요. AI 에이전트가 변이를 잡았다고 보고해도 직접 확인하세요.

8. 에이전트 설계와 신뢰성 (축 ③, 25 점)

항목	무엇을 보이나
그래프 설계 판단	왜 그래프여야 하는지, 노드마다 책임이 무엇인지
멀티에이전트 역할 분리와 컨텍스트 격리, 단일 에이전트 대비 효과	Supervisor 와 서브에이전트 2 개 이상, 같은 평가셋으로 단일과 비교
재검색과 도구 호출 루프	근거가 부족할 때 질의를 고쳐 다시 찾는 경로
사람 승인과 체크포인트 재개	`interrupt` 로 멈추고 체크포인트로 이어서 재개
인용 검증 (코드로)	답변의 인용이 실제 검색된 문서에 있는지 코드로 확인
신뢰 경계, 인젝션, 실패 경로	외부 문서 속 지시문에 어떻게 대응하는지, 실패 시 어디로 떨어지는지

단일 에이전트부터 만듭니다

일정은 이 순서를 가정합니다. D4(11/5)에 단일 에이전트 그래프(질의 라우팅, 문서 채점, 질의 재작성, 재검색)를 만들고 기준 점수를 잽니다. D5(11/6)에 Supervisor 와 서브에이전트로 나누고 같은 평가셋으로 다시 재서 비교해요. 단일을 건너뛰면 "멀티에이전트가 나았다"를 증명할 기준이 없습니다. 개선이 없었다면 없었다고 기록하세요. 정직한 기록도 점수가 됩니다. 11/6 중간 점검에서는 Supervisor 와 서브에이전트로 나눈 설계와 단일 대비 재측정 여부를 묻습니다. 단일 에이전트로 되돌리는 것은 허용하지 않고, 끝까지 나누지 못하면 G4 감점(-10)이 그대로 적용되며 그 사실을 최종 제출물에 적어야 합니다.

신뢰 경계: 코퍼스도 믿지 않습니다

RAG 는 외부 문서를 프롬프트에 붓는 구조라서, 문서 안에 숨은 문장이 지시로 읽히면 계약도 검증도 소용이 없습니다. 2 차에서 배운 신뢰 경계를 이어 씁니다.

- 코퍼스에 간접 인젝션(문서 속에 심은 지시문)을 심어 시험 3 건 이상, 출처 유출 시험도 포함합니다 (D7, 11/10).
- 막지 못한 것은 코드로 보강하고, 무엇을 시도했고 무엇이 막혔고 무엇이 남았는지 적습니다.
- "통과"는 안전하다는 뜻이 아니라 우리가 아는 방식으로는 안 뚫렸다는 뜻입니다. 막았다고 쓰지 말고 남은 것을 쓰세요.

9. 배포와 운영 (축 ④, 20 점)

항목	무엇을 보이나
compose 재현성	새로 clone 한 저장소에서 `docker compose up` 한 줄로 10 분 안에 뜸
클라우드 배포 동작과 API 설계	Cloud Run 주소에 평가용 테스트 계정으로 로그인해 열림, /health, 질의 엔드포인트, 오류 응답 규약
관측: 트레이스 연결, 비용과 응답 시간 실측	Langfuse 트레이스, 현재 코드로 단계별 비용과 지연 실측
CI 에 넣은 회귀 테스트	평가 회귀를 GitHub Actions 등에서 자동으로 돌림

