

# 김희주 (HuiJu, Kim)

문제를 구조화하며, 답을 찾는 개발자



## CONTACT

- 📞 Tel: 010-9340-0582  
✉️ E-Mail: ajtwlsgmlwn@naver.com  
💻 GitHub: <https://github.com/huizoo>

### 개발 프로젝트를 진행할 때 무엇을 가장 중요하게 생각하나요?

저는 완성도 높은 개발을 최우선으로 생각합니다. 기능을 빠르게 만드는 것보다, 사용자의 흐름과 경험까지 고려한 신뢰감 있는 결과물을 만드는 데 집중합니다. 단순히 동작하는 코드보다, 쓰는 사람을 위한 제품을 만드는 것이 목표입니다.

### 문제를 해결할 때 어떤 접근 방식을 선호하나요?

문제를 보면 먼저 흐름과 구조부터 파악합니다. 예외가 어디서 생길 수 있는지, 나중에 유지보수가 쉬울지를 먼저 고민하고, 그 위에 코드를 쓰습니다. 이 방식은 속도는 느릴 수 있지만, 결과적으로 팀 전체의 속도를 높이는 기반이 된다고 생각합니다.

### 팀과 협업할 때 특히 강조하는 가치는 무엇인가요?

저는 소통과 공유를 가장 중요하게 생각합니다. 구현보다 먼저 맥락을 맞추고, 함께 합의한 기준을 지키는 것을 중시합니다. 또 사용자 경험에 대한 고민을 팀과 나누며, 모두가 같은 방향으로 나아갈 수 있도록 기여하려 합니다.

## SKILLS



Python



AI



- AI 서버 개발에 주력 언어로 사용하며, 2번의 프로젝트에서 전체 서버 구조를 직접 설계하고 구현했습니다.
- 비동기 처리, 작업 큐, 병렬 처리 등 Python의 기능을 상황에 맞게 적용할 수 있으며, 필요에 따라 라이브러리 내부 동작을 이해하며 활용합니다.
- 공식 문서 및 다양한 오픈소스 구현을 참고해 새로운 라이브러리를 빠르게 습득하고 응용할 수 있습니다.

- 자연어 질문 처리와 개인화 응답 생성을 위한 RAG 파이프라인을 단독 설계하고 구현할 수 있습니다.
- 날짜 파싱, 필터링 전략 등 사용자 질의 전처리를 통해 검색 정확도를 높이는 파이프라인 튜닝 경험이 있습니다.
- KoBART 등 공개된 AI 모델의 구조와 입력 형식을 이해하고, 서비스 목적에 맞게 조정·조합하여 실제 환경에 적용할 수 있습니다.



JS  
TS  
React



- Zustand 기반 상태 분기 설계로 앱 재진입 시 온보딩 단계 자동 복원 로직을 구현해 사용자 이탈률을 낮춘 경험이 있습니다.
- React Query 기반의 캐싱 및 병합 전략으로 불필요한 API 호출을 줄이고 응답 속도를 향상시켰습니다.
- 컴포넌트 구조는 FSD 패턴을 참고해 모듈화하고, 요구사항에 따라 커스텀훅과 조건부 렌더링을 적극 활용해 UI 응답성과 재사용성을 높였습니다.

- MongoDB 해시 색딩을 통해 사용자 ID 기반 로그를 분산 저장하고, 사용자 수 증가에도 쿼리 병목을 최소화한 구조를 설계했습니다.
- 도메인 모델링 단계에서 ERD 설계에 기여하며, 정규화/비정규화 기준과 관계 설정에 대해 팀과 토의한 경험이 있습니다.
- RDBMS, NoSQL에 대한 이해를 바탕으로 프로젝트에 알맞는 스킬을 선택할 수 있습니다.



Git



- Git-flow에 익숙하며, 브랜치 전략 및 커밋 컨벤션 등 협업에서의 규칙을 준수합니다.
- 협업 환경에서 충돌을 효과적으로 관리하고, Git 명령어를 활용해 병합 이슈를 빠르게 해결한 경험이 있습니다.

