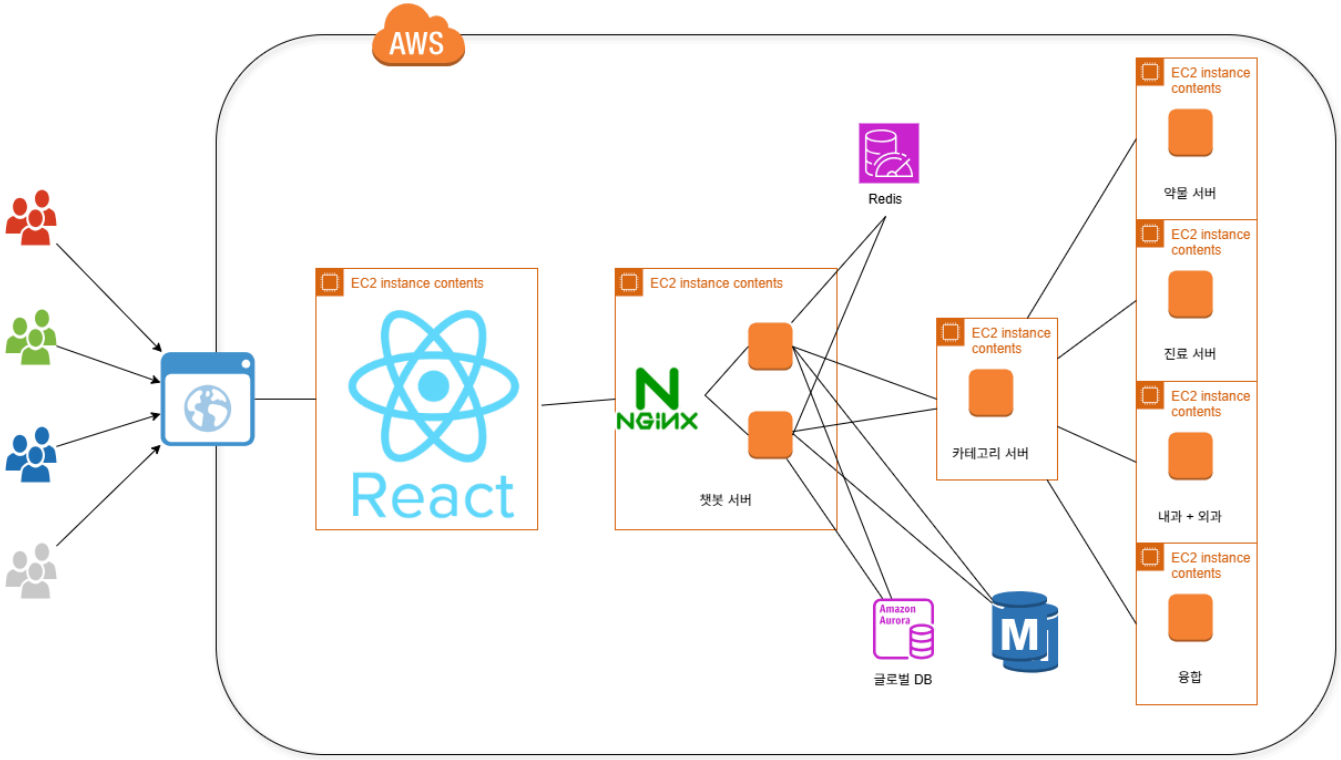
시스템 아키텍처 및 구성도

1. 전체 시스템 개요

- 본 시스템은 의료 정보 제공 및 챗봇 서비스를 위한 마이크로서비스 아키텍처(MSA) 기반의 웹 플랫폼입니다.
- 프론트엔드는 React로 구현되어 있으며, 백엔드는 FastAPI 기반의 여러 개별 서비스(챗봇, 카테고리, 진료, 약물, 응급지원, 내외과 등)로 구성되어 있습니다.
- 각 서비스는 AWS EC2 인스턴스에 배포되며, Nginx를 통해 트래픽을 분산합니다.
- 데이터 저장은 글로벌 DB(Amazon Aurora/MySQL), 샤드 DB, Redis 캐시를 활용합니다.

