

온프레미스 AI + RPA: 지능화된 ALL in ONE 업무 최적화 시스템

#On-Device AI #RAG #Vector Database
#WebSocket #실시간 협업

Team : 2조

팀 프로젝트 주소 : https://github.com/01nJun/Fullstack_AIDesk

팀장 | 김민식 : <https://github.com/minsik321>

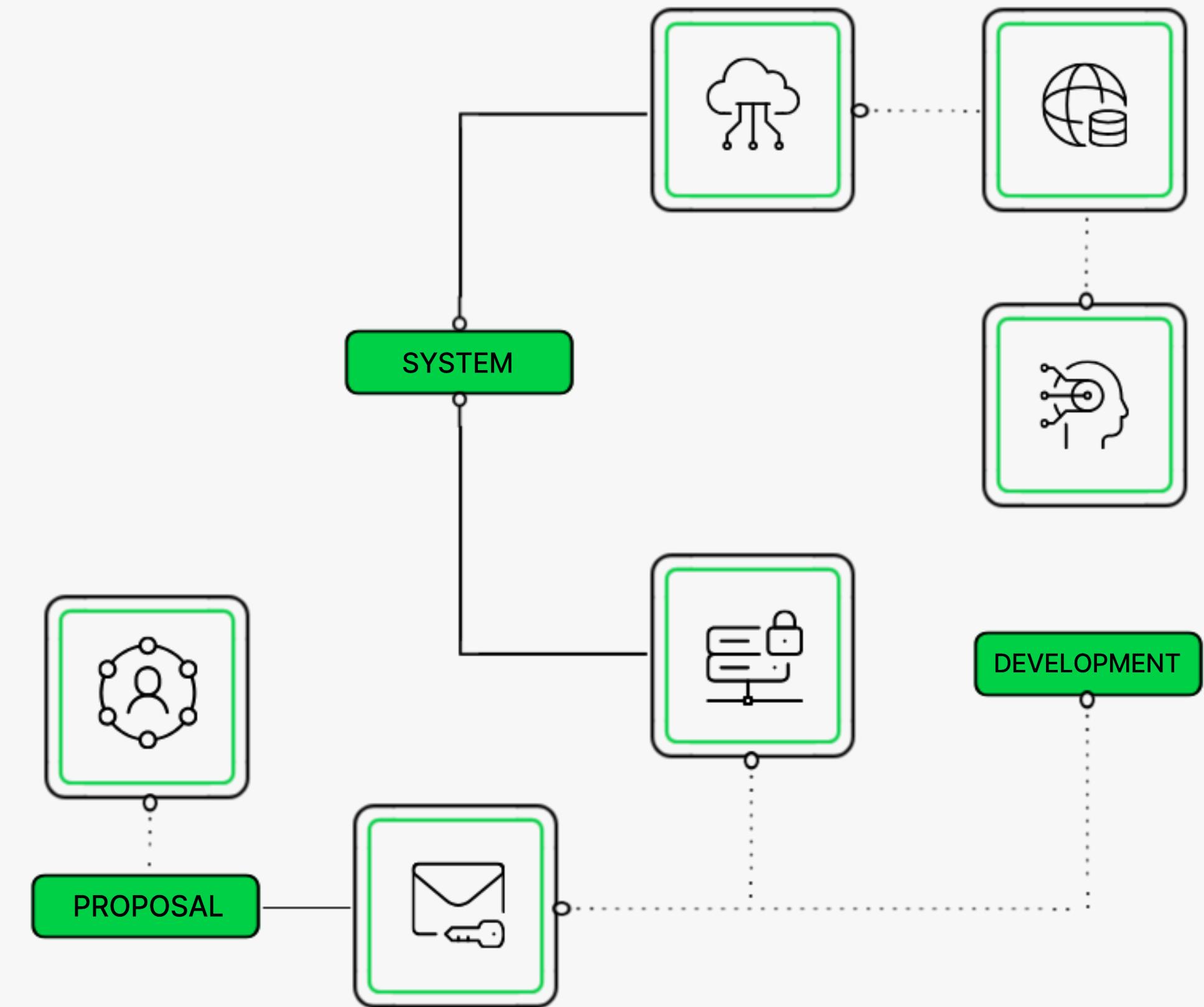
팀원 | 한정연 : <https://github.com/DOT-SOY>

팀원 | 박태오 : <https://github.com/teomichaelpark-glitch>

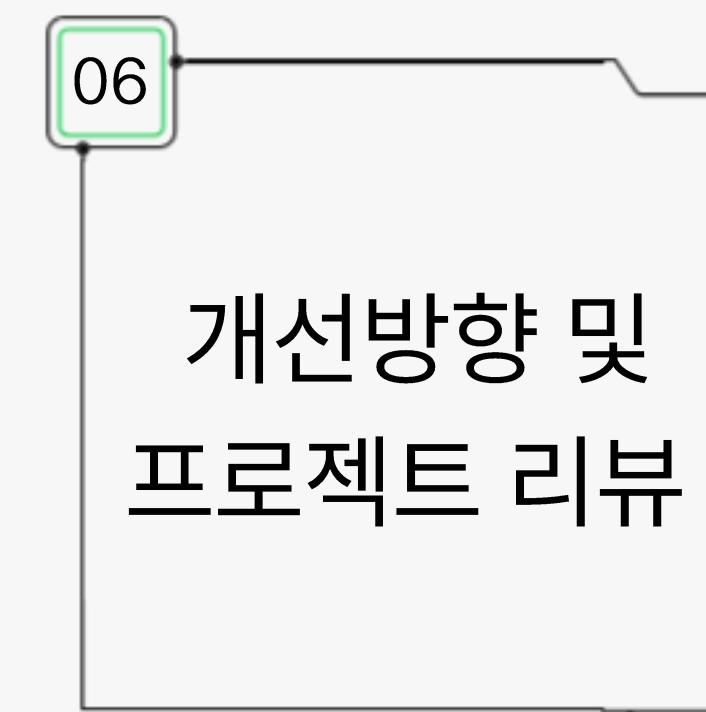
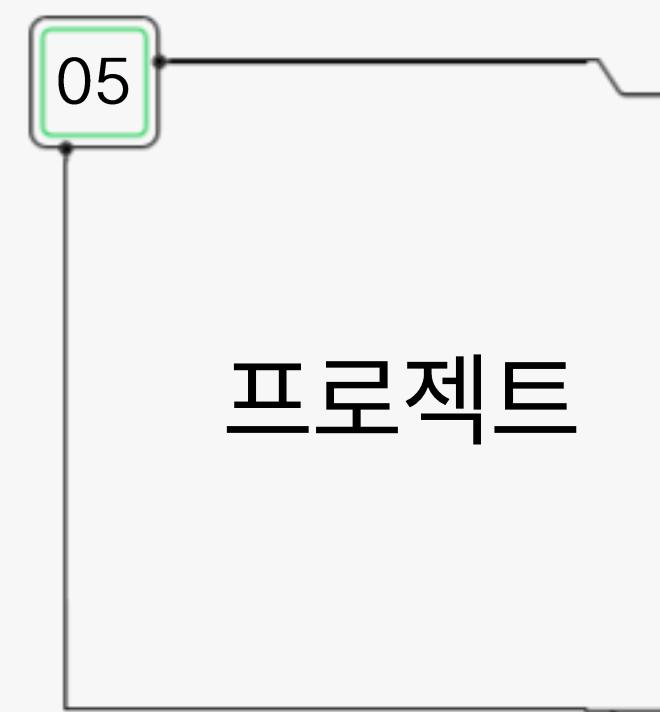
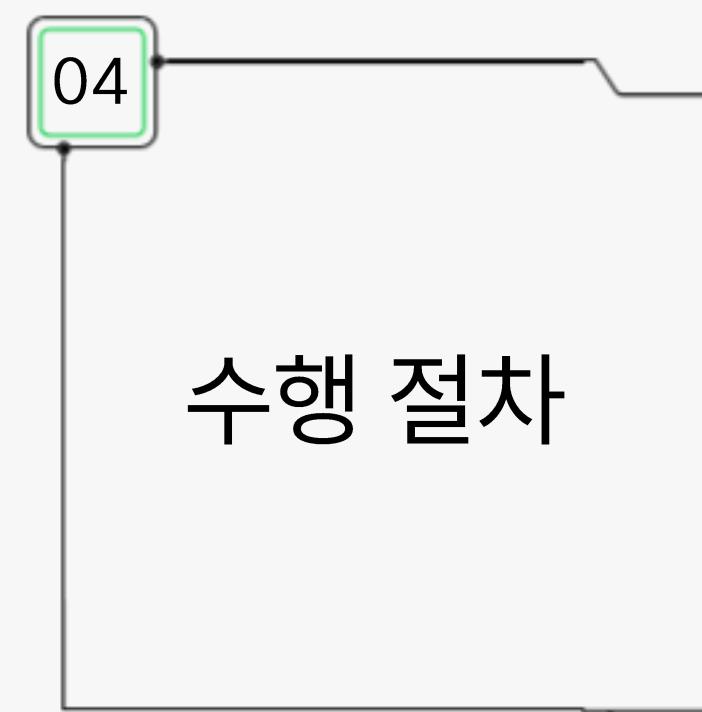
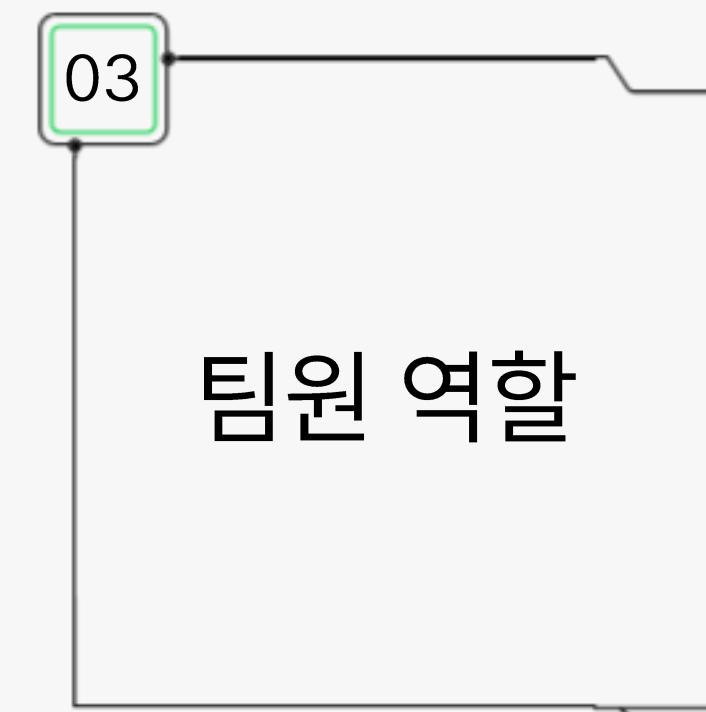
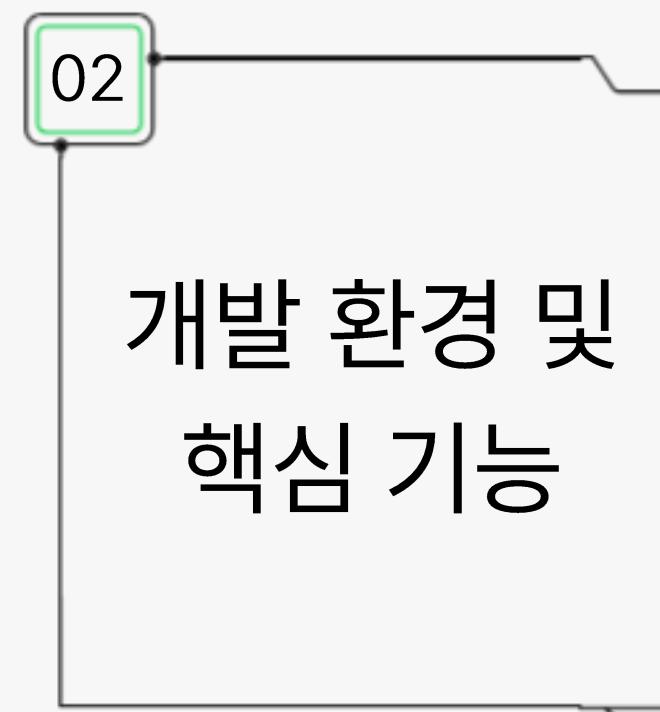
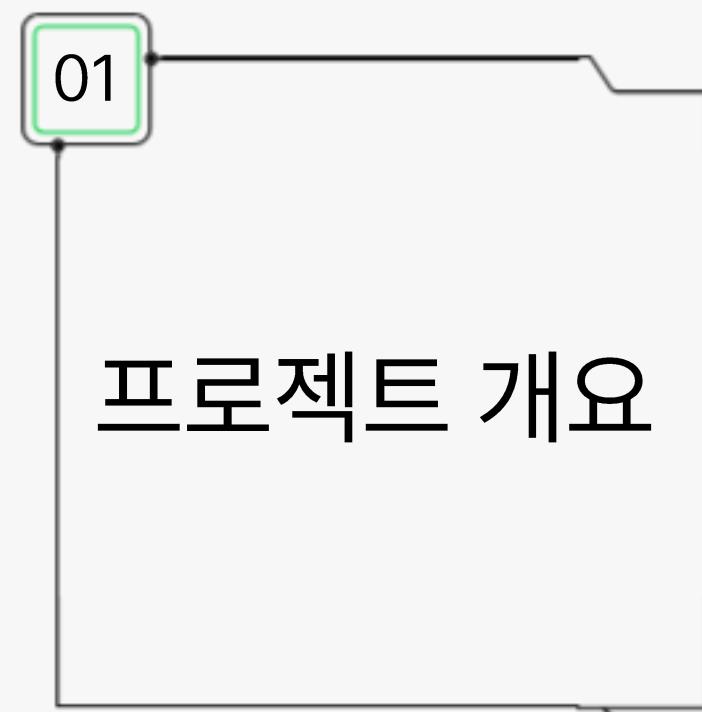
팀원 | 박건영 : <https://github.com/keonyeong4550>

