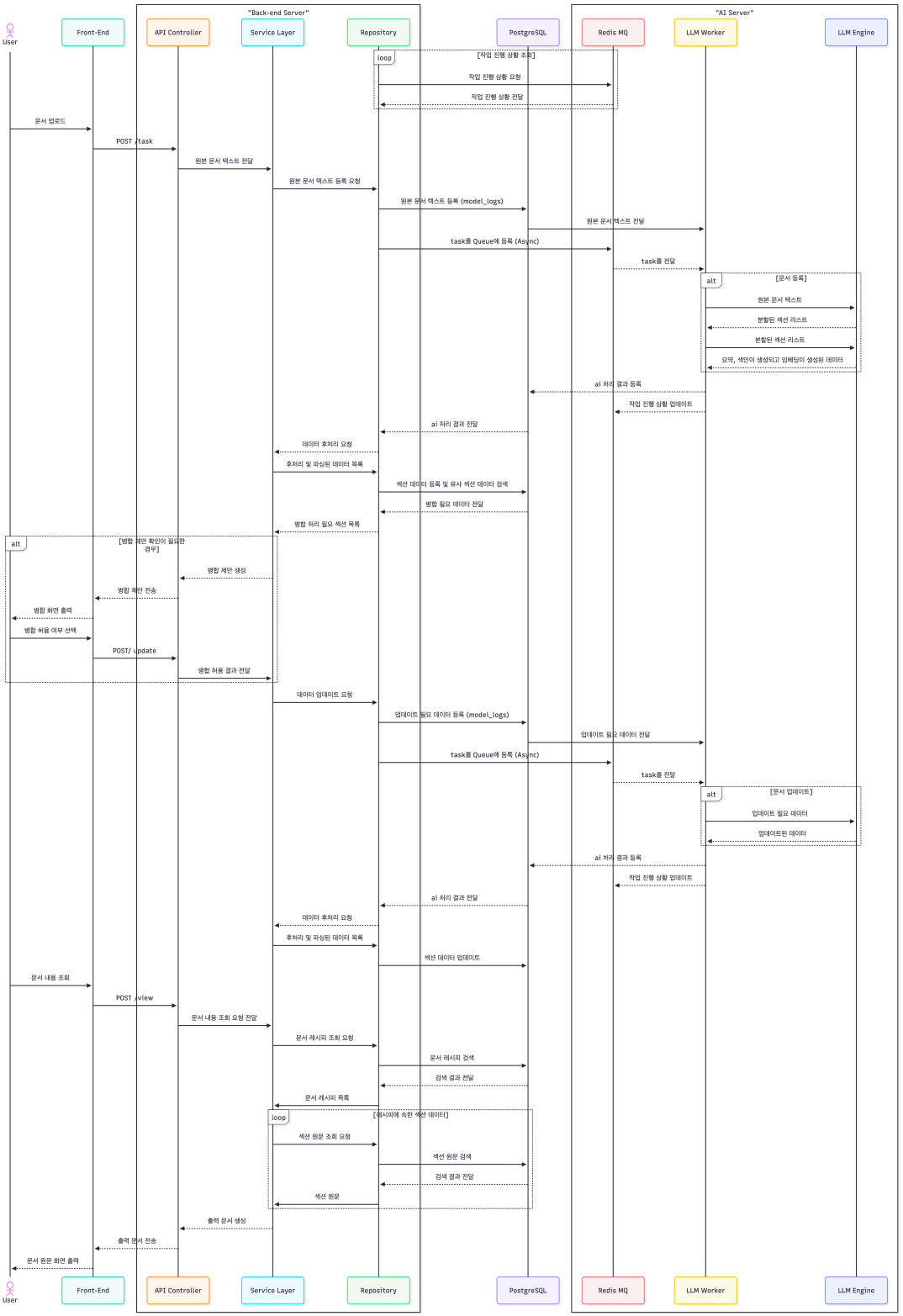
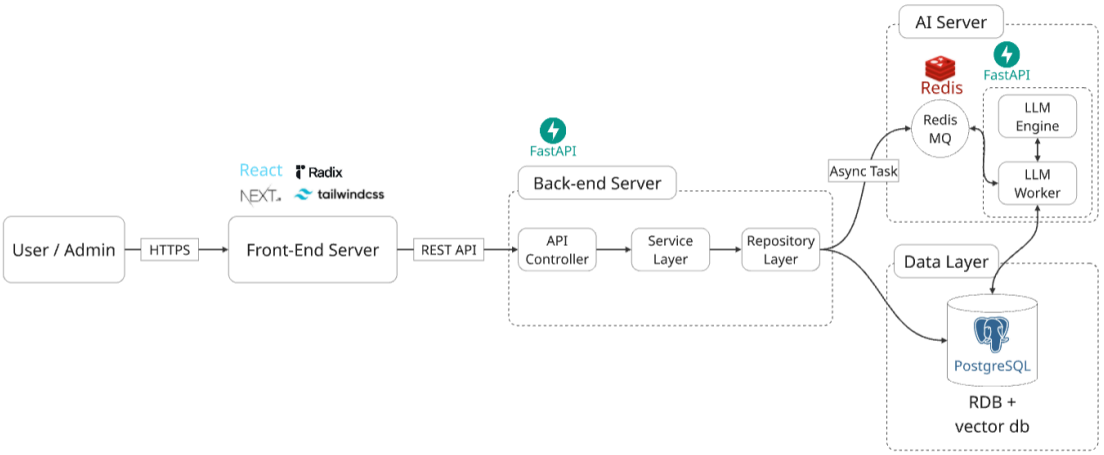
시스템 아키텍처


