2025년도 <종단형 PBL> 성과 발표

2025.12.5.(금) 10:30~14:30

관리번호	2025-

프로젝트 개요

프로젝트 명칭		
	국문명	기업용 이메일 PII(개인 식별 정보) 마스킹 AI agent
	영문명	Corporate Email PII Masking AI Agent

프로젝트 팀	팀명	헨젤과 그레텔	지도교수	김형종

팀구성	학과	학번	이름	휴대폰번호	E-mail 주소
팀원1(팀장)	정보보호학과	2022111346	정지윤	010-3767-1740	eileenjjy@naver.com
팀원2	정보보호학과	2023111394	육은서	010-4687-4599	yes0823bs@swu.ac.kr
팀원3	정보보호학과	2023111399	이시온	010-3118-9927	tldhs122484@gmail.com
팀원4					
팀원5					
팀원6					

오픈소스 탑재 URL 명시 (GitHub, SourceForge)	GitHub: https://github.com/guardcap/MASKIT-pub

프로젝트 요약

MASKIT은 PII 탐지·정책 판단·파일 마스킹까지 전 과정을 자동화하는 AI agent이다. 사람이 문서를 열어 직접 검증하는 수동 방식을 AI agent로 대체함으로써 업무 효율을 올리고 이메일 전송을 간편화하고자 진행하였다.

핵심 기술 구성은 하이브리드 PII 탐지 엔진(Regex + 한국어 특화 NER + OCR), RAG 기반 LLM 정책 판단 시스템, 좌표 기반 무손실 파일 마스킹, 멀티모달 정책 자동 처리, 역할 기반 접근 제어(RBAC) 등 총 5개의 독자적 기능으로 이루어진다. 특히 단순 패턴 탐지가 중심인 기존 Presidio 기반 시스템과 달리, MASKIT은 "수신자·전송 맥락·사내 보안 규정"을 LLM이 종합 판단하여 마스킹 필요 여부를 결정하는 지능형 구조를 갖추고 있어 실제 업무 환경에서의 정확성과 실용성을 크게 높였다.

정책 관리 기능 역시 기존 시스템과 뚜렷한 차별성을 갖는다. 관리자가 PDF나 이미지 형태의 정책 문서를 업로드하면, LLM이 문서의 구조를 분석해 핵심 요약, 적용 규칙, 관련 엔티티 등을 자동으로 추출하고 이를 Vector Store에 즉시 반영한다. 덕분에 별도의 개발 지식 없이도 조직의 보안 정책을 지속적으로 최신 상태로 유지할 수 있다. 또한 비정형 데이터 처리 측면에서는 OCR과 좌표 기반 탐지 기법을 활용하여 PDF와 이미지 내 민감 정보를 정밀하게 식별·마스킹하면서도 원본 문서의 레이아웃을 그대로 보존하여 실제 업무 환경에서의 활용성을 극대화하였다.

시스템은 FastAPI·MongoDB·OpenAI Vector Store·React 기반으로 구축되었으며, JWT 기반 RBAC, 감사 로그, SMTP 이메일 전송, 엔티티 커스터마이징 기능까지 완전한 프로토타입을 완성하였다. 최종적으로 MASKIT은 개인정보 검증 프로세스를 자동화하여 업무 효율을 높이고, 데이터 기반 보안 감사 환경을 구축함으로써 기업 내부 보안 수준을 실질적으로 향상시킬 수 있도록 설계하였다.

프로젝트 상세 설명

프로젝트 명칭	국문명	기업용 이메일 PII(개인 식별 정보) 마스킹 AI agent		
	영문명	Corporate Email PII Masking AI Agent		
오픈소스 URL		GitHub: https://github.com/guardcap/MASKIT-pub		
프로젝트 팀명		헨젤과 그레텔	지도교수	김형종
팀원(이름·학과·학번)		육은서·정보보호학과·2023111394 이시온·정보보호학과·2023111399 정지윤·정보보호학과·2022111346		

1. 프로젝트 개요

1.1 개발 필요성

개인정보보호위원회 통계에 따르면 2024년 국내 개인정보 유출 신고 건수는 총 307건으로, 이 중 업무 과실이 30%를 차지하며 기관·기업 내 실무 과정에서 발생하는 단순 실수들이 심각한 보안 사고로 이어지고 있는 것으로 나타났다. 특히 업무 과실 유형 중 '개인정보 파일 게시(30%)', '개인정보 파일 오첨부 및 오발송(총합 20% 이상)'과 같은 내부자의 정보 유출은 조직 내에서 가장 빈번하게 발생하는 사고로 지적된다.