팀원 | 오인준 : <https://github.com/01nJun>

팀 노션 주소 : <https://www.notion.so/2cce455ae480803d8925c84065cf4f0b>



목차 소개



프로젝트 개요

01

기획 배경 및 문제 의식

- 직장 내 소통 오류로 인한 갈등과 스트레스
- 부정확한 소통은 불필요한 마찰과 업무 속도, 완성도의 저하 -> 생산성 감소
- 회의 후 정리 및 배포까지의 시간 소요로 인한 업무 실행 지연

해결 방안

- 자동 완성
 - 사용자의 언어를 AI가 분석하여 업무 제목, 내용, 담당자 등을 자동으로 추출해 업무 요청 초안을 생성.
- 역질문
 - "언제까지 해야 하나요?", "이 작업의 목적은 무엇인가요?" 등 필수 정보가 빠졌을 때 AI가 먼저 질문하여 완성도 높은 요청서를 제작
- 감정 필터링
 - 채팅에서 격한 표현이나 비속어가 감지 시, AI가 자동 필터링하여 팀 내 감정 싸움을 예방.
- AI 회의록 자동화
 - 회의 녹취 mp3 파일을 올리면 AI가 이를 텍스트로 변환 후 요약, PDF로 제작.



https://www.fnnews1.com/news/articleView.html?idxno=103794&utm_source=chatgpt.com

개발 환경

분야	개발도구	개발 기능
AI	Ollama (On-Device AI, Qwen3:8B), OpenAI API (Whisper STT), Spring AI, Python AI Server, PostgreSQL pgvector	<ul style="list-style-type: none"> - 온디바이스 AI 기반 대화형 티켓 생성 (Ollama Qwen3:8B) - RAG 기반 지식 검색 (벡터 임베딩 + pgvector) - 실시간 메시지 필터링 (Ollama) - STT 기반 음성 자동화 (OpenAI Whisper-1) - 얼굴 인식 벡터 추출 (Python AI Server)
Front	HTML, CSS, JavaScript, React, Axios, Redux Toolkit, React Router, Tailwind CSS, SockJS, STOMP	<ul style="list-style-type: none"> - React SPA 기반 반응형 웹 페이지 - Axios를 활용한 비동기 호출로 프론트엔드와 백엔드 원활한 통신 - WebSocket을 활용한 실시간 채팅 시스템 - 무한 스크롤 및 게시판/티켓 페이지네이션 - Redux Toolkit으로 로그인 상태 및 핀 상태 전역 관리 - Tailwind CSS를 활용한 UI
Back	Java , Spring Boot , Spring Security, Spring Data JPA, JWT, QueryDSL, MariaDB, Redis, WebSocket/STOMP, Lombok, ModelMapper, iText7	<ul style="list-style-type: none"> - Spring Security 및 JWT Token을 활용한 보안 강화 및 관리자 권한 검사 - MariaDB를 활용한 메인 데이터베이스 운영 및 JPA를 통한 ORM - PostgreSQL을 벡터 데이터베이스로 활용한 얼굴 인식 벡터 저장 - Redis를 활용한 Refresh Token 저장 및 세션 관리, 화이트리스트 기반 안전한 토큰 재발급 - QueryDSL을 활용한 동적 쿼리 - WebSocket(STOMP)을 활용한 실시간 양방향 통신 - iText7를 활용한 PDF 생성 및 처리 기능 - 대용량 파일 업로드 지원(최대 100MB)
API	Kakao Login API	<ul style="list-style-type: none"> - 카카오 API를 활용한 소셜 로그인 시스템 구현
배포	Aws , 젠킨스 CI/CD	<ul style="list-style-type: none"> - EC2,RBS,IAM,S3,VPC,Elastic Beanstalk를 활용하여 인스턴스 배포 진행 - CI / CD 자동 배포

팀원 역할

이름	Part 및 상세 내용
김민식(팀장)	<ul style="list-style-type: none">- 프로젝트 총괄- 로그인 및 회원가입 : JWT 이용 보안 처리 및 백엔드 Spring Security 사용하여 구현- 소셜 로그인 통합 : 카카오 로그인을 JWT 기반 인증으로 통합하고 관리자 승인 정책 동일 적용- 임베딩 벡터 기반 얼굴 인식 로그인 구현 및 MariaDB + PostgreSQL DB 분리 설계- Redis 이용한 Refresh Token Rotation 정책 구현, 화이트리스트 기반 토큰 재발급- Redis를 이용한 일정 횟수 로그인 실패 시 일정 시간 로그인 잠기는 기능 구현)- 관리자 페이지 : 승인, 삭제, 검색 및 필터, 페이징 처리 기능 구현)- 업무 관리 시스템 : 업무 요청서 목록, 필터링, 중요업무(Pin) 기능 구현)- 파일 관리 시스템 : 파일 업로드/다운로드, 파일함 기능 구현)- 메인 대시보드(중요 업무, 읽지 않은 업무 등 사용자 정보 통합 대시보드 구현)- Redux Toolkit을 활용한 중요업무(Pin) 전역 상태 관리 및 사용자별 동기화- 프로젝트 총괄
한정연	<ul style="list-style-type: none">- Notion / Github 형상관리-SockJS + STOMP 기반 실시간 채팅 구현, JWT 기반 인증이 적용된 WebSocket 통신 구현- 채팅 메시지 무한 스크롤(Spring Data JPA페이지네이션 기능 활용)- Ticket CRUD 및 단건 조회 API- 업무 관리 시스템 백엔드 CRUD- AI 메시지 필터링(Ollama API 연동을 통한 메시지 필터링, On / Off 토글 가능하도록 구현)- WebClient 기반 AI 비동기 처리(Spring WebClient로 Ollama API 비동기 호출, 논블로킹 구조 적용)
박태오	<ul style="list-style-type: none">- AI 업무티켓: Routing → 담당자 확정(DB 검증) → RAG+JSON 인터뷰 기반 자동 작성- AI 업무티켓: JSON 템플릿 기반반복 패턴 템플릿 우선 매칭 + LLM fallback- AI 파일조회: JAVA 규칙 기반으로 1차 파싱, 애매한 입력은 AI로 보완하는 하이브리드 검색 로직 구현- AI 파일조회 ACL: 티켓/채팅 파일 view/download 권한 검증 통합, 보안누수 방지- AI 비서함 통합 UI: 채팅·업무티켓·파일조회 모달/위젯 단일 진입점 및 전환 UX- AI 채팅 가드/제재 : 채팅 사고 방지 안전장치를 구현- 채팅 파일 첨부(드래그&드랍) 구현
박건영	<ul style="list-style-type: none">- AI 기반 회의록 자동 생성 및 PDF 변환- OLLAMA AI를 이용한 텍스트 정규화(Normalization) + 문장 최적화 후 파일 자동 첨부 및 다운로드 구현- JAVA 정규 표현식 + 파일 기반 시스템 불용어(Stopword) 필터링
오인준	<ul style="list-style-type: none">- 공지사항 + 댓글(대댓글) CRUD(권한별 제한) 구현- 공지사항 페이지 제목 / 카테고리 별 검색 구현, 페이징 처리- AI 비서 회의록 MP3 파일 업로드 및 STT 구현(Spring AI를 통하여 Open AI(whisper-1)사용)- 회의록 / 수행일지 작성

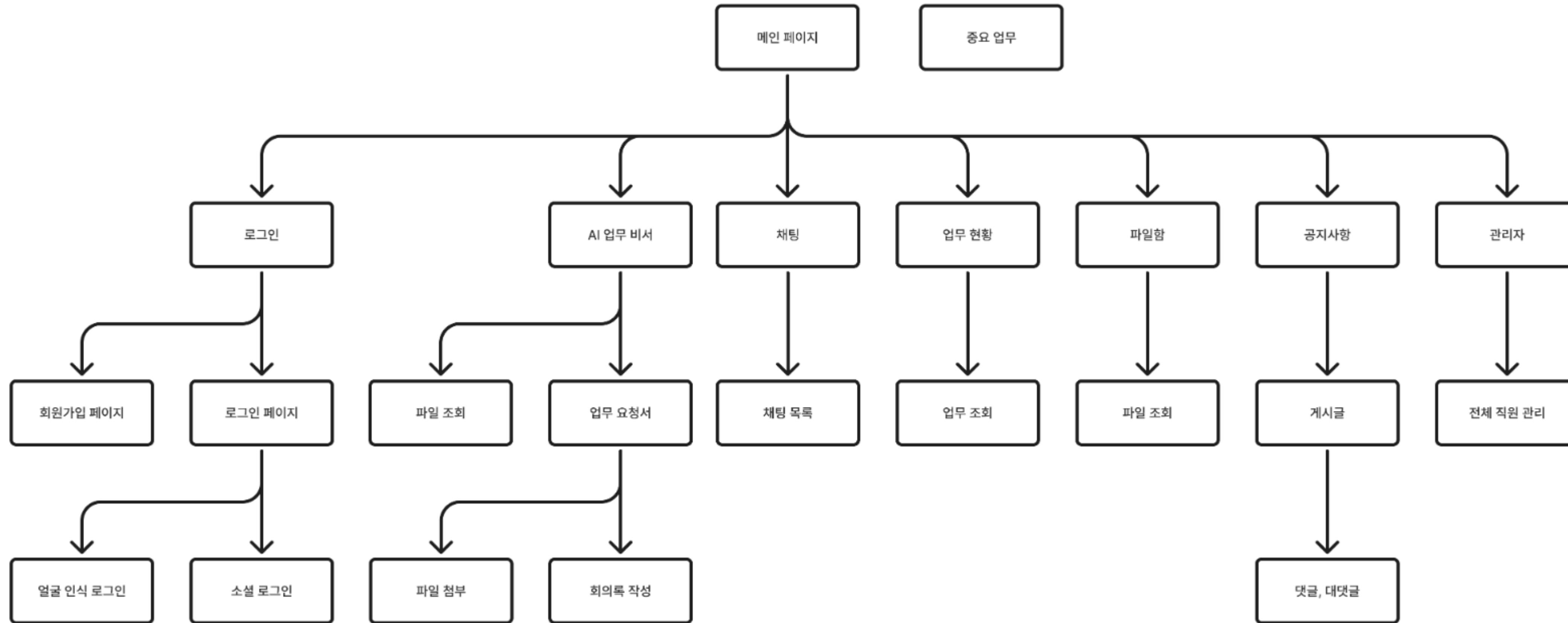
핵심 AI 기능

구현기능	기능 설명
AI 임베딩	얼굴 이미지를 Python AI 서버로 벡터 변환 후 PostgreSQL pgvector에 저장, L2 거리 기반 유사도 검색으로 얼굴 인식 (Spring WebFlux 활용)
AI 언어 필터링	Ollama(Qwen3:8B)로 채팅 메시지의 욕설/비속어를 정중한 업무용 문장으로 자동 변환, Spring WebFlux 기반 실시간 처리
AI 회의록 요약	MP3 음성 파일을 OpenAI Whisper-1(STT)로 텍스트 변환, Ollama(Qwen3:8B)로 요약, iText7로 PDF 자동 생성 원스톱 프로세스
AI 파일 조회	Ollama(Qwen3:8B)로 자연어 질문 분석, Komoran 형태소 분석기로 키워드 추출, Spring Data JPA로 관련 파일 검색
AI 업무 작성	Ollama(Qwen3:8B) 기반 3단계 대화형 프로세스(라우팅→담당자→인터뷰)로 티켓 정보 자동 수집 및 구조화, RAG 기반 지식 검색으로 누락 정보 자동 탐지 후 질문 생성