## Education

### 삼성 청년 SW·AI 아카데미 13기 (SSAFY)

- 2025.01 - 2025.12
- 1학기 성적우수상 수상
- 1학기 프로젝트 최우수상
- 2학기 프로젝트 우수상 3회

### 전남대학교

- 2016.03 - 2024.02
- 전기공학과 학사

## Awards

|         |                                       |
|---------|---------------------------------------|
| 2025.06 | 삼성 청년 SW·AI 아카데미 성적우수상(3등/23명)        |
| 2025.06 | 삼성 청년 SW·AI 아카데미 관통 프로젝트 최우수상(1등/11팀) |
| 2025.08 | 삼성 청년 SW·AI 아카데미 공통 프로젝트 우수상(2등/9팀)   |
| 2025.10 | 삼성 청년 SW·AI 아카데미 특화 프로젝트 우수상(2등/8팀)   |
| 2025.11 | 삼성 청년 SW·AI 아카데미 자율 프로젝트 우수상(2등/6팀)   |

## Certificates

## Experience

|                   |                     |
|-------------------|---------------------|
| 2021.09 - 2023.09 | 카페 바리스타 (Part-time) |
|-------------------|---------------------|

## Project Summary

|  |   |
|--|---|
| <b>ToGather</b> 2025.10.10 - 2025.11.20<br>AI, Frontend Lead | 커플이 일상을 기록하고 공유하는 모바일 라이프로그 서비스 (3p) <ul style="list-style-type: none"><li>• 개인화된 AI 챗봇</li><li>• 자연어 날짜 정규화 기반 Advanced RAG(Time-aware)</li><li>• Qdrant 단일 컬렉션 + 메타데이터 필터 구조</li><li>• 온보딩 상태 기반 전역 상태 설계 및 앱 재진입 시 단계 복원 로직 구현</li></ul> |
| <b>Picky</b> 2025.08.25 - 2025.10.01<br>AI/Data Engineer, 팀장 | 브라우징 로그 기반 개인화 컨텐츠 추천 마이크로 러닝 서비스 (7p) <ul style="list-style-type: none"><li>• 크롬 확장프로그램 기반 사용자 로그 수집</li><li>• O(1) 벡터 업데이트 기반 개인화 프로필 관리</li><li>• 크롤링·KoBART 뉴스 요약 파이프라인</li><li>• MongoDB 샤딩 기반 대용량 데이터 설계</li></ul>                    |
| <b>Ottereview</b> 2025.07.14 - 2025.08.18<br>Frontend Lead   | 주니어 개발자의 PR·리뷰 진입 장벽을 낮춘 실시간 협업 서비스 (10p) <ul style="list-style-type: none"><li>• GitHub 기반 PR 생성·리뷰 플로우 UI 설계</li><li>• Zustand 기반 복합 리뷰 상태 관리</li><li>• 코드 Diff 멀티라인 선택·스레드형 댓글 구조</li><li>• SSE 기반 push 이벤트 알림 처리</li></ul>              |



## ToGather

2024.10.14 - 2024.11.19 (6주/6인)

커플이 일상을 기록하고 공유하며, 두 사람의 기록을 기반으로 개인화된 경험을 제공하는 모바일 라이프로그 서비스



## 역할

AI(60%), Frontend(40%)

## 특이사항

삼성 청년 SW 아카데미 프로젝트 우수상 (2등/6팀)

## 팀원 구성

Frontend 2 | Backend 3 | Infra 1 (AI: FE 1, BE 1 참여)

## 개발환경

Frontend React Native (Expo), TypeScript, Zustand, TanStack Query

Backend Java 17, Spring Boot, JPA, Spring Security

Database MySQL, Redis

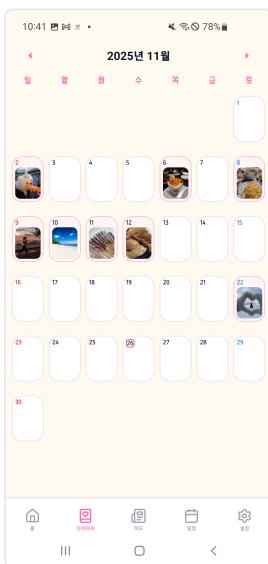
AI Python 3.11, FastAPI, Google Gemini API, OpenAI API, Qdrant, Pillow