"Back-end Server"
"AI Server"
User
Front-End
API Controller
Service Layer
Repository
PostgreSQL
Redis MQ
LLM Worker
LLM Engine
loop
[작업 진행 상황 조회]
작업 진행 상황 요청
작업 진행 상황 전달
문서 업로드
POST /task
원본 문서 텍스트 전달
원본 문서 텍스트 등록 요청
원본 문서 텍스트 등록 (model_logs)
원본 문서 텍스트 전달
task를 Queue에 등록 (Async)
task를 전달
alt
[문서 등록]
원본 문서 텍스트
분할된 섹션 리스트
분할된 섹션 리스트
요약, 섹인이 생성되고 임베딩이 생성된 데이터
ai 처리 결과 등록
작업 진행 상황 업데이트
ai 처리 결과 전달
데이터 후처리 요청
후처리 및 파싱된 데이터 목록
섹션 데이터 등록 및 유사 섹션 데이터 검색
병합 필요 데이터 전달
병합 처리 필요 섹션 목록
alt
[병합 제안 확인이 필요한 경우]
병합 제안 생성
병합 제안 전송
병합 화면 출력
병합 허용 여부 선택
POST/ update
병합 허용 결과 전달
데이터 업데이트 요청
업데이트 필요 데이터 등록 (model_logs)
업데이트 필요 데이터 전달
task를 Queue에 등록 (Async)
task를 전달
alt
[문서 업데이트]
업데이트 필요 데이터
업데이트된 데이터
ai 처리 결과 등록
작업 진행 상황 업데이트
ai 처리 결과 전달
데이터 후처리 요청
후처리 및 파싱된 데이터 목록
섹션 데이터 업데이트
문서 내용 조회
POST /view
문서 내용 조회 요청 전달
문서 레시피 조회 요청
문서 레시피 검색
검색 결과 전달
문서 레시피 목록
loop
[레시피에 속한 섹션 데이터]
섹션 원문 조회 요청
섹션 원문 검색
검색 결과 전달
섹션 원문
출력 문서 생성
출력 문서 전송
문서 원문 화면 출력



Front-End

1. 언어

- TypeScript (TSX 파일 기반으로 작성)
  - Next.js에서 1급 지원(최우선 기술 스택)
  - 정적 타입 기반 안정성 확보
- HTML5 / CSS3

