LogMate

Software Requirements Specification (SRS)

Version	2.0
Authors	강찬욱, 양승원, 정주연
Date	2025.10.18

Table of Contents

3.2.1.6.5 HTTP 메서드 기반 필터링
3.2.1.7 Log Exporter
3.2.1.7.1 로그 전송
3.2.1.7.2 배치 단위처리
3.2.1.7.3 전송 재시도
3.2.1.7.4 중복 전송 방지
3.2.1.7.5 보안 전송 및 인증
3.2.1.7.6 선택적 압축 전송
3.2.1.7.7 서버 응답 처리
3.2.1.8 Fallback & Recovery
3.2.1.8.1 Fallback 저장소 생성
3.2.1.8.2 로그 직렬화 및 저장
3.2.1.8.3 저장소 읽기
3.2.1.8.4 손상된 로그 무시
3.2.1.8.5 Fallback 로그 재전송
3.2.1.8.6 저정소 초기화

3.2.2 Streaming Server
3.2.2.1 Ingress
3.2.2.1.1 로그 수신
3.2.2.1.2 인증 및 권한 검증
3.2.2.1.3 유효성 검사
3.2.2.1.4 표준화 처리
3.2.2.1.5 압축 데이터 해제
3.2.2.2 Messaging
3.2.2.2.1 메시지 브로커 전송
3.2.2.2.2 메시지 브로커 로그 소비
3.2.2.2.3 DLQ 관리
3.2.3.1 Pipeline
3.2.2.3.1 AI 분석 요청
3.2.2.3.2 로그 저장
3.2.2.3.3 실시간 로그 스트리밍
3.2.4.1 Search
3.2.2.4.1 로그 검색

3.2.3 API Server
3.2.3.1 사용자 인증 및 접근
3.2.3.1.1 로그인 요청 처리
3.2.3.1.2 회원가입 요청 처리
3.2.3.1.3 로그아웃 처리
3.2.3.2 팀 워크스페이스 및 대시보드 관리
3.2.3.2.1 팀 목록 조회
3.2.3.2.2 대시보드 목록 및 상태 조회
3.2.3.2.3 대시보드 생성 및 수정
3.2.3.2.4 파싱 룰 등록
3.2.3.2.5 파싱 룰 미리보기 처리
3.2.3.2.6 필터링 룰 등록
3.2.3.3 팀 관리 및 사용자 정보 관리
3.2.3.3.1 팀 정보 수정
3.2.3.3.2 팀 멤버 권한 수정
3.2.3.3.3 팀 초대 URL 발급
3.2.3.3.4 팀 생성 처리
3.2.3.4 마이페이지 기능
3.2.3.4.1 사용자 정보 조회
3.2.3.4.2 사용자 정보 수정
3.2.3.4.3 사용자 탈퇴 처리
3.2.3.5 알림 기능
3.2.3.5.1 알림 목록 조회
3.2.3.5.2 알림 읽음 상태 업데이트
3.2.3.5.3 알림 페이지네이션 처리
3.2.3.6 제3자 어플리케이션 연동 알림

1. Introduction

1.1. Purpose

- 본 문서의 목적은 초간단 로그 수집 및 AI 분석 SaaS 서비스 LogMate에 대한 요구사항을 명확히 정의하는 것이다. 이 문서는 개발팀, 이해 관계자 및 유지보수 팀이 시스템의 주요 기능, 제약 사항 및 개발 범위를 이해하고, 이를 기반으로 효율적이고 안정적인 로그 수집 및 모니터링 시스템을 구축 및 관리하는 데 필요한 정보를 제공한다.

1.2. Scope

- LogMate는 경량 로그 수집·분석 SaaS로서, Agent가 수집한 로그를 Streaming Server로 전송하고, 내부 Kafka 파이프라인을 통해 OpenSearch(저장·검색) 및 AI Server(이상탐지)로 전달한다. Frontend는 WebSocket으로 실시간 로그를 표출하며, API Server는 인증·설정·대시보드·알림을 제공한다.
- 이 SRS는 LogMate의 소프트웨어 요구사항에 대한 전반적인 설명을 포함하며, 설치 환경, 네트워크 구성, 보안 인증 방식 등 일부 인프라 설계 및 배포 세부사항은 본 문서 범위에 포함되지 않는다.

1.3. Definitions, acronyms, and abbreviations

- LogMate: 본 프로젝트의 정식 명칭이자 서비스 이름
- Agent: 로그 수집을 담당하는 프로그램. 설정 주입 후 로그를 Tailing 하여 전송
- API Server: 사용자 인증, 팀/대시보드 설정 등을 담당하는 핵심 API 제공 서버
- Streaming Server: 실시간 로그를 수신 및 저장하고, WebSocket을 통해 사용자에게 전달하는 서버
- Frontend Server: 사용자 인터페이스를 제공하는 서버
- AI Server: AI 기반 로그 이상탐지를 수행하며, 데이터를 분석
- Tail: 로그 파일의 마지막 라인을 지속적으로 감시하여 변경된 내용을 수집하는 방식
- JWT: JSON Web Token, 인증 토큰 형식
- Webhook: 외부 시스템으로 이벤트 알림을 전송하기 위한 URL 기반 통신 규격
- WebSocket: 브라우저와 서버 간의 양방향 통신 프로토콜

1.4. References

- IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications
- LogMate GitHub 문서: https://github.com/TEAM-LOGMATE
- 수업자료: 2024년 소프트웨어공학 (교수자: 유준범)

1.5. Overview

- 본 문서는 다음과 같은 구성으로 이루어져 있다.
- 제1장은 시스템 개요 및 문서 목적을 설명한다.
- 제2장은 시스템 전반 구조와 컴포넌트 간 상호작용을 포함한 개요를 설명한다.
- 제3장은 각 컴포넌트별 기능 요구사항(Functional Requirements)을 세부적으로 기술하며, Agent, API, Streaming, Frontend, AI Server 별 요구사항을 명확히 분리하여 제시한다.

2. Overall Description

2.1. Product Perspective

- LogMate는 로그 수집, 분석, 시각화를 위한 경량 SaaS 기반 모니터링 시스템으로, 기존의 복잡한 설정과 인프라 구성을 단순화하여 개발자 친화적인 로그 운영 환경을 제공하는 것을 목표로 한다.
- 이 시스템은 다섯 개의 주요 모듈 (Agent, Streaming Server, API Server, FrontendServer, AI Server) 로 구성되며, 각 모듈은 독립적으로 실행된다.

- LogMate 는 다양한 OS 및 언어 환경에서 실행되는 애플리케이션의 로그 읽기를 지원하며, 초기 설정 없이 동작 가능한 Agent 를 제공하고, Web UI 를 통한 동적 설정 주입으로 운영 중에도 무중단 설정 변경이 가능한다.