Infra Docker, GitLab CI/CD, Jenkins, Nginx, Ubuntu, EC2, S3

## Preview



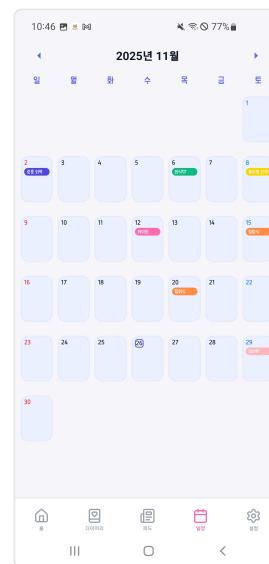
홈 화면



다이어리 화면



피드 화면



일정 화면



설정 화면

## 담당 업무

### 1. AI 챗봇 (개인화 RAG 파이프라인)

커플의 기록을 기반으로 개인화된 답변을 제공하는 AI 챗봇 시스템 전체 설계 및 구현

- Qdrant 단일 컬렉션 + type(profile/diary/plan) 분리 저장 구조 설계
- 3단계 분리 검색: profile 직접 조회 → diary 벡터 검색 → plan 벡터 검색
- 프롬프트 구성에 Context Sandwich 기법을 적용하고, 프로필은 고정하며 검색 결과는 diary와 plan 태그로 구분

### 2. Date Parser (Query Transformation)

자연어 날짜 표현을 정규화하여 날짜 기반 질문에 대한 답변 정확도 향상

- 정규 표현식과 규칙 기반으로 12가지 날짜 패턴(예: '저번 주', '작년' 등)을 파싱
- Unix timestamp 범위 변환 → Qdrant Range 필터 적용
- 날짜 기반 질문의 정확도 99% 달성

### 3. 온보딩 상태 복원 로직

앱 재진입 시 커플 상태, 프로필, 초대 코드에 따라 올바른 단계로 복원

- AsyncStorage 기반 상태 복원 및 분기 로직 설계
- coupleStatus, hasProfile, invitationCode 조합에 따라 화면을 분기해, 앱 재진입 시 적절한 화면으로 이동
- 온보딩 Context로 전역 상태 관리

### 4. SSE 실시간 커플 연동

Server-Sent Events를 활용한 커플 매칭 실시간 동기화

- 피초대자가 코드를 입력하면 초대자의 화면이 자동으로 홈으로 전환되도록 실시간 동기화
- 펫 선택 화면에서도 한 명이 선택 시 상대 화면에 실시간 반영되도록 구현
- 실시간 동기화로 연동 대기 시간을 최소화하고, 온보딩 흐름을 간소화해 이탈 가능성 감소

