




# CBT-Diary project review report Ver-0.2

검토 기준: CBT-Diary-develop-0620-044450 소스 코드  
작성 일자: 2025년 6월 21일

## 1. 개요





본 문서는 CBT Diary 프로젝트의 전체 소스 코드를 기반으로 작성된 기술 분석 보고서입니다. 프로젝트는 Spring Boot 기반의 **Auth-server(백엔드)**, Python/FastAPI 기반의 **AI 서버**, 그리고 React-Native 기반의 **\*\*CBT-front(모바일 앱)\*\***로 구성된 MSA(Microservice Architecture) 구조를 명확히 따르고 있습니다. 전반적으로 Docker를 활용한 개발 환경 통일화, 상세한 API 설계 문서 등 좋은 기반을 갖추고 있습니다. 본 보고서는 현재 시점의 코드를 기준으로 각 서비스의 완성도를 평가하고, 안정성과 확장성, 보안 강화를 위한 구체적인 개선 방안을 제안합니다.










### 범례

-  **구현 완료**: 기능이 안정적으로 구현되어 요구사항을 충족하는 상태
-  **개선 필요**: 기능은 존재하나, 리팩토링, 안정화 또는 보안 강화가 필요한 상태
-  **신규 제안**: 프로젝트의 완성도를 높이기 위해 추천하는 신규 기능 또는 아키텍처

## 2. Auth-server (백엔드 - Spring Boot)

인증 및 핵심 비즈니스 로직을 담당하는 서버입니다. 대부분의 기능이 구현되어 있으나, 토큰 관리 정책과 예외 처리의 고도화가 필요합니다.

구분	기능	검토 내용 및 현황	개선 제안	관련 소스
사용자 관리	회원가입/로그인	 일반 회원가입, 소셜 로그인 후 프로필 정보 입력 및 이메일 인증 프로세스 구현 완료.   아이디/비밀번호 찾기 기능은 있으나, 임시 비밀번호 발급 로직의 보안 강화가 필요합니다.	 <b>(보안)</b> 임시 비밀번호 발급 후, 사용자가 다음 로그인 시 즉시 비밀번호를 변경하도록 강제하는 로직을 추가해야 합니다.   <b>(UX)</b> 회원 탈퇴 시, 데이터를 즉시 삭제하는 대신 유예 기간(e.g., 30일)을 두고 비활성화 처리하여 사용자가 복구를 원할 경우를 대비하는 로직을 고려할 수 있습니다.	UserService, AuthController, EmailService

토큰 관리	JWT 발급/재발급	 Access/Refresh 토큰 발급 및 재발급(api/token/refresh) 로직 구현 완료.   Redis를 이용해 Refresh 토큰을 저장하여 서버 재시작에도 인증을 유지할 수 있도록 설계되었습니다.	 <b>(보안)</b> Refresh 토큰 탈취 시 해당 토큰을 무효화(blacklist) 처리하는 로직이 부재합니다. 토큰 재발급 시 이전 Refresh 토큰을 비활성화하는 <b>'Refresh Token Rotation'</b> 전략 도입을 적극 검토해야 합니다.   <b>(안정성)</b> Redis 연결 실패 시 발생하는 RedisConnectionFailureException 등에 대한 예외 처리 로직을 보강해야 합니다.	TokenService, TokenController, RedisService
일기 관리	CRUD	 일기 생성, 조회, 수정, 삭제 기능 구현 완료.   QueryDSL을 사용하여 동적 쿼리 및 검색 기능을 일부 구현하였습니다. (DiaryRepository)	 <b>(성능)</b> findAll 메소드는 페이징 처리가 되어 있으나, 향후 데이터 증가 시 인덱스 전략이 중요해집니다. created_at 등 자주 조회되는 컬럼에 인덱스를 추가하는 것을 고려해야 합니다.   <b>(기능)</b> 일기 내용에 대한 자동 저장(임시 저장) 기능을 추가하여 사용자 경험을 개선할 수 있습니다.	DiaryService, DiaryController, DiaryRepository
보안	인증/인가	 SecurityConfig를 통해 URL 기반 접근 제어 설정 및 JwtVerificationFilter	 <b>(보안)</b> GlobalExceptionHandler에서 500 Internal Server Error 발생 시,	SecurityConfig, JwtVerificationFilter, GlobalExceptionHandler

		er를 통한 JWT 토큰 검증 로직 구현. ⚠️ publicAPI에 정의된 /api/users/join, /api/auth/login 등 필수 경로 외에, 일부 엔드포인트의 접근 권한을 재검토할 필요가 있습니다.	민감한 스택 트레이스 정보가 노출되지 않도록 별도의 로깅 처리 후 일반적인 에러 메시지만 사용자에게 반환해야 합니다.	
--	--	--	--	--

### 3. AI 서버 (Python - FastAPI)




CBT 분석 모델을 API 형태로 제공하는 서버입니다. 외부 LLM을 호출하는 방식으로 구현되어 있습니다.