- 응답 시간은 단계별로 잽니다. 검색, 리랭킹, 생성, 승인 대기 중 어디가 느린지 보여야 해요. 코드를 바꿨다면 그 코드로 다시 재고, 기준을 중간에 낮추지 않습니다.
- 재현 명령: 평가 결과는 원시 결과와 "한 줄 재현 명령"을 같이 냅니다. 평가자가 샘플 몇 문항을 다시 돌려 숫자가 맞는지 봅니다.
- 데모 모드(필수): docker compose up 만으로 키 없이 뜨는 모드를 두세요. 로컬 pgvector 컨테이너에 스냅샷을 적재해서 Supabase 계정 없이 뜨게 하고, 키가 없으면 기록해 둔 응답을 재생합니다. 로그인이 필수이므로 데모 모드에는 데모 전용 테스트 계정 2 개를 시드로 넣어 키 없이도 로그인해 볼 수 있어야 합니다(제출하는 평가용 계정과 다른 값으로 두고, 실서비스 계정의 비밀번호를 저장소에 적지 않습니다). 평가자는 이 모드로 한 줄 기동(G2)과 시나리오 흐름을 키 없이 확인합니다. 실제 호출 성능은 발표일 현장 입력 3 개와 재현 명령으로 봅니다.
- CI 는 조 저장소에서 먼저 도는지 확인하세요. 2 차에서 조직 결제 제한으로 Actions 가 시작되지 않은 조가 있었습니다.

10. 축별 배점과 가산점

10-1. 5 축별 배점 (합계 100)

축	평가 내용	배점
① 문제와 시장 근거	목표 직무의 공고 근거, 페르소나 및 페르소나 저니 맵, 대체재 대비 우위, 지불 주체와 실재성	15
② RAG 품질	코퍼스 설계와 파싱 검증, 청킹과 임베딩 선택, 하이브리드와 리랭킹 개선 전후, 평가셋, 검색 실패 분석	25
③ 에이전트 설계와 신뢰성	그래프 설계 판단, 멀티에이전트 구조, 재검색 루프, 사람 승인과 재개, 인용 검증, 신뢰 경계	25
④ 배포와 운영	compose 재현성, 클라우드 배포와 API 설계, 비용과 응답 시간 실측, CI 회귀 테스트	20
⑤ 증거 패키지와 발표	발표자료, 시연 영상, 발표력	15

강사 현장 입력 3 개: 발표일에 강사가 즉석에서 정상, 경계, 방어 유형의 입력 3 개를 넣어 서비스를 라이브로 호출합니다. 세 유형에서 의도한 동작을 하는지로 채점하고, 응답 속도는 ④의 실측에서 따로 봅니다. 또한 위 평가를 위한 추가적인 코드 점검 및 실행 점검을 진행합니다.

10-2. 가산점

필수 게이트를 모두 충족한 조에게만 가산됩니다. 합계는 최대 +16 이지만 상한 +10 을 적용합니다. 필수를 부실하게 하고 가산을 채우는 것은 감점 요인입니다.

가산	점수	인정 기준
증분 수집과 정해진 시각 실행 (자동화)	+5	새 문서만 받아 추가와 갱신을 반영하고(삭제는 목록을 완전히 받았을 때만), 중복 실행을 막고, 정해진 시각 실행 기록과 실패 알림이 있다
파인튜닝	+5	LoRA, QLoRA, PEFT 등으로 도메인 특화 모델을 만들고 베이스 모델과 같은 평가셋으로 비교. 학습 데이터 구성 설명 필수. RAG 맥락에서는 임베딩 모델이나 리랭커의 파인튜닝도 인정
GraphRAG 또는 지식 그래프	+3	벡터 검색이 지는 질문 유형을 정의하고 그래프 검색으로 개선 전후를 보임 (특강 2)
MCP 와 Skill 통합	+3	둘 다 해야 인정됩니다. (1) 3 차 RAG 검색 등을 MCP 서버(FastMCP)로 노출하고 LangGraph 에이전트가 MCP 클라이언트로 호출합니다. (2) 도메인 판단 규칙을 Skill 문서(SKILL.md)로 분리해 에이전트가 쓰고, 같은 평가셋으로 Skill 이 있을 때와 없을 때의 점수 차이를 보입니다. 2 차에서 만든 MCP 서버나 Skill 을 그대로 가져온 것은 인정하지 않고, 3 차에서 새로 만들었거나 RAG 와 이어 붙인 부분만 봅니다 (참고: 2 차의 MCP 와 Skill 자료, Part III 2 주 17, 20 강)

멀티에이전트는 필수 조건이라 가산으로 중복 계산하지 않습니다. "파인튜닝했습니다"만으로는 가산되지 않고, 베이스와 같은 평가셋으로 잰 숫자가 있어야 해요. 또한 실제 출력결과 비교표(기존 모델과 파인튜닝 모델)도 필요합니다.