## Time-aware RAG 파이프라인(Advanced RAG)

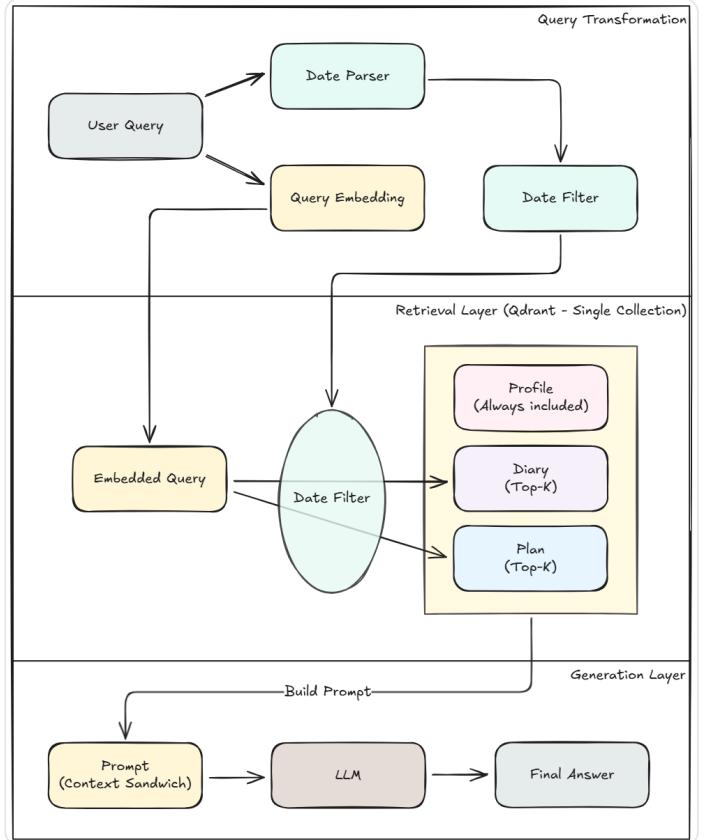
### 도입 배경

챗봇 초기에는 날짜 중심 질문에서 정확도가 낮았습니다.  
텍스트 기반 질문은 잘 매칭되었지만,  
날짜 기반 질문에서는 날짜를 의미로 인식하지 못해  
단순 문자열 유사도로 다른 날짜의 기록을 반환하거나,  
'저번 주' 같은 상대 날짜 표현을 처리하지 못했습니다.

해당 문제의 원인은 벡터 검색이  
날짜 개념을 이해하지 못하고,  
상대 날짜 표현은 정규 날짜로 변환되지 않아  
검색 자체가 불가능하기 때문입니다.

이를 해결하기 위해 자연어 날짜를 정규화하는  
Query Transformation(Date Parser)을 도입해  
모든 날짜 표현을 Unix timestamp 범위로 변환하고  
메타데이터 필터를 우선 적용하는 구조로 개선했습니다.

오른쪽 RAG 파이프라인은  
이러한 문제를 해결하기 위해 설계한 구조입니다.



### 주요 구현

- Qdrant 단일 컬렉션 + type(profile/diary/plan) 분리 저장: 프로필 검색 없이 직접 조회, diary/plan 각 5개씩 벡터 검색
- 날짜 인식 문제 해결: 자연어 날짜 파싱 → Query Transformation → 메타데이터 필터 우선 적용 RAG 파이프라인 구축

### 기술적 포인트

- 3단계 분리 검색 파이프라인:
  - ① profile(couple\_id+type 필터 직접 조회) → ② diary 벡터 검색(limit=5) → ③ plan 벡터 검색(limit=5)
- Date Parser 정규식 + 규칙 기반 12가지 패턴 파싱 후 Unix timestamp 범위로 변환 → Qdrant Range 필터 적용
- Context Sandwich 프롬프트: [응답 지침] → [안전 규칙] → [커플 프로필] → [커플 추억 기록] → [사용자 질문] 순서로 구조화
- 커플 추억 기록은 diary와 plan 검색 결과를 태그로 구분하여 LLM이 출처를 인식하고 답변 생성
- point\_id 기반 자동 업서트: couple\_{couple\_id}\_{type}\_{id} 형식으로 수정 시 벡터 자동 생성

### 성과

- 날짜 기반 질의 정확도 99% 달성 (12가지 날짜 패턴 적용)
- "저번 주에 뭐 했어?", "지난달 데이트 어디 갔지?" 같은 상대 날짜 표현도 정확하게 검색 가능
- 사용자 맞춤형 답변 생성으로 챗봇 만족도 향상 및 서비스 차별화 포인트 확보