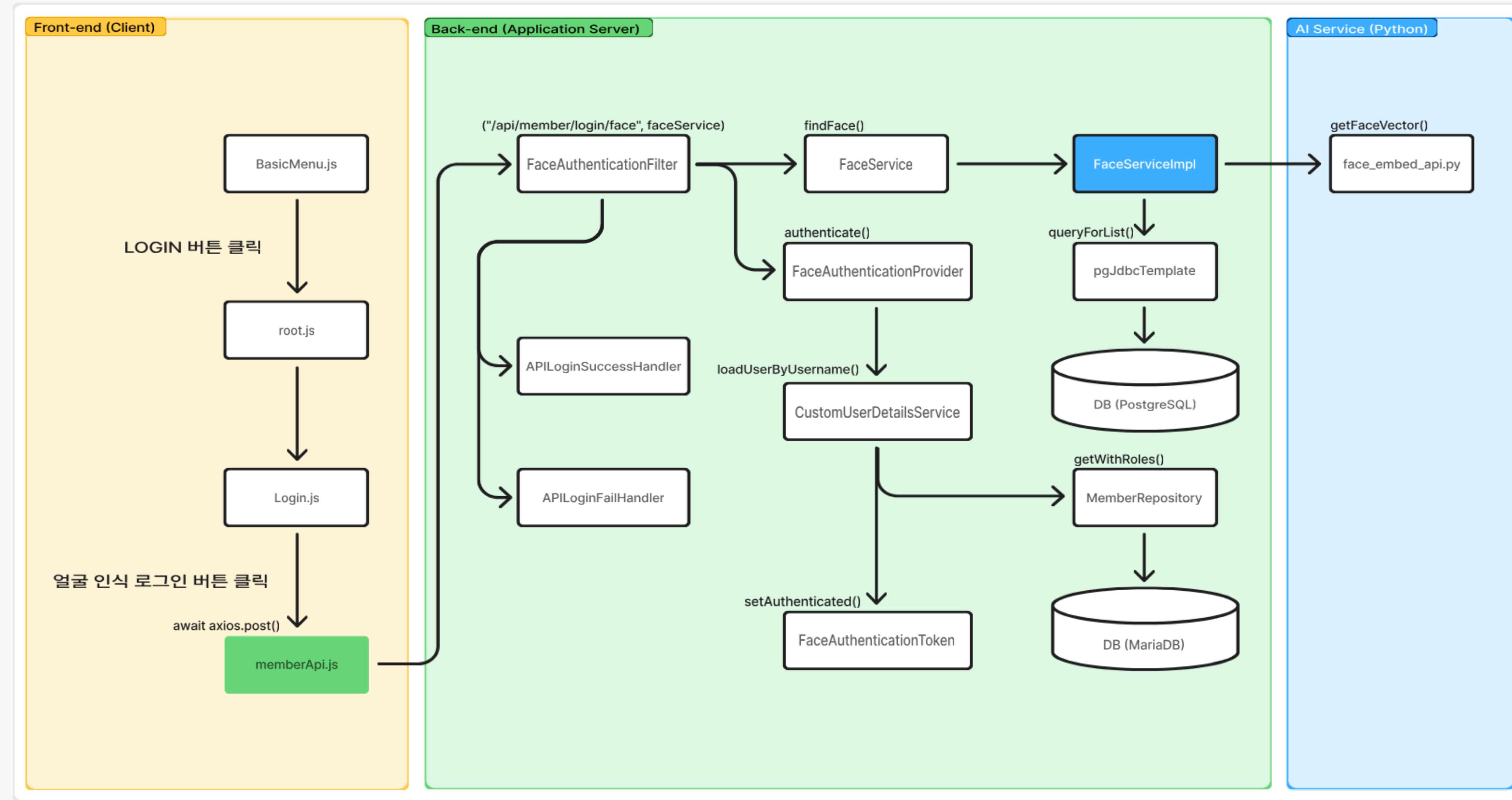
수행 기록

구분	기간	활동	비고
기획	12/18 ~ 12/20	프로젝트 주제 선정 및 기획 자연어 기반 AI 티켓 생성 시스템 아이디어 도출 팀원 역할 분담 기술 스택 조사 (Ollama, OpenAI등) 주요 기능 정의	온디바이스 AI(Ollama) 활용 방안 검토 API 연동 방안 검토
설계	12/21 ~ 12/27	시스템 아키텍처 설계 데이터베이스 설계 Ollama, STT, RAG 기술 테스트 하이브리드 AI 파싱 전략 설계 API 설계 및 프롬프트 설계 UI/UX 설계 보안 설계	Ollama 모델 설정 및 테스트 RAG 시스템 설계 (벡터 임베딩 + 지식 베이스)
구현	12/28 ~ 1/10	인증/보안 구현 핵심 AI 기능 구현 (임베딩, 채팅, 업무티켓 (RAG), AI 파일 검색, 음성 회의록 자동 요약(STT+PDF)) 채팅 시스템 구현 게시판 구현 파일 관리 구현 관리자 기능 구현	1/6(월) 1차 통합 테스트 1/10(금) 2차 통합 테스트
최종 구현	1/11 ~ 1/15	통합 테스트 및 버그 수정 성능 최적화 UI/UX 개선 보안 점검 및 취약점 보완 PDF 문서 제작 및 데모 영상 촬영	주요 기능 점검 및 개선 보안 설정 점검 (JWT, Refresh Token, CORS 등)

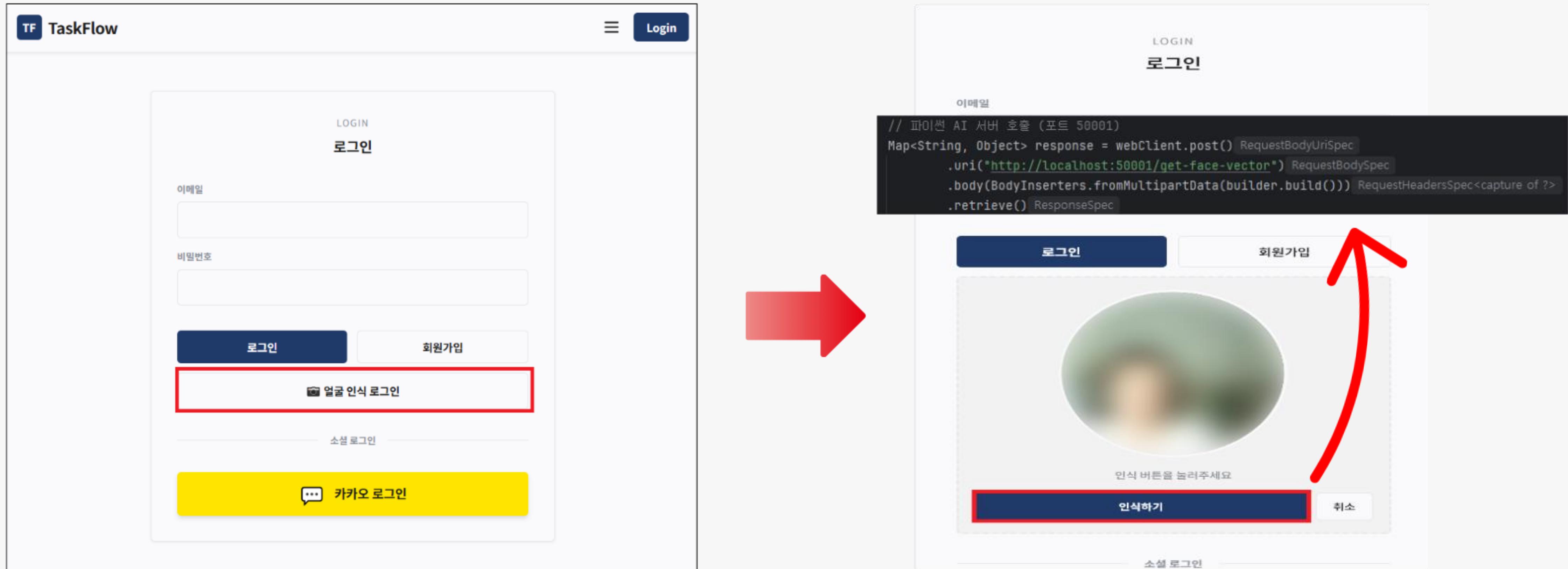
전체 유스다이어 그램



임베딩(얼굴인식) UML



임베딩(얼굴인식)