가산을 모두 하려고 하지 마세요. 상한이 +10 이라서 필수를 탄탄히 하는 쪽이 이득입니다.

11. 제출물

코드와 설정 동결: 11 월 16 일 (월) 자정, 최종 제출: 11 월 17 일 (화) 자정

항목	형식	비고
GitHub 레포	KDT 7 기 org 의 조 저장소	`.env.example` 포함, 비밀값 커밋 금지
README	레포 내	한 줄 정의, 지표 표, docker compose up 실행 절차 (10 분 안에 재현), 키 없이 뜨는 데모 모드 실행 방법
`SPEC.md`, `AGENTS.md`	레포 내	정본 문서와 에이전트 지침
클라우드 배포 주소	GCP Cloud Run	평가용 테스트 계정으로 로그인해 핵심 시나리오가 열려야 함. /health 동작
평가용 테스트 계정 2 개	비공개 제출 (제출 방법은 공지)	ID 와 비밀번호를 저장소에 적지 않음. 사용자 A 와 B 의 대화와 기억이 섞이지 않는지 평가자가 확인
인덱스 목록 (manifest)	레포 내	서로 다른 청크(본문 sha256 기준) 1 만 개 이상, 문서 50 건 이상을 확인할 수 있게. 원문 길이, 보존 길이, 잘림 여부 포함
평가셋과 원시 결과, 재현 명령	레포 내	평가셋 v2 50 건 이상(관련 25 이상, 비관련 25 이상, 비관련은 4 종을 섞음), v1 과 v2 의 동결 해시, 제외 파일(있다면), 한 줄 재현 명령
평가 리포트	레포 내 `EVAL_REPORT.md`	개선 전후 지표표와 달라진 문항, 지표 5 개 분리 보고, 단일 대비 멀티에이전트, 대체재 비교표 (같은 질문 5 개), 검색 실패 유형 분석
신뢰 경계 1 장	`EVAL_REPORT.md` 안	무엇을 믿지 않기로 했는지, 인젝션 시험 3 건의 전후, 아직 막지 못한 것
Langfuse 대시보드	스크린샷 3 장 이상	트레이스 상세 1 장 포함, 비용과 응답 시간
시연 영상	링크, 3 분 이내	정상 경로 1 개 + 실패 경로 1 개, 화면마다 한 문장 해설
발표 자료	PPT/PDF	문제 정의 → 구조 → 지표 → 한계 순. 전원이 파트를 맡음

실패 경로 영상을 요구하는 이유: 정상 경로만 찍은 영상은 어느 팀이나 만듭니다.
장애 상황에서 서비스가 어떻게 반응하는지 보여 주는 팀이 준비된 팀이에요.

⚠️ 11/13(D10)에 제출물 점검을 합니다. 새 노트북(또는 깨끗한 환경)에서 한 줄 기동을 시험하고, 시연 영상 초안과 외부 시선 1 명의 검토를 반영하세요. 제출물을 마지막 날 처음 모으는 팀은 항상 한두 개가 빠집니다.

12. 발표에서 반드시 나올 질문

미리 답을 준비해 두세요. 이 질문들이 곧 채점 기준입니다.

