졸업 프로젝트

Software Requirements Specification(SRS)
+
System Test Plan(STP)

과목명	졸업프로젝트
팀원	강찬욱, 양승원, 정주연
담당교수	유준범
제출일자	2025.05.24

Table of Contents

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

- 본 시스템의 정식 명칭은 LogMate 이다. LogMate는 에이전트 서버를 통해 다양한 서버 환경에서의 애플리케이션 로그를 수집하고, 이를 스트리밍 서버 및 API 서버로 전송하여 실시간 로그 뷰, 사용자 알림, AI 기반 이상 탐지 등의 기능을 제공한다. 주요 구성 요소로는 Agent Server, Streaming Server, API Server, Frontend Server, AI Server 가 있으며, 각 구성 요소는 독립적으로 동작한다.
- 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 Server: 로그 수집을 담당하는 서버. 설정 주입 후 로그를 Tailing 하여 전송
- API Server: 사용자 인증, 팀/대시보드 설정, 로그 수신 및 저장 등을 담당하는 핵심 API 제공 서버

- Streaming Server: 실시간 로그를 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 프로젝트 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 Server, 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 Server를 제공하고, Web UI 를 통한 동적 설정 주입으로 운영 중에도 무중단 설정 변경이 가능한다.

2.2. Product Functions

- 로그 수집 에이전트: 설정 없이 바로 실행되며, 설정 주입 후 로그 수집 시작
- 실시간 로그 전송 및 스트리밍: 메시지큐를 통해 스트리밍 서버로 로그 전송 후 WebSocket으로 사용자에게 전달
- 직관적인 Web UI 설정: 로그 경로, 파서, 전송 주기, 필터링 조건 등을 브라우저에서 쉽게 구성 가능
- 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.
- Slack 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.