파일 첨부 전 개인정보 포함 여부 미확인, 동보 발송 기본 설정, 수신자 입력 실수, 마스킹 누락, 내부 문서 관리 소홀 등 단순하고 반복적인 인간 행동 패턴에서 유출 사고로 이어지는 사례가 잦다.

이러한 문제를 단순히 직원의 수동 작업으로 해결하기에는 업무 효율에 있어서 많은 문제를 야기한다. 단순 실수를 유발할 가능성을 높일 뿐만 아니라 핵심 업무에 집중할 시간을 앗아가는 구조적 병목으로 작용한다. 따라서 조직 내 이메일, 첨부 문서 등 업무 커뮤니케이션 과정에서 민감정보를 자동으로 검출하고 마스킹하는 지능형 에이전트(Agent)의 도입은 보안 사고 예방은 물론 실무자의 업무 효율성을 획기적으로 개선하고자 프로젝트를 진행했다.

1.2 프로젝트 목적 및 용도

자동화된 개인정보 탐지 및 보호: 이메일에서 정규표현식(Regex)과 한국어 특화 NER을 결합하여 PII를 식별한다. 단순 텍스트뿐만 아니라 이미지, PDF 등 비정형 데이터도 OCR 전처리를 통해 PII 탐지를 지원한다.

지능형 마스킹 의사결정 지원: 단순히 일방적인 차단이 아닌, AI가 RAG(검색 증강 생성)를 사용하여 사내 보안 정책과 법률, 상황을 분석한 뒤 마스킹 권고안을 제시함으로써 사용자가 안전하게 데이터를 전송하도록 돕는다.

1.3 유사 기술과의 차별성/독창성

1.3.1 기술적 독창성: Microsoft Presidio 대비 llm 결합을 통한 지능형 탐지 시스템 구축

MASKIT은 오픈소스 라이브러리인 Presidio가 제공하는 단순 탐지 기능을 넘어, '정책에 근거한 판단'이 결합된 지능형 보안 시스템이다.

구분	MASKIT	Presidio : PII 탐지 오픈소스 라이브러리
탐지 기능	단순 탐지와 문맥에 따른 위반 여부를 추론하는 agent	단순 식별 도구
엔티티 추가	GUI 제공으로 간편하게 추가	하드 코딩이나 설정 파일을 수정 등 높은 기술적 장벽이 존재
비정형 데이터	원본 파일의 레이아웃을 훼손하지 않고 민감 정보만 가리는 무손실 마스킹을 지원	원본 서식을 유지한 채 마스킹하는 기능이 부족
한국 최적화	한국어 특화 NER 모델을 탑재	한국어의 고유한 언어적 특성 처리에 취약

표 1 . MASKIT과 Presidio 비교

1.3.2 운영적 차별성: 기존 상용 DLP 대비 혁신

MASKIT은 '통제와 차단' 중심의 기존 DLP 패러다임을 '협업과 유연성' 중심으로 전환하였다.

구분	MASKIT	기존 DLP
문맥 이해를 통한 유연한 마스킹	데이터의 맥락에 따른 AI의 마스킹 판단 근거 제시	규칙은 무조건 차단하거나 격리하는 Strict 방식
시각적 마스킹	문서 내의 민감 정보(PII) 영역만 마스킹	파일을 통째로 차단하거나 암호화