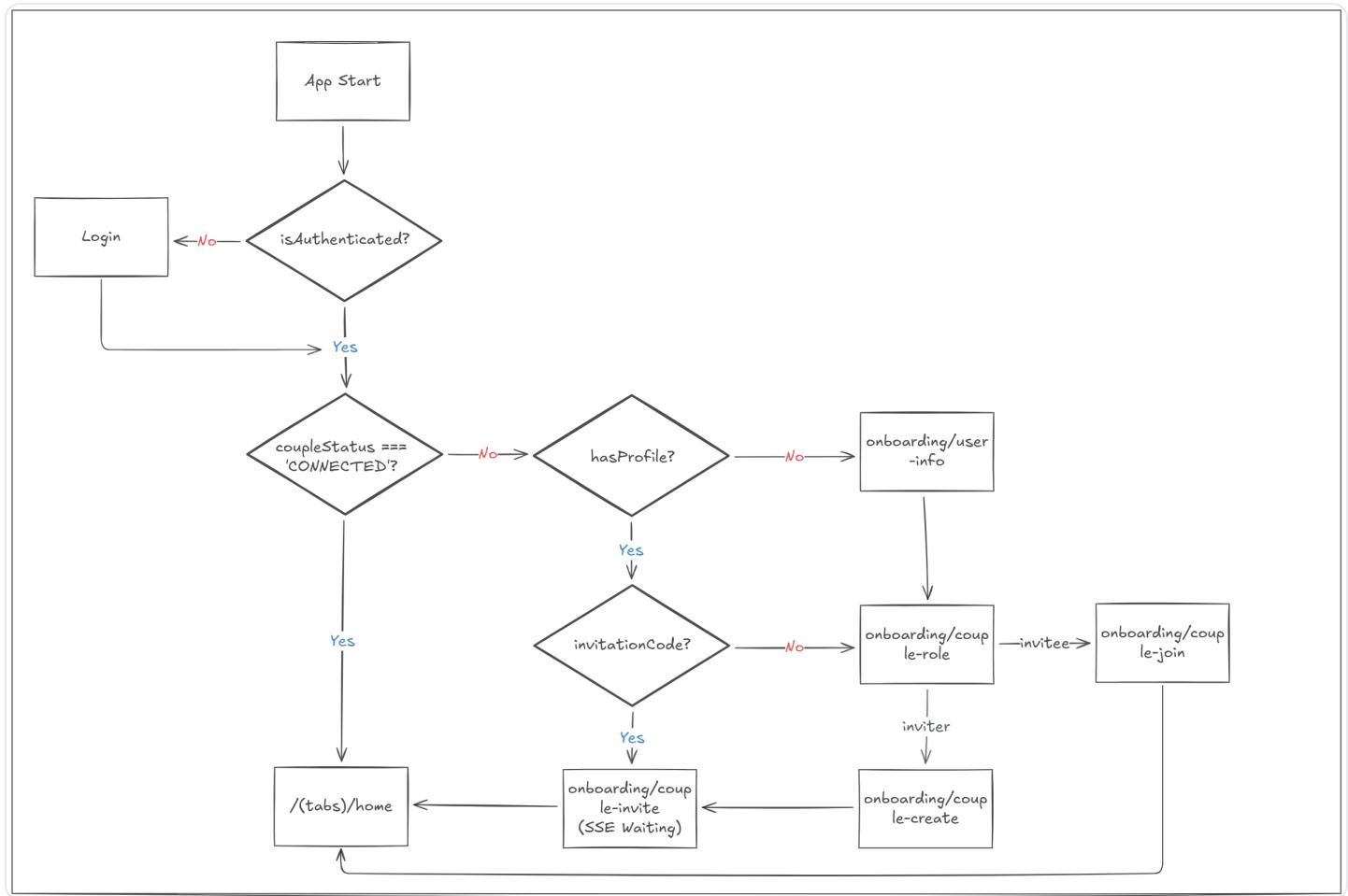
## 온보딩 상태 복원 로직

### 도입 배경

커플 상태, 프로필 정보 입력 여부, 초대 코드 존재 여부에 따라

사용자가 앱을 재진입했을 때 올바른 단계로 복원해주는 온보딩 분기 설계가 필요했습니다.

아래는 앱의 온보딩·로그인 플로우를 나타낸 플로우차트입니다.



### 주요 구현

- 온보딩 Context 기반 전역 상태 관리 및 앱 재진입 시 올바른 단계 복원 로직 구현
- SSE 기반 커플 연동 실시간 동기화: 피초대자 코드 입력 → 초대자 자동 흘 이동

### 기술적 포인트

AsyncStorage 기반 상태 복원 및 SSE 실시간 동기화

### 상태 기반 분기

- coupleStatus === 'CONNECTED' → 홈 화면
- hasProfile (nickname, birthday, mbti) → 프로필 완성 여부
- invitationCode 존재 → 초대 대기 화면

# 김희주 (HuiJu, Kim)

문제를 구조화하며, 답을 찾는 개발자



## Picky

2024.08.26 - 2024.10.04 (6주/6인)

브라우징 로그를 기반으로 사용자 관심사를 분석하고  
개인화된 뉴스·퀴즈를 추천하는 크롬 확장 기반 마이크로 러닝 서비스

쓸어지는 뉴스 속 당신만의 뉴스를 찾다

개인 맞춤형 콘텐츠 추천

크롬 확장 마이크로 러닝 서비스

**PICKY**



## 역할

AI/Data Engineer, 팀장

## 특이사항

삼성 청년 SW 아카데미 프로젝트 우수상 (2등/8팀)