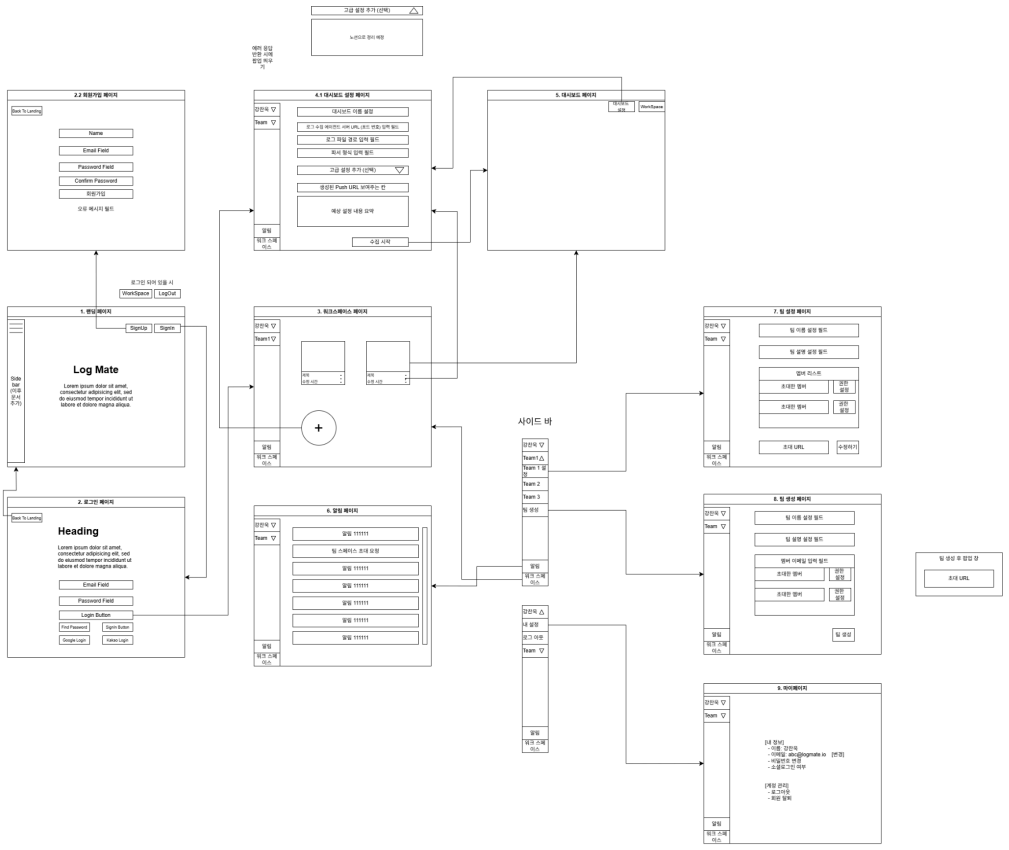
2.5. Assumptions and Dependencies

- 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



Hardware Interfaces

- 해당 사항 없음

Software Interfaces

- 로그 수집 인터페이스 (Agent server -> Streaming server)
  - Agent가 수집한 로그데이터를 실시간으로 streaming server에 전송한다.
  - 프로토콜: HTTP POST(JSON 형식)
- 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
  - Frontend 는 로그 필터 조건을 streaming server에게 전송하고, streaming server는 frontend에게 실시간 로그 메시지, 필터 조건에 부합하는 로그를 전송한다.
  - 프로토콜: WebSocket
- 로그 분석 인터페이스 (Streaming Server -> AI Server)
  - 수신된 로그를 AI 분석 서버로 전달하여 이상 여부를 판단한다.
  - 프로토콜 : 내부 REST API
- 이상탐지 결과 통보 인터페이스 (AI Server -> API Server)
  - 이상 탐지 결과를 API server에 전달하여 저장 및 후속 처리를 가능하도록 한다.
  - 알림 트리거 처리, 대시보드에 표시할 시각화 데이터를 포함한다.
  - 프로토콜 : HTTP POST(JSON 형식)
- 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 : HTTP REST API
  - 보안 : JWT 기반 인증, HTTPS

Communications Interfaces

통신주체	설명
Agent Server → API Server	사용자 설정 요청 수신 및 검증, 설정 갱신 결과를 반환한다.
Agent Server → 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.
Streaming Server → AI Server	웹 로그를 수신하면 AI Server로 전달하고, AI는 이상 여부 결과를 반환한다.
AI Server → API Server	이상 탐지 결과를 API 서버를 통해 사용자 대시보드에 전달한다.
API Server → Frontend	사용자 인증, 대시보드 설정, 알림 설정, 로그 조회 등 모든 요청을 처리한다.
API Server → 외부 Webhook (Slack 등)	사용자 설정에 따라 외부 시스템으로 이벤트 알림을 전송한다.

3.2. Functional Requirements

3.2.1.1. Agent Server

3.2.1.1.1. Agent Initialization

3.2.1.1.1.1. 설정 주입 서버 실행 **(FR-1.1.1)**

- 시스템은 실행 시 설정을 외부에서 주입받기 위한 서버를 스레드 형태로 시작해야 한다.
- 이 서버는 외부 설정 주입 요청을 수신하고, 프로그램 종료까지 대기 상태를 유지한다.

3.2.1.1.1.2. 로컬 설정 불러오기 **(FR-1.1.2)**

- 시스템은 시작 시 로컬에 저장된 설정 파일을 자동으로 로드해야 한다.
- 로드된 설정은 유효성 검사 대상으로 사용되며, 로컬 설정이 존재하지 않는 경우 설정 주입 서버를 통해 외부 설정 수신을 기다려야 한다.

3.2.1.1.1.3. 설정 유효성 검사 **(FR-1.1.3)**

- 시스템은 로컬에서 로드되었거나 외부로부터 수신된 설정에 대해 유효성 검사를 수행해야 한다.
- 설정이 유효할 경우, 에이전트는 내부 상태를 초기화하고 로그 수집 프로세스를 시작해야 한다. 설정이 유효하지 않을 경우, 시스템은 기존의 상태를 유지해야 한다.

3.2.1.1.2. Configuration Injection

3.2.1.1.2.1. 설정 수신 및 재시작 처리 **(FR-1.2.1)**

- 시스템은 웹 UI를 통해 사용자로부터 설정이 전송되면, 현재 수행 중인 로그 수집 작업을 안전하게 중단하고, 수신된 설정에 대해 유효성 및 보안 검사를 수행해야 한다.
- 검사를 통과한 설정은 시스템에 반영되며, 로그 수집은 중단되었던 로그파일 위치부터 재개되어야 한다.

3.2.1.1.2.2. 설정 수신 시 보안 검사 **(FR-1.2.2)**

- 시스템은 설정 전송 요청이 수신되었을 때, 요청의 헤더 또는 포함된 토큰을 기반으로 인증 및 권한 검사를 수행해야 한다.
- 인증에 실패하거나 권한이 없는 요청일 경우, 시스템은 설정 반영을 거부하고 적절한 오류 응답을 반환해야 하며 기존의 상태를 유지해야 한다

3.2.1.1.3. Log Collection

3.2.1.1.3.1. 실시간 로그 수집 **(FR-1.3.1)**

- 시스템은 유효한 설정이 주입되면, 설정된 로그 파일을 대상으로 실시간 tailing을 수행하여 새로운 로그 항목을 감지해야 한다. 감지된 로그는 메모리에 저장되어 후속 처리 대상으로 대기해야 한다

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

- 시스템은 로그 수집을 시작하기 전에, 대상 로그 파일 경로가 접근 권한이 허용된 디렉터리에 속하는지 확인해야 한다.
- 허가되지 않은 경로는 로그 수집 대상에서 제외되어야 한다.

