프로젝트 계획서

1. 프로젝트 개요

1.1 프로젝트 명

- 에러 리포트 자동화 시스템

1.2 프로젝트 목적

- 다양한 이벤트들을 실시간으로 로그 분석 서버로 전송하고, 이를 **AI** 기반 시스템이 즉각적으로 분석하여 정상적인 접근인지, 혹은 잠재적인 사이버 공격인지를 판별하는 시스템 구축
- AI가 분석하여 공격 또는 에러로 판단한 모든 비정상적인 활동에 대해 자동으로 상세 리포트를 생성. 잠재적인 공격 시도나 시스템 오류를 탐지하는 즉시, 해당 이벤트의 상세 정보를 담은 보고서를 자동으로 생성

1.3 프로젝트 기간

- 2025.10.x ~ 2025.10.x
- 총 기간 : 8일

2. 팀 구성

이름	역할	담당업무
김**xx**	**PM, DevOps**	프로젝트 일정 관리 Kubernetes, Docker Image 제작, 관리 IP, Domain, SSL 관리 GCP VM NFS Server 관리 방화벽 설정
서**xx**	**Frontend (메인)** **Backend (보조)**	Client 페이지 구현(화면 구성 및 API 연동) Figma 와이어프레임 구성 프론트 프로젝트 세팅 백엔드 일부 기술 지원 및 오류 해결 협업
최**xx**	**Frontend**	Figma 와이어프레임 구성 관리자 페이지 구현(화면 구성 및 API 연동) 시연 영상 및 피피티 제작 개발 명세서 작성

박**xx**	**Backend**	회원 계정 관련 구현 보고서 처리 Flask서버 구현 DB 설계 및 구현
김**xx**	**Backend**	Client 페이지 CRUD API 구현 관리자 페이지 CRUD API 구현 Flask 연동 백엔드 관련 문서 작성 백엔드 부분 피피티 제작

3. 기술 스택

Backend

- Framwork
    - Springboot 3.5.3
    - Spring security
    - Spring JPA
    - Spring Web (Rest API)
    - Spring Data Redis
- 언어: Java
- DB, 저장소
    - MariaDB 11.4
    - Redis Latest
    - NFS (Network File System)
- 통신: RestClient
- 보안: JWT Token 인증 기반
- 빌드: Maven

Frontend

- Framework: React 19
- 언어: JavaScript
- 상태 관리: Zustand
- 디자인: Figma
- 스타일링: CSS, Lucide React, react-datepicker
- 데이터 시각화: Chart.js + react-chartjs-2, Recharts
- 라우팅 라이브러리: React Router v7
- Lint 도구: ESLint
- 빌드: Vite

DevOps

- 배포: kubernetes 1.32, gcp vm
- kubernetes cluster:
    - node : 4
    - cpu : 8
    - memory : 16gb
    - control plain location : us-central1-a
- 이미지:
    - file server
    - mariadb server
    - ai server
    - api server
    - redis server
    - client server
- HTTPS
    - IP : VPC Network
    - Domain : Square Space
    - SSL : GTS CA WR3
- Storage
    - GCP VM Ubuntu Server 22.04 NFS Server
- Health Check
    - kubernetes backend config
    - nginx /health endpoint

4. 일정 계획

2조	**(Backend)** 김**xx**	**(Frontend(메인), Backend(보조))** 서**xx**	**(PM, DevOps)** 박**xx**	**(Backend)** 김**xx**	**(Frontend)** 최**xx**
1일차 (7월 4일 금)	Entity 설계서 고도화 진행 초기 환경 세팅 생성된 maria db 및 docker 연동 문서 crud 기능 생성 중	프론트 레포 생성 및 파일 생성 피그마 와이어프레임 제작 완 로그인 페이지 생성 doc-viwer페이지 기능 추가중 { sidebar mode 수정중 , DocCard, 다른 기능}	개발 기능 총 정리 역할분담, 일정 데드라인 설정 서버 예시 구성 파일 서버 kubernetes nfs 테스트	SpringBoot Project 생성, 회원 관리를 위한 Jwt, Redis를 추가하고 이를 docker-compose로 구성하여 편리하게 서비스를 시작할 수 있도록 MariaDB와 Redis를 구성. User Service, Contoller 구성하여	ERD 제작 관리자 페이지 UI 작성(Figma) 관리자 대시보드 제작