2.2. Product Functions

- 로그 수집 에이전트**:** 보안식별자 이외의 설정 없이 바로 실행되며, 설정 주입 후 로그 수집을 시작한다.
- 실시간 로그 전송 및 스트리밍**:** 메시지큐를 통해 스트리밍 서버로 로그 전송 후 WebSocket으로 사용자에게 전달한다.
- 직관적인 **Web UI** 설정**:** 로그 경로, 파서, 전송 주기, 필터링 조건 등을 브라우저에서 쉽게 구성 가능하다.
- 다중 로그 파일 수집 및 멀티라인 병합**:** 복수 로그파일 동시 수집 및 멀티라인 로그 병합 지원
- 보안 통신 및 인증 검증**:** HTTPS, JWT 토큰 기반 인증, 설정 주입 시 보안 검사
- **AI** 이상 탐지 및 시각화**:** 비정상 로그를 탐지하고 대시보드 형태로 시각화
- 메신저 어플과 연동 알림**:** 외부 알림 전송 가능

2.3. User Characteristics

- LogMate는 인프라 지식이 부족한 사용자도 쉽게 사용할 수 있도록 UI 중심으로 설계되었으며, 직관적인 설정 마법사 및 시각화 대시보드를 제공한다.
- DevOps 및 엔터프라이즈 환경을 위한 고급 설정 옵션도 지원한다.
- 사용자 유형은 프리랜서 개발자, 스타트업 교육기관 등으로 다양하게 구성될 수 있다.

2.4. Constraints

- 로그 파일 수집은 로컬 파일 시스템 기반이며, 파일 접근 권한이 있어야 한다.
- 로그 전송은 HTTPS 를 통해 수행되며, 에이전트 인증 토큰이 없을 경우 설정/전송 모두 거부된다.
- 로그 수집 및 처리는 실시간으로 이루어져야 하며, 지연은 최대 3초를 초과하면 안 된다.
- 시스템은 멀티 테넌트 환경을 지원하며, 각 테넌트 간 데이터의 접근 및 관리 권한이 명확하게 분리되어야 한다.
- Webhook 알림 기능은 시스템 내 중대 이상 상황 발생 시 1분 이내로 작동하여 알림을 제공해야 한다.
- 시스템의 가용성은 프로젝트 기간 동안 95% 이상을 유지해야 하며, 이보다 긴 장애는 발생하면 안 된다.
- 서비스는 별도의 복잡한 초기 설정을 필요로 하면 안 된다.

2.5. Assumptions and Dependencies

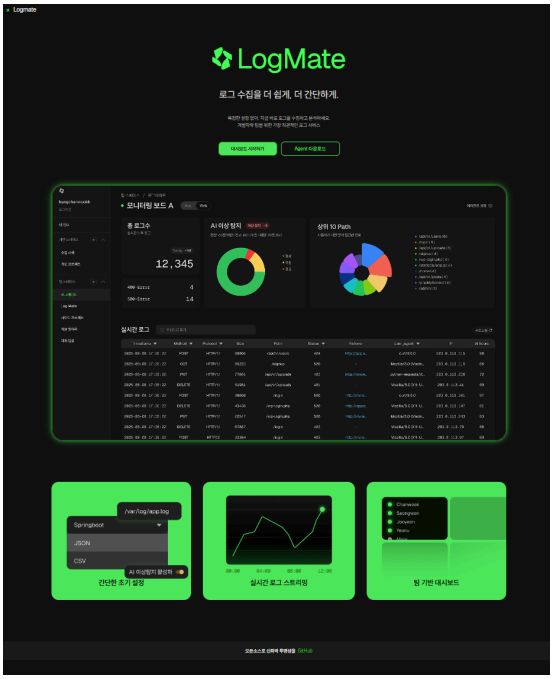
- 시스템은 안정적인 인터넷 환경에서 운영된다고 가정한다. 네트워크 단절이나 극단적인 속도 저하는 고려하지 않는다.
- 로그를 전송하는 클라이언트 애플리케이션은 올바르게 구성되어 있으며, 형식에 맞는 데이터를 전송한다고 가정한다.
- 로그 수집 서버와 데이터베이스, AI 분석 서버는 정상적으로 동작하고 있다고 가정한다.
- 외부 시스템(Slack 등)과의 연동에 필요한 API 키 및 인증 정보는 사전에 정상적으로 등록되어 있다고 가정한다.