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

- 시스템은 설정된 복수 개의 로그 파일 경로에 대해 병렬로 tailing을 수행해야 하며, 각각의 로그 파일에 동일한 수집 및 파싱 로직을 적용해야 한다

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

- 시스템은 .log확장자가 없는 로그 파일도 수집 대상에 포함해야 하며, 확장자 여부와 관계없이 지정된 파서를 적용하여 로그를 분석해야 한다.

3.2.1.1.3.5. 로그 파싱 **(FR-1.3.5)**

- 시스템은 수집된 로그에 대해 사전에 설정된 파서를 적용하여 로그를 구조화된 형식으로 변환해야 한다.
- 변환된 로그는 필터링 및 전송 전처리를 위한 상태로 유지되어야 한다.

3.2.1.1.3.6. 로그 필터링 **(FR-1.3.6)**

- 시스템은 파싱된 로그에 대해 사전 설정된 조건에 따라 필터링을 수행해야 하며, 필터링을 통과한 로그만 전송 준비 상태로 전환해야 한다.

3.2.1.1.3.7. 멀티라인 로그 병합 **(FR-1.3.7)**

- 시스템은 연속된 로그 라인 중 특정 시작 패턴을 만족하는 경우, 해당 라인들을 하나의 로그 항목으로 병합하여 구성해야 한다.

3.2.1.1.3.8. 멀티라인 병합 실패 처리 **(FR-1.3.8)**

- 시스템은 멀티라인 병합 조건이 충족되지 않거나 병합에 실패한 경우, 해당 로그를 원문 그대로 보존한 채 지정된 실패 태그를 추가하여 전송 준비 상태로 전환해야 한다.

3.2.1.1.3.9. 비형식 로그 처리 **(FR-1.3.9)**

- 시스템은 수집된 로그가 설정된 파서 형식과 일치하지 않는 경우, 해당 로그를 손실 없이 저장하며 지정된 태그를 부여해 구분 가능하도록 처리해야 한다

3.2.1.1.4. Log Exporting

3.2.1.1.4.1. 로그 전송 **(FR-1.4.1)**

- 시스템은 수집된 로그가 전송 준비 상태가 되면, JSON 형식으로 직렬화하여 Streaming Server의 API 엔드포인트로 전송해야 한다

3.2.1.1.4.2. 전송 실패 대응 **(FR-1.4.2)**

- 시스템은 로그 전송 시 서버 응답 오류 또는 네트워크 장애가 발생할 경우, 해당 로그를 로컬 버퍼에 저장하고, 설정된 재시도 정책에 따라 주기적으로 전송을 재시도해야 한다.

3.2.1.1.4.3. 중복 전송 방지 **(FR-1.4.3)**

- 시스템은 각 로그 항목에 고유 식별자를 부여하고, 스트리밍 서버는 이를 이용해 중복 여부를 판별해야 한다.
- 이를 통해 동일 로그의 중복 저장을 방지해야 한다

3.2.1.1.4.4. 보안 전송 및 인증 **(FR-1.4.4)**

- 시스템은 로그 전송 시 HTTPS 암호화 채널을 사용해야 하며, 요청의 헤더 또는 포함된 토큰을 기반으로 인증된 Agent만 로그를 전송할 수 있어야 한다.

3.2.1.1.4.5. 선택적 압축 전송 **(FR-1.4.5)**

- 시스템은 설정에 따라 로그 데이터를 압축하여 전송할 수 있어야 한다.

3.2.1.2. Streaming Server

3.2.1.2.1. Log Streaming

3.2.1.2.1.1. 로그 수신 및 실시간 전달 **(FR-2.1.1)**

- 시스템은 메시지 큐에 로그 메시지가 도착하면, 해당 메시지를 비동기적으로 수신하고, 각 메시지의 소속 사용자 또는 팀 정보를 기준으로 해당 사용자와 연결된 클라이언트들에게만 실시간으로 로그를 전달해야 한다.

3.2.1.2.1.2. 사용자 접속 상태 관리 **(FR-2.1.2)**

- 시스템은 사용자가 WebSocket 연결을 시도할 경우 클라이언트와의 세션을 생성 및 등록해야 하며, 연결 종료 시 해당 세션을 적절히 제거해야 한다.
- 이를 통해 클라이언트는 안정적인 실시간 로그 수신 상태를 유지할 수 있어야 한다.

3.2.1.2.1.3. 사용자별 로그 필터링 **(FR-2.1.3)**

- 시스템은 클라이언트가 필터 조건(예: 특정 키워드, 로그 레벨, 소스 ID 등)을 포함한 WebSocket 메시지를 전송하면, 해당 조건을 각 클라이언트 세션에 저장하고, 이후 브로드캐스트되는 로그 메시지 중 조건에 부합하는 로그만 해당 클라이언트에게 전송해야 한다. 조건은 접속 중인 동안 유지된다.

3.2.1.2.1.4. 압축 데이터 해제 후 저장 **(FR-2.1.4)**

- 시스템은 수신한 로그가 압축된 데이터(Gzip 등)인 경우, 해당 데이터를 해제한 후 내부 처리 파이프라인으로 전달하거나 저장 가능한 형태로 변환해야 한다.

