AiGov 설계 문서 (2025.09.22 Draft)

1. 프로젝트 개요

1.1 배경 및 목적

기업에서 ChatGPT, Claude, Gemini 등 생성형 AI 서비스를 안전하게 활용하기 위한 보안 거버넌스 솔루션 개발. Proxy Server 방식으로 통합 Web UI를 구축하고, 사용자 프롬프트를 보안 정책에 따라 필터링하여 허용된 AI 서비스로 전달하며, 내부 LLM을 통해 응답을 정리/마스킹 처리하여 사용자에게 제공.

1.2 시장 동향 분석

- **Shadow AI 사용률**: 전체 오피스 근로자의 50%가 승인되지 않은 AI 사용
- **데이터 유출 증가**: Shadow AI로 인한 기업 데이터 노출이 30배 증가
- **시장 규모**: AI 거버넌스 시장이 2025년 0.34억 달러에서 2030년 1.21억 달러로 28.80% CAGR 성장 예상

1.3 핵심 해결 과제

1. 비인가 AI 서비스 사용 (Shadow AI) 탐지 및 차단
2. 민감 정보 유출 방지를 위한 프롬프트 필터링
3. AI 사용 현황 모니터링 및 거버넌스 정책 관리
4. 컴플라이언스 요구사항 자동 대응

2. 솔루션 아키텍처

2.1 아키텍처 구성 방식

도메인 기반 마이크로서비스 구조를 채택하여 각 비즈니스 도메인별로 독립적인 개발/배포가 가능하도록 설계.

2.2 핵심 설계 원칙

- **독립적 개발/배포**: 각 도메인별 팀이 독립적으로 개발 가능
- **확장성**: 모듈별 독립적 스케일링 및 기술 스택 선택
- **장애 격리**: 한 모듈 장애가 전체 시스템에 미치는 영향 최소화
- **비즈니스 적합성**: AI 거버넌스의 각 도메인이 명확히 분리됨