3. Specific Requirements

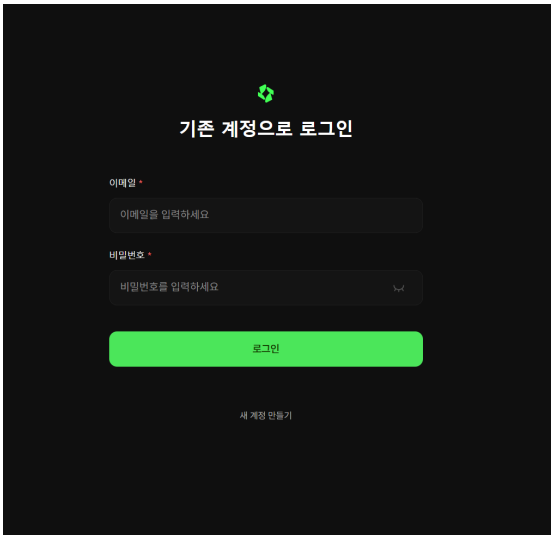
3.1. External Interfaces

User Interfaces

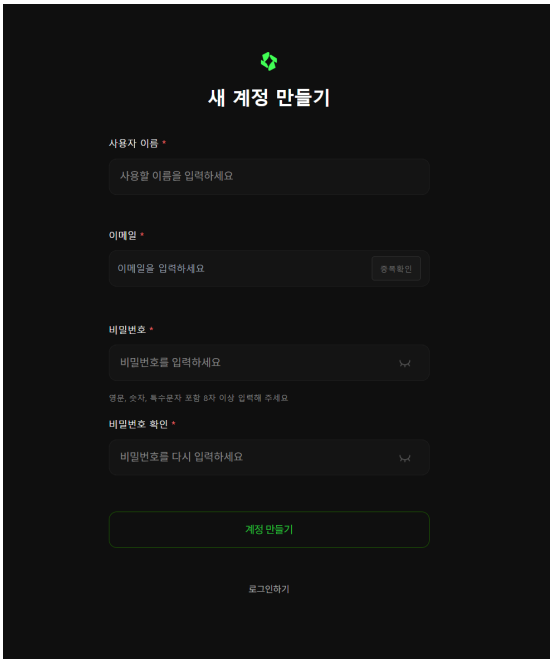
1. 온보딩 화면	**2.** 로그인 화면



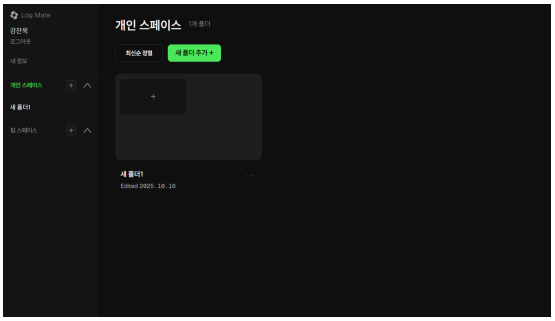
3. 회원가입 화면



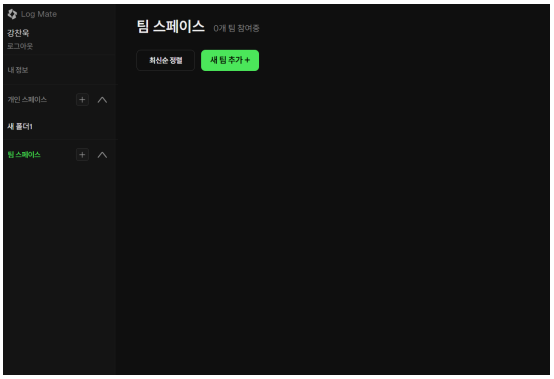
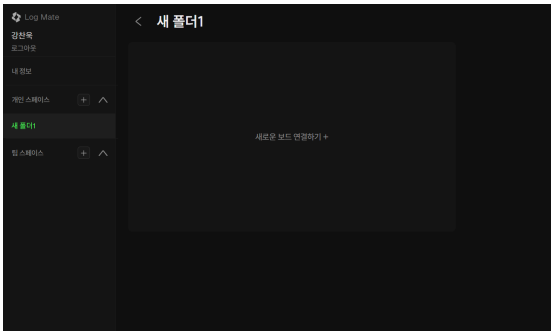
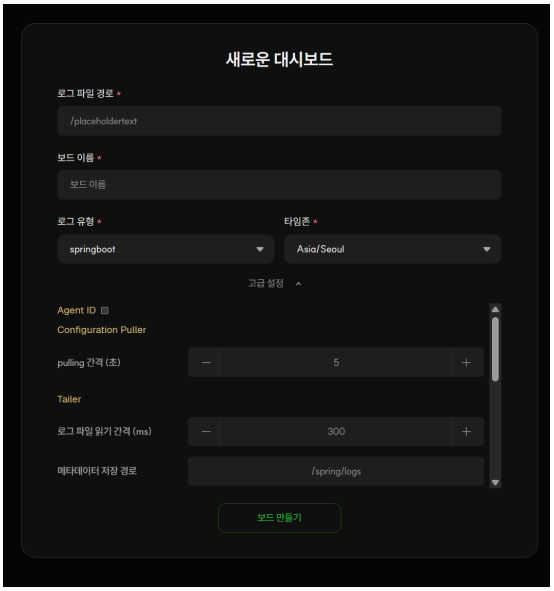
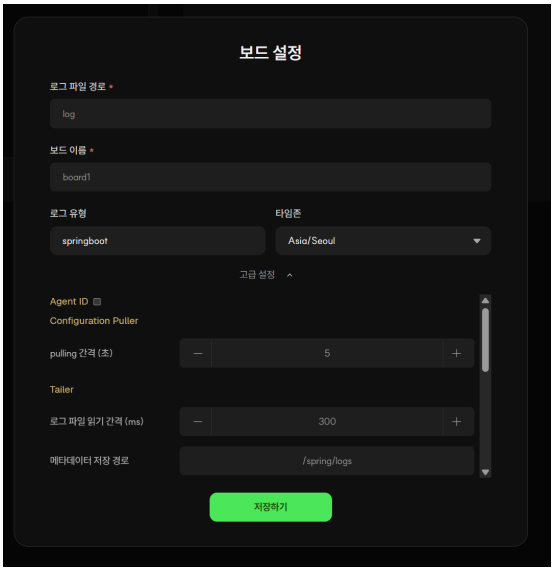
4. 개인 스페이스 화면

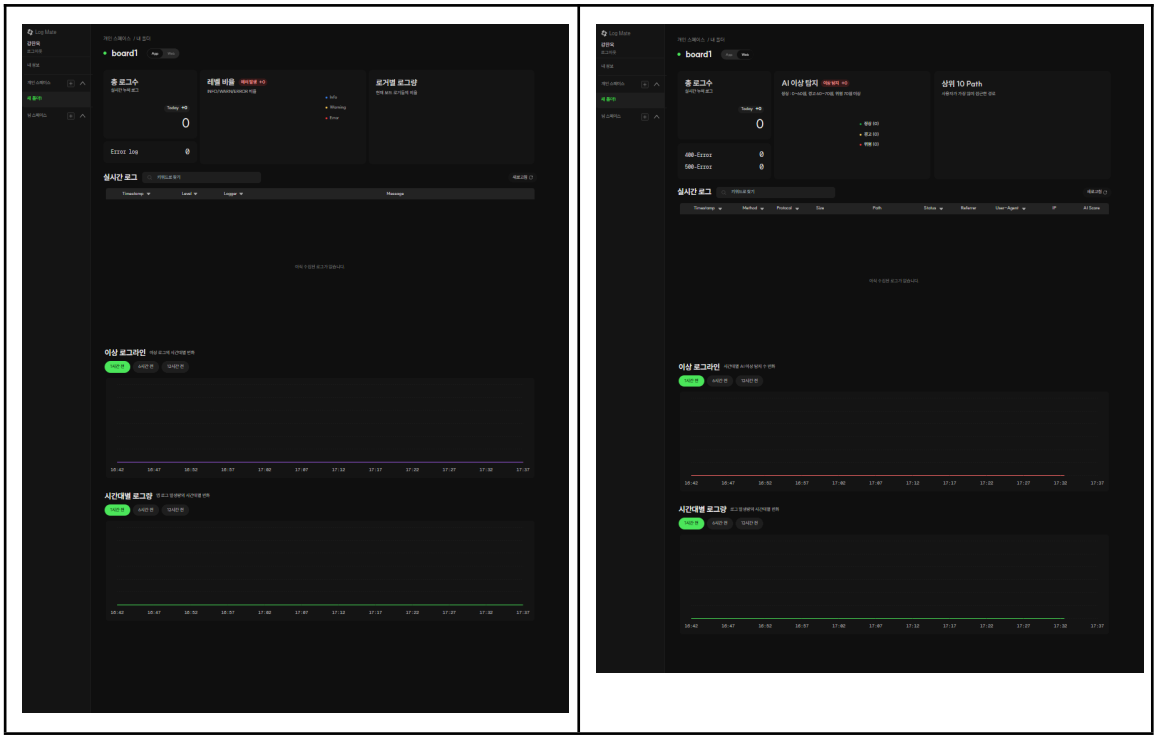


5. 팀 스페이스 화면



6. 폴더 화면

<table>
<tr>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>7. 대시보드 만들기 화면</td>
<td>8. 대시보드 수정 화면</td>
</tr>
<tr>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>9. Springboot 로그 대시보드</td>
<td>10. Tomcat Access 로그 대시보드omcat Access 로그 대시보드</td>
</tr>
</table>



Software Interfaces

- 로그 수집 인터페이스 (Agent -> Streaming server)
  - Agent가 수집한 로그데이터를 실시간으로 Streaming server에 전송한다.
  - 프로토콜: HTTPS
- 실시간 로그 스트리밍 인터페이스 (Streaming Server <-> Frontend)
  - WebSocket 을 통해 실시간 로그 메세지를 사용자에게 전달하고, 사용자는 필터 조건을 서버로 전송한다.
  - JSON 메시지 기반의 양방향 통신
  - 프로토콜: WebSocket
- 로그 조회 인터페이스 (Streaming Server <-> Frontend)
  - DB 에 저장된 로그를 조건에 대해 검색한다.
  - 프로토콜: HTTPS
- 로그 분석 인터페이스 (Streaming Server -> AI Server)
  - 수신된 로그를 AI 분석 서버로 전달하여 이상 여부를 판단한다.
  - 프로토콜 : HTTPS
- 사용자 인증 및 설정 관리 인터페이스 (Frontend <-> API Server)
  - 로그인, 회원가입 등 사용자 인증 기능을 제공한다.
  - 설정 등록, 수신 및 반영 기능을 제공한다.
  - 대시보드 관리 기능을 제공한다.
  - 팀 및 유저 권한을 관리한다.
  - 프로토콜 : HTTPS

Communications Interfaces

통신주체	설명
Agent → API Server	Agent 설정을 요청하며 인증을 진행한다.