구분	기능	검토 내용 및 현황	개선 제안	관련 소스
API	일기 분석	 /analyze 엔드포인트에서 텍스트를 입력받아 분석 결과를 JSON으로 반환하는 기능 구현 완료.   OpenAIService를 통해 외부 LLM(GPT)을 호출하여 분석을 수행합니다.	 <b>(성능/비용)</b> 자체적인 경량화 모델(e.g., KoBERT 기반 감정 분석 모델)을 파인튜닝하여 서빙하거나, 비용 효율적인 다른 LLM API(e.g., Claude, Gemini) 사용을 검토하여 비용 및 속도를 최적화할 수 있습니다.   <b>(기능)</b> 단순 분석 결과 반환 외에, 사용자의 이전 일기 내용을 기반으로 한 대화형 피드백 기능(chatbot.py의 고도화)을 추가할 수 있습니다.	ai/ai_api/api_server.py
모델/프롬프트	CBT 분석	 prompts.py에 역할(Role), 지시사항(Instructions), 예시(Examples) 등을 포함한	 <b>(고도화)</b> 현재 프롬프트는 일반적인 분석에 초점이 맞춰져 있습니다. <b>인지적 왜곡 발견, 대안적</b>	ai/ai_api/prompts.py, ai/ai_api/openai_service.py



		체계적인 프롬프트가 정의되어 있습니다.	<b>사고 제안</b> 등 CBT의 핵심 요소를 더 명확하게 지시하는 정교한 프롬프트 엔지니어링이 필요합니다.  <b>(기능)</b> 사용자의 피드백(e.g., "이 분석은 도움이 되었어요")을 받아 프롬프트를 동적으로 개선하는 RAG(Retrieval-Augmented Generation) 개념을 도입해볼 수 있습니다.	
--	--	-----------------------	---	--

## 4. Frontend (React-Native)

실제 소스 코드(CBT-front 디렉토리)를 기반으로 분석 및 재작성된 내용입니다.

구분	기능	검토 내용 및 현황	개선 제안	관련 소스
API 연동	API 호출 방식	 <b>(개선 필요)</b> AuthContext.tsx, WriteScreen.tsx 등 여러 파일에서 fetch API를 직접 사용하고 있습니다. 이 방식은 인증 토큰 추가, 에러 처리 등 공통 로직이 중복되고, API 명세 변경 시 유지보수가 어렵습니다.	 <b>(신규 제안)</b> axios 라이브러리를 도입하여 <b>API 클라이언트를 중앙에서 관리하는 모듈</b> (src/api/client.ts)을 생성해야 합니다. <b>요청/응답 인터셉터</b> 를 활용하여 Access Token 자동 추가 및 401 에러 시 <b>자동 토큰 재발급 로직</b> 을 구현하면 코드 중복을 제거하고 안정성을 크게 향상시킬 수 있습니다.	AuthContext.tsx, WriteScreen.tsx
상태 관리	전역 상태	 <b>(구현 완료)</b> AuthContext를	 <b>(신규 제안)</b> 서버 상태 관리를 위해	AuthContext.tsx, MainScreen.tsx

		<p>통해 사용자 인증 정보(토큰, 유저 정보)와 관련 함수(signIn, signOut)를 전역으로 관리하는 구조는 좋습니다.&lt;br&gt;⚠️ (개선 필요) 일기 목록, 분석 결과 등 서버로부터 받아오는 데이터(Server State)를 일반 useState, useEffect로 관리하고 있습니다. 이는 로딩, 에러 상태, 캐싱 처리를 매번 수동으로 구현해야 해 보일러플레이트 코드가 많아집니다.</p>	<p><b>React-Query (TanStack Query) 또는 SWR 도입을 강력히 권장합니다.</b> 캐싱, 재요청, 로딩/에러 상태 분기 등을 자동화하여 코드 복잡도를 크게 낮추고 사용자 경험을 향상시킬 수 있습니다.</p>	
컴포넌트	구조 및 재사용	<p>⚠️ (개선 필요) screens 디렉토리 중심으로 파일이 구성되어 있으며, 공통으로 재사용될 수 있는 UI 컴포넌트(e.g., 버튼, 입력창, 카드)들이 분리되어 있지 않습니다.</p>	<p>🚀 (신규 제안) 재사용 가능한 UI 컴포넌트를 모아두는 src/components 디렉토리를 생성하고, Atomic Design(Atoms, Molecules 등) 원칙에 따라 구조화하는 것을 권장합니다.&lt;br&gt;🚀 (신규 제안) Storybook을 도입하여 UI 컴포넌트를 독립적으로 개발하고 시각적으로 테스트할 수 있는</p>	src/screens/**/*.ts x