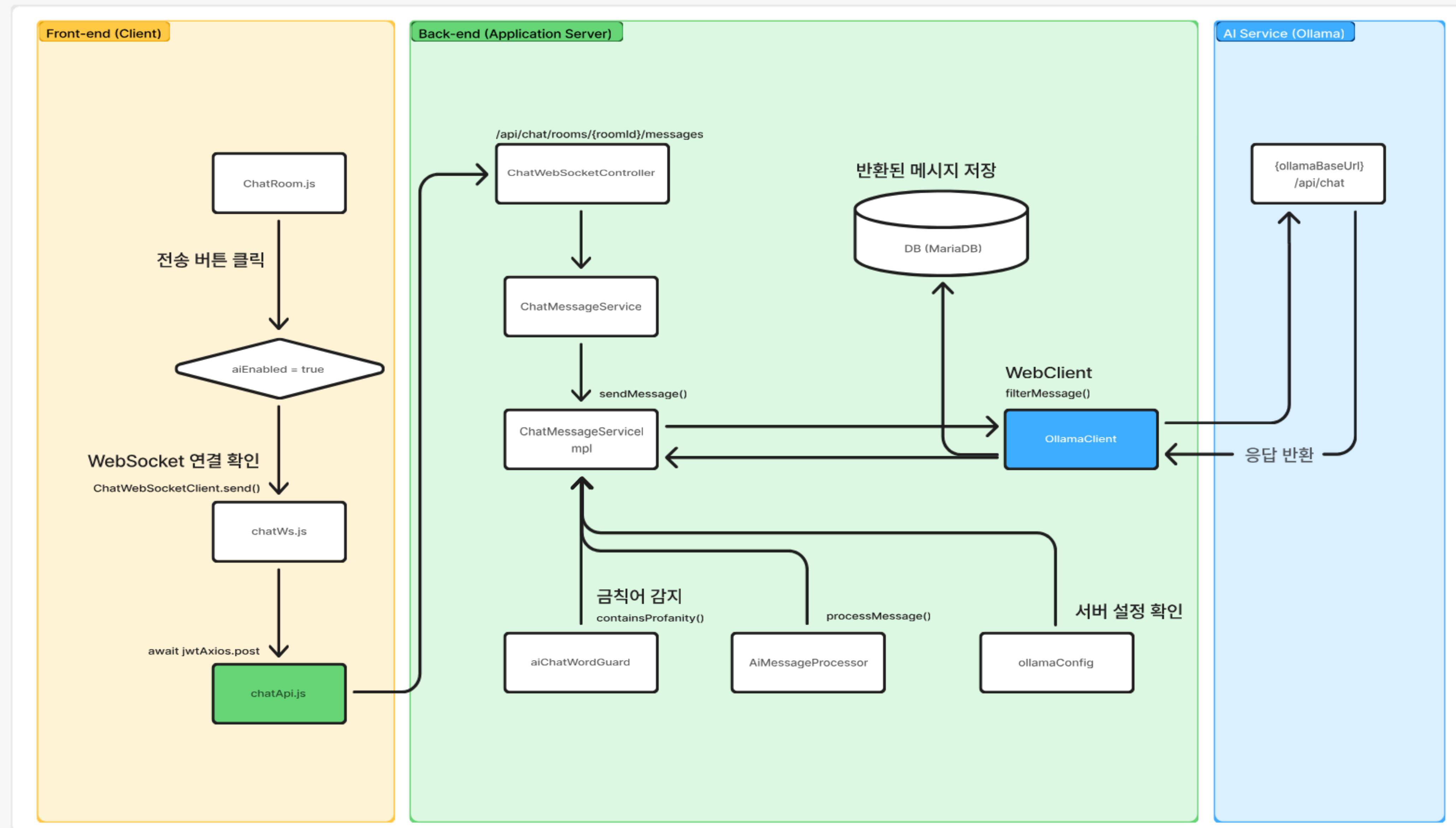
1. 정확한 얼굴 인식을 위해 검출과 식별로 분리하여 처리

- YOLOv8-Face 모델로 이미지에서 얼굴 영역 검출
- ArcFace 모델을 통해 이미지를 임베딩 벡터로 변환

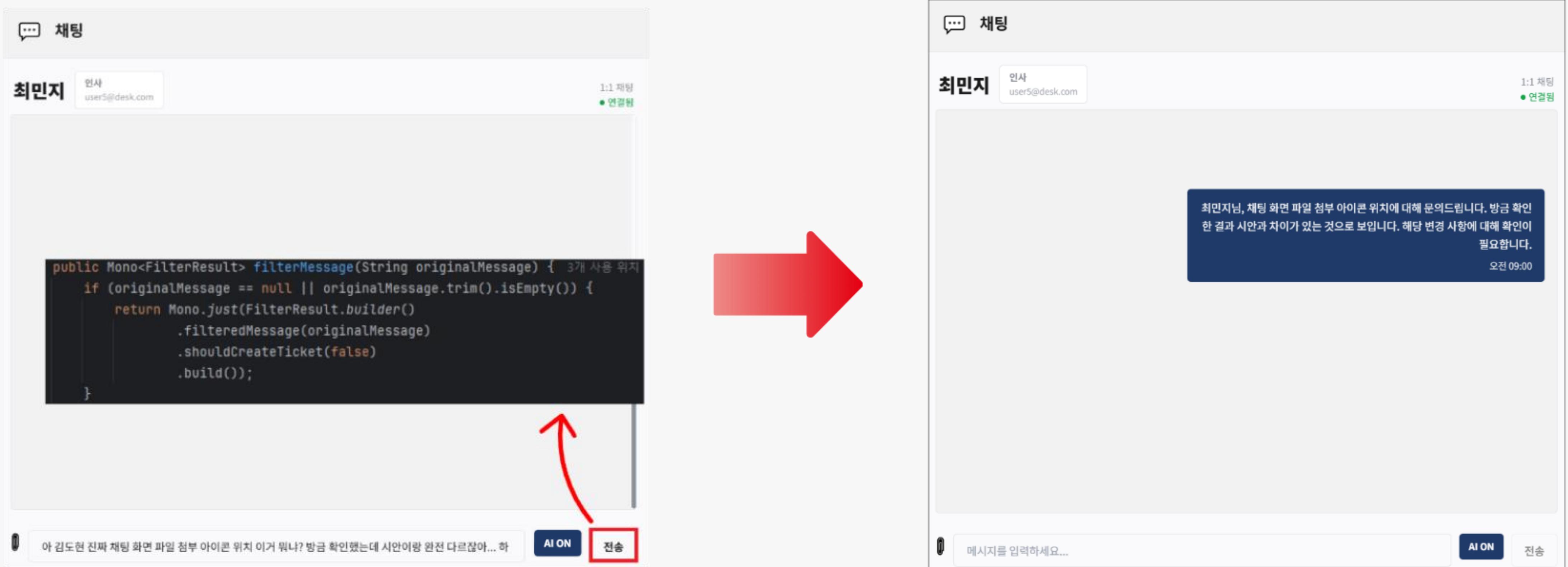
2. 마이크로 서비스 아키텍처

- 메인 로직에 영향을 주지 않도록, Python(FastAPI)기반 독립 서버 구축
- ArcFace 모델을 통해 이미지를 임베딩 벡터로 변환

채팅AI UML

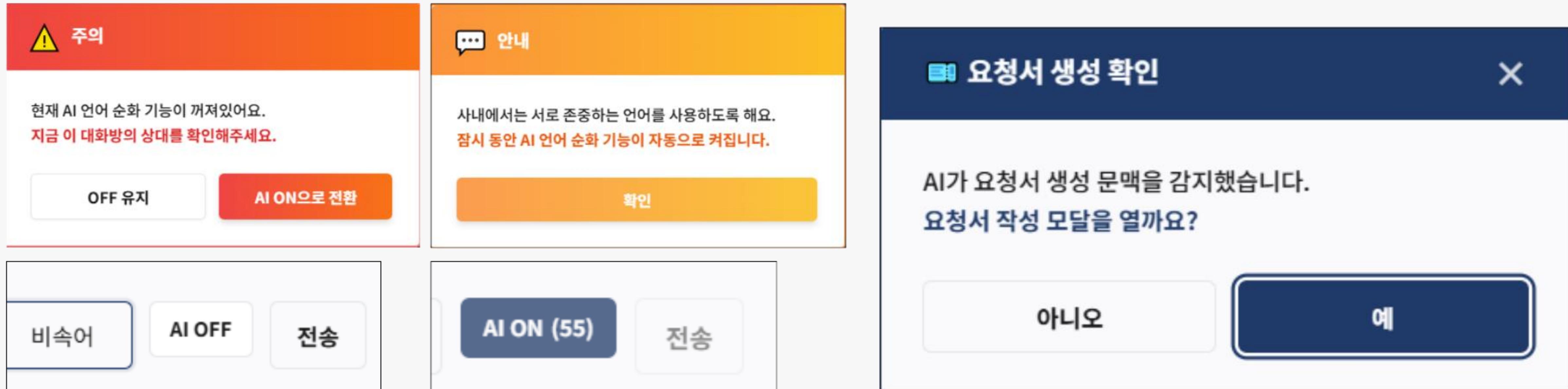


채팅 AI



- Case A (AI ON): 모든 문장을 LLM 프로세스로 전달합니다.
평범한 문장도 "정중한 업무용 어조"로 커뮤니케이션 품격을 높입니다.
- Case B (AI OFF + Clean): AI가 꺼져 있고 금칙어가 없다면, Java 를 엔진만 통과한 후 즉시 전송합니다.

채팅 AI



- Case C (AI OFF + Dirty)

AI OFF 일때 금칙어가 감지되면, 시스템이 이를 Case A(LLM 프로세스)로 경로를 변경합니다.

- 반복 욕설 제재 시스템 : 10초 내 위반 카운트 2회 -> 경고 팝업

● 팝업 이후 위반시 60초간 강제로 AI 필터 ON, 클라이언트 조작으로 해제 불가능.

-

- 채팅으로 업무 요청 → 확인 모달 → 기능 화면 이동 (사용자 흐름 중심의 유기적 기능 설계)

-

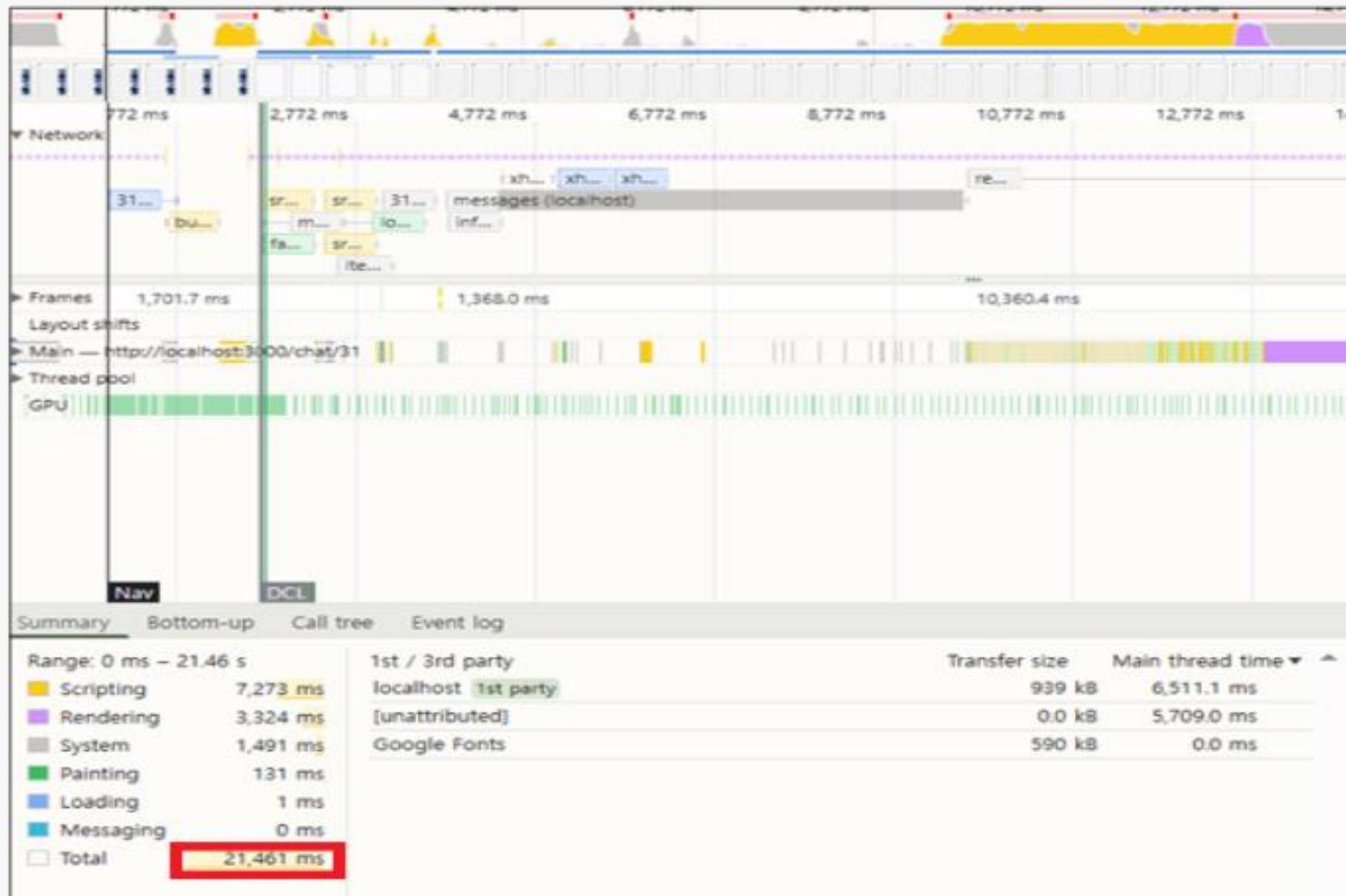
1. 하이브리드 메시지 정제 파이프라인 (The Filtering Logic)

- 모든 메시지를 AI에 태우는 비효율을 없애고, 사용자 설정(ON/OFF)과 위험도에 따라 처리 경로를 동적으로 스위칭하는 조건부 라우팅 알고리즘을 적용

2. 능동형 채팅 가드 & 제재 매커니즘(Active Guard & Penalty)

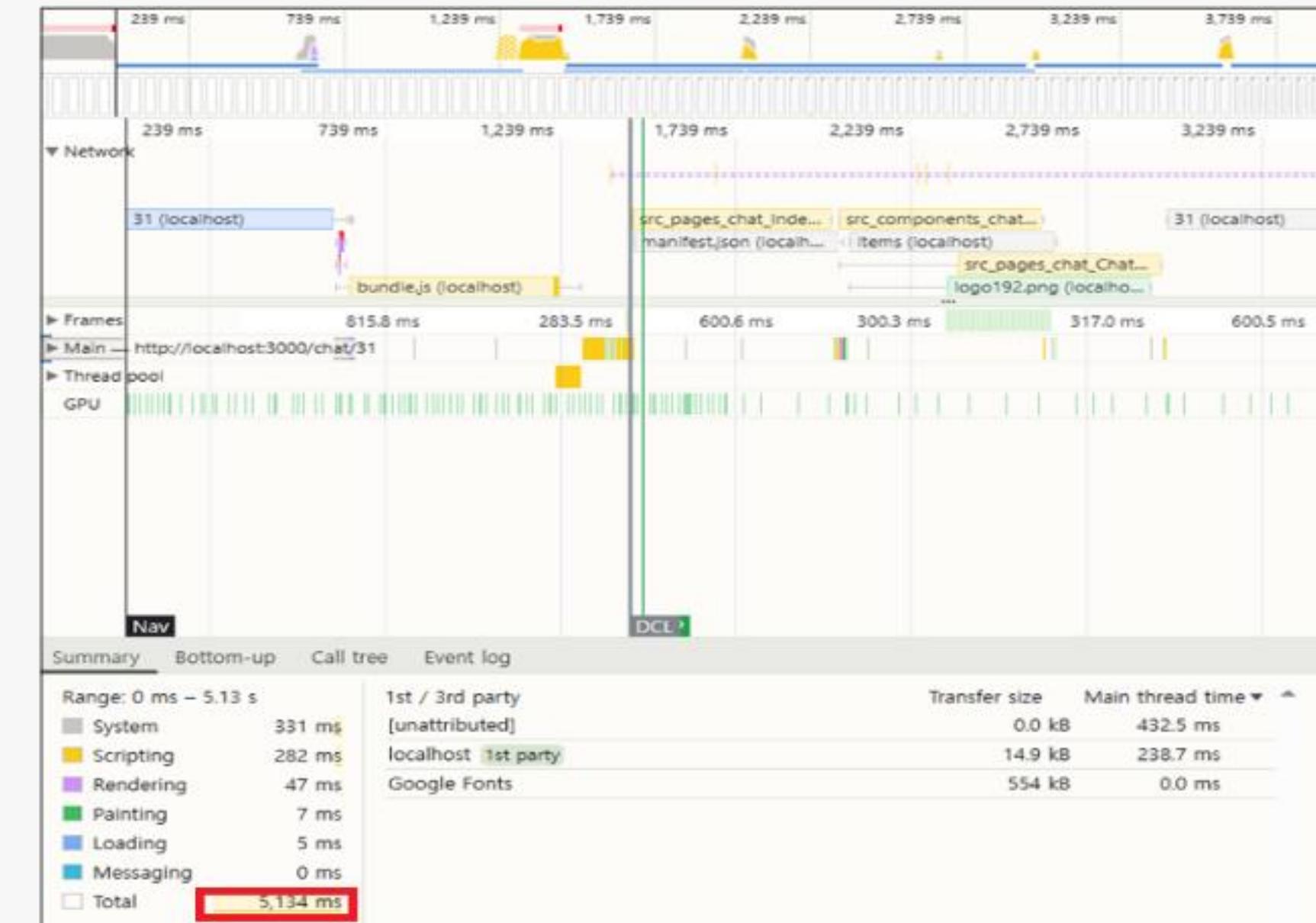
- 클라이언트가 AI_OFF 플래그를 보내더라도, 백엔드 인터셉터 계층에서 금칙어가 감지되는 순간 해당 세션의 속성을 AI_FORCE_ON 상태로 강제 전환 합니다.

무한 스크롤 성능 비교



무한 스크롤 적용 전

무한스크롤 적용 후 전체 성능이 압도적으로 개선됨
특히 Scripting Time, Rendering Time, DOM 개수가 극적으로 감소
대량 메시지를 한 번에 렌더링하는 방식은 브라우저 성능에 매우 치명적이다.
무한스크롤 적용으로 DOM 개수를 23,655 → 224로 줄였고, Total Time은 약 72%,
Scripting Time은 약 96% 감소했다.