Agent → Streaming Server	실시간 로그 데이터를 스트리밍 서버로 전송한다.
Streaming Server → Web Client (Frontend)	클라이언트에게 실시간 로그를 전달한다. 클라이언트는 필터 조건을 포함한 메시지를 서버로 전송할 수 있다. 로그 검색 요청에 응답한다.
Streaming Server → AI Server	웹 로그를 수신하면 AI Server로 전달하고, AI는 이상 여부 결과를 반환한다.
API Server → Frontend	사용자 인증, 대시보드 설정, 알림 설정, 로그 조회 등 모든 요청을 처리한다.

3.2. Functional Requirements

3.2.1.1. Agent

3.2.1.1.1. Agent Initialization

3.2.1.1.1.1. 실행 인자 추출 **(FR-1.1.1)**

- 시스템은 실행 시 전달된 프로그램 인자(args)를 파싱하여 데이터 클래스로 변환해야 한다.
- 잘못된 인자가 전달된 경우 에러를 기록하고 초기화를 중단해야 한다.

3.2.1.1.1.2. **Config Puller** 시작 **(FR-1.1.2)**

- 시스템은 초기화가 완료되면 Config Puller 를 실행하여 외부 설정 주입 요청을 수신해야 한다.
- 설정 변경 이벤트가 발생하면 무중단으로 반영할 수 있어야 한다.

3.2.1.1.1.3. 로컬 설정 불러오기 및 초기화 **(FR-1.1.3)**

- 시스템은 YAML 설정 파일을 로드하여 Agent 설정을 초기화해야 한다.

3.2.1.1.1.4. 설정 유효성 검사 **(FR-1.1.4)**

- 시스템은 불러온 설정값에 대해 유효성 검사를 수행하고, 오류가 있는 경우 기본값을 적용하거나 실행을 중단해야 한다.

3.2.1.1.1.5. **Agent** 인증 수행 **(FR-1.1.5)**

- 시스템은 ID, Password를 통해 인증 서버(API Server) 에 검증을 요청해야 한다.
- 인증 실패 시 초기화를 중단해야 한다.

3.2.1.1.2. Configuration Injection

3.2.1.1.2.1. 주기적 설정 **Pull** 수행 **(FR-1.2.1)**

- 시스템은 일정 주기마다 API 서버에 HTTP 요청을 보내 최신 설정을 가져와야 한다.

3.2.1.1.2.2. **eTag** 기반 변경 감지 **(FR-1.2.2)**

- 시스템은 마지막으로 수신한 설정의 eTag 값을 서버에 전달하여, 변경된 설정이 있을 때만 새 설정을 수신해야 한다.
- 변경이 없을 경우 API 서버는 응답을 생략하거나 캐시 히트 응답을 반환할 수 있다.

3.2.1.1.2.3. 인증 기반 요청 **(FR-1.2.3)**

- 시스템은 설정 Pull 요청 시 Access Token을 포함하여 인증된 요청만 수행해야 한다.
- 인증 실패 시 Puller 동작을 중단하고 오류를 기록해야 한다.

3.2.1.1.2.4. 설정 반영 **(FR-1.2.4)**

- 시스템은 새로운 설정을 수신하면 `ConfigUpdater`를 통해 무중단으로 애플리케이션에 적용해야 한다.
- 적용 완료 후 최신 eTag 값으로 교체해야 한다

3.2.1.1.2.5. 설정 유효성 검사 **(FR-1.2.5)**

- 시스템은 불러온 설정값에 대해 유효성 검사를 수행하고, 오류가 있는 경우 기본값을 적용하거나 실행을 중단해야 한다.

3.2.1.1.2.6. 변경 없음 처리 **(FR-1.2.6)**

- 시스템은 새로운 설정이 없을 경우 로그에 “변경 없음” 메시지를 기록하고 기존 설정을 유지해야 한다.

3.2.1.1.3. Log Tailer

3.2.1.1.3.1. 실시간 로그 수집 **(FR-1.3.1)**

- 시스템은 지정된 경로의 로그 파일을 tailing 방식으로 실시간 수집해야 한다.

3.2.1.1.3.2. 로그 파일 접근 제어 **(FR-1.3.2)**

- 시스템은 보안상 민감한 디렉토리 접근을 차단해야 한다.

3.2.1.1.3.3. 다중 로그 파일 처리 **(FR-1.3.3)**

- 시스템은 동시에 여러 개의 로그 파일을 tailing할 수 있어야 한다.

3.2.1.1.3.4. 확장자가 없는 로그 파일 수집 **(FR-1.3.4)**

- 시스템은 확장자가 없는 로그 파일도 인식하여 수집할 수 있어야 한다.

3.2.1.1.3.5. 로그 롤링**/**삭제 대응 **(FR-1.3.5)**

- 시스템은 로그 파일이 롤링되거나 삭제 후 재생성되는 경우를 감지하고, 처음부터 다시 읽어야 한다.
- 변환된 로그는 필터링 및 전송 전처리를 위한 상태로 유지되어야 한다.

3.2.1.1.3.6. 인코딩 처리 **(FR-1.3.6)**

- 시스템은 설정에 정의된 문자셋(예: UTF-8, EUC-KR)에 맞춰 로그를 해석해야 한다.

3.2.1.1.3.7. 메타데이터 기반 위치 저장 **(FR-1.3.7)**

- 시스템은 마지막으로 읽은 위치를 메타데이터 파일에 저장하여, 재시작 시 이어서 읽을 수 있어야 한다.

3.2.1.1.3.8. 종료**/**재시작 처리 **(FR-1.3.8)**

- 시스템은 Tailer 쓰레드를 안전하게 종료하고 재시작할 수 있는 인터페이스를 제공해야 한다.

3.2.1.1.4. Log Merger

3.2.1.1.4.1. 멀티라인 로그 병합 **(FR-1.4.1)**

- 시스템은 스택트레이스 등 여러 줄에 걸친 로그를 하나의 이벤트로 병합해야 한다.

3.2.1.1.4.2. 로그 병합 미사용 **(FR-1.4.2)**

- 시스템은 enabled = false 로 설정된 경우, 모든 로그를 원본 그대로 출력해야 한다.

3.2.1.1.4.3. 병합 조건 판별 **(FR-1.4.3)**

- 시스템은 LogParser의 판단 로직을 사용하여 한 줄 로그인지, 멀티라인에 포함될 라인인지 판별해야 한다.