3.2.1.2.1.5. 로그 영속화 **(FR-2.1.5)**

- 시스템은 수신한 로그 메시지를 지정된 데이터 저장소에 영구 저장해야 한다.
- 저장된 로그는 추후 조회 및 분석에 활용될 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 ID를 반환해야 한다. 이메일 중복 여부를 검사해야 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 목록을 반환해야 한다.

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

- 시스템은 사용자가 선택한 팀에 속한 대시보드들의 목록과 각 대시보드의 상태 정보를 반환해야 한다

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

- 시스템은 대시보드 이름, 로그 경로, 전송 주소 등의 설정 정보를 수신하고, 신규 생성 또는 기존 대시보드 설정을 갱신해야 하며 갱신된 정보를 등록된 Agent 서버로 전송해야 한다.

3.2.1.3.2.4. 파싱 룰 등록 **(FR-3.2.4)**

- 시스템은 대시보드별로 정규표현식, 키 매핑 등의 파싱 룰을 등록하고 저장해야 한다.

3.2.1.3.2.5. 파싱 룰 미리보기 처리 **(FR-3.2.5)**

- 시스템은 클라이언트로부터 전달받은 샘플 로그와 파싱 룰을 이용해 파싱 결과를 반환해야 한다.

3.2.1.3.2.6. 필터링 룰 등록 **(FR-3.2.6)**

- 시스템은 로그 레벨, 키워드 포함/제외 등 필터링 조건을 저장하고, 해당 조건에 따라 로그 필터링 처리를 수행할 수 있도록 해야 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.

3.2.1.3.3.3. 팀 초대 **URL** 발급 **(FR-3.3.3)**

- 시스템은 해당 팀에 새 사용자를 초대할 수 있도록 고유한 초대 URL을 생성하고 반환해야 한다

3.2.1.3.3.4. 팀 생성 처리 **(FR-3.3.4)**

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

3.2.1.3.4. 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. 알림 기능

3.2.1.3.5.1. 알림 목록 조회 **(FR-3.5.1)**

- 시스템은 로그인한 사용자에게 발송된 알림 목록을 최신순으로 반환해야 한다.

3.2.1.3.5.2. 알림 읽음 상태 업데이트 **(FR-3.5.2)**

- 시스템은 특정 알림 ID를 읽음 상태로 변경 요청 시, 해당 상태를 저장해야 한다.

3.2.1.3.5.3. 알림 페이지네이션 처리 **(FR-3.5.3)**

- 시스템은 클라이언트가 요청한 페이지 또는 커서 기반으로 알림 목록을 잘라서 반환해야 한다.

3.2.1.3.6. 제**3**자 어플리케이션 연동 알림

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

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

3.2.1.3.6.2. 외부 알림 전송 **(FR-3.6.2)**

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

3.2.1.3.7. 실시간 로그 스트리밍

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

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

3.2.1.3.7.2. 세션별 필터 적용 **(FR-3.7.2)**

- 시스템은 각 사용자 세션에 저장된 필터 조건에 따라 필터링된 로그만 해당 사용자에게 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. 제**3**자 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.

4. System Test Plan

4.1. 개요

4.1.1. 목적

- 이 문서는 Log Mate 시스템의 전반적인 기능이 요구사항에 따라 제대로 동작하는지 검증하기 위한 시스템 테스트 계획을 정의한다.
- 주요 목표는 다음 모듈의 기능이 SRS (소프트웨어 요구사항 명세서) 에 따라 정확히 작동하는지 확인하는 것이다.
    - Agent Server
    - Streaming Server
    - API Server
    - Frontend Server
    - AI Server

4.1.2. 범위

- 사용자 인증, 대시보드 설정, 로그 수집 및 전송, 알림 처리, 외부 연동 등의 전체 기능을 테스트 범위로 포함한다.
- Agent Server, API Server, Streaming Server, Frontend Server, AI Server의 모든 주요 기능에 대해 단위 테스트(Unit Testing)와 통합 테스트(Integration Testing)를 수행한다.
- 성능 테스트는 로그 수집 및 전송의 병목, WebSocket 처리량, DB 응답시간 등을 측정 대상으로 한다.