2. 시스템 구성도



---

3. 소프트웨어 설계와 마이크로서비스

- 각 서비스(챗봇, 카테고리, 진료, 약물, 내외과, 응급지원)는 FastAPI로 구현되어 있으며, 독립적으로 배포 및 운영됩니다.
- 서비스 간 통신은 HTTP REST API로 이루어집니다.
- Redis는 세션 및 대화 히스토리 관리에 사용됩니다.
- 데이터베이스는 글로벌 DB(중앙 관리)와 샤드 DB(사용자별 데이터 분산)로 구성되어 있습니다.
- 모든 서비스는 공통 템플릿 패턴(`template/`)을 상속하여 일관된 구조를 유지하고, 서비스 레이어(`service/`)에서 DB, 캐시, HTTP, AI 연동 등 공통 기능을 제공합니다.
- 이러한 소프트웨어적 설계 덕분에, 각 서비스가 독립적으로 동작하고, 유지보수 및 확장성이 용이한 마이크로서비스 구조가 구현되었습니다.

---

4. 히스토리 기반 챗봇

- 챗봇 서버는 사용자의 대화 히스토리를 Redis에 저장하고, 응답 생성 시 해당 히스토리를 불러와 맥락을 반영합니다.
- 대화 히스토리는 사용자가 메시지를 보낼 때마다 Redis에 저장되며, 챗봇 응답 생성 시 Redis에서 최근 대화 내용을 조회합니다.

---

5. 주요 기술 스택

- 프론트엔드: React, React Router, CSS3
- 백엔드: Python, FastAPI, aiomysql, aiohttp, dotenv
- 데이터베이스: MySQL(Aurora), Redis
- 인공지능/검색: LangChain, FAISS(VectorDB)
- 인프라: AWS EC2, Nginx
- 배포: Shell Script, 환경변수 기반 설정

---

6. 서비스별 주요 엔드포인트

- 챗봇 서버: `/account`, `/chatbot`
- 카테고리 서버: `/category`
- 진료 서버: `/clinic`
- 약물 서버: `/drug`
- 응급지원 서버: `/emergency-support`
- 내외과 서버: `/internal_external`

---

7. 실제 시스템 구성도 (AWS 기반, 그림 참고)

- 사용자 → 웹(React) → Nginx(API Gateway) → 각 FastAPI 서버(챗봇, 카테고리, 진료, 약물, 내외과, 응급지원)
- 각 서버는 글로벌 DB, 샤드 DB, Redis, VectorDB(FAISS)와 연동
- 모든 인프라는 AWS EC2 인스턴스에 배포

8. 미흡한 점 및 개선방안

미흡한 점

- 서비스 간 인증 및 보안: 서비스 간 통신 시 인증/인가 체계가 단순할 수 있음.
- 모니터링 및 장애 대응: 각 서비스의 상태 모니터링, 장애 발생 시 자동 복구 체계가 미흡할 수 있음.
- 배포 자동화: 서비스별 배포 자동화 및 롤백 체계가 부족할 수 있음.
- 트래픽 증가 대응: 대규모 트래픽 발생 시, 수평 확장 및 로드밸런싱 전략이 추가적으로 필요함.

개선방안

- 서비스 간 인증 강화: JWT, OAuth2 등 표준 인증 방식 도입 및 서비스 간 인증 체계 강화
- 모니터링 도입: Prometheus, Grafana 등 모니터링 도구 도입 및 알림 시스템 구축
- 장애 대응 자동화: 헬스체크, 오토스케일링, 자동 롤백 등 인프라 자동화 적용
- 데이터 일관성 관리: 분산 트랜잭션, 이벤트 소싱, CQRS 등 데이터 일관성 보장 기법 도입 검토
- **CI/CD** 구축: Github Actions, Jenkins 등으로 서비스별 자동 배포 및 롤백 체계 구축
- 트래픽 분산: 로드밸런서, 캐시, CDN 등 인프라 확장 및 최적화
- **AI** 품질 개선: 챗봇의 대화 품질 향상을 위해 추가적인 자연어 처리 모델 및 피드백 루프 도입