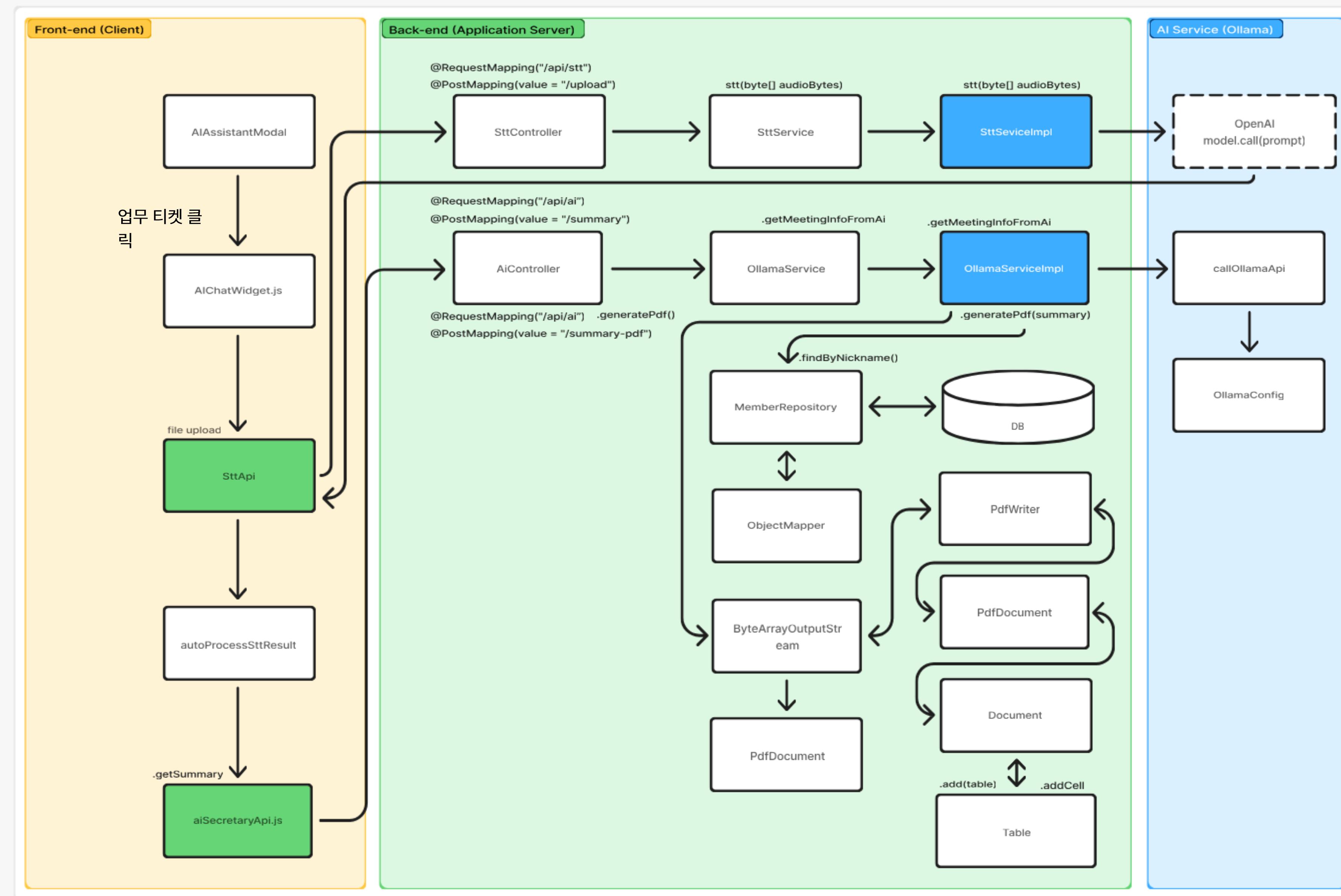


무한 스크롤 적용 후

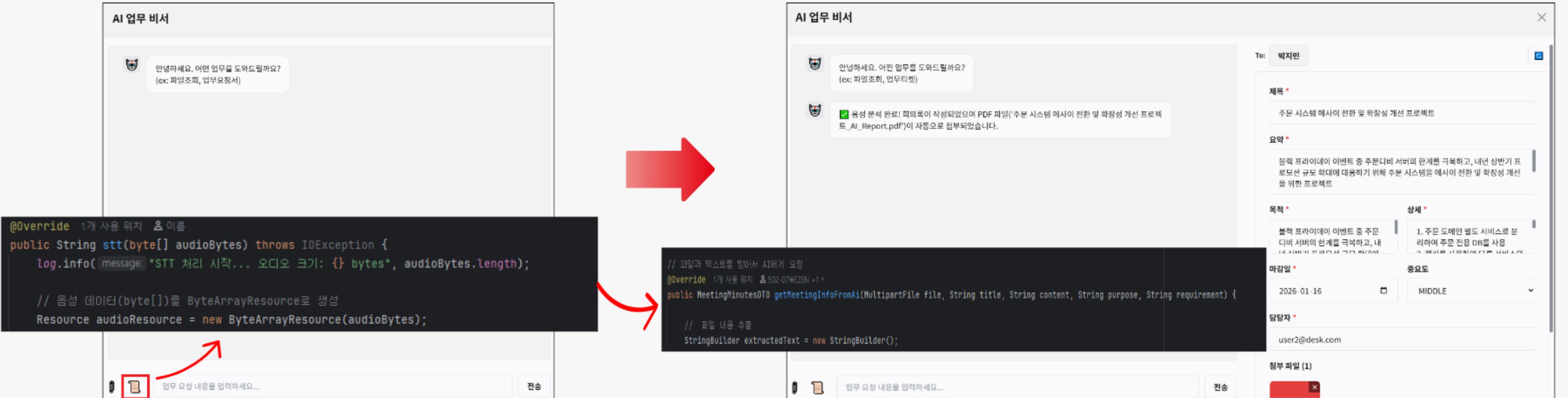
성능 비교

항목	적용 X	적용 O
메세지 로드 방식	1만개 한번에 로드	20개씩 분할로드
Total Time	18,193 ms	5,134 ms
Scripting Time	6,805 ms	282 ms
Rendering Time	3,304 ms	47 ms
DOM Nodes	23,655개	224개
체감 성능	매우 느림/끊김 발생	부드러움

AI 회의록 UML



AI 자동화 회의록



1. STT 및 텍스트 정제 (STT & Text)

■ **AI transcription**: 오디오 파일을 OpenAI Whisper 모델이 고정밀 텍스트로 변환.

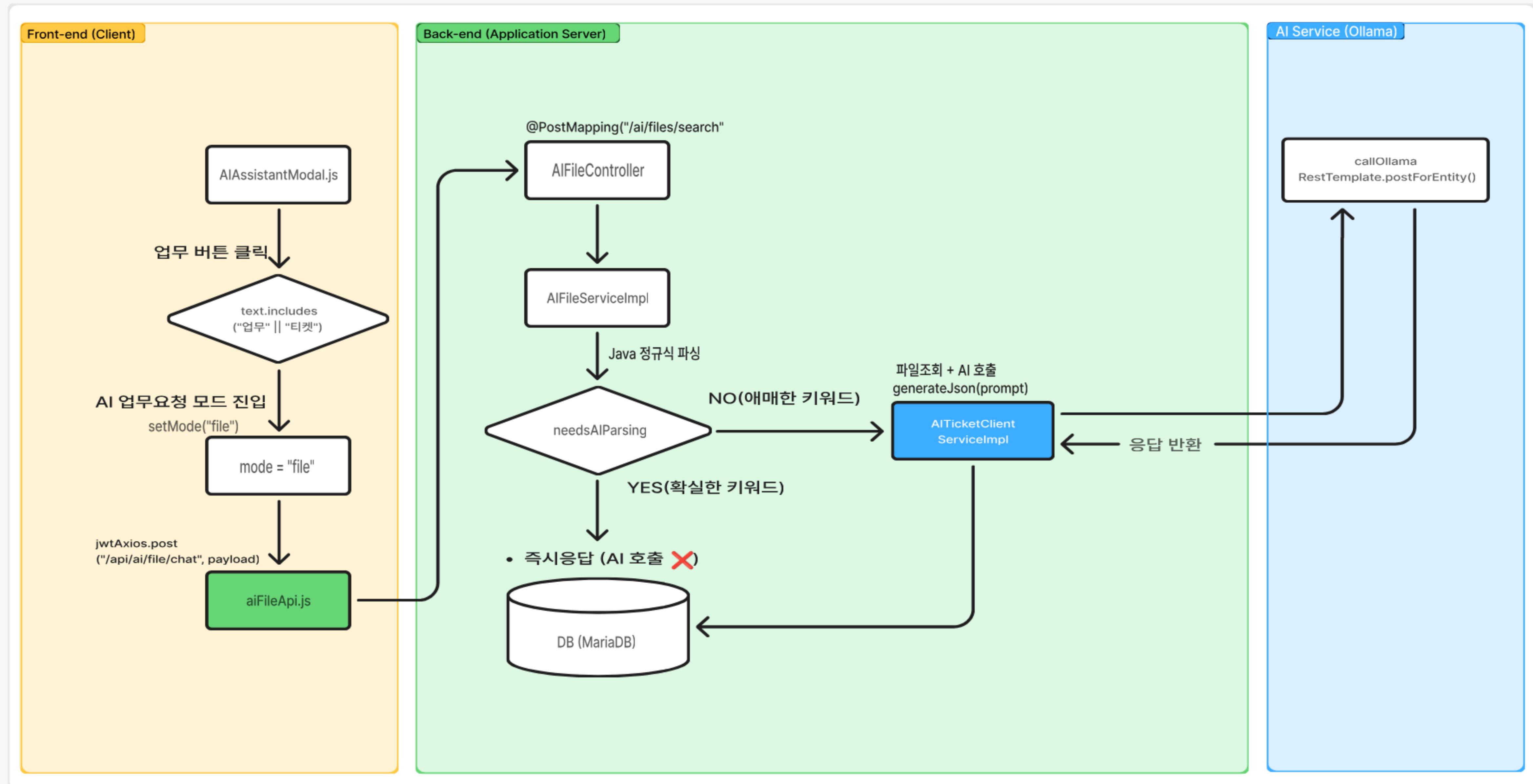
- JAVA 정규 표현식 + 파일 기반 시스템으로 불용어를 제거.

2. 구조화된 요약 및 정보 추출 (Structured Summarization)

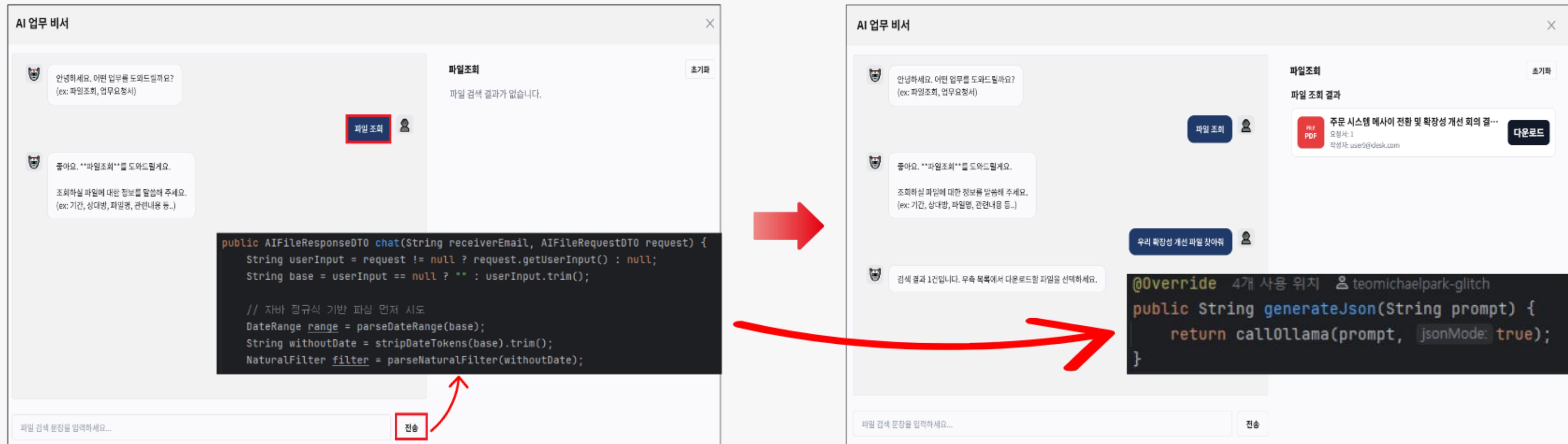
■ **AI summarization**: "요약해줘"라고 하지 않고, 명확한 JSON 스키마를 제시하여 정보를 추출

- PDF 자동 생성 및 업무요청서에 통합

AI 파일 조회 UML



AI 파일 조회



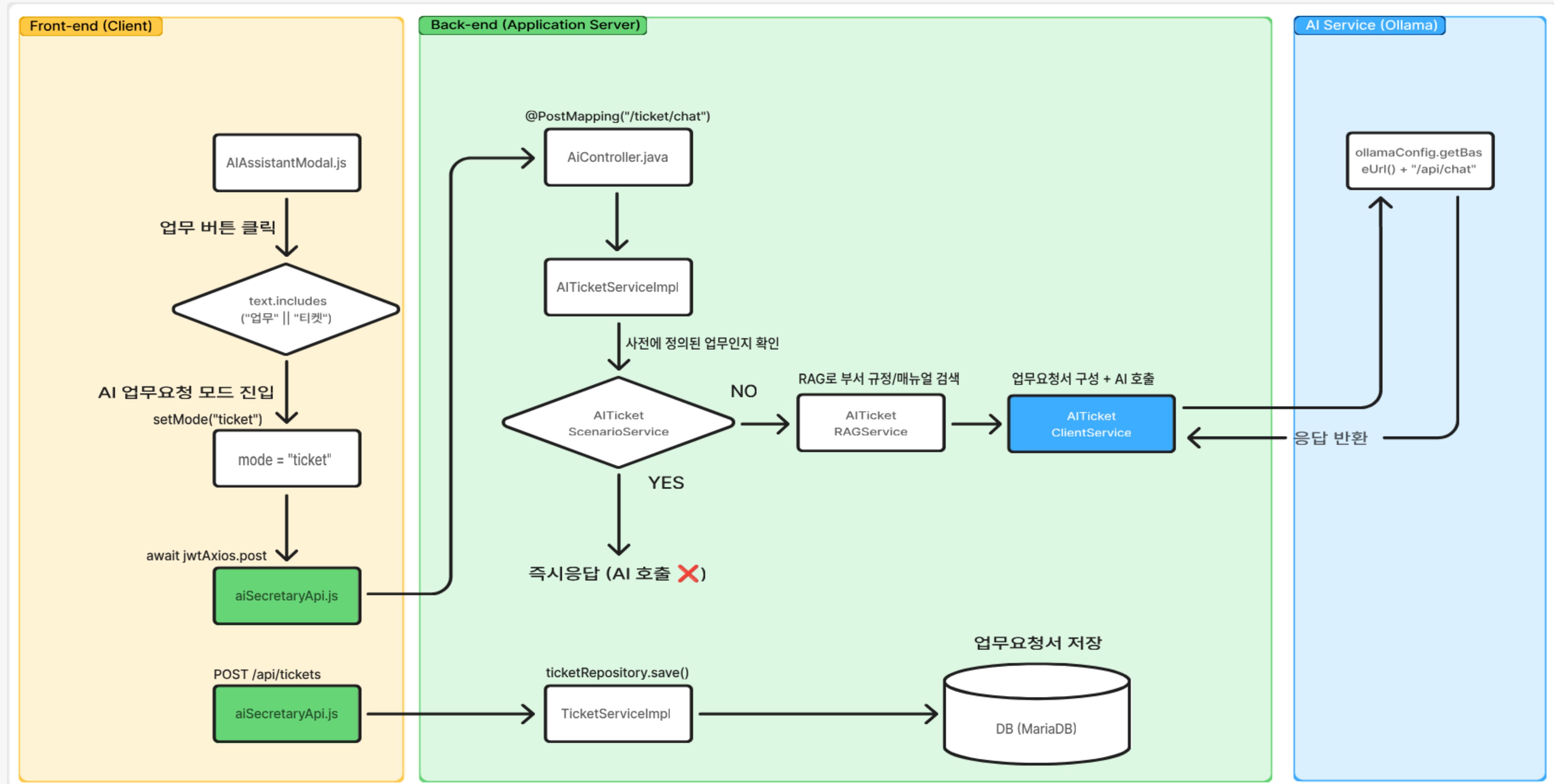
1. 하이브리드 파싱 전략 (Hybrid Parsing Strategy)