3.2.1.1.4.4. 최대 병합 라인 제한 **(FR-1.4.4)**

- 시스템은 병합 버퍼의 병합 가능 최대 라인 수를 초과할 경우, 강제로 병합하여 출력해야 한다

3.2.1.1.4.5. 버퍼 플러시 정책 **(FR-1.4.5)**

- 시스템은 새 로그가 들어오거나 입력이 끝날 때, 병합 중인 버퍼가 남아 있으면 즉시 플러시해야 한다.

3.2.1.1.4.6. 병합 실패 처리 **(FR-1.4.6)**

- 병합 실패 시 원본 라인을 보존하고, fallback 저장소에 기록해야 한다

3.2.1.1.5. Log Parser

3.2.1.1.5.1. 로그 파싱 **(FR-1.5.1)**

- 시스템은 SpringBoot, Tomcat Access 등 사전 정의된 포맷에 따라 로그를 구조화해야 한다.

3.2.1.1.5.2. 타임존 변환 **(FR-1.5.2)**

- 시스템은 로그의 timestamp를 항상 UTC 기준(LocalDateTime, 초 단위)으로 변환해야 한다.
- Timezone이 누락된 경우 사용자가 설정한 기본 Timezone을 사용해야 한다.

3.2.1.1.5.3. 비형식 로그 판별 **(FR-1.5.3)**

- 시스템은 로그 한 줄이 정의된 패턴에 맞는지 여부를 판별할 수 있어야 한다.

3.2.1.1.5.4. 비형식 로그 처리 **(FR-1.5.4)**

- 시스템은 정의되지 않은 포맷 로그도 Raw 형태로 보관 가능해야 한다.

3.2.1.1.6. Log Filter

3.2.1.1.6.1. 필터 적용 범위 구분 **(FR-1.6.1)**

- 시스템은 로그 유형(Spring Boot 로그, Tomcat Access 로그 등)에 따라 서로 다른 필터 로직을 적용해야 한다.

3.2.1.1.6.2. 파싱 실패 로그 보존 **(FR-1.6.2)**

- 시스템은 파싱이 실패한 로그 라인은 사용자 설정과 상관없이 무조건 accept 처리해야 한다.
- 이는 스택트레이스·멀티라인 로그 등 디버깅 필수 정보의 손실을 방지하기 위함이다.

3.2.1.1.6.3. 로그 레벨 기반 필터링 **(FR-1.6.3)**

- Spring Boot 로그에 대해서는 설정을 기반으로 로그 레벨(INFO, WARN, ERROR 등)을 허용하거나 차단해야 한다.
- 허용 레벨이 지정되지 않은 경우, 모든 로그 레벨을 수용해야 한다.

3.2.1.1.6.4. 키워드 기반 필터링 **(FR-1.6.4)**

- Spring Boot 로그에 대해서는 설정을 기반으로 로그 메시지에 특정 키워드가 포함된 경우에만 accept 처리해야 한다.
- 설정이 비어 있는 경우 모든 메시지를 수용해야 한다.

3.2.1.1.6.5. **HTTP** 메서드 기반 필터링 **(FR-1.6.5)**

- Tomcat Access 로그에 대해서는 설정을 기반으로 GET, POST 등 허용된 HTTP 메서드만 accept 처리해야 한다.

- 허용 메서드가 비어 있는 경우 모든 메서드를 수용해야 한다.

3.2.1.1.7. Log Exporter

3.2.1.1.7.1. 로그 전송 **(FR-1.7.1)**

- 시스템은 파싱된 로그를 API 서버(HTTPS)로 전송해야 한다.

3.2.1.1.7.2. 배치 단위 처리 **(FR-1.7.2)**

- 시스템은 로그를 전송할 때 일정 크기에 따라 배치로 분할하여 전송해야 한다.

3.2.1.1.7.3. 전송 재시도 **(FR-1.7.3)**

- 시스템은 네트워크 오류, 서버 다운 등의 상황에서 재시도 정책을 적용해야 한다.

3.2.1.1.7.4. 중복 전송 방지 **(FR-1.7.4)**

- 시스템은 같은 로그가 여러 번 전송되지 않도록 전송 상태를 추적해야 한다.

3.2.1.1.7.5. 보안 전송 및 인증 **(FR-1.7.5)**

- 시스템은 JWT 기반 인증과 TLS 암호화를 적용해야 한다.

3.2.1.1.7.6. 선택적 압축 전송 **(FR-1.7.6)**

- 시스템은 설정에 따라 로그 전송 시 압축을 적용할 수 있어야 한다.

3.2.1.1.7.7. 서버 응답 처리 **(FR-1.7.7)**

- 시스템은 서버 응답 코드가 2xx 범위일 경우 성공으로 간주하고, 그렇지 않은 경우 실패로 처리해야 한다.

3.2.1.1.8. Fallback & Recovery

3.2.1.1.8.1. **Fallback** 저장소 생성 **(FR-1.8.1)**

- 시스템은 전송 실패한 로그를 로컬 파일로 저장해야 한다.

3.2.1.1.8.2. 로그 직렬화 및 저장 **(FR-1.8.2)**

- 시스템은 실패한 로그를 JSON 포맷으로 직렬화하여 한 줄 단위로 파일에 append해야 한다**.**

3.2.1.1.8.3. 저장소 읽기 **(FR-1.8.3)**

- 시스템은 저장소에 기록된 로그를 순차적으로 모두 읽어와 메모리에 적재할 수 있어야 한다.
- 파싱 불가능한 라인은 무시하고 경고 로그를 남겨야 한다.

3.2.1.1.8.4. 손상된 로그 무시 **(FR-1.8.4)**

- 시스템은 저장된 로그가 일부만 손상되더라도 나머지 정상 로그를 불러올 수 있어야 한다.

3.2.1.1.8.5. **Fallback** 로그 재전송 **(FR-1.8.5)**

- 시스템은 네트워크나 서버 복구 시 Fallback 저장소에 있는 로그를 재전송해야 한다.

3.2.1.1.8.6. 저장소 초기화 **(FR-1.8.6)**

- 시스템은 로그가 성공적으로 복구·전송되면 저장소를 삭제하거나 초기화해야 한다.

3.2.1.2. Streaming Server

3.2.1.2.1. Ingress

3.2.1.2.1.1. 로그 수신 **(FR-2.1.1)**

- 시스템은 Agent가 보낸 로그를 수신해야 한다.

3.2.1.2.1.2. 인증 및 권한 검증 **(FR-2.1.2)**

- 시스템은 요청 헤더의 Access Token을 검증하여 유효한 Agent만 수락해야 한다.

3.2.1.2.1.3. 유효성 검사 **(FR-2.1.3)**

- 시스템은 요청에 대한 유효성을 검증해야 한다.
- 유효하지 않은 로그는 수용하지 않아야 한다.

3.2.1.2.1.4. 표준화 처리 **(FR-2.1.4)**

- 시스템은 수신 로그를 내부 표준 DTO로 변환해야 한다.
- 로그 유형(SpringBoot, Tomcat 등)에 따라 parser를 달리 적용해야 한다.

3.2.1.2.1.5. 압축 데이터 해제 **(FR-2.1.5)**

- 시스템은 압축된 로그 데이터를 감지하여 압축 해제할 수 있어야 한다.
- 해제된 데이터를 Streaming Process 로 진입시켜야 한다.

