프로젝트 : **LangChain** 및 **RAG** 활용 의료 **LLM** 개발 **(MediChain)**

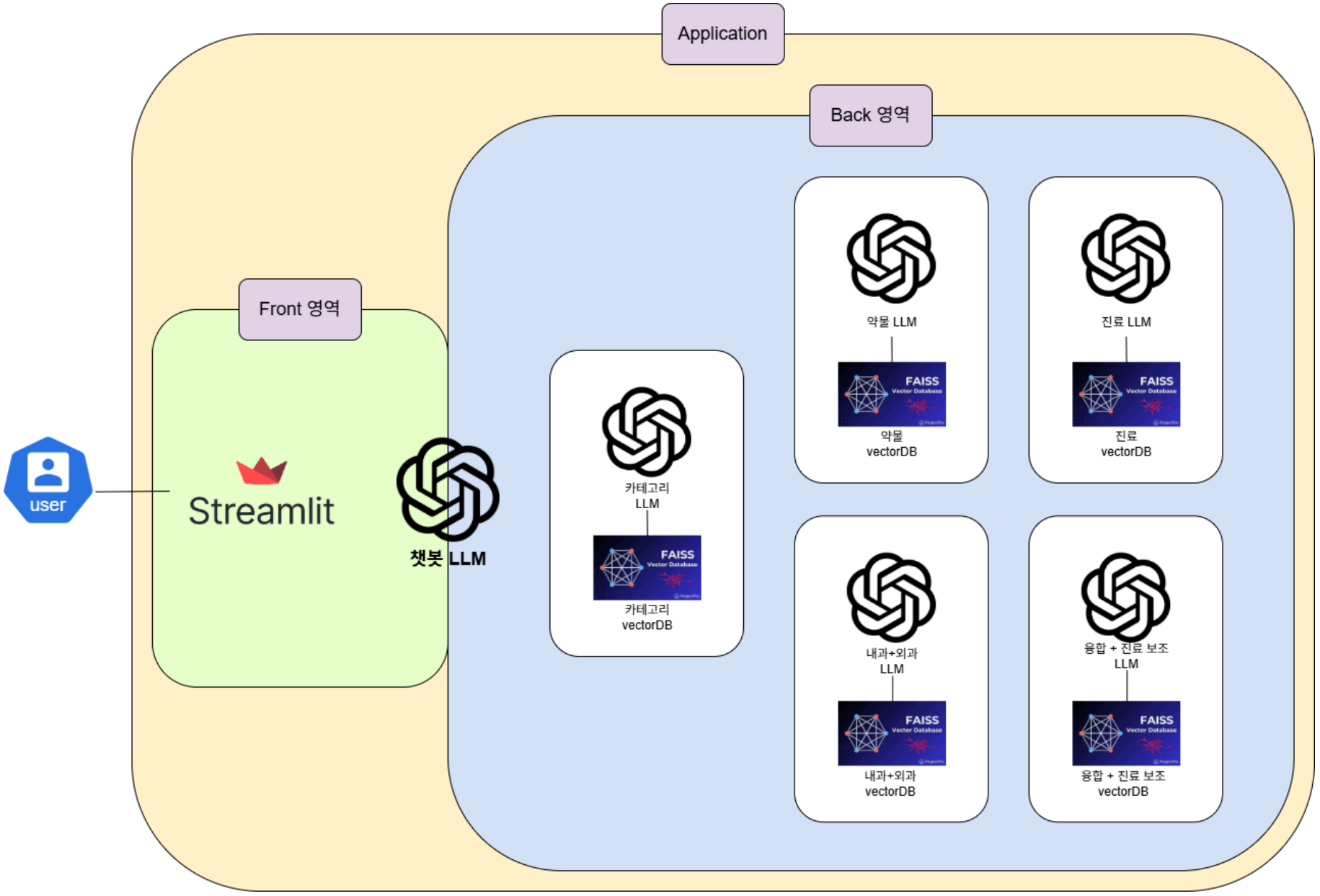
팀명 : windows
팀원 : 권성호, 남의헌, 이준배, 이준석, 손현성