- 누가, 무엇 때문에, 얼마나 절실하게 막혀 있습니까? 범용 챗봇과 같은 질문 5 개를 나란히 놓으면 무엇이 다릅니까?
- 코퍼스는 무엇이고, 왜 이 문서들입니까? 청크는 몇 개이고 파싱은 원문과 대조했습니까?
- 청킹과 임베딩, 리랭킹을 어떻게 골랐고, 한 번에 하나만 바꿔 개선 전후가 얼마나 달라졌습니까?
- 평가셋은 몇 건이고(관련과 비관련 각각 몇 건) 정답은 누가 어떤 기준으로 정했습니까? 동결했습니까?
- 개선 전후 차이는 몇 문항 차이였습니까? 사전에 적어 둔 기대가 틀렸을 때 무엇을 했습니까?
- 테스트가 통과한다는 것을 어떻게 믿습니까? 코드를 한 줄 깨뜨리면 실패하는 것을 확인했습니까?
- 왜 이 구조가 그래프여야 하고, 멀티에이전트가 단일 에이전트보다 나았습니까? (아니었다면 왜였습니까?)
- 사람 승인 지점에서 멈추고, 재시작해도 이어서 재개됩니까?
- 코퍼스에 심은 인젝션 시험에서 무엇이 막혔고 무엇이 남았습니까?
- 요청 한 번에 얼마가 들고, 단계별로 어디가 가장 느립니까?
- 지금 이 서비스의 가장 큰 약점은 무엇입니까?

마지막 질문에 "없습니다"라고 답하는 팀은 자기 서비스를 모르는 팀입니다.
약점을 정확히 아는 것이 그 자체로 실력이에요.

13. 이 프로젝트를 취업에 쓰는 법

13-1. 이 프로젝트가 커버하는 채용 요구

채용 공고에 나오는 표현	이 프로젝트에서 해당하는 것
"RAG 파이프라인 구축 경험"	축 ②, 필수 조건 G3
"LangGraph, 멀티에이전트"	축 ③, G4
"문서 파싱, 하이브리드 검색, 리랭킹"	축 ②의 파싱 검증, 하이브리드와 리랭킹
"FastAPI 등 백엔드 서빙 경험"	축 ④, G1
"Docker 컨테이너 환경 경험"	축 ④ compose 재현성, G2
"클라우드 배포 경험"	축 ④ Cloud Run
"LLM 평가 체계 수립", "LLMOps, 관측"	축 ② 평가셋, 축 ④ Langfuse, 평가 리포트
"LoRA, PEFT 파인튜닝"	파인튜닝 가산

13-2. GitHub README 에 반드시 넣을 것

채용 담당자가 레포를 여는 시간은 길지 않습니다. 아래 세 가지가 상단에 없으면 읽히지 않아요.

- 한 줄 정의: "누구의 어떤 문제를 어떻게 푸는가". 기술 나열이 아니라 문제로 시작합니다.
- 지표 표: 평가셋 통과율, 개선 전후, 단계별 응답 시간, 요청당 비용. 숫자가 있는 README 는 드뭅니다.
- 실행 방법: `docker compose up` 한 줄. 이게 되면 "돌려볼 수 있는 프로젝트"가 됩니다.

아키텍처 다이어그램과 데모 GIF 는 그다음입니다. "FastAPI, Docker, LangGraph 사용"은 누구나 씁니다. 숫자와 문제 정의가 변별력이에요.

13-3. 면접에서 이 프로젝트로 답할 수 있는 질문

면접 질문	이 프로젝트의 어느 부분으로 답하나
"RAG 검색 품질을 어떻게 높이셨나요?"	하이브리드와 리랭킹의 개선 전후, 검색 실패 유형 분석
"답변이 근거 없이 나오는 문제는 어떻게 다뤘나요?"	인용 검증 코드, 근거 부족 시 재검색과 거절
"에이전트를 왜 그래프로 설계하셨나요?"	그래프 설계 판단, 단일 대비 멀티에이전트 비교

"품질을 어떻게 검증하셨나요?"	평가셋 50 건(관련 25, 비관련 25), 동결, 두 번 이상 재측정
"비용과 지연은 어떻게 관리하셨나요?"	단계별 응답 시간과 요청당 비용 실측
"보안은 어떻게 고려하셨나요?"	간접 인젝션 시험 3 건과 남은 위험
"협업은 어떻게 하셨나요?"	스탠드업, 작은 PR, 역할과 보조 담당 축

13-4. 부풀리지 마세요

면접관은 금방 압니다. 실제로 한 것만 정확하게 적으세요.