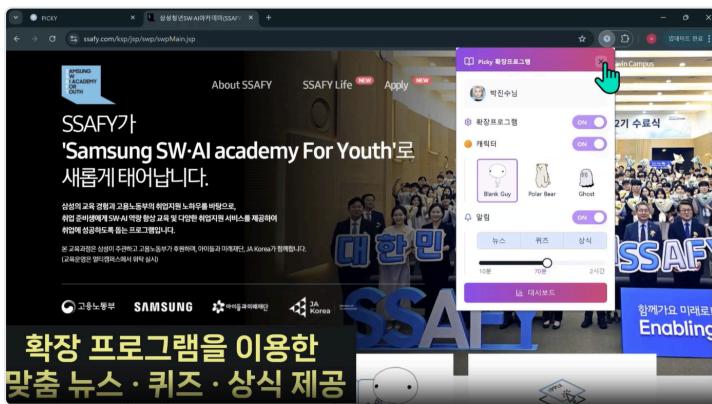
## 팀원 구성

Frontend 2 | Backend 2 | Infra 1 | AI/Data 1

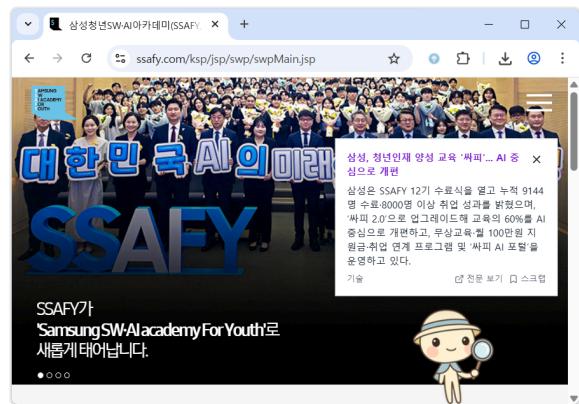
## 개발환경

|          |  |
|----------|--|
| Frontend | React, TypeScript, Zustand, TailwindCSS, Chrome Extension (MV3)          |
| Backend  | Java 17, Spring Boot, JPA, Spring Security                               |
| Database | MySQL, MongoDB, Redis  |
| AI/Data  | Python 3.11, FastAPI, OpenAI API, KoBART, Qdrant, Chrome Extension (MV3) |
| Infra    | Docker, GitLab CI/CD, Jenkins, Nginx, Ubuntu, EC2                        |

## Preview



확장 프로그램



캐릭터 & 뉴스

## 담당 업무

### 1. 크롬 확장 프로그램 로그 수집 시스템

Manifest V3 기반 크롬 확장 프로그램으로 사용자 브라우징 데이터 수집 파이프라인 설계 및 구현

- 체류 시간, 스크롤 깊이, 접속 시간대 등 브라우징 행동 메트릭을 실시간으로 수집
- 수집된 로그를 백엔드로 전송하고 MongoDB에 저장하는 데이터 파이프라인 구축
- 크롬 확장 백그라운드 서비스 워커와 콘텐츠 스크립트 간 메시지 통신 구조 설계

### 2. 초기 프로필 생성을 위한 히스토리 데이터 수집

콜드 스타트 문제 해결을 위해 브라우징 히스토리 URL의 실제 콘텐츠를 추출하는 시스템 설계 및 구현

- Offscreen Document API를 활용해 백그라운드에서 히스토리 URL을 독립적으로 로드하는 구조 설계
- Readability 라이브러리를 통합하여 페이지 본문 추출 파이프라인 구축
- Service Worker와 Offscreen Document 간 메시지 통신으로 추출된 콘텐츠 전달 및 저장
- 콘텐츠 추출 실패 시(SPA 등) 스kip 처리 로직으로 안정적인 데이터 수집 파이프라인 구축