목차

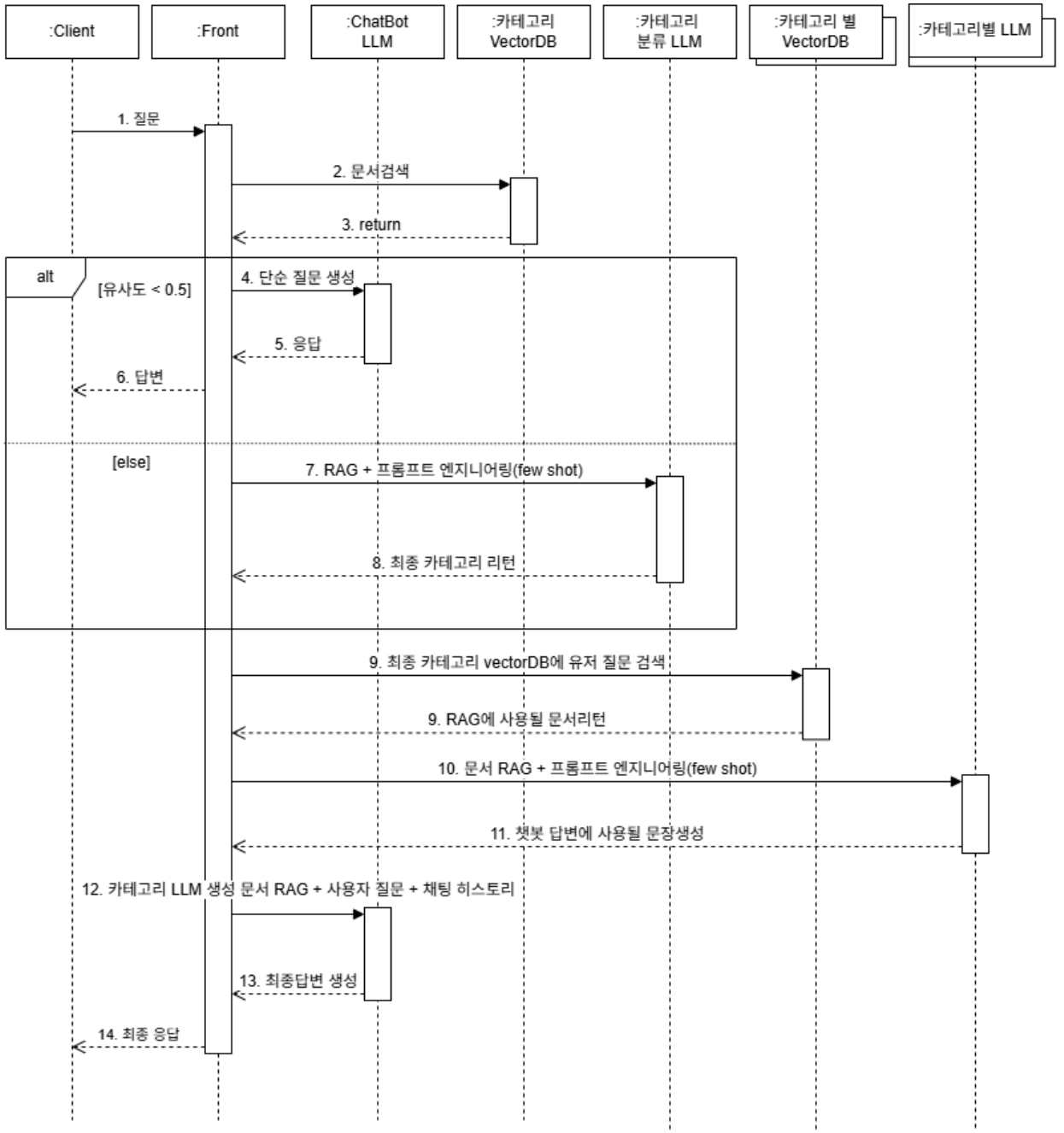
---

1. 전체 시스템 구성도

현 시스템 구성



시퀀스 다이어그램


:Client
:Front
:ChatBot LLM
:카테고리 VectorDB
:카테고리 분류 LLM
:카테고리 별 VectorDB
:카테고리별 LLM
1. 질문
2. 문서검색
3. return
alt
[유사도 < 0.5]
4. 단순 질문 생성
5. 응답
6. 답변
[else]
7. RAG + 프롬프트 엔지니어링(few shot)
8. 최종 카테고리 리턴
9. 최종 카테고리 vectorDB에 유저 질문 검색
9. RAG에 사용될 문서리턴
10. 문서 RAG + 프롬프트 엔지니어링(few shot)
11. 챗봇 답변에 사용될 문장생성
12. 카테고리 LLM 생성 문서 RAG + 사용자 질문 + 채팅 히스토리
13. 최종답변 생성
14. 최종 응답

2. 모듈별 역할 및 데이터 흐름

2.1 입력 및 **UI**

- Streamlit 기반 웹 인터페이스에서 사용자 질문 입력
- (세션별, 멀티 사용자 지원)