- Java 정규식과 규칙 엔진이 1차적으로 투입.
(ex : 기간, 이메일, 대상)
- 애매한 표현이나, 1차 파싱 결과가 거의 없을 때만 AI를 호출합니다.

2. 검색 전략 고도화: 0건 방지 UX (Zero-Result Prevention)

- Strict Mode** : 조건을 완벽하게 만족하는 파일을 1차로 찾습니다.
- AI Re-parsing** : 결과가 0건이면, AI를 호출하여 다시 시도합니다
- Overlap Mode** : 가장 조건이 겹치는 유사 결과를 찾아냅니다

AI 업무 요청 UML



AI 업무 요청

AI 업무 비서

To: 황시우

제목: 포스터 디자인 제작 요청

요약:

목적: 신제품 런칭 행사용 A1 포스터 디자인

상세: 행사/프로모션 포스터 제작

마감일: 2026-01-22

중요도: HIGH

담당자: design01@desk.com

첨부 파일 (0)

업무 요청 내용을 입력하세요...

전송

AI 업무 비서

말씀하신 내용을 바탕으로 업무 티켓을 구성했습니다.

업무 티켓 개요

- 제목: 신제품 런칭 행사용 A1 포스터 디자인
- 목적: 신제품 홍보 및 행사장 거치용 인쇄물 제작
- 담당: 황시우님 (디자인팀)

[상세 요구사항 정리]

- 컨셉: 쉽하고 자유분방한 무드 (진정적인 기업 느낌 시양)
- 레이아웃: 넥스트 최소화 / 제품 이미지 크게 강조
- 제작 사양: A1 사이즈 / 인쇄용 고해상도 파일
- 문제해결: 배경(IVORY, Warm White) + 포인트(Brand Blue)
- 핵심 문구: 'NEW PRODUCT LAUNCH' (상단/Bold)
- 참고: 침부된 레퍼런스 이미지 무드 반영

위 내용으로 진행하겠습니다.
마감기한과 중요도를 함께 알리주세요. (예: 다음주 금요일, 보통)

알겠습니다. 마감기한은 다음주 금요일로 설정하겠습니다.

중요도는 어떻게 되나요? (보통 / 긴급)

다음주 금요일

업무 요청 내용을 입력하세요...

전송

업무 요청서 전송

1. 하이브리드 파싱 전략 + RAG 기반 지식 주입

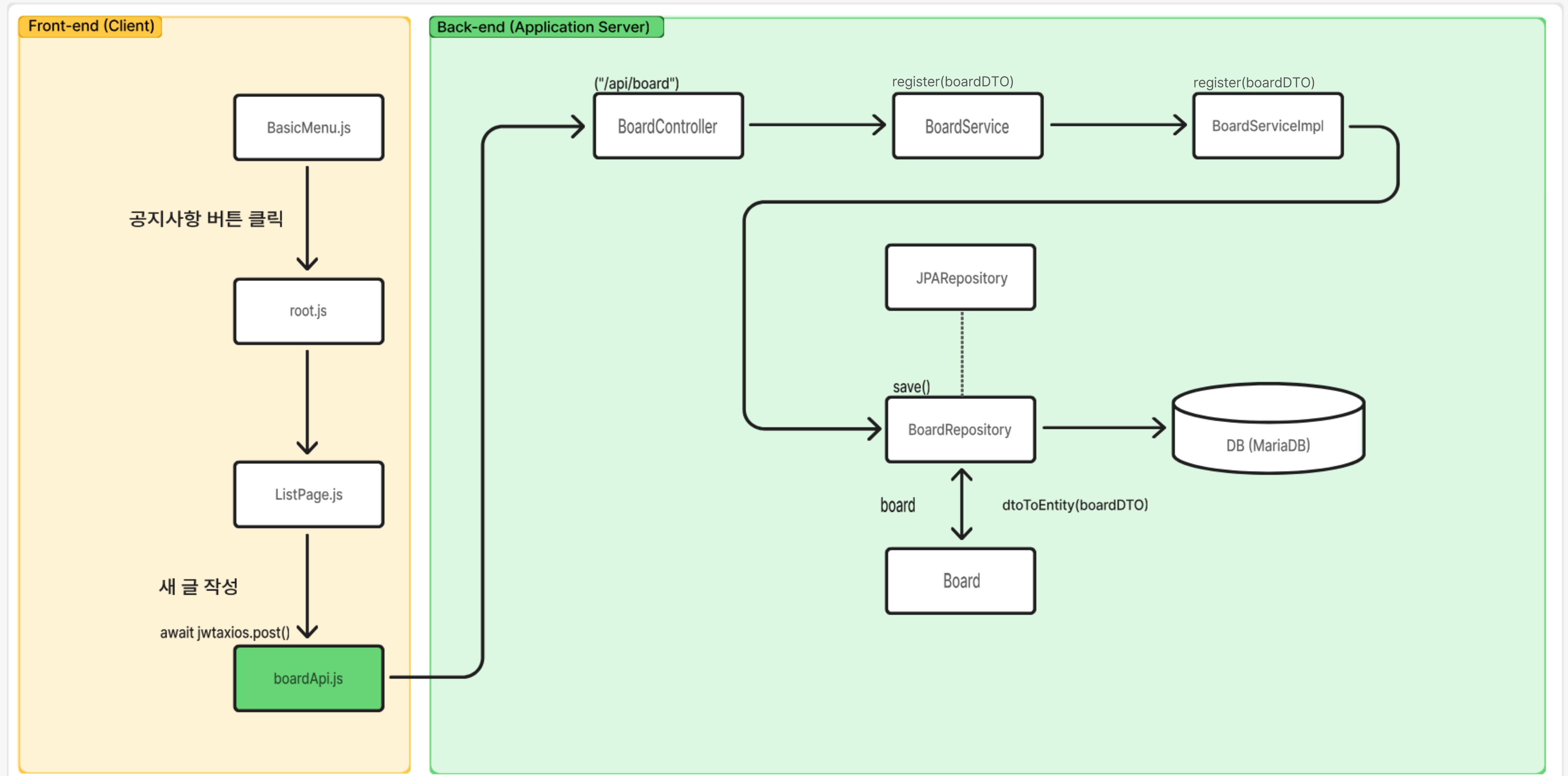
- 사내 업무 규정이나 반복되는 요청(예: "휴가 신청")은 사전 정의된 파일과 대조합니다.
- 사용자 입력이 유사도가 높다고 판단되면 즉시 해당 템플릿 폼을 렌더링합니다

2. LLM Fallback (Slow-Path)

- 템플릿에 없는 복잡한 요청(Long-tail)이나 문맥 파악이 필요한 경우에만 LLM 파이프라인으로 진입합니다.

```
public List<Double> getEmbedding(String text, String embeddingModel) {  
    String apiUrl = ollamaConfig.getBaseUrl() + "/api/embeddings";  
    String requestJson = String.format("{\"model\": \"%s\", \"prompt\": \"%s\"}", embeddingModel, text.replace(target, replacement));  
  
    try {  
        HttpHeaders headers = new HttpHeaders();  
        headers.setContentType(MediaType.APPLICATION_JSON);  
        HttpEntity<String> entity = new HttpEntity<>(requestJson, headers);  
  
        ResponseEntity<String> response = restTemplate.postForEntity(apiUrl, entity, String.class);  
    } catch (Exception e) {  
        // Handle exception  
    }  
}
```

공지사항 UML



일, 회원, 티켓, 관리자, 채팅, 티켓 핀)

TaskFlow

Welcome 관리자님 Logout

BOARD

게시판 목록

전체 공지사항 가이드 FAQ

제목을 입력하세요...

검색

카테고리	제목	작성자	작성일
공지사항	사내 행사 및 이벤트 안내	오재용	2026-01-08
공지사항	재택근무 규정 변경 사항 공지	남주민	2026-01-08
공지사항	연차 및 휴가 사용 정책 안내	백기진	2026-01-08
공지사항	개인정보 보호 정책 개정 안내	조류아	2026-01-08
공지사항	새로운 협업 도구 도입 안내	박아성	2026-01-08
공지사항	연말연시 업무 일정 변경 안내	강충보	2026-01-08
공지사항	월례 경매회의 일정 안내 (12월)	신유지	2026-01-08

공지사항 List

TaskFlow

Welcome 관리자님 Logout

BOARD

게시글 수정

게시글 수정

카테고리: 공지사항 작성자: 오재용

제목: 사내 행사 및 이벤트 안내

내용:

사내 행사 및 이벤트 안내입니다.

- 12월 생일자 축하행사: 매일 마지막 금요일

- 사내 체육대회: 2024년 봄 예정

- 회식 및 네트워킹: 분기별 1회

- 자격증 취득 지원금 지급 안내

많은 참여 부탁드립니다.

삭제하기 취소 저장하기

공지사항 Update,Delete

TaskFlow

Welcome 관리자님 Logout

BOARD

게시판 목록

전체 공지사항 가이드 FAQ

제목을 입력하세요...

검색

게시판 목록

FAQ 1

사내 시스템 비밀번호를 잊어버렸습니다

최영업 2026-01-08

FAQ 1

재택근무 중에도 출근 기록을 해야 하나요?

오재용 2026-01-08

+ (Red circle with a plus sign)

모바일 버전

TaskFlow

대시보드 채팅 업무 현황 파일함 공지사항 관리자

Welcome 관리자님 Logout

BOARD

게시판 목록

전체 공지사항 가이드 FAQ

제목을 입력하세요...

검색

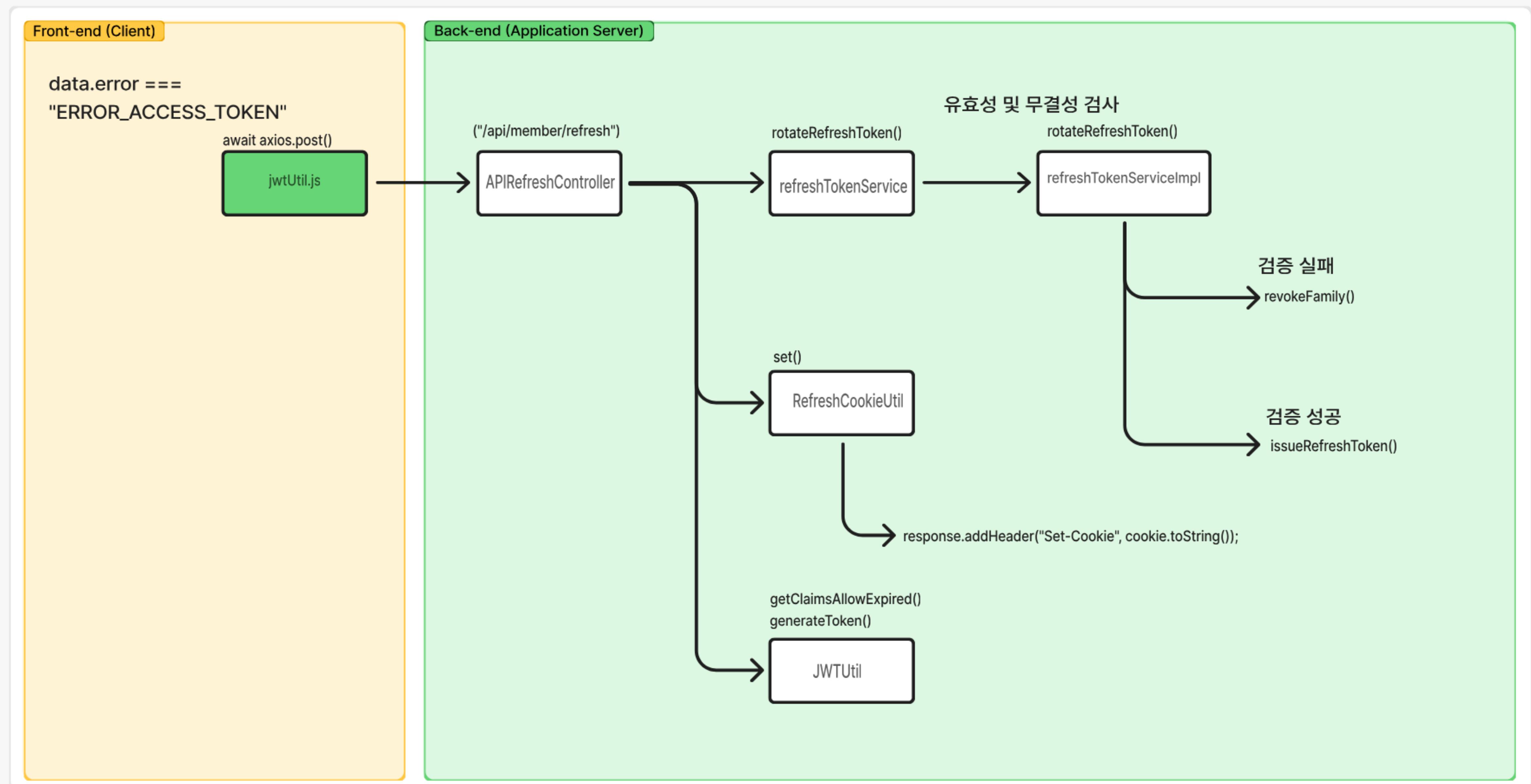
필터처리

1 2 3 4 5 6 7 8 9 10

페이지 처리

```
public Long register(BoardDTO boardDTO) {  
    log.info(" --- 게시글 등록 ---");  
    Board board = dtoToEntity(boardDTO);  
    Board savedBoard = boardRepository.save(board);  
    return savedBoard.getBno();  
}
```