### 3. 관심사 프로필 벡터 생성 및 MongoDB 색인

수집된 브라우징 로그를 기반으로 사용자 관심사 프로필 벡터를 생성하고, MongoDB 색인으로 확장 가능한 저장소 구축

- 초기 히스토리 로그 전체를 문장 임베딩으로 변환해 사용자 관심사 초기 벡터 생성
- 사용자 ID 기반 해시 색인으로 MongoDB 컬렉션 분산 저장, 사용자 수 증가 시에도 조회 성능 유지
- 체류 시간·스크롤 깊이 가중치 메타데이터를 활용해 O(1) 충분 업데이트로 실시간 프로필 갱신

### 4. KoBART 뉴스 요약 파이프라인

사용자가 방문한 뉴스 콘텐츠를 KoBART 모델로 요약하여 마이크로 러닝 콘텐츠로 제공하는 배치 시스템 구축

- KoBART 생성 요약 모델을 활용해 긴 뉴스 기사를 핵심 문장으로 압축하는 배치 파이프라인 구축
- 요약된 콘텐츠를 개인화된 뉴스 및 퀴즈 형태로 재구성하여 마이크로 러닝 경험 제공
- 비동기 배치 처리로 대량의 뉴스 콘텐츠 요약 작업을 효율적으로 처리

## Offscreen Document API 콘텐츠 추출 트러블슈팅

### 도입 배경

사용자의 브라우징 히스토리를 기반으로 관심사를 분석하려면,  
단순히 URL 목록뿐만 아니라 각 페이지의 실제 콘텐츠(본문 텍스트)를 추출해야 했습니다.  
chrome.history API로 히스토리 URL 목록은 얻을 수 있었지만,  
해당 URL의 페이지 콘텐츠를 가져오는 과정에서 기술적 문제가 발생했습니다.

### 문제 발생

초기 접근 방식: Content Script를 사용해 히스토리 URL에 접근 시도

- Content Script는 현재 활성화된 탭의 DOM에 접근하는 방식으로 동작
- 히스토리 URL(예: naver.com)의 콘텐츠를 가져오려면, 현재 열린 페이지(예: google.com)에서 다른 origin의 페이지에 접근해야 하는 상황 발생
- 브라우저의 Same-Origin Policy로 인해 cross-origin 요청 차단 → 콘텐츠 추출 불가
- fetch API 사용 시에도 CORS 정책으로 다른 도메인의 HTML 콘텐츠를 직접 가져올 수 없음

### 해결 과정

Offscreen Document API를 활용한 독립적인 백그라운드 페이지 로딩

- Offscreen Document: 사용자에게 보이지 않는 독립적인 문서 컨텍스트를 백그라운드에서 생성
- Service Worker가 Offscreen Document를 생성하고, 히스토리 URL을 로드하도록 지시
- Offscreen Document 내부에서 Readability 라이브러리로 로드된 페이지의 본문 추출
- 추출된 콘텐츠를 chrome.runtime.sendMessage로 Service Worker에 전달하여 저장

### 기술적 포인트

- Offscreen Document는 현재 탭과 독립적인 환경으로, cross-origin 제약 없이 URL 로드 가능
- chrome.offscreen.createDocument() API로 백그라운드 문서 생성 및 생명주기 관리
- Readability 라이브러리: Mozilla의 Reader View 알고리즘 기반으로 페이지 본문만 정확히 추출
- SPA(CSR) 한계: JavaScript 실행 전 HTML이 비어있어 콘텐츠 추출 불가 → 콘텐츠 없으면 스킵 처리
- 배치 처리: 여러 히스토리 URL을 순차적으로 로드하며 성능 최적화 및 브라우저 부하 관리

### 성과

- cross-origin 제약 없이 SPA를 제외한 대부분의 사이트에서 히스토리 URL 콘텐츠 추출 가능
- 뉴스, 블로그, 위키 등 주요 사이트에서 안정적으로 텍스트 수집
- 추천 모델의 데이터 커버리지가 대폭 확대되어 더욱 정확한 사용자 관심사 프로필 생성 가능

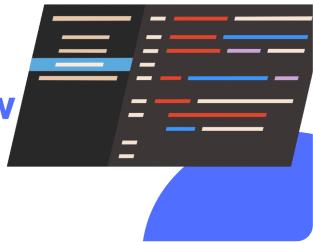


## Ottereview

2024.07.14 - 2024.08.18 (6주/6인)