2.2 질문 임베딩

- Ko-SRoBERTa 등 SentenceTransformer로 입력 질문을 벡터화

2.3 카테고리 벡터**DB** 검색 **(FAISS)**

- 입력 임베딩과 카테고리별 임베딩 DB(FAISS)에서 유사도 검색
- 유사도 0.5 이상일 경우만 context로 사용

2.4 카테고리 분류 **LLM**

- OpenAI LLM이 context 예시들을 참고해
  질문을 4가지 카테고리 중 하나로 분류

2.5 카테고리별 벡터**DB** 재검색 **(FAISS)**

- 분류된 카테고리 전용 벡터DB에서 추가 유사도 검색
- 해당 카테고리에서 유사도 0.5 이상 문서 context 추출

2.6 카테고리별 전문 **LLM**

- context, 질문을 인풋으로 해당 전문 LLM에서 1차 답변 생성

2.7 최종 챗봇 **LLM**

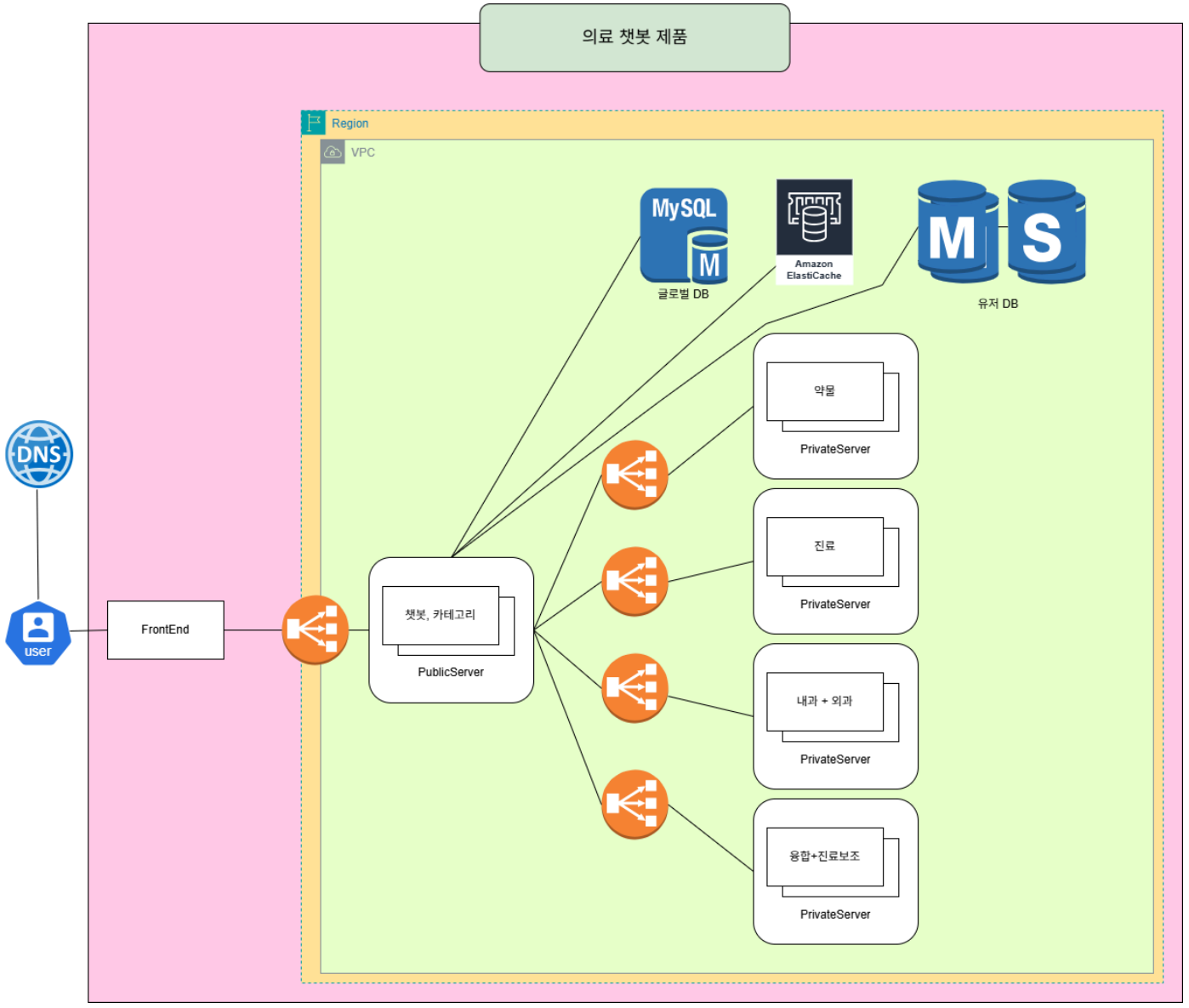
- 사용자 질문, 1차(전문가) 답변, 히스토리를 종합하여
  '최종 자연어 답변' 생성 및 사용자에게 반환