3.2.1.2.2. Messaging

3.2.1.2.2.1. 메시지 브로커 전송 **(FR-2.2.1)**

- 시스템은 표준화된 로그를 Kafka 토픽에 produce해야 한다.
- 전송 실패 시 재시도/백업 처리해야 한다.

3.2.1.2.2.2. 메시지 브로커 로그 소비 **(FR-2.2.2)**

- 시스템은 Kafka 토픽에서 로그를 구독(consume)해야 한다.
- 메시지 offset 관리 및 재처리를 지원해야 한다.

3.2.1.2.2.3. DLQ 관리 **(FR-2.2.3)**

- 시스템은 파이프라인에서 실패한 로그를 DLQ(Dead Letter Queue) 에 produce 해야 한다.

3.2.1.2.3. Pipeline

3.2.1.2.3.1. AI 분석 요청 **(FR-2.3.1)**

- 시스템은 소비한 로그를 AI 서버에 전달하여 이상 탐지/스코어링을 요청할 수 있어야 한다.
- 요청 실패 시 로그는 정상 파이프라인 저장소로 흘려보내야 한다(Fail-safe).

3.2.1.2.3.2. 로그 저장 **(FR-2.3.2)**

- 시스템은 소비한 로그를 OpenSearch 등 검색/분석 가능한 저장소에 적재해야 한다.
- 적재 시 인덱스 정책(시간 단위, 레벨별 구분 등)을 적용할 수 있어야 한다.

3.2.1.2.3.3. 실시간 로그 스트리밍 **(FR-2.3.3)**

- 시스템은 WebSocket/SSE를 통해 소비자(UI 대시보드 등)에게 실시간 로그를 push해야 한다.
- 연결 관리(구독/해제)와 backpressure 제어를 지원해야 한다.

3.2.1.2.4. Search

3.2.1.2.4.1. 로그 검색 **(FR-2.4.1)**

- 시스템은 DB에 저장했던 로그를 검색하여 제공할 수 있어야 한다.

3.2.1.3. API Server

3.2.1.3.1. 사용자 인증 및 접근

3.2.1.3.1.1. 로그인 요청 처리 **(FR-3.1.1)**

- 시스템은 사용자의 이메일과 비밀번호를 수신하여 인증을 수행하고, 성공 시 액세스 토큰과 사용자 정보를 반환해야 한다. 실패 시 오류 메시지를 반환해야 한다.

3.2.1.3.1.2. 회원가입 요청 처리 **(FR-3.1.2)**

- 시스템은 이름, 이메일, 비밀번호를 수신하여 신규 사용자를 생성하고, 성공 시 사용자 ID와 등록한 이름, 이메일을 반환해야 한다. 이메일 중복 여부를 검사하며 이미 동일한 이메일이 DB에 등록돼있는 경우 오류 메시지를 반환해야 한다.

3.2.1.3.1.3. 로그아웃 처리 **(FR-3.1.3)**

- 시스템은 사용자의 토큰 폐기 요청을 수신하고, 서버 측 세션 정보 또는 토큰 블랙리스트에 등록하여 로그아웃 처리를 완료해야 한다.

3.2.1.3.2. 팀 워크스페이스 및 대시보드 관리

3.2.1.3.2.1. 팀 목록 조회 **(FR-3.2.1)**

- 시스템은 로그인된 사용자의 소속 팀 목록(팀 이름과 팀 설명)을 반환해야 한다.

3.2.1.3.2.2. 대시보드 목록 및 상태 조회 **(FR-3.2.2)**

- 시스템은 사용자가 선택한 팀에 속한 대시보드들 / 개인의 특정 폴더에 속한 대시보드들의 목록과 각 대시보드의 이름과 로그 경로, 마지막 수정일, 대시보드의 상태 (로그 수집 전, 수집 중 등) 을 반환해야 한다

3.2.1.3.2.3. 대시보드 생성 및 수정 **(FR-3.2.3)**

- 시스템은 대시보드 이름, 로그 경로 등의 설정 정보를 수신하고, 신규 생성 또는 기존 대시보드 설정을 갱신해야 한다.

3.2.1.3.2.4. 대시보드 고급정보 설정 및 수정 **(FR-3.2.4)**

- 시스템은 대시보드 고급 설정 정보 (파서, 필터, 멀티라인 등)를 수신하고, 신규 생성 또는 기존 설정을 갱신하며 갱신의 경우는 etag를 새롭게 발급해야 한다.

3.2.1.3.2.5. 대시보드 고급정보 설정 조회 **(FR-3.2.5)**

- 시스템은 사용자가 선택한 팀이나 개인의 폴더에 속한 대시보드의 고급정보 설정 값을 반환해야 한다

3.2.1.3.2.6. 대시보드 고급정보 설정 **agent pulling** 지원 **(FR-3.2.6)**

- agent server의 요청 파라미터로 agentId와 etag(nullable)를 수신하여 해당 agentId로 등록된 모든 고급 설정 정보를 JSON 타입으로 반환해야 한다. 한 agentId에 속하는 로그파일(대시보드)이 여러 개인 경우 해당하는 logPipelineConfig를 모두 반환한다. 요청 받은 etag와 DB의 etag가 다른 경우 설정값이 변경된 것으로 간주하고 응답을 보내주며, 동일한 경우에는 변경된 것이 없으므로 JSON응답을 따로 보내지 않으며 304 Not Modified 오류를 전송한다.

3.2.1.3.2.7. 대시보드 삭제 처리 **(FR-3.2.7)**

- 시스템은 삭제 요청을 수신하면 대시보드 삭제 처리를 한다.

3.2.1.3.3. 팀 관리 및 사용자 정보 관리

3.2.1.3.3.1. 팀 정보 수정 **(FR-3.3.1)**

- 시스템은 팀 이름, 설명 등의 변경 요청을 수신하고, 해당 정보를 갱신해야 한다.

3.2.1.3.3.2. 팀 멤버 권한 수정 **(FR-3.3.2)**

- 시스템은 팀 멤버 리스트와 각 권한 변경 요청을 수신하여 저장하고, 권한에 따라 접근 제어가 이루어지도록 해야 한다. ADMIN 권한의 사용자만 권한 변경이 가능하다.

3.2.1.3.3.3. 팀 생성 처리 **(FR-3.3.4)**

- 시스템은 팀 이름, 설명, 초기 멤버 정보, 권한 정보를 수신하여 팀을 생성하고, 사용자를 해당 팀에 등록해야 한다.

3.2.1.3.3.4. 팀 삭제 처리 **(FR-3.3.5)**

- 시스템은 팀 삭제 요청을 수신하면 팀 삭제 처리를 한다. 팀 관리자 (ADMIN)만 삭제 처리가 가능하다.

3.2.1.3.4. 마이페이지 기능

3.2.1.3.4.1. 사용자 정보 조회 **(FR-3.4.1)**

- 시스템은 로그인한 사용자의 프로필 정보를 반환해야 한다.

3.2.1.3.4.2. 사용자 정보 수정 **(FR-3.4.2)**

- 시스템은 이메일 또는 비밀번호 변경 요청을 수신하고, 유효성 검사를 수행한 후 정보를 갱신해야 한다

3.2.1.3.4.3. 사용자 탈퇴 처리 **(FR-3.4.4)**

- 시스템은 탈퇴 요청을 수신하면 사용자 데이터를 비활성화 처리하거나 삭제하고, 관련 리소스를 정리해야 한다.