4.1.3. 참조 문서

- Software Requirements Specification (SRS) - LogMate
- 수업자료: 2025 객체지향개발방법론 (교수자: 유준범)

4.2. 테스트 전략

4.2.1. **Unit Testing**

- 각 기능별 메서드 및 클래스에 대해 JUnit 을 이용한 단위 테스트 수행
- 각 유닛 함수는 독립적으로 테스트되며, 외부 의존성은 mocking 처리

4.2.2. **Integration Testing**

- 모듈 간 통신 검증
- 실제 구성 요소를 통합한 상태에서 동작 여부 확인

4.2.3. **System Testing**

- 전체 시스템을 단일 제품으로 간주하여 요구사항을 기준으로 테스트
- 로그인, 로그 수집, 필터링, 알림, 외부 연동 등의 전체 사용자 시나리오 수행

4.2.4. **Performance Testing**

- 로그 수집 속도, WebSocket 처리량, Kafka 소비율, DB 쓰기/읽기 속도 측정
- Apache JMeter 등의 성능 측정 도구를 이용

4.3. 테스트 환경

4.3.1. **Agent Server** 테스트 환경

- OS 별 테스트: Linux, Windows, MacOS
- 로그 파일 접근 테스트: 다양한 확장자(.log, .txt, .out, 확장자 없음 등)

4.3.2. 이 외 공통 테스트 환경

- Docker Compose 기반 테스트 환경 (Kafka, Spring, WebFlux, MySQL 포함)
- 운영체제: Ubuntu
- 브라우저: Chrome, Edge 최신 버전

4.4. 테스트 케이스

4.4.1. Unit Test

UT ID	테스트 항목	절차	예상 결과	관련 FR
UT-001	설정 주입 서버 실행	에이전트 실행 후 설정 주입 서버 확인	설정 서버가 대기 상태로 실행됨	FR-1.1.1
UT-002	로컬 설정 불러오기 성공	에이전트 실행 시 설정 파일 존재	설정 내용이 로드되어 상태 초기화 진행	FR-1.1.2
UT-003	로컬 설정 불러오기 실패	설정 파일이 존재하지 않음	설정 주입 대기 상태 진입	FR-1.1.2
UT-004	설정 유효성 검사 성공	유효한 설정 파일 로드 또는 주입	로그 수집 시작	FR-1.1.3
UT-005	설정 유효성 검사 실패	잘못된 설정 파일 주입	로그 수집 시작 안됨, 대기 유지	FR-1.1.3
UT-006	설정 수신 및 재시작 성공	`POST /config` 호출로 새 설정 전송	기존 작업 중단, 새 설정 반영, 로그 재시작	FR-1.2.1
UT-007	설정 수신 유효성 실패	필수 값 누락된 설정 전송	`400 Bad Request` 반환, 설정 무시	FR-1.2.1
UT-008	설정 보안 검사 성공	유효한 토큰 포함 요청 전송	설정 반영 허용	FR-1.2.2
UT-009	설정 보안 검사 실패	없는 토큰 또는 권한 부족 요청	403 응답 반환, 설정 반영 안됨	FR-1.2.2
UT-010	실시간 로그 수집 시작	유효 설정 적용 후 로그 생성	로그가 메모리에 저장됨	FR-1.3.1
UT-011	로그 파일 접근 허용	허용된 경로 설정 후 감시 시작	정상 tailing 수행	FR-1.3.2

UT-012	로그 파일 접근 거부	비허용 디렉토리 설정	로그 수집 실패	FR-1.3.2
UT-013	다중 로그 파일 수집	여러 경로 설정	각 파일 병렬 tailing	FR-1.3.3
UT-014	확장자 없는 파일 수집	`syslog`, `output` 등 등록	정상 수집 및 파싱	FR-1.3.4
UT-015	로그 파싱 성공	파서 설정 후 로그 수집	파싱된 구조화 로그 생성	FR-1.3.5
UT-016	로그 필터링 적용	키워드/레벨 설정 후 수집	필터 통과 로그만 전송 대상	FR-1.3.6
UT-017	멀티라인 로그 병합	StackTrace 포함 로그 수집	병합된 JSON 생성	FR-1.3.7
UT-018	멀티라인 병합 실패 처리	패턴 불일치 StackTrace 수집	원문 보존, 태그 추가 전송	FR-1.3.8
UT-019	비형식 로그 처리	파서 미적용 로그 수집	태깅하여 저장	FR-1.3.9
UT-020	로그 전송 성공	수집 후 JSON 직렬화 전송	Streaming 서버 수신 확인	FR-1.4.1
UT-021	로그 전송 실패 대응	서버 오류 시 전송	버퍼 저장 후 재시도	FR-1.4.2
UT-022	중복 전송 방지	UUID 중복 로그 전송	중복 저장 안됨	FR-1.4.3
UT-023	보안 전송 확인	HTTPS + Bearer 토큰 포함	정상 인증 및 전송	FR-1.4.4
UT-024	압축 전송 옵션	Gzip 옵션 활성화 후 전송	`Content-Encoding` 확인됨	FR-1.4.5
UT-025	로그 수신 및 실시간 전달	Kafka 토픽에 로그 메시지를 발행한 후, 해당 팀 클라이언트 WebSocket 수신 여부 확인	해당 팀에 연결된 클라이언트만 로그 수신	FR-2.1.1