3. AiGov 솔루션 프로젝트 구조

3.1 전체 프로젝트 구조

```
AiGov/
├── 📦 PromptGate/                    # 핵심 프록시 & 프롬프트 필터링
```

```
│   ├── frontend/
│   │   ├── public/
│   │   │   ├── assets/              # 정적 자산
│   │   │   │   ├── images/          # 회사 로고, 파비콘 등
│   │   │   │   │   ├── logos/       # 회사 로고 (logo.png, logo.svg)
│   │   │   │   │   ├── icons/       # 아이콘 파일들
│   │   │   │   │   └── branding/    # 브랜딩 이미지들
│   │   │   │   ├── fonts/           # 폰트 파일들
│   │   │   │   └── documents/       # 정적 문서 (PDF 등)
│   │   │   ├── favicon.ico
│   │   │   ├── manifest.json
│   │   │   └── index.html
│   │   ├── src/
│   │   │   ├── assets/              # 개발용 자산
│   │   │   │   ├── images/          # 컴포넌트용 이미지
│   │   │   │   ├── styles/          # SCSS/CSS 파일들
│   │   │   │   └── icons/           # SVG 아이콘 컴포넌트
│   │   │   ├── components/
│   │   │   │   ├── common/          # 공통 컴포넌트
│   │   │   │   ├── prompt/          # 프롬프트 관련 컴포넌트
│   │   │   │   ├── filter/          # 필터링 관련 컴포넌트
│   │   │   │   └── layout/          # 레이아웃 컴포넌트
│   │   │   ├── pages/
│   │   │   │   ├── Chat/            # 채팅 인터페이스
│   │   │   │   ├── Dashboard/       # 사용자 대시보드
│   │   │   │   └── Settings/        # 설정 페이지
│   │   │   ├── hooks/               # React 커스텀 훅
│   │   │   ├── services/            # API 호출 서비스
│   │   │   ├── utils/               # 유틸리티 함수
│   │   │   ├── store/               # 상태 관리 (Redux/Zustand)
│   │   │   └── types/               # TypeScript 타입 정의
│   │   ├── package.json
│   │   ├── tsconfig.json
│   │   ├── tailwind.config.js
│   │   └── vite.config.ts           # Vite 설정
│   ├── backend/
│   │   ├── src/
│   │   │   ├── controllers/         # API 컨트롤러
│   │   │   ├── services/            # 프록시 서비스, 필터링 엔진
│   │   │   ├── middleware/          # 인증, 로깅 미들웨어
│   │   │   ├── models/              # 데이터 모델 (ORM)
│   │   │   ├── routes/              # API 라우트
│   │   │   ├── utils/
│   │   │   ├── types/
│   │   │   └── app.ts               # Express 앱 설정
│   │   ├── config/                  # 서버 설정
│   │   │   ├── database.ts          # DB 연결 설정
│   │   │   ├── redis.ts             # Redis 설정
│   │   │   ├── qdrant.ts            # Qdrant 설정
│   │   │   └── env.ts               # 환경변수 설정
│   │   ├── package.json
│   │   ├── tsconfig.json
│   │   └── nodemon.json
│   ├── database/                    # 데이터베이스 구성
│   │   ├── postgresql/              # PostgreSQL 관련
│   │   │   ├── migrations/          # DB 마이그레이션 파일
│   │   │   ├── seeds/               # 초기 데이터
│   │   │   ├── schemas/             # 스키마 정의
│   │   │   ├── queries/             # 자주 사용하는 SQL 쿼리
│   │   │   └── backup/              # 백업 스크립트
│   │   ├── redis/                   # Redis 설정
│   │   │   ├── config/              # Redis 설정 파일
```

```
│   │   │   ├── scripts/              # Redis Lua 스크립트
│   │   │   └── init/                 # 초기 설정 스크립트
│   │   └── qdrant/                   # Qdrant 벡터 DB
│   │       ├── collections/          # 컬렉션 설정
│   │       ├── config/               # Qdrant 설정
│   │       └── init/                 # 컬렉션 초기화 스크립트
│   ├── docker/
│   │   ├── Dockerfile.frontend
│   │   ├── Dockerfile.backend
│   │   ├── docker-compose.yml        # 로컬 개발환경
│   │   ├── docker-compose.prod.yml   # 프로덕션 환경
│   │   └── scripts/                  # 초기화 스크립트
│   ├── tests/                        # 테스트 파일
│   │   ├── unit/                     # 단위 테스트
│   │   ├── integration/              # 통합 테스트
│   │   ├── e2e/                      # E2E 테스트
│   │   └── fixtures/                 # 테스트 데이터
│   ├── README.md
│   ├── .env.example
│   ├── .gitignore
│   └── Makefile
│
├── 📦 SolMan/                        # 솔루션 관리 & 정책 관리
│   ├── frontend/                     # 관리자 포탈 UI
│   ├── backend/                      # 사용자 관리, 정책 엔진
│   ├── database/                     # 사용자, 조직, 정책 DB
│   └── docker/
│
├── 📦 DashIQ/                        # 대시보드 인텔리전스 & 로그관리
│   ├── frontend/                     # 실시간 대시보드, 리포트 UI
│   ├── backend/                      # 로그 수집, 분석 엔진, 메트릭스
│   ├── database/                     # 로그 저장소, 메트릭 DB
│   └── docker/
│
├── 📦 ShadowEye/                     # Shadow AI 탐지 & 차단
│   ├── frontend/                     # 탐지 현황 모니터링 UI
│   ├── backend/                      # 트래픽 분석, AI 탐지 엔진
│   ├── database/                     # 탐지 로그, 패턴 분석 DB
│   └── docker/
│
├── 📦 TrustLLM/                      # 내부 LLM & 응답 처리
│   ├── backend/                      # LLM 엔진, 응답 포매팅/마스킹
│   ├── models/                       # AI 모델 파일 및 가중치
│   ├── database/                     # 모델 메타데이터, 학습 데이터
│   └── docker/
│
├── 📦 Common/                        # 공통 컴포넌트 & 라이브러리
│   ├── tools/                        # 개발 도구, 스크립트
│   ├── modules/                      # 공통 모듈 (인증, 로깅 등)
│   ├── utils/                        # 유틸리티 함수들
│   ├── security/                     # 보안 관련 공통 기능
│   ├── services/                     # 공통 서비스 (API 클라이언트 등)
│   ├── types/                        # TypeScript 타입 정의
│   ├── constants/                    # 상수 및 설정값
│   └── tests/                        # 공통 테스트 유틸리티
│
├── 📦 Developer/                     # 개발 지원 자료
│   ├── docs/                         # 개발 문서
│   │   ├── api-specs/                # API 명세서
│   │   ├── architecture/             # 아키텍처 문서
│   │   ├── deployment/               # 배포 가이드
│   │   └── troubleshooting/          # 트러블슈팅 가이드
```

```
│   ├── assets/                    # 이미지, 아이콘 등 자료
│   │   ├── images/
│   │   ├── icons/
│   │   └── diagrams/
│   ├── references/                # 참조 자료
│   │   ├── research-papers/       # 연구 자료
│   │   ├── best-practices/        # 모범 사례
│   │   └── external-docs/         # 외부 문서
│   ├── logs/                      # 개발 로그
│   │   ├── development.md         # 개발 일지
│   │   ├── decisions.md           # 기술적 결정사항
│   │   └── issues.md              # 이슈 및 해결방안
│   └── templates/                 # 개발 템플릿
│       ├── component-template/    # 컴포넌트 템플릿
│       ├── service-template/      # 서비스 템플릿
│       └── api-template/          # API 템플릿
│
├── 📦 api-gateway/                # 전체 시스템 API 게이트웨이
│   ├── config/                    # 게이트웨이 설정
│   ├── middleware/                # 공통 미들웨어
│   └── docker/
│
├── 📦 infrastructure/             # 인프라 및 배포
│   ├── kubernetes/                # K8s 매니페스트
│   │   ├── deployments/
│   │   ├── services/
│   │   └── ingress/
│   ├── terraform/                 # 인프라 코드
│   ├── ansible/                   # 배포 자동화
│   ├── monitoring/                # 프로메테우스, 그라파나 설정
│   └── scripts/                   # 배포 스크립트
│
├── 📄 README.md                   # 프로젝트 소개
├── 📄 docker-compose.yml          # 전체 서비스 오케스트레이션
├── 📄 package.json                # 루트 패키지 설정
├── 📄 .env.example                # 환경변수 예시
├── 📄 .gitignore                  # Git 무시 파일
└── 📄 Makefile                    # 빌드 및 배포 명령어
```