3.2.1.3.5. 제**3**자 어플리케이션 연동 알림

3.2.1.3.5.1. **WebHook URL** 등록 **(FR-3.6.1)**

- 시스템은 사용자로부터 Webhook 전송 대상 URL(Slack, Discord, Custom)을 수신하고, 저장하여 추후 이벤트 전송에 사용할 수 있도록 해야 한다.

3.2.1.3.5.2. 외부 알림 테스트 전송 **(FR-3.6.2)**

- 사용자가 WebHook URL 등록 전, 유효한 URL인지 확인 가능한 기능이다. DB에 URL 등록 여부와 관계 없이 테스트 알림이 전송된다.

3.2.1.3.5.3. 외부 알림 전송 **(FR-3.6.3)**

- 시스템은 이벤트 발생 시 저장된 Webhook URL로 알림 메시지를 POST 요청으로 전송해야 하며, 실패 시 재시도 로직 또는 실패 기록을 남겨야 한다

3.2.1.3.6. 실시간 로그 스트리밍

3.2.1.3.6.1. 실시간 로그 브로드캐스트 **(FR-3.7.1)**

- 시스템은 로그 수집 파이프라인으로부터 수신한 로그를 구독 중인 사용자 세션에게 실시간으로 전달해야 한다.

3.2.1.4. Frontend Server

3.2.1.4.1. 사용자 인증 및 접근

3.2.1.4.1.1. 사용자 로그인 **(FR-4.1.1)**

- 시스템은 사용자가 이메일과 비밀번호를 입력할 수 있는 로그인 화면을 제공하고, 입력값을 API 서버에 인증 요청으로 전송해야 한다.

3.2.1.4.1.2. 사용자 회원가입 **(FR-4.1.2)**

- 시스템은 사용자가 이름, 이메일, 비밀번호를 입력할 수 있는 회원가입 화면을 제공하고, 입력값을 API 서버에 계정 생성 요청으로 전송해야 한다

3.2.1.4.1.3. 사용자 로그아웃 **(FR-4.1.3)**

- 시스템은 로그아웃 버튼을 제공하고, 클릭 시 토큰을 삭제하여 사용자를 비인증 상태로 전환해야 한다

3.2.1.4.2. 팀 워크스페이스/대시보드 관리

3.2.1.4.2.1. 팀 선택 기능 **(FR-4.2.1)**

- 시스템은 사용자가 소속된 팀 목록을 선택할 수 있는 UI를 제공하고, 선택된 팀의 정보를 API 서버로부터 불러와 화면에 표시해야 한다.

3.2.1.4.2.2. 대시보드 상태 표시 **(FR-4.2.2)**

- 시스템은 API 서버에서 조회한 대시보드 목록을 화면에 표시하고, 각 대시보드의 상태 정보를 함께 표시해야 한다.

3.2.1.4.2.3. 대시보드 추가 및 수정 **(FR-4.2.3)**

- 시스템은 사용자가 대시보드를 추가하거나 수정할 수 있는 버튼을 제공하고, 각 동작에 대한 입력 화면으로 연결되어야 한다.

3.2.1.4.2.4. 대시보드 설정 **(FR-4.2.4)**

- 시스템은 사용자가 대시보드 이름, 로그 경로, 전송 주소 등을 입력할 수 있는 화면을 제공하고, 설정 값을 API 서버에 전송해야 한다.

3.2.1.4.2.5. 파싱 룰 등록 **(FR-4.2.5)**

- 시스템은 주요 로그 형식에 따른 프리셋을 제공하고, 사용자가 정규표현식을 입력해 파싱 룰을 등록할 수 있도록 입력 필드를 제공해야 하며, 설정된 파싱 룰은 API 서버에 전송되어야 한다.

3.2.1.4.2.6. 파싱 룰 미리보기 **(FR-4.2.6)**

- 시스템은 사용자가 샘플 로그를 입력하면, 현재 설정된 파싱 룰에 따라 구조화된 결과를 시각적으로 보여줘야 한다.

3.2.1.4.2.7. 필터링 조건 등록 **(FR-4.2.7)**

- 시스템은 사용자가 키워드 포함/제외, 로그 레벨 등 조건을 입력할 수 있는 UI를 제공하고, 설정된 필터 조건은 API 서버에 저장되어야 한다

3.2.1.4.3. 팀 관리 및 사용자 정보 기능

3.2.1.4.3.1. 팀 정보 수정 **(FR-4.3.1)**

- 시스템은 팀 이름과 설명을 수정할 수 있는 입력 필드를 제공하고, API 서버에 정보 수정 요청을 전해야 한다.

3.2.1.4.3.2. 팀 멤버 권한 설정 **(FR-4.3.2)**

- 시스템은 팀 구성원 리스트와 권한 설정 버튼을 제공하고, 설정 변경 시 API 서버에 권한 변경 요청을 전송해야 한다.

3.2.1.4.3.3. 팀 초대 링크 제공 **(FR-4.3.3)**

- 시스템은 API 서버에서 발급받은 초대 URL을 사용자에게 표시하고, 복사 또는 공유할 수 있는 버튼을 제공해야 한다.

3.2.1.4.3.4. 팀 생성 **(FR-4.3.4)**

- 시스템은 팀 이름, 설명, 구성원 이메일 및 권한을 입력할 수 있는 화면을 제공하고, 입력된 정보를 API 서버에 팀 생성 요청으로 전송해야 한다.

3.2.1.4.4. 마이페이지 기능

3.2.1.4.4.1. 사용자 기본 정보 조회 **(FR-4.4.1)**

- 시스템은 API 서버에서 조회한 사용자 정보를 화면에 표시하고, 이메일, 이름, 비밀번호 상태 등을 보여줘야 한다.

3.2.1.4.4.2. 이메일/비밀번호 변경 **(FR-4.4.2)**

- 시스템은 이메일 또는 비밀번호를 변경할 수 있는 입력 필드 및 버튼을 제공하고, 입력된 정보를 API 서버에 전송해야 한다.

3.2.1.4.4.3. 계정 관리 기능 **(FR-4.4.3)**

- 시스템은 로그아웃 버튼과 탈퇴 버튼을 제공하고, 계정 탈퇴 시 확인 메시지를 띄운 후 API 서버에 탈퇴 요청을 전송해야 한다.

3.2.1.4.5. 알림 기능

3.2.1.4.5.1. 알림 목록 조회 **(FR-4.4.1)**

- 시스템은 API 서버로부터 사용자의 알림 목록을 조회하여 화면에 최신순으로 표시해야 한다

3.2.1.4.5.2. 알림 유형 구분 **(FR-4.5.2)**

- 시스템은 일반 알림과 중요 알림을 시각적으로 구분하여 렌더링해야 한다.