				회원 로그인 기능 추가	
2일차 (7월 7일 월)	카테고리 DB 분리 오류 예외 처리 구현 중 문서 api 오류 보완 문서 api 코드 리팩토링 중	토큰값이 undefined형태로 저장되는 오류발생 - 해결 로그인 연동 등록 연동 zustand 구성 header 텍스트 동적으로 변경 열람기능 50% 완성	kubernetes docker hub 이미지 빌드, 배포 테스트 ingress nginx domain route 테스트, ssl 적용 nfs 파일 공유 테스트 nginx conf 제작, api 접근 테스트 metallb 적용 테스트	관리자 계정이 사용하는 기능 (유저 전체 조회, 유저 상세 조회)구현, Flask 서버에 Spring가 전달할 json형식의 데이터를, AI 모델이 LangChain과 LangGraph로 분석하여 처리할 수 있도록 방법을 찾고 학습 중.	관리자 회원 목록 조회 및 개별 조회 ui 수정 회원 전체 조회 및 개별 조회 api 연동 상태관리 구성(zustand)
3일차 (7월 8일 화)	에러 리포트 조회, post api 구현 flask 로직 중복 호출 에러 발생 보고서 물리 저장 경로 에러 발생	새로고침 제작 디자인수정 권한에 따른 보고서 열람 확인 도중 에러 발생 백엔드 Resolved [java.nio.file.FileSystemException: /mnt: Read-only file system] 에러 발견 및 수정중.. + 열람 기능도 에러발생(같은 이유)	GKE 서버 설정 -> 실패 -> 지역바꿔서 재시동 NFS 방화벽 테스트 GKE 생성 (standard 모드), NFS 연동 테스트	Flask 서버에서 사용할 언어 모델과 프롬프트 형식을 결정하고, Spring 서버로부터 메시지를 받는 로직을 구현. 메시지를 전달받은 다음 모델에서 결과 보고서를 작성하는 로직 구현	상태 관리 리팩토링 완료 로그인할 때 회원 정보 로드 로직 'CEO'만 관리자 페이지에 접근 가능 로직 추가 에러 리포트 전체 목록 조회 api 연동 전체적인 ui 수정
4일차 (7월 9일 수)	에러 리포트 데이터 샘플 추가 flask 연동 테스트 -> 아직 전체적인 확인 및 수정 전체 코드 리팩토링	카테고리별 이미지 매핑 client 페이지 연동 완료 날짜 필터링 기능 수정 문서 등록 기능 수정 코드 리팩토링	전체 기능 교통정리 ingress 적용 테스트 -> 성공 ssl 적용 테스트 -> 실패 (managedCertificate 오류) -> managedCertificate 해결 후, CORS 적용 도메인 접근 테스트 (google 도메인) -> 성공	Spring 서버와 Flask 서버 간 연동 Flask에서 DB의 error_report 테이블을 생성하고, 보고서에 따라 record를 테이블에 저장하도록 구현	수정된 error_report 엔티티에 맞춰 출력 컬럼 수정 추가된 GET 매핑 연동 관리자 대시보드 -상태별 & 카테고리별 에러 리포트 통계 추가 에러 리포트 목록 페이지 -일별 에러 리포트 카운트 그래프 추가 에러 리포트 상세 페이지 -상태 변경 및 comment 입력