- ✗ "대규모 문서 처리": 청크 1 만 2 천 개를 처리했다면 그 숫자를 씁니다
- ✗ "검색 정확도 95% 달성": 평가셋 몇 건인지, 전에는 얼마였는지 없으면 의미가 없습니다
- ✓ "자체 평가셋 50 건(관련 25, 비관련 25) 기준 검색 Recall 0.62 → 0.81, 단계별 지연은 생성이 70%"

한계를 아는 사람으로 보이는 편이 만능으로 보이는 것보다 낫습니다.

14. 자주 나오는 질문

주제와 범위

Q. 주제를 아직 못 정했어요.

추천 산업 6 개 중에서 먼저 골라 보세요. 직무까지 좁히기 어려우면 산업을 먼저 정하고, 그 산업에서 사람들이 문서를 읽고 판단하는 반복 업무를 찾으면 됩니다. 공고 읽기는 주제 선정 근거로 따로 합니다. 10/29 까지 주제서 1 장이 필요합니다.

Q. 추천 산업이 아닌 주제를 하고 싶어요.

가능합니다. 단 10/29 까지 사전 논의를 신청해서 승인을 받아야 합니다. 코퍼스 소스 URL 3 개 이상과 예상 문서 수의 근거, 이용 조건 확인 한 줄을 한 장에 적어 오세요. 두 가지가 모두 있으면 승인합니다.

Q. 2 차에서 만든 서비스를 이어서 해도 되나요?

됩니다. LLM 호출부, compose 틀, Langfuse 설정은 그대로 가져오고, RAG 와 LangGraph 를 얹는 범위를 주제서에 적으세요. 다만 2 차에서 이미 평가받은 부분은 3 차 점수에 반영되지 않고, 3 차의 5 축으로 새로 평가합니다.

기술

Q. 임베딩은 로컬 모델과 API 중 무엇을 쓰나요?

조가 고릅니다. 로컬은 비용이 들지 않지만 시간이 걸리고, API 는 빠르지만 예산에서 빠집니다. 프로젝트 초반에 문서 100 건으로 처리 속도를 재 보고 정하세요.

Q. 청크가 1 만 개에 안 닿습니다.

산업의 공개 문서(공시, 법령, 가이드라인, 기술 문서)를 늘리세요. 서로 다른 청크여야 하고 복제나 약간의 변형은 세지 않습니다. 주제를 정하기 전에 문서가 충분한지 먼저 확인하세요.

Q. Supabase Free 에 청크가 안 들어갈 것 같아요.

1 만~2 만 청크는 벡터만 41~82MB 라서 들어갑니다. 5 만 청크 이상은 500MB 에 가까워집니다. 4-2 의 용량표를 보고 규모를 정하세요. 다른 pgvector 나 벡터 DB 를 써도 되고, 선택 이유와 한계를 적으면 됩니다.

Q. 평가셋의 비관련 질문은 어떻게 만드나요?

주제 안의 질문이지만 코퍼스에 근거가 없는 질문을 만드세요. 예를 들어 코퍼스가 특정 연도까지의 공시라면 그 이후의 전망을 묻는 식입니다. 질문을 만든 뒤 실제로 검색해 보고, 답이 우연히 있는 질문은 관련 질문으로 옮기세요. 비관련 질문에서 서비스가 답을 지어내지 않고 거절하는지가 이 질문들이 재는 것입니다. 비관련 질문은 도메인 밖, 코퍼스에 없는 대상, 없는 항목, 기간이 다른 질문의 네 종류를 섞으세요.