UT-026	사용자 접속 상태 관리 - 연결	WebSocket 연결 요청을 보내고 세션 등록 로그 확인	세션이 서버에 등록되고, 로그 수신 가능 상태가 됨	FR-2.1.2
UT-027	사용자 접속 상태 관리 - 종료	WebSocket 연결을 종료한 후, 세션 제거 여부 확인	세션이 정상 제거되고 더 이상 로그 수신되지 않음	FR-2.1.2
UT-028	사용자별 로그 필터링	클라이언트가 필터 조건(WebSocket 메시지) 전송 → 조건에 맞는 로그만 수신되는지 확인	조건에 부합하는 로그만 수신됨	FR-2.1.3
UT-029	압축 데이터 해제 후 저장	Gzip으로 압축된 로그를 Kafka에 발행 → 서버에서 해제 및 처리되는지 확인	압축 해제 후 정상적으로 처리 및 저장	FR-2.1.4
UT-030	로그 영속화	로그 수신 후 DB 또는 파일 시스템에 저장 확인	수신 로그가 데이터 저장소에 저장됨	FR-2.1.5
UT-031	로그인 성공	이메일/비밀번호 입력	토큰 발급 및 대시보드 이동	FR-3.1.1
UT-032	로그인 실패	잘못된 정보 입력	인증 실패 메시지 표시	FR-3.1.1
UT-033	회원가입 성공	정상 입력 후 등록 요청	회원 등록 및 로그인 이동	FR-3.1.2
UT-034	회원가입 실패	중복 이메일로 요청	오류 메시지 표시	FR-3.1.2
UT-035	로그아웃 성공	로그아웃 버튼 클릭	토큰 삭제 후 로그인 화면 이동	FR-3.1.3
UT-036	팀 목록 조회	토큰 포함 API 호출	팀 리스트 정상 반환	FR-3.2.1
UT-037	대시보드 목록 조회	팀 선택 후 목록 요청	대시보드 목록 + 상태 반환	FR-3.2.2
UT-038	대시보드 생성 및 수정	이름/경로/주소 입력 후 저장	대시보드 갱신 완료	FR-3.2.3
UT-039	팀 생성	팀 이름, 설명, 멤버 이메일 및 권한 입력 후 저장	팀이 생성되고 사용자에게 할당	FR-4.3.4

UT-040	사용자 기본 정보 조회	마이페이지 진입 시 사용자 정보 표시 확인	API 서버에서 받은 정보가 정확히 화면에 출력	FR-4.4.1
UT-041	이메일/패스워드 변경	이메일 또는 패스워드 변경 입력 후 저장 버튼 클릭	입력된 정보가 저장되고 사용자 정보가 갱신됨	FR-4.4.2
UT-042	계정 관리	로그아웃 또는 탈퇴버튼 클릭	로그아웃 또는 탈퇴 처리가 완료됨	FR-4.4.3
UT-043	알림 목록 조회	알림 화면 진입 후 목록 불러오기 확인	알림이 최신순으로 정렬되어 화면에 표시	FR-4.5.1
UT-044	알림 유형 구분	일반 알림과 중요 알림 시각적 구분 확인	중요 알림은 강조된 스타일로 표시	FR-4.5.2
UT-045	알림 읽음 처리	알림 클릭시 상태 변경	알림 상태가 읽음으로 바뀌고 API에 반영	FR-4.5.3
UT-046	알림 스크롤 페이징	스크롤 혹은 더 보기 클릭 후 다음 알림 페이지 요청	다음 알림 목록이 정상적으로 출력됨	FR-4.5.4
UT-047	외부 알림 전송 설정	Webhook URL 입력 후 저장	URL이 저장되어 외부 전송 준비 완료	FR-4.6.1
UT-048	알림 발생 시 외부 전송	이벤트 발생 후 Webhook으로 알림 전송 확인	외부 채널로 알림이 전송, 실패시 재시도 혹은 실패정보 기록	FR-4.6.2
UT-049	실시간 로그 수신	스트리밍 서버와 연결된 채널로 로그 수신 확인	로그가 실시간으로 UI에 출력됨	FR-4.7.1
UT-050	필터 적용	필터조건(키워드, 레벨 등)을 입력 후 로그 확인	필터링된 로그만 화면에 표시	FR-4.7.2
UT-051	이상탐지	비정상 웹 로그 시나리오에서 탐지 동작 확인	AI 모델이 비정상 로그를 탐지하고 표시	FR-5.1.1
UT-052	실시간 로그 목록 표시	탐지된 정상/비정상 로그가 대시보드에 표시되는지 확인	분류된 로그가 실시간으로 시각화됨	FR-5.1.2
UT-053	상태코드 필터링	2xx, 4xx 등 상태코드 기준 필터링 적용 후 확인	필터링 조건에 맞는 로그만 시각화	FR-5.1.3
UT-054	IP 기반 사용자 집계	웹 로그의 IP 기준으로 시간 단위 사용자 수 표시 확인	시간별 사용자 수가 시각화되어 제공	FR-5.1.4
UT-055	비정상 로그 알림	비정상 로그 탐지 시 대시보드 및 외부 채널로 알림 전송 확인	비정상 로그 발생 시 알림이 실시간으로 전송됨	FR-5.1.5