			환경을 구축하면 개발 효율성이 크게 향상됩니다.	
테스트	코드 안정성	 <b>(개선 필요)</b> _tests_/App.test.tsx 파일 외에 유의미한 테스트 코드가 전무합니다. 이는 기능 추가 및 리팩토링 시 코드의 안정성을 보장하기 어렵게 만듭니다.	 <b>(신규 제안)</b> <b>Jest와 React Native Testing Library</b> 를 사용하여 주요 컴포넌트(인증 폼, 일기 작성 폼 등)의 렌더링 및 사용자 인터랙션에 대한 단위/통합 테스트 코드 작성이 시급합니다.	tests_/App.test.tsx

## 5. 공통 및 아키텍처

구분	기능	검토 내용 및 현황	개선 제안	관련 소스
컨테이너화	Docker	 docker-compose.yml을 통해 전체 서비스를 한 번에 실행할 수 있도록 구성하여 개발 환경 통일성과 배포 편의성을 확보했습니다.   각 서비스(auth-server, ai-server)를 위한 개별 Dockerfile이 프로젝트 내에 존재하지 않습니다.	 <b>(신규 제안)</b> 각 서비스 루트에 Dockerfile을 작성해야 합니다. 특히 <b>Multi-stage builds</b> 를 사용하여 최종 이미지 크기를 줄이고, 불필요한 빌드 캐시를 제거하여 이미지 효율을 최적화해야 합니다.	docker-compose.yml
모니터링	Prometheus/Grafana	 docker-compose.yml에 모니터링 시스템(Prometheus, Grafana, Promtail)을 도입하려는 구성이 존재합니다.	 <b>(구체화 필요)</b> 현재 설정은 기본 설정에 가깝습니다. 각 서버에서 비즈니스적으로 중요한 메트릭(e.g., API 응답 시간, 에러율, 사용자 수)을	monitoring/*

			Micrometer(Spring), prometheus-fast api-instrumentator(Python) 등을 통해 노출하고 Grafana 대시보드에서 시각화하는 작업이 필요합니다.	
CI/CD	자동화	⚠ (개선 필요) .github/workflows 와 같은 CI/CD 설정 파일이 없어, 현재 수동으로 빌드 및 배포하는 것으로 보입니다.	🚀 (신규 제안) <b>GitHub Actions</b> 를 도입하여 develop 브랜치에 코드가 푸시될 때마다 [테스트 → 빌드 → Docker 이미지 생성/푸시 → 서버 배포] 과정을 자동화하는 파이프라인 구축이 시급합니다. 이는 개발 생산성을 극대화하고 휴먼 에러를 줄여줍니다.	-

## 6. 종합 결론 및 우선순위 권장 사항

CBT Diary는 MSA의 기본 철학을 잘 이해하고 최신 기술 스택을 적극적으로 도입하려는 노력이 돋보이는 잠재력 높은 프로젝트입니다.

프로젝트의 안정성과 확장성을 한 단계 더 높이기 위해 아래 사항들을 우선적으로 개선할 것을 권장합니다.

### ● P0 (가장 시급)

- **CI/CD 파이프라인 구축**: GitHub Actions를 이용한 테스트/빌드/배포 자동화.
- **Refresh Token 보안 강화**: 백엔드에 'Refresh Token Rotation' 전략 도입.

### ● P1 (중요)

- **Frontend API 클라이언트 중앙화**: axios와 인터셉터를 사용한 API 모듈 구현.
- **Frontend 테스트 코드 작성**: React Native Testing Library를 이용한 핵심 기능 테스트.

### ● P2 (권장)

- **Frontend 서버 상태 관리 도입**: React-Query 또는 SWR 도입.
- **모니터링 시스템 고도화**: 실제 비즈니스 메트릭을 Grafana 대시보드에 연동.

위 제안들을 반영한다면, CBT Diary는 기술적으로 더욱 견고하고 안정적인 서비스로 발전할 수 있을

것입니다.