3. RAG, 벡터 DB, LLM 연동 구조의 적절성

- RAG 구조로 최신/정확/출처 기반 답변 지원
- 대용량 문서에 FAISS 벡터DB를 사용해 빠르고 효율적
- 카테고리별 DB/LLM 분리로 전문성, 확장성, 유지보수 용이
- 카테고리 분류 LLM/전문 LLM/최종 챗봇 LLM 분리로
  다양한 조합의 파이프라인 설계 및 운영 가능

---

4. 확장성 및 유지보수 관점 **(**마이크로서비스 구조 기반**)**



- 카테고리별 마이크로서비스 분리
  약물, 진료, 내과+외과, 융합+진료보조 등 각 분야를 개별 PrivateServer(마이크로서비스)로 설계
  각 PrivateServer는 독립적으로 개발, 배포, 확장, 장애대응 가능
  서비스 단위로 신규 LLM·RAG·벡터DB 도입, 업그레이드, 교체가 용이

- 공통 서비스**(PublicServer)** 일원화
  챗봇, 카테고리 분류, 세션/대화 관리, 인증 등 전체 공통 로직은 PublicServer에서 처리
  모든 요청을 PublicServer가 수신·분기 처리 → 운영 편의성, 관리 효율 극대화

- 로드밸런서/게이트웨이 도입
  모든 트래픽은 로드밸런서/게이트웨이를 통해 적절한 PrivateServer로 자동 분산
  장애 시 유연한 대처, 서비스 수평 확장 및 핫스왑 가능

- 데이터 계층 분리 및 확장
  MySQL(글로벌 DB), ElastiCache(캐시), 벡터DB, 유저DB(M, S) 등 역할별 분리
  대용량 데이터, 벡터 검색 분산·확장에 최적화된 구조
  설정파일, 환경변수 기반으로 경로·DB정보·모델명 통합 관리

- **CI/CD** 및 자동화 적용
  각 서비스별 독립 자동화 파이프라인 구성
  신규 서비스 배포, 스케일링, 버전 업그레이드 등 무중단/자동화 지원

---

5. 예상 확장 포인트 예시 **(**실제 아키텍처 기반**)**

- 새로운 의료 분야 추가
    - 예: 피부과, 정신과, 건강검진 등
      → PrivateServer 추가, 벡터DB/LLM만 연동하면 즉시 확장
- **RAG·LLM** 교체 및 고도화
    - 최신 임베딩모델, LLM, 외부 벡터DB(FAISS/PGVector 등)로 서비스별 교체 가능
- 외부 시스템**/**데이터 연동
    - 파일 업로드, 문서 검색, 의료데이터 연동 등 별도 마이크로서비스로 추가
- 운영·모니터링·알림 통합
    - 서비스별 모니터링, 장애 알림, 운영 로그 통합 대시보드화
- 사용자 맞춤 기능 확장
    - 요약, 파일 다운로드, 챗봇 이력, 검색 결과 하이라이트 등 부가 서비스 모듈화

---

6. 유지보수 관점 주요 포인트 (실제 시스템 구조 기준)

- 마이크로서비스 단위 장애 격리 및 롤백
  - 각 PrivateServer, PublicServer 장애 시 서비스 단위로 롤백/복구
  - 전체 시스템에 영향 최소화, 운영 효율 극대화
- 설정파일/환경변수 기반 일원화 관리
  - 경로, DB, LLM, 모델명 등 모든 설정 파일·환경변수로 통합
  - 운영환경 변경·이관이 용이
- 테스트, 로깅, 모니터링
  - 서비스별 로깅, 통합 모니터링/지표/오류 추적 자동화
  - 장애·이슈 발생 시 신속한 원인 파악 및 조치 가능
- 자동화된 배포 및 확장
  - CI/CD로 각 서비스 독립적 자동 배포
  - 오토스케일링, 핫스왑, 무중단 서비스 운영 지원

---

7. 부록 **:** 전체 아키텍처 흐름 요약

1. 질문 → 임베딩
2. 카테고리DB 유사도 검색 (0.5↑)
3. 카테고리 분류 LLM
4. 카테고리 전용DB 유사도 검색 (0.5↑)
5. 전문 LLM → 전문가 답변
6. 챗봇 LLM → 최종 답변
7. Streamlit UI → 사용자