4.4.2. Integration Test

IT ID	테스트 항목	절차	예상 결과	관련 FR
IT-001	설정 주입 → Agent 설정 적용 확인	1. API 서버에서 대시보드 설정 변경 2. Agent에 설정 주입 요청	Agent 로그 수집 설정이 변경되며 재시작됨	FR-1.2.1, FR-3.2.3
IT-002	로그 수집 → 전송 → Kafka 적재	1. Agent에서 로그 파일 생성 2. 실시간 수집 3. Kafka 전송	Kafka 토픽(logs)에 JSON 로그 메시지 도착	FR-1.3.x, FR-1.4.1
IT-003	Kafka → Streaming Server → WebSocket	1. Kafka에 로그 메시지 직접 발행 2. Streaming Server가 수신 후 WebSocket 클라이언트에 전달	실시간 로그가 클라이언트에 전송됨	FR-2.1.1, FR-2.1.2
IT-004	WebSocket 필터 적용	1. 클라이언트가 특정 키워드 필터 등록 2. 필터와 맞는 로그만 수신	조건에 맞는 로그만 표시됨	FR-2.1.3
IT-005	비정상 로그 탐지 → 알림 발생	1. AI 서버가 Kafka 로그 수신 2. 이상 탐지 후 알림 발생 요청 전송	알림 서버에 이상 알림 생성됨	FR-5.1.1, FR-5.1.5
IT-006	이상 알림 → Webhook 연동	1. 알림이 Webhook 설정된 Slack/Discord URL로 전송	외부 채널에 메시지 수신됨	FR-3.6.2, FR-4.6.2
IT-007	로그인 → 대시보드 설정 → Agent 동기화	1. 유저 로그인 후 대시보드 설정 2. Agent에 설정 전송 3. 로그 수집 시작됨	전체 흐름이 정상 작동하며 로그가 Web에 표시됨	FR-3.1.1 ~ 3.2.4, FR-1.2.x, FR-1.3.x
IT-008	로그 수집 → 필터 → 이상탐지 → 알림 전파	1. 로그 생성 → 수집 → 필터 → AI 이상 탐지 2. Web/Slack으로 알림 전파	전체 파이프라인 동작 확인	FR-1.x ~ FR-5.x 종합

4.4.3. System Test

ST ID	테스트 항목	절차	예상 결과	관련 FR
ST-001	신규 사용자 회원가입 및 로그인	1. 이메일, 비밀번호, 이름 입력 2. 로그인 시도	회원가입 성공, 로그인 후 토큰 발급 및 대시보드 이동	FR-3.1.1, FR-3.1.2, FR-4.1.1
ST-002	대시보드 생성 및 설정	1. 팀 선택 후 2. 대시보드 setting 입력	신규 대시보드 생성, Agent 설정 반영됨	FR-3.2.3, FR-4.2.4
ST-003	로그 수집 및 실시간 표시	1. 설정된 경로에 로그 발생 2. Web에서 실시간 확인	로그가 실시간 UI에 도착하고 시각화됨	FR-1.3.x, FR-4.7.1
ST-004	로그 필터 적용	1. 필터링 키워드 입력 2. 로그 스트리밍 시작	필터 조건을 만족하는 로그만 표시됨	FR-2.1.3, FR-4.7.2
ST-005	비정상 로그 탐지 및 경고	1. AI 탐지 모델이 이상 로그 감지	대시보드 경고 발생, 알림 리스트에 추가	FR-5.1.1, FR-5.1.5
ST-006	Slack 외부 알림 연동	1. Webhook URL 등록 2. 이상 탐지 트리거 발생	Slack 채널에 비정상 로그 알림 수신	FR-3.6.2, FR-4.6.2
ST-007	로그 전송 실패 시 재시도	1. 네트워크 차단 후 로그 수집 2. 연결 복구	로그가 재전송되어 Kafka로 도착함	FR-1.4.2
ST-008	확장자 없는 로그 수집	1. output 파일 생성 후 로그 발생	해당 파일 로그도 수집되고 파싱됨	FR-1.3.4
ST-009	팀 초대 및 권한 설정	1. 초대 링크 생성 2. 사용자 가입 후 팀 접속	권한에 맞는 UI 기능 노출됨	FR-3.3.3, FR-4.3.3
ST-010	사용자 탈퇴 처리	1. 마이페이지 > 탈퇴 버튼 클릭	사용자 정보 삭제 및 로그아웃 처리됨	FR-3.4.3, FR-4.4.3