3.2 각 모듈별 핵심 역할

모듈명	핵심 기능	주요 책임
PromptGate	프롬프트 게이트웨이	사용자 입력 필터링, 외부 AI 서비스 프록시
SolMan	솔루션 매니저	사용자/정책 관리, 시스템 운영 관리
DashIQ	대시보드 인텔리전스	실시간 모니터링, 로그 분석, 리포팅
ShadowEye	섀도우 AI 탐지기	비인가 AI 사용 탐지 및 차단
TrustLLM	신뢰할 수 있는 LLM	내부 LLM 운영, 응답 후처리

3.3 Common 폴더 활용 전략

- **modules/**: 인증, 로깅, 암호화 등 모든 서비스에서 사용하는 핵심 모듈
- **security/**: JWT 토큰, API 키 관리, 보안 정책 등 보안 관련 공통 기능
- **utils/**: 날짜 처리, 문자열 유틸, 데이터 변환 등 범용 유틸리티
- **services/**: 외부 API 클라이언트, 이메일 발송 등 공통 서비스

3.4 Developer 폴더 활용 방안

- **실시간 개발 문서화**: API 변경사항, 기술적 결정 등을 즉시 기록
- **지식 공유**: 팀원 간 개발 경험 및 문제 해결 과정 공유
- **자료 중앙화**: 외부 참조 자료, 이미지 등을 체계적으로 관리
- **템플릿 활용**: 일관된 코드 스타일과 구조 유지

3.5 데이터베이스 구조 설명

PromptGate 데이터베이스 아키텍처

- **PostgreSQL**: 사용자 정보, 프롬프트 로그, 필터 규칙 등 구조화된 데이터
- **Redis**: 세션 캐시, API 응답 캐시, 실시간 통계 등
- **Qdrant**: 프롬프트 임베딩, 유사 프롬프트 검색, AI 응답 벡터화