Q. 평가셋을 동결한 뒤에 코퍼스를 바꿨더니 정답이 달라진 문항이 있어요.

평가셋 파일은 고치지 마세요. 동결 해시가 바뀌면 CI 가 실패합니다. 대신 evals/exclusions.json 에 영향받은 문항 id 와 사유를 적어 지표에서만 빼면 됩니다. 사유를 공개하면 감점하지 않아요. 코퍼스를 다 모은 뒤에 동결하면 이런 일이 줄어듭니다.

Q. 프레임워크 없이 직접 구현하면 가산점이 있나요?

없습니다. 3 차의 평가는 프레임워크 위에서의 설계와 측정에 있어요.

Q. Docker 가 Apple Silicon 에서 안 떠요.

이미지 아키텍처 불일치가 가장 흔한 원인입니다. `platform: linux/amd64` 를 명시하거나 arm64 용 베이스 이미지를 쓰세요. 2 차에서 겪었다면 같은 방법입니다.

평가와 비용

Q. LLM API 비용은 누가 부담하나요?

조당 30 달러(OpenRouter 키 한도)가 제공됩니다. 키는 조당 하나이고 모델은 자유입니다. 받는 방법은 킥오프에서 안내합니다. 비용을 줄이는 것 자체가 ④의 평가 항목이니, 개발 중에는 작은 모델로 검증하고 평가셋 실행처럼 꼭 필요한 때만 큰 모델을 쓰세요.

Q. 게이트는 가점인가요?

아닙니다. 게이트는 못 지키면 깎는 항목입니다. 지켜도 점수가 오르지 않아요. 점수는 5 축 100 점에서 받고, 게이트 감점은 최대 -50 입니다. 가산점은 게이트를 모두 지킨 조만 받습니다.

Q. 평가셋 점수가 개선되지 않았습니다.

그대로 제출하세요. 무엇을 시도했고 왜 안 됐는지 분석하면 점수를 받습니다. 아무 실패 없이 "다 잘 됐습니다"라는 리포트가 오히려 의심받아요. 사전 등록한 기대와 결과가 달라도 불이익이 없습니다. 달라진 문항 목록과 원인 분석을 같이 적으세요.

Q. 기능을 다 못 만들 것 같습니다.

기능을 줄이세요. 이 프로젝트는 기능 개수로 평가하지 않습니다. 기능 하나가 근거를 붙여 답하고, 실패하면 우아하게 떨어지고, 측정된 상태인 것이 기능 다섯 개가 가끔 되는 것보다 높은 점수를 받습니다. 11/6 중간 점검에서 "무엇을 뺄 것인가"를 반드시 이야기합니다.

Q. 발표 현장 입력 3 개는 무엇인가요?

발표일에 강사가 정상, 경계, 방어 유형의 입력 3 개를 즉석에서 넣어 라이브로 서비스를 호출합니다. 의도한 동작을 하는지로 채점해요. 시연 직전에 키의 남은 한도와 배포 주소의 응답을 확인해 두세요.

부록. 참고 자료

주제서 양식 (1 장)

항목	적을 것
조, 구성원	
목표 직무와 산업	추천 산업 6 개 중 하나 (또는 사전 논의 신청)
누가	이 서비스를 쓰는 사람
어떤 업무에서	그 산업에서 반복되는 "문서를 읽고 판단하는 일" 하나
무엇을 묻는가	대표 질문 3 개
지금은 어떻게 하는가	지금의 방법과 불편한 점
범용 챗봇과 무엇이 다른가	근거 문서가 있다는 점을 포함해서
누가 쓰고 누가 지불하는가	가설이면 가설이라고 적음
코퍼스 후보	공개 문서의 출처, 이용 조건
(2 차를 이어가는 경우)	이어 가는 부분과 새로 얹는 범위

일별 목표와 연결 강의 (Part IV)