주니어 개발자의 PR·리뷰 진입 장벽을 낮추기 위해  
GitHub 기반 코드 리뷰 과정을 재구성한 실시간 협업 웹 서비스

더 편한 협업  
**Ottereview**



## 역할

Frontend Lead

## 특이사항

삼성 청년 SW 아카데미 프로젝트 우수상 (2등/9팀)

## 팀원 구성

Frontend 2 | Backend 3 | Infra 1

## 개발환경

|          |  |
|----------|--|
| Frontend | React, TypeScript, Zustand, TailwindCSS, Monaco Editor, tldraw, STOMP/SockJS |
| Backend  | Java 17, Spring Boot, JPA, Spring Security, GitHub API                       |
| Database | MySQL, Redis   |
| AI       | Python, FastAPI, OpenAI API, Pinecone, WhisperAI                             |
| Infra    | Docker, GitLab CI/CD, Jenkins, Nginx, Ubuntu, EC2, S3                        |

## Preview

안녕하세요, kidreamx님!  
효율적인 코드 리뷰로 팀의 생산성을 높여보세요!

실시간 채팅방

PR 리뷰

코드 리뷰 워크스페이스

Repository

Pull Request

accessTokenGenerate 메서드 수정

[FEATURE]- Swagger 설정 및 카카오 유저 정보 수...

메인 홈 화면

PR 리뷰

private final DiaryService diaryService;

public DiaryController(DiaryService diaryService) {  
 this.diaryService = diaryService;  
}

상성자 주입의 장점을 고려해보세요. 의존성 관리와 테스트 용이성을 높일 수 있습니다.

생성자 주입을 사용하는 이유를 설명해주시면 좋을 것 같습니다. 소스 코드의 가독성과 유지보수성을 고려해볼 때, 다른 주입 방식과의 차별점을 명확히 해주시기 바랍니다.

코드 Diff 리뷰 화면

## 담당 업무

### 1. PR 생성/리뷰 플로우 UI 구현

주니어 개발자를 위한 직관적인 5단계 PR 생성 플로우 및 리뷰 인터페이스 설계·구현

- 브랜치·기존 PR 검증 → 컨벤션·제목 피드백 → 리뷰어 선택 → 코드/음성 코멘트 관리로 이어지는 5단계 PR 생성 플로우
- PR 리뷰·머지·닫기·재오픈 화면 구현 및 Monaco Editor 기반 코드 Diff 뷰어 통합
- Repository 목록/상세 화면 구현 (브랜치·상태 필터링, 잘못된 repold 접근 시 404 처리)

### 2. Zustand 전역 상태 관리 아키텍처

Zustand를 활용한 전역 상태 관리 구조 설계로 컴포넌트 간 데이터 흐름 최적화

- 사용자 인증 상태, 알림 목록, 저장소 데이터를 Zustand 스토어로 전역 관리
- 도메인별로 스토어를 분리하여 관심사 분리 및 유지보수성 향상
- Props drilling 없이 깊은 컴포넌트 트리에서도 효율적인 상태 접근 가능

### 3. 재사용 가능한 공통 컴포넌트 시스템

일관된 UI/UX를 위한 공통 컴포넌트 라이브러리 및 커스텀 훅 구축

- 헤더, 버튼, 모달, 토스트, 스텝 인디케이터 등 재사용 가능한 UI 컴포넌트 라이브러리 구축
- useCommentManager 훅으로 코멘트 작성·수정·삭제 상태를 캡슐화하여 리뷰 화면 로직 단순화
- beforeunload/popstate 이벤트 처리로 리뷰 작성 중 페이지 이탈 시 경고 및 임시 저장 기능 구현

### 4. SSE 실시간 알림 시스템

Server-Sent Events를 활용한 실시간 GitHub 이벤트 알림 시스템 구현

- useSSE 커스텀 훅으로 SSE 연결 생명주기 관리 및 이벤트 리스너 추상화
- GitHub 푸시 이벤트를 실시간으로 수신해 토스트 알림 및 알림 센터에 자동 반영
- 읽음/안읽음 상태 관리 및 알림 센터 UI 구현으로 팀원 간 코드 리뷰 반응 속도 향상