보안 UML



1. 고도화된 JWT 인증 전략 (Advanced JWT Strategy)

- Access/Refresh 토큰 분리

보안성과 편의성을 모두 잡기 위해 유효기간이 짧은 Access Token과 긴 Refresh Token을 분리하여 운영합니다. 모든 API 요청은 Access Token으로 인증하며, 만료 시 Refresh Token으로 재발급받습니다.

- RTR (Refresh Token Rotation)

Refresh Token을 재사용하지 않습니다. 토큰 재발급 요청이 들어오면 기존 토큰은 폐기하고 무조건 새로운 Refresh Token을 발급합니다. 이를 통해 탈취된 토큰의 재사용을 원천 차단합니다.

2. Redis 기반 보안 제어 (Redis Security Control)

- 화이트리스트(Whitelist) 기반 관리

유효한 Refresh Token을 Redis에 화이트리스트로 저장하여 관리합니다. 서버는 요청이 올 때마다 Redis를 확인 하므로, 필요시 특정 토큰을 삭제하여 강제 로그아웃 시킬 수 있는 제어권을 가집니다

- 로그인 실패 잠금 (Account Lock):

Redis의 카운터 기능을 활용해 로그인 실패 횟수를 기록합니다. 5회 이상 실패 시 계정을 30분간 자동으로 잠금 처리하여 무차별 대입 공격(Brute-force)을 방어합니다

```
String newRefreshToken = refreshTokenService.rotateRefreshToken(refreshToken, request, expectedEmail);
RefreshCookieUtil.set(request, response, newRefreshToken, maxAgeSeconds);
```

```
// 토큰 저장 (TTL 포함) - 한 번의 호출로 저장과 TTL 설정
redisTemplate.opsForValue().set(tokenKey, tokenData, refreshTtl);
```

```
// Family Set에 추가 및 TTL 설정 - 최적화: TTL이 필요할 때만 설정
Long addedCount = redisTemplate.opsForSet().add(familyKey, jti);
```

Redis / DB 저장 성능 비교

```
xec-1] c.desk.security.filter.JWTCheckFilter : check uri...../api/tickets/pins/items
xec-1] c.desk.security.filter.JWTCheckFilter : -----JWTCheckFilter-----
xec-1] c.desk.security.filter.JWTCheckFilter : JWT Check Error.....
xec-1] c.desk.security.filter.JWTCheckFilter : Expired
xec-5] c.desk.security.filter.JWTCheckFilter : check uri...../api/member/refresh
xec-3] c.desk.security.filter.JWTCheckFilter : check uri...../api/files/list
xec-3] c.desk.security.filter.JWTCheckFilter : -----JWTCheckFilter-----
xec-3] c.desk.security.filter.JWTCheckFilter : JWT Check Error.....
xec-3] c.desk.security.filter.JWTCheckFilter : Expired
xec-5] c.d.s.token.RefreshTokenRedisService : [Redis Performance] get tokenKey: 22 ms
xec-5] c.desk.controller.APIRefreshController : === Refresh Token Rotation Performance ===
xec-5] c.desk.controller.APIRefreshController : Storage: redis | Rotation: 34 ms | Total: 34 ms
xec-6] c.desk.security.filter.JWTCheckFilter : check uri...../api/tickets/pins/items
xec-6] c.desk.security.filter.JWTCheckFilter : -----JWTCheckFilter-----
xec-6] c.desk.security.filter.JWTCheckFilter : -----
xec-7] c.desk.security.filter.JWTCheckFilter : check uri...../api/member/refresh
xec-6] c.desk.security.filter.JWTCheckFilter : MemberDTO(email=user9@desk.com, pw=PROTECTED, nickname=관리자9, social=false, department=[ROLE_ADMIN, ROLE_USER])
xec-6] c.desk.security.filter.JWTCheckFilter : -----JWTCheckFilter-----
```

Redis 성능 테스트 결과

```
xec-10] c.desk.controller.APIRefreshController : === Refresh Token Rotation Performance ===
xec-10] c.desk.controller.APIRefreshController : Storage: db | Rotation: 119 ms | Total: 119 ms
xec-3] c.desk.security.filter.JWTCheckFilter : check uri...../api/tickets/pins/items
xec-3] c.desk.security.filter.JWTCheckFilter : -----JWTCheckFilter-----
xec-6] c.desk.security.filter.JWTCheckFilter : check uri...../api/files/list
xec-6] c.desk.security.filter.JWTCheckFilter : -----JWTCheckFilter-----
xec-3] c.desk.security.filter.JWTCheckFilter : -----
xec-3] c.desk.security.filter.JWTCheckFilter : MemberDTO(email=user9@desk.com, pw=PROTECTED, nickname=관리자9, social=false, department=[ROLE_ADMIN, ROLE_USER])
xec-3] c.desk.security.filter.JWTCheckFilter : -----
xec-6] c.desk.security.filter.JWTCheckFilter : MemberDTO(email=user9@desk.com, pw=PROTECTED, nickname=관리자9, social=false, department=[ROLE_ADMIN, ROLE_USER])
xec-6] c.desk.security.filter.JWTCheckFilter : -----JWTCheckFilter-----
```

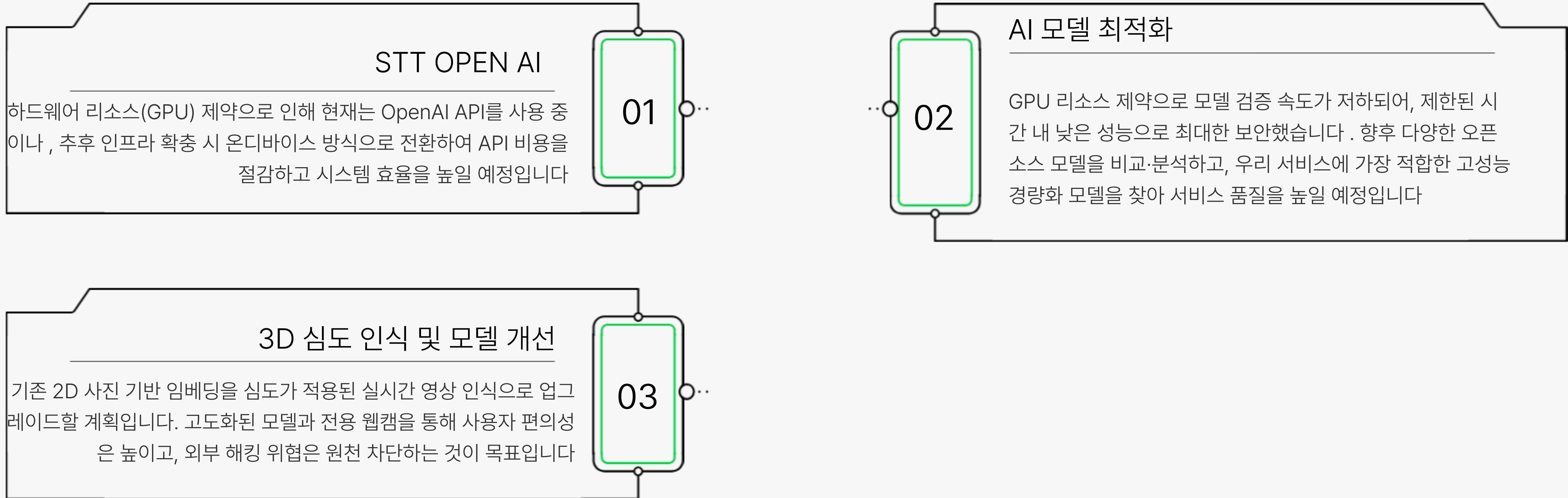
DB 성능 테스트 결과

성능 분석 및 이유

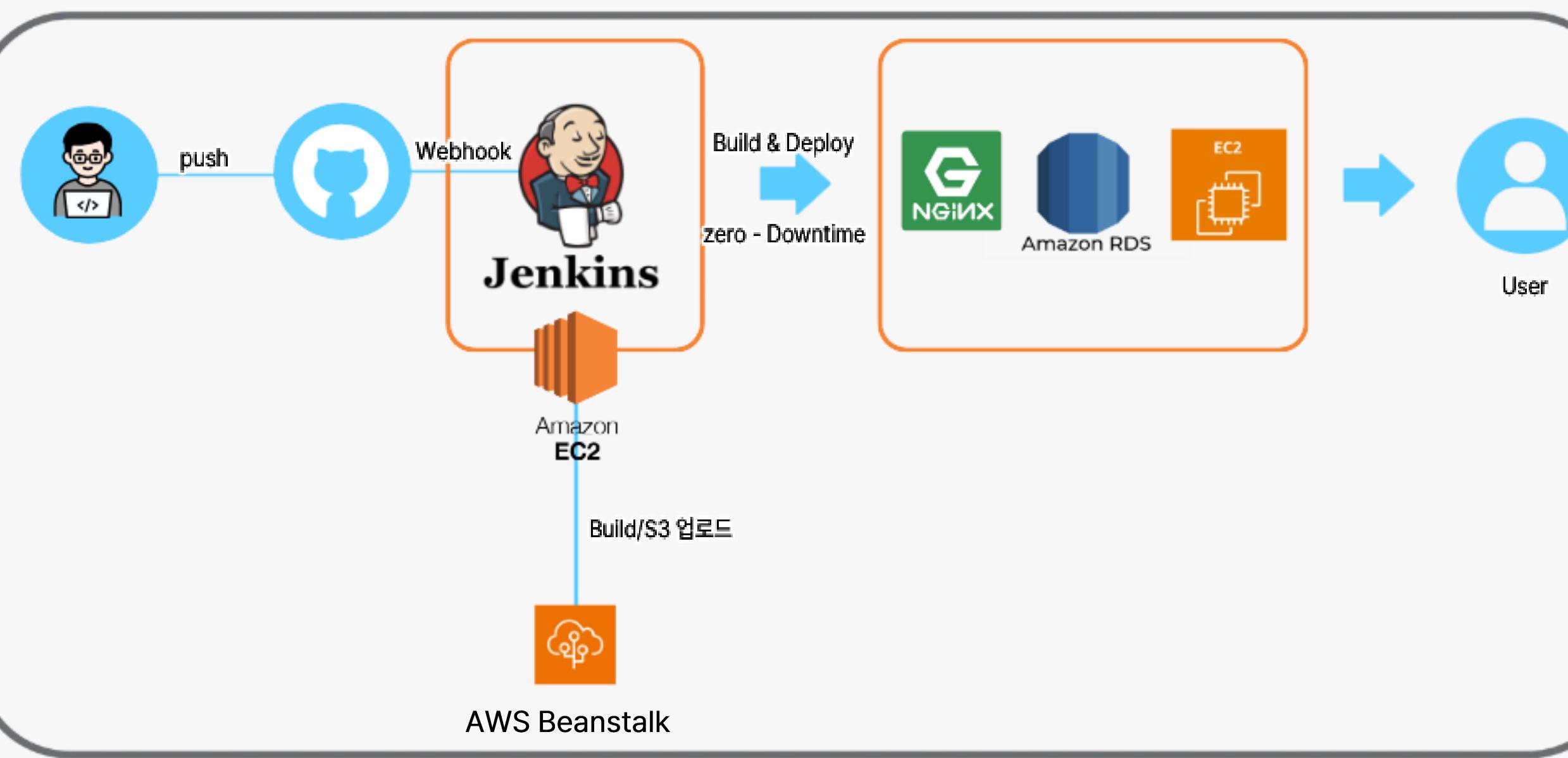
저장소	평균 처리 시간	특징
Redis	34 ms	메모리 기반 → 디스크 I/O 없음 → 빠른 접근, TTL 자동 관리
DB	119 ms	디스크 I/O 발생 → 느린 읽기/쓰기, TTL 불가, 배치 필요