일	날짜	목표	연결 강의
D0 ★	10/30 오후	Spec 과 숫자 기준 합의, 역할과 담당 축, 저장소 정리, 빈 FastAPI 를 compose 로 띄워 배포 주소 확보	3 주
D1	11/2	코퍼스 수집, 문서 파싱 (PDF, 표, 스캔)과 원문 대조 3 건 이상, 청킹	특강 1 01~03, 10, 1 주 08
D2	11/3 ★	RAG 베이스라인과 검색 평가 (Recall, MRR), 평가셋 v1 30 건(관련 15, 비관련 15) 사람 판정과 동결 (sha256), 베이스라인 점수 기록	1 주 13, 3 주 04~06
D3	11/4	하이브리드 검색, 리랭킹, 메타데이터 필터. 실험 계획을 사전 등록하고 한 번에 하나만 바꿔 재측정	1 주 06, 12, 14, 3 주 10
D4	11/5	단일 에이전트 그래프 v1 (질의 라우팅, 문서 채점, 질의 재작성, 재검색 루프), 단일 기준 점수	2 주 01~04, 17~19
D5	11/6 ★	평가셋 v2 50 건(관련 25, 비관련 25) 확장과 동결, Supervisor 와 서브에이전트로 분리하고 단일과 멀티를 v2 로 재측정, 사람 승인과 체크포인트 재개. 중간 점검	2 주 05, 09~12, 16
D6	11/9	서비스화: FastAPI 엔드포인트, 오류 응답 규약, 로그인과 사용자별 thread 와 Store namespace, compose 정식, Supabase 연결, Langfuse 연동	3 주
D7	11/10	신뢰 경계: 코퍼스에 심은 간접 인젝션과 출처 유출 시험 3 건 이상, Cloud Run 배포 주소 최종 점검	3 주
D8	11/11 ★	비용과 응답 시간을 현재 코드로 단계별 실측, 평가 회귀 테스트를 CI 에 연결	3 주
D9	11/12	검색 실패 유형 분석 후 개선 2 회차 재측정, 대체재 비교표 (같은 질문 5 개)	1 주 14, 3 주 13
D10	11/13 ★	제출물 점검: 한 줄 기동 시험, 시연 영상 초안, 외부 시선 1 명의 검토 반영	-
D11	11/16 ★	전날부터 빌드 유지, 최소 수정만. 24:00 코드와 설정 동결	-
D12	11/17	실패 사례 분석 글, 포트폴리오 1 장, AI 코딩 도구 사용 기록, 발표 자료. 24:00 최종 제출	-
발표	11/18	조당 발표와 강사 현장 입력 3 개	-

★은 그날 못 넘기면 이후가 밀리는 항목입니다.

강의 자료 위치

주제	위치
문서 수집, 파싱, 청킹	Part IV / 1. RAG 파이프라인 / 07, 08, 10 강 + 특강 1
하이브리드 검색, 리랭킹	Part IV / 1. RAG 파이프라인 / 06, 12, 14 강
임베딩 모델 (bge-m3, KURE)	Part IV / 1. RAG 파이프라인 / 18 강
LangGraph 상태 그래프, 체크포인트, 승인	Part IV / 2. LangGraph 런타임 / 01~05, 09~12, 16~19 강 + 특강 3
시멘틱 메모리 (Store)	Part IV / 2. LangGraph 런타임 / 07 강
GraphRAG	Part IV / 4. 특강 / 특강 2
평가셋, 회귀, baseline	Part IV / 3. 평가·회귀·보안 /
간접 인젝션과 출처 유출	Part IV / 3. 평가·회귀·보안 /
FastAPI 에 RAG 얹기, trace 와 비용	Part IV / 3. 평가·회귀·보안 /
캡스톤	Part IV / 3. 평가·회귀·보안 /
Docker, Langfuse, FastAPI	Part III 특강 실습 (2 차에서 사용한 자료)