3.2.1.4.5.3. 알림 읽음 처리 **(FR-4.5.3)**

- 시스템은 사용자가 알림을 클릭하면 해당 알림을 읽음 처리하고, 그 상태를 API 서버에 전송해야 한다

3.2.1.4.5.4. 알림 스크롤 페이징 **(FR-4.5.4)**

- 시스템은 알림 목록을 무한 스크롤 또는 '더 보기' 방식으로 로드하고, 다음 알림 페이지를 API 서버에 요청해야 한다.

3.2.1.4.6. 제3자 어플리케이션 연동 알림

3.2.1.4.6.1. 외부 알림 전송 설정 **(FR-4.6.1)**

- 시스템은 사용자가 Slack, Discord 등의 Webhook URL을 입력할 수 있는 설정 화면을 제공하고, 설정 값을 API 서버에 전송해야 한다.

3.2.1.4.6.2. 알림 발생 시 외부 전송 **(FR-4.6.2)**

- 시스템은 Webhook 설정이 등록되어 있는 경우, 이벤트 발생 시 API 서버로 알림 전송 요청을 보내 외부 채널로 전달되도록 해야 한다.

3.2.1.4.7. 실시간 로그 스트리밍

3.2.1.4.7.1. 실시간 로그 수신 **(FR-4.7.1)**

- 시스템은 스트리밍 서버와 연결된 채널을 통해 전달되는 로그 메시지를 화면에 실시간으로 출력해야 한다

3.2.1.4.7.2. 필터 적용 **(FR-4.7.2)**

- 시스템은 사용자가 입력한 키워드, 로그 레벨 등 필터 조건을 스트리밍 서버에 전달하고, 필터링된 로그만 화면에 표시해야 한다.

3.2.1.5. AI Server

3.2.1.5.1. 이상탐지 **(FR-5.1.1)**

- 시스템은 스트리밍 서버에서 보내주는 웹 로그 데이터들을 기반으로 학습된 모델이 비 정상적인(공격 시도) 웹 로그를 탐지한다.

3.2.1.5.2. 실시간 로그 목록 표시 **(FR-5.1.2)**

- 탐지된 정상로그와 비정상 로그를 대시보드화하여 시각화로 보여줘야 한다.

3.2.1.5.3. 상태코드 기반 필터링 **(FR-5.1.3)**

- 웹 로그의 상태코드(2xx,3xx,4xx 등)를 기준으로 필터링과 시각화 가능을 제공해야 한다

3.2.1.5.4. **IP** 기반 사용자 추정 기능 **(FR-5.1.4)**

- 웹 로그에 포함된 IP주소로 접속한 사용자 수를 확인하여 대략적인 수치를 x시간,x일을 기준으로 보여줘야 한다.

3.2.1.5.5. 비정상 로그 시각화 및 알림 **(FR-5.1.5)**

- 모델이 판단한 비정상 로그를 사용자에게 공개함과 동시에 알람기능을 제공해야 한다.

3.3. Performance Requirements

- 로그 전송 실패 시, Agent는 최대 3회 재시도를 수행하며, 실패한 로그는 로컬 버퍼에 안전하게 저장되어야 한다.
- Streaming Server는 로그 수신 후 3초 이내에 해당 로그를 실시간 스트리밍 및 AI 분석 서버로 전달해야 한다.
- 설정이 완료되고 유효한 상태인 경우, Agent는 외부 종료 명령 또는 시스템 오류가 없는 한 로그 수집을 중단 없이 지속해야 한다.
- 사용자 필터 조건 변경 요청은 1초 이내에 반영되어야 하며, 조건에 맞는 로그만 전송되어야 한다.

3.4. Design Constraints

- 시스템은 클라우드 환경에서 동작하도록 설계되어야 하며, 사용자 측에서 별도의 서버 인프라나 복잡한 설치 과정 없이 서비스를 시작할 수 있어야 한다.
- 복잡한 초기 설정 없이 로그 수집을 시작할 수 있도록, 설정 주입 및 에이전트 배포 방식은 단순하고 직관적으로 구성되어야 한다.
- 실시간 로그 처리는 WebSocket 기반으로 구현되어야 하며, 사용자별 필터 조건 적용 기능을 포함해야 한다.
- 로그 수집 에이전트는 .log 확장자 여부와 관계없이 로그 파일을 처리할 수 있어야 하며, 다중 파일 수집 및 멀티라인 병합 기능이 지원되어야 한다.
- 로그 전송 및 분석은 비동기 방식으로 이루어져야 하며, 전송 실패 시 로그 손실을 방지하기 위한 로컬 버퍼 저장 및 재시도 정책이 필수적으로 적용되어야 한다.
- 시스템은 멀티 테넌시 구조를 반드시 지원해야 하며, 각 테넌트의 데이터와 설정 정보는 논리적으로 완전하게 분리되어야 한다.
- 로그 전송 및 분석은 비동기 방식으로 이루어져야 하며, 전송 실패 시 로그 손실을 방지하기 위한 로컬 버퍼 저장 및 재시도 정책이 필수적으로 적용되어야 한다.

3.5. Software System Attributes

Reliability

- 로그 수집 중 예외 발생 시에도 시스템 전체가 중단되지 않고 복구 가능해야한다.
- 로그 유실을 방지하기 위해 로컬 버퍼 및 재전송 정책을 적용한다.

- Agent, Streaming Server, AI Server 등 주요 컴포넌트는 독립적인 예외 처리 로직을 갖추어야 한다.

Availability

- 로그 수집 및 전송, 실시간 스트리밍은 항상 동작 가능한 상태를 유지해야 하며, 서버중단 시 자동 재시작 또는 연결 재시도 기능이 제공되어야 한다.
- Kafka 기반 메시지 처리 구조와 웹소켓 연결 유지 전력(WebSocket ping/pong, 백오프 재시도 등)을 통해 연속적인 로그 처리 환경을 제공한다.
- Agent, Streaming Server는 독립적으로 실행되며, 하나의 컴포넌트 장애가 전체 시스템 중단으로 이어지지 않도록 설계되어야 한다.

Security

- 모든 API 통신은 HTTPS 프로토콜을 통해 암호화된다.
- 민감한 설정 변경 및 로그 전송 시에는 JWT 또는 bearer Token 기반의 인증 및 권한 검사를 수행한다.
- 외부 Webhook으로 전송되는 알림은 사용자 설정을 기반으로 하며, 무단 전송을 방지하기 위한 보안 토큰 검사를 수행한다.
- 로그 데이터에 포함된 개인정보 혹은 민감 정보가 있는 경우, 마스킹 혹은 저장제한 정책이 적용될 수 있다.

Maintainability

- Agent, API, Streaming, AI, Frontend 서버는 모듈화된 구조로 개발되어, 개발 서비스 수정 시 전체 시스템에 영향을 주지 않는다.
- 설정 기반 동작 구조를 채택하여 시스템 재시작 없이 주요동작(로그 수집 대상, 필터링 조건 등)을 변경할 수 있다.
- 로그, 상태코드, 모니터링 지표를 수집하여 운영 중 이상 상태에 대한 원인 파악과 디버깅이 가능해야 한다.

Portability

- 로그 수집 대상 경로 및 API URL은 설정 파일 또는 환경 변수로 지정되어 환경 간 이식성이 높아야 한다.
- 웹 프론트엔드는 표준 브라우저 기반으로 작동하며, 별도 설치 없이 접근 가능해야 한다.