SQL 파일 저장 위치

- **마이그레이션**: `database/postgresql/migrations/` - 스키마 변경 이력
- **스키마 정의**: `database/postgresql/schemas/` - 전체 DB 구조
- **쿼리 모음**: `database/postgresql/queries/` - 재사용 가능한 SQL 쿼리
- **초기 데이터**: `database/postgresql/seeds/` - 기본 데이터 삽입

3.6 Frontend 이미지 파일 저장 전략

1. **`public/assets/images/`** (권장)
    - **장점**: 빌드 후에도 직접 URL 접근 가능
    - **사용예**: `<img src="/assets/images/logos/company-logo.png" />`
    - **적합한 파일**: 회사 로고, 파비콘, 정적 브랜딩 이미지
2. **`src/assets/images/`**
    - **장점**: 빌드 시 최적화됨 (압축, 해시 등)
    - **사용예**: `import logo from './assets/images/logo.png'`
    - **적합한 파일**: 컴포넌트에서 동적으로 사용하는 이미지

4. 기술 스택 권장사항

구분	기술 스택	근거
Frontend	React + TypeScript + Tailwind CSS	관리 포탈의 복잡한 UI, 실시간 대시보드 구현에 적합
Backend	Node.js/FastAPI + TypeScript/Python	AI 모델 통합과 실시간 처리에 최적화
Database	PostgreSQL + Redis + ClickHouse	관계형 데이터 + 캐싱 + 로그 분석용 컬럼형 DB
Vector DB	Qdrant	프롬프트 임베딩 및 유사도 검색
Message Queue	Apache Kafka	대용량 로그 스트리밍 및 실시간 이벤트 처리
Container	Docker + Kubernetes	마이크로서비스 오케스트레이션
API Gateway	Kong/Envoy Proxy	AI 거버넌스 특화 기능 지원

5. 단계별 개발 로드맵

Phase 1: MVP (3-4개월)

1. **PromptGate** 구축
    - 기본 프록시 서버 및 ChatGPT/Claude 연동
    - 간단한 프롬프트 필터링
    - 통합 Web UI 프로토타입
2. **SolMan** 기초
    - 사용자 인증/권한 관리
    - 기본 정책 설정 기능

Phase 2: 확장 기능 (2-3개월)

3. **ShadowEye**
    - 네트워크 트래픽 분석
    - AI 서비스 패턴 탐지
4. **DashIQ**
    - 실시간 대시보드
    - 기본 로그 관리

Phase 3: 고도화 (3-4개월)

5. **TrustLLM**
    - 내부 LLM 모델 통합
    - 고도화된 응답 포매팅/마스킹
6. **Advanced Features**
    - AI 보안 정책 자동화
    - 고도화된 Shadow AI 탐지 알고리즘

6. Git/GitHub 활용 전략

```
# 브랜치 전략 예시
main                    # 프로덕션 배포용
├── develop             # 개발 통합 브랜치
├── feature/prompt-ui   # 기능별 브랜치
├── feature/proxy-api   # 기능별 브랜치
└── hotfix/security-patch # 긴급 수정
```

7. 핵심 차별화 포인트

1. **실시간 프롬프트 보안 검사**: 사용자 입력을 실시간으로 스캔하여 민감정보 차단
2. **멀티 LLM 통합 프록시**: ChatGPT, Claude, Gemini 등 주요 AI 서비스 통합 지원

3. **지능형 Shadow AI 탐지**: 네트워크 패턴 분석을 통한 비인가 AI 사용 실시간 탐지
4. **컴플라이언스 자동화**: GDPR, 개인정보보호법 등 규제 요구사항 자동 대응

8. 다음 단계 제안

1. **기술 스택 최종 확정** 및 개발환경 구축
2. **MVP 상세 설계** 및 프로토타입 개발 착수
3. **보안 정책 프레임워크** 설계
4. **파일럿 고객 확보** 및 피드백 수집 계획

문서 버전: 2025.09.22 Draft
작성자: JAKE 대표
최종 업데이트: 2025.09.22