4.4.4. Performance Test

PT ID	테스트 항목	절차	예상 결과	검증 지표
PT-001	로그 수집 처리량	1초에 1000줄 이상 로그가 append되는 파일 tail 시 수집 성능 확인	수집 누락 없이 Kafka 전송 완료	수집률 ≥ 99%
PT-002	WebSocket 동시 접속 처리	1000개 클라이언트가 동시에 WebSocket 연결 후 로그 수신	모든 클라이언트에게 끊김 없이 로그 브로드캐스트	평균 지연 ≤ 1초
PT-003	Kafka 소비 처리량	초당 1만 메시지를 Kafka로 입력 → WebFlux로 소비	소비 지연 없이 스트리밍 처리	소비율 ≥ 95%
PT-004	API 응답속도 측정	로그인/회원가입/대시보드 생성 등 주요 API 호출 반복	평균 응답 시간 500ms 이하	응답시간 ≤ 500ms
PT-005	로그 전송 압축 성능	압축 미사용 vs Gzip 압축 옵션 활성화 후 전송	Gzip 사용 시 네트워크 사용량 절감	전송량 30%↓, 처리시간 비슷
PT-006	로그 병합 처리 비용	멀티라인 로그(10줄 이상)가 지속적으로 발생하는 상황 측정	병합 실패 없이 처리 완료	CPU 사용률 < 70%
PT-007	DB 로그 저장 지연	ElasticSearch 에 초당 1000건 이상 로그 삽입 시	저장 누락 없이 처리 완료	평균 쓰기 시간 ≤ 100ms
PT-008	Agent 재시작 회복 시간	로그 수집 중 Agent 강제 종료 → 5초 뒤 재시작	재시작 후 중단 시점부터 재수집 성공	누락 없음
PT-009	AI 서버 실시간 분류 처리량	초당 100건 이상 로그가 AI 서버에 전달됨	탐지 실패율 ≤ 1%, 지연 ≤ 2초	탐지 성공률 ≥ 99%

4.5. 테스트 진행 절차

4.5.1. 테스트 준비 단계

4.5.1.1. 테스트 환경 초기화

- 각 모듈 환경 구성 및 부팅 확인
- Kafka, MySQL 등 외부 의존 모듈 연결 확인

4.5.1.2. 테스트 데이터 세팅

- 회원가입된 테스트 유저, 팀, 대시보드 데이터 생성
- 로그 수집용 가상 로그 파일 및 샘플 로그 생성 스크립트 준비
- Webhook, 필터링 조건 등 주요 설정 시나리오 사전 구성

4.5.1.3. 테스트 툴 준비

- Postman, WebSocket Client, JMeter, JUnit 등 툴의 사전 설치 및 구성

4.5.2. 테스트 실행 단계

4.5.2.1. **Unit Test**

- 각 기능별 모듈 또는 클래스 단위에서 독립적인 테스트 수행

4.5.2.2. **Integration Test**

- 각 모듈에서 정상 동작 후 서로 호출 및 반응이 예상대로 이루어지는지 확인

4.5.2.3. **System Test**

- 최종 배포와 유사한 환경에서 사용자의 전체 사용 흐름 시나리오를 수행

4.5.2.4. **Performance Test**

- 전체 구성 요소의 부하 테스트 수행

4.5.3. 테스트 결과 기록 방법

4.5.3.1. 결과 기록 방식

- 각 테스트 케이스별 결과를 Pass/Fail/Skipped 상태로 기록
- 테스트 일지 형식으로 Notion 페이지에 작성

4.5.3.2. 스크린샷/캡처

- Web UI 기반 오류나 알림 표시 관련 기능은 스크린샷 첨부

4.5.3.3. 로그 저장

- 각 모듈의 콘솔 로그와 API 응답 로그는 파일로 저장

4.5.4. 결함 보고 및 수정 절차

4.5.4.1. 결함 보고서 포맷

항목	내용
이슈 ID	BUG-202406-UT1
제목	로그 필터 적용 안됨
발견 모듈	Streaming Server
발생 조건	필터에 error 키워드 설정 후 정상 로그도 수신됨
예상 동작	error 키워드 포함 로그만 수신되어야 함
실제 동작	모든 로그가 전달됨
심각도	High
스크린샷/로그	stream.log 첨부

테스트자	강찬욱
상태	Open / Fixed / Retest / Closed

4.5.4.2. 수정 및 재테스트 절차

- 개발자가 수정 후 hotfix 브랜치에 푸시
- 배포 후 테스트 케이스를 다시 수행
- 문제가 해결되었을 경우 Closed 아니라면 Reopen 후 반복

4.6. 테스트 결과 산출물

- 테스트 케이스 문서
- 실행 로그 및 서버 콘솔 로그
- 화면 캡처본 및 API 응답
- 성능 측정 그래프
- 테스트 요약 리포트

4.7. 테스트 승인 기준

- 주요 기능 단위, 통합 테스트 모두 통과
- 성능 기준 만족
- 시스템 테스트 모두 통과