2. JS 라이브러리

React 19.2.0

- 보안 취약점(CVE-2025-55182)을 개선한 최신 버전 사용
- 컴포넌트 재사용, 적용이 쉬우며 다양한 라이브러리가 존재
- 단순 SPA가 아닌 서버-클라이언트 혼합 구조

3. 풀스택 웹 프레임워크

Next.js 16.0.10

- 미들웨어 우회(CVE-2025-29927), 서버 컴포넌트 RCE(CVE-2025-55182)을 개선한 최신 버전 사용
- SSR (server side rendering)에 적합
- API Routes 및 서버 로직 포함
- Node.js 기반 실행 환경

3. 스타일링 / 아이콘 / UI 컴포넌트

3.1 Tailwind CSS ^4.1.9

- 유틸리티 퍼스트 CSS
- PostCSS + Autoprefixer 기반

- 빠른 디자인 시스템 구축에 적합

3.2 Lucide React ^0.454.0

- 다양한 SVG 아이콘을 지원하는 라이브러리

3.3 Radix UI (버전관리가 컴포넌트별로 이뤄지기에 버전정보는 미기입)

- Headless, unstyled UI 컴포넌트 라이브러리
- 접근성(A11y) 우선

3. 문서 렌더링

React-Markdown 10.1.0

- 마크다운 문서 뷰어 기능을 구현하기 위해 사용

4. 상태 / 폼 / 검증

react-hook-form ^7.60.0

- 퍼포먼스 중심 폼 관리
- uncontrolled input 기반

zod 3.25.76

- 스키마 기반 유효성 검사
- 타입 추론 지원
- 서버/클라이언트 공통 검증 가능

@hookform/resolvers ^3.10.0

- react-hook-form ↔ zod 연결

통신규약

REST API

- 프론트-백엔드 간 통신 규약을 명확히 하여 의존성 감소 및 독립적 배포 가능

Back-End Server

SOLID 원칙을 적용한 계층형 아키텍처 (Layered Architecture)

FastAPI

- Python 기반의 AI 라이브러리 호환성 확보
- 높은 동시성 처리를 지원하여 I/O 작업 효율화
- Swagger 자동 생성으로 API 문서화 및 협업 비용 절감

계층형 아키텍처 (Controller - Service - Repository)

- SRP (단일 책임 원칙) 준수
  - **Controller**: 요청 검증 및 응답 처리 담당
  - **Service**: 순수 비즈니스 로직 담당
  - **Repository**: 실제 DB 쿼리 및 데이터 접근 담당
- **DIP (의존성 역전 원칙) 적용**: Service가 구체적인 DB 구현체가 아닌 추상화된 로직에 의존하도록 설계하여, 추후 DB 변경 시 사이드 이펙트 최소화
- **테스트 용이성**: 계층 분리로 Mocking을 활용한 단위 테스트 작성이 수월함

AI Server

시스템 안정성(Stability)과 확장성(Scalability)을 고려한 비동기 작업 처리 구조

RedisMQ (Message Broker)

- 무거운 LLM 추론 작업을 비동기 큐로 넘겨 API 응답 지연(Blocking) 방지
- 작업 실패 시 재시도(Retry) 로직을 통해 안정성 확보

LLM Worker 분리

- **장애 격리 (Fault Tolerance)**: API 서버(Engine)와 추론(Worker) 프로세스를 분리하여, 모델 과부하(OOM 등) 발생 시 전체 서비스 다운 방지
- **유연한 확장**: 트래픽 증가 시에는 API 서버만, 연산량 증가 시에는 Worker(GPU)만 독립적으로 증설 가능

Database

PostgresSQL

- **관리 포인트 단일화**: 별도 Vector DB 구축 없이 `pgvector` 확장 기능을 사용
- 관계형 데이터와 벡터 데이터 간의 조인(Join) 효율성 및 데이터 무결성 유지