					기능 추가 -리포트 삭제 기능 추가(Soft Delete)
5일차 (7월 10일 목)	Boot와 Flask 연동 테스트 완료 에러 리포트 로그 제외 완료 전체 코드 주석 추가 로그 full url 추가	백엔드 직렬화 에러 수정(오전) 헤더 UI 수정 사이드바 토글추가 전반적인 UI수정 관리자 페이지 헤더 디자인 리팩토링하여 통일	SSL 발급 성공, CORS Origin 해결 File Server Nginx 설정 변경 -> 성공 Frontend, Backend, ai 이미지 빌드 테스트 -> 성공 GKE Loadbalance health check 추가 nginx health chekck endpoint 추가, 오류 디버깅 redis 서버 구성, 백엔드 연동 테스트 전체 서버 배포 테스트 전체 기능, 연결 테스트 노드 풀 늘리기	Boot와 Flask 연동 테스트 완료 Flask에서 Spring으로부터 전달받은 요청을 처리할 때 쿼리 파라미터를 추가하여 보고서 판별 기준을 좀 더 엄격하고 정확하게 처리하도록 함	에러 리포트 상태 및 코멘트 저장, 삭제 구현 -(PATCH / DELETE 매핑 테스트 완료) 공격 에러 리포트 페이지 -공격 에러 리포트 조회 -일별 카운트 선 그래프 시각화 구현 관리자 페이지 전체적인 ui 수정 마무리 에러 리포트 상세 페이지 -Base Url 추가하여 File Path 경로 수정
6일차 (7월 11일 금)	flask url 프로퍼티스로 분리, 파일 nfs 다운로드 경로 수정	디자인 수정 및 리팩토링 및 다크모드 제거 등록 이후 alert 추가 및 자동 새로고침 추가 아이콘 매핑 에러 수정	Frontend 파일 경로 설정 AI 서버 DB 연결 오류 해결 nginx cors preflight 설정 페이지 전체 디버깅	파트 통합 후, 테스팅 작업	관리자 디렉토리 코드 리팩토링 및 주석 추가 백엔드 서버 연동테스트 에러 리포트 테이블 간격 ui 조정 에러 리포트 전체 조회 매핑 연동 시연 영상 및 PPT 프론트엔드 파트 제작

5. 주요 기능

사용자 경험

- 실시간 위협 인지 및 대응**:** 시스템은 문제가 발생한 순간 바로 알림을 받고 대응.
- 직관적인 비정상 활동 보고서**:** AI가 공격 또는 에러로 판단한 모든 비정상 활동에 대해 상세한 보고서가 자동으로 생성됩니다. 이 보고서는 관련 이벤트의 상세 정보와 **AI** 분석 결과를 명확하게 제공하여, 사용자가 문제의 원인을 빠르고 정확하게 파악하고 적절한 조치를 취할 수 있도록 돕습니다.
- 보안 전문가의 부담 경감**:** 복잡한 로그 분석과 잠재적 위협 탐지 작업 자동화

기술

- 실시간 로그 분석 : Langchain 을 이용한 실시간 로그 분석
- 에러 리포트 자동화 : 로그에 관한 에러 판별, 리포트 자동 작성

6. 기대 효과

- 문서화 시간 단축 : 문서화 하는 시간 2시간 -> 0초로 단축
- 미해결 에러에 대해 즉각 반응 : 문서화 시간에 줄어듦에 따라 미해결 에러에 관해 즉시 반응 가능
- 강화된 보안 태세**:** AI 기반의 실시간 분석을 통해 기존에는 놓칠 수 있었던 미묘한 패턴의 사이버 공격 시도나 내부 시스템 오류를 조기에 탐지하여 선제적으로 대응 가능
- 피해 최소화 및 비용 절감**:** 데이터 유출, 서비스 중단 등으로 인한 금전적 및 비금전적 피해를 최소화
- 신뢰도 및 신용도 향상**:** 시스템의 보안 및 안정성이 강화됨에 따른 신뢰도 향상