- Redis가 약 3.5배 빠름
- DB는 디스크 I/O 때문에 대량 처리 시 지연 발생

AI 관련 개선 사항



AWS 배포

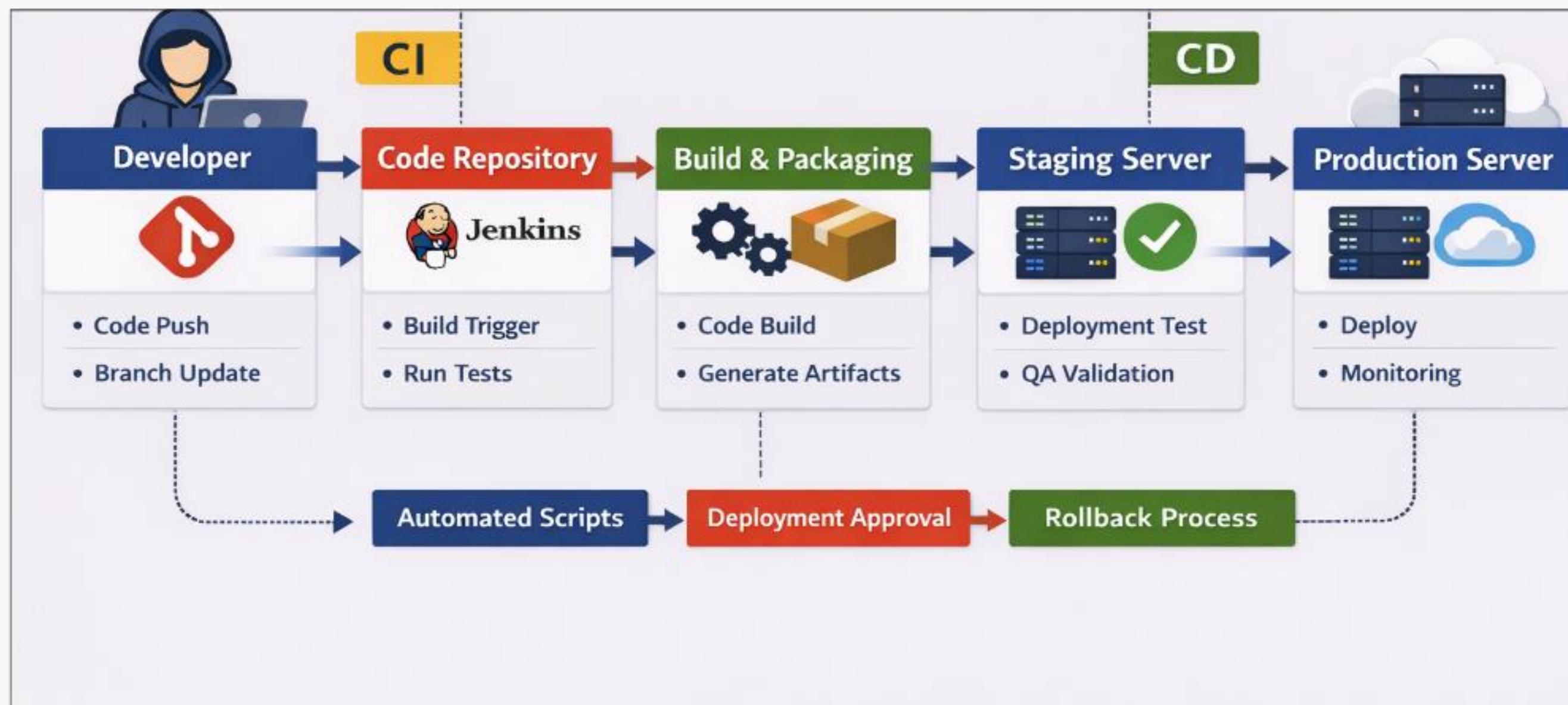


AWS 배포 흐름도

Two screenshots of the AWS Elastic Beanstalk console are shown. The top screenshot displays the 'APIServer1-env' environment. It shows the environment is 'OK' with an 'Elastic Beanstalk' icon. The logs tab indicates a successful deployment update. The bottom screenshot shows the 'Mall-env' environment, which has 'No Data - 상태 문제 1개'. Both environments include sections for '환경 개요', '도메인', '애플리케이션 이름', and '플랫폼' information.

배포 완료 화면

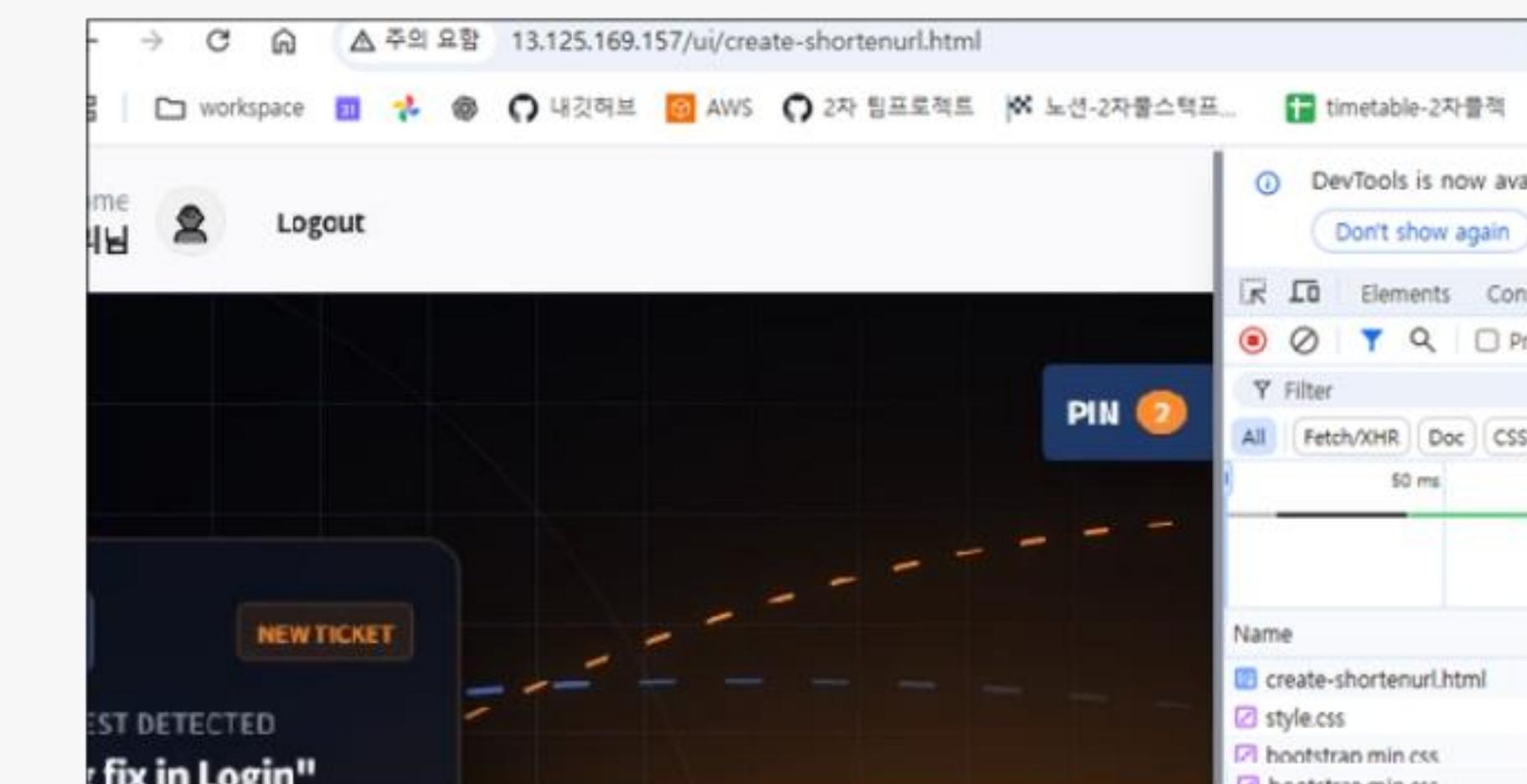
jenkins CI / CD 자동화 배포



jenkins CI/CD 흐름도

```
Script ?  
22 stage('Deploy to All Servers') {  
23   steps {  
24     script {  
25       def jarFile = 'build/libs/shortenurlservice-0.0.1-  
26       def serverIps = [  
27         '52.79.95.30', application-1  
28         '13.209.43.104', application-2  
29         '3.34.181.165' application-3 ip주소  
30     ]  
31     def deployDir = '/root'  
32     // 서버에 전달할 deploy.sh 스크립트 내용  
33     def deployScript = """  
34       #!/bin/bash  
35       cd $deployDir  
36       rm -rf shortenurlservice  
37       mvn clean package  
38       cp target/shortenurlservice-0.0.1.jar shortenurlservice  
39       nohup java -jar shortenurlservice-0.0.1.jar > log.txt &  
40     """  
41   }  
42 }
```

서버 증설 및 무중단 배포 설정



서버 과부화 실험 - 웹페이지 무중단으로 정상작동

프로젝트 리뷰

구분	내용
어려움 극복	<p>Refresh Token 쿠키 설정 문제: 로컬 개발 환경과 프로덕션 환경에서 다른 쿠키 설정이 필요한 상황에서, <code>request.isSecure()</code>를 활용한 환경별 동적 쿠키 설정으로 문제를 해결했습니다. HTTP 환경에서는 <code>Secure=false, SameSite=Lax</code>를, HTTPS 환경에서는 <code>Secure=true, SameSite=None</code>을 적용하여 브라우저 호환성 문제를 해결했습니다.</p> <p>Access Token 재발급 Race Condition: 동시 다중 요청 시 Refresh Token 재발급 과정에서 Race Condition이 발생하여 <code>REFRESH_REPLAY_DETECTED</code> 에러가 발생했습니다. Subscriber 패턴과 요청 대기열(Queue)을 적용하여 첫 번째 요청만 Refresh API를 호출하고, 나머지 요청은 대기 후 새 토큰을 공유하도록 구현하여 네트워크 효율성을 높이고 에러를 해결했습니다.</p>
잘한 점	<p>보안 및 인증 시스템의 고도화: JWT 기반 Refresh Token Rotation(RTR) 정책을 구현하여 보안성을 강화하고, Redis를 활용한 성능 최적화를 통해 12,000건 기준 평균 34ms의 빠른 처리 속도를 달성, 얼굴 인식 로그인을 임베딩 벡터 기반으로 구현하여 보안과 편의성을 동시에 고려했습니다.</p> <p>데이터베이스 분리 설계: 일반 트랜잭션 데이터(MariaDB)와 AI 벡터 데이터(PostgreSQL)를 물리적으로 분리하여 각 DB의 특성에 맞는 최적화를 실현했습니다. PostgreSQL의 pgvector 확장 기능을 활용한 벡터 유사도 검색으로 얼굴 인식 성능을 향상시켰습니다.</p> <p>실시간 통신 및 AI 통합: WebSocket/STOMP 기반 실시간 채팅 시스템을 구축하고, Ollama AI를 연동하여 실시간 메시지 필터링 및 부적절한 대화 자동 정제 기능을 구현, RAG 기반 AI 티켓 코파일럿을 통해 자연어 대화를 분석하여 업무 요청서를 자동으로 생성하는 기능을 제공했습니다.</p> <p>성능 최적화: 채팅 무한 스크롤 최적화를 통해 10,000건 기준 DOM 개수를 약 99% 감소시키고, Scripting Time을 약 96% 단축시켜 대용량 데이터 처리 성능을 크게 개선했습니다. Redux Toolkit을 활용한 전역 상태 관리로 사용자별 중요업무(Pin) 상태를 효율적으로 동기화했습니다.</p> <p>사용자 경험 개선: OpenAI Whisper 기반 음성 텍스트 변환(STT) 및 AI 요약 기능을 통한 회의록 PDF 자동 생성 시스템을 구현하여 업무 효율성을 향상시켰습니다. Access Token 자동 재발급을 위한 Subscriber 패턴 적용으로 사용자 경험 중단 없이 인증을 유지했습니다.</p>
아쉬운 점	<p>초반 설계 및 모델링의 중요성: 프로젝트 진행 과정에서 초기 설계 단계에서 엔티티 간 관계와 API 설계를 더 상세하게 검토했어야 한다는 점이 아쉬웠습니다. 특히 Ticket과 TicketPersonal, TicketFile 간의 양방향 연관관계 설정과 Cascade 설정을 초반에 명확히 정립했더라면 중간 수정 작업을 줄일 수 있었을 것입니다.</p> <p>파일 관리 시스템의 구조적 한계: 파일 업로드 시 UUID 기반 파일명 저장 방식을 사용했지만, 파일 관리 디렉토리 구조를 초기에 체계적으로 설계하지 못해 파일 탐색 및 관리에 시간이 소요되었습니다. 향후 프로젝트에서는 파일 저장 경로를 날짜별, 사용자별로 구조화하여 관리하는 방안을 고려해야 할 것입니다.</p> <p>AI 모델 정확도 개선 여지: Ollama Qwen 모델을 활용한 AI 티켓 코파일럿의 부서 라우팅 및 담당자 배정 정확도가 때때로 낮았습니다. RAG 시스템의 지식베이스 구축과 프롬프트 엔지니어링을 더 체계적으로 진행했다면 정확도를 향상시킬 수 있었을 것입니다.</p>

감사합니다.