표 2 . MASKIT과 기존 DLP 비교

2. 프로젝트 구성

2.1 시스템 전체 구성도 및 주요 기능 설명

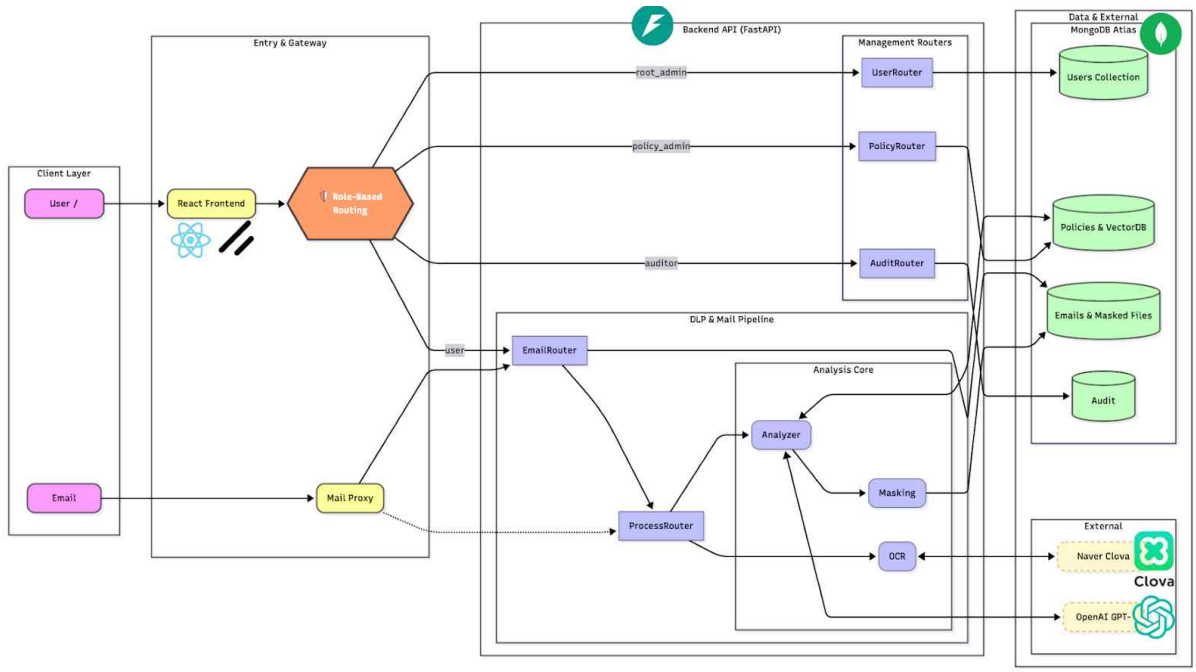

그림 1 . 시스템 아키텍처

2.1.1. 역할 기반 접근 제어 (RBAC) 설계

MASKIT은 기업 환경에서의 효율적인 개인정보 관리를 위해 5단계 역할 기반 접근 제어 시스템을 채택하였다. 각 역할은 명확히 분리된 권한과 책임을 가지며, 최소 권한 원칙(Principle of Least Privilege)에 따라 설계되었다.

역할	역할 정의 및 권한	핵심 기능
Sysytem Admin (시스템 관리자)	전체 시스템 관리 및 사용자 계정 관리	• 사용자 계정 삭제 및 정보와 권한 열람 • 사용자 역할 변경 및 권한 부여 • 접근 범위: 권한 관리 로그 + 본인 활동 로그
Auditor (감사자)	개인정보 처리 활동 감사 및 컴플라이언스 모니터링	• 모든 계정의 프라이버시보호 이력 열람 가능 • 접근 범위: 읽기 전용, 전체 로그
Policy Admin (정책 관리자)	개인정보 보호 정책 및 탐지 엔티티 관리	• 정책 문서 수정, 업로드, 삭제 • 커스텀 엔티티 생성 및 삭제 • 접근 범위: 정책 및 엔티티 관리, 본인 활동 로그
User (일반 사용자)	이메일 작성 및 마스킹 데이터 전송	• 이메일 작성 및 첨부파일 추가 • AI 분석 기반 마스킹 권장 확인 • 커스텀 마스킹 설정 선택 • 마스킹된 결과 미리보기 • 본인 이메일 전송 이력 조회 • 접근 범위: 본인 이메일, 본인 활동 로그

표 3 . RBAC에서 정의한 role

2.1.2 기업 맞춤형 엔티티 관리 시스템

MASKIT은 내장된 PII 엔티티뿐 아니라 기업의 특수한 개인정보 유형을 유연하게 관리하기 위해 엔티티 커스터마이징 기능을 제공한다. 정책 관리자는 민감정보 유형을 직접 정의 및 추가함으로써 시스템을 기업에 최적화 된 상태로 운영 가능하다.

엔티티 기본 정보	• 엔티티 ID : recognizer에서 해당 엔티티를 구분할 이름 • 표시 이름: 사용자에게 보여지는 이름
탐지 규칙 설정	• Regex 패턴: 정규표현식 기반 자동 탐지 규칙 • 키워드: 컨텍스트 분석용 연관 키워드

표 4 . 엔티티 추가 양식

2.1.4 맥락 기반 마스킹 분석

이메일을 작성하면 마스킹 페이지에서 커스텀 패널이 뜬다. 사내/사외 발신인지, 검색 시 어떤 규칙을 우선하는지 등 세부적인 커스텀이 가능하다. 사용자가 선택 사항과 이메일 메타데이터 기반으로, VectorDB에서 거리가 가까운 가이드라인을 검색한다. 이 가이드라인을 기반으로 LLM이 각 PII 항목에 대해 마스킹이 필요한지 여부를 판단한다

.

2.1.5 첨부파일 마스킹

이메일 본문뿐 아니라 비정형 데이터에 대해서도 마스킹을 진행 후, 마스킹된 파일로 메일 전송이 가능하다. 비정형 데이터는 PII 엔티티마다 좌표를 매핑해두어 블랙박스로 마스킹을 진행한다.

파일 유형	마스킹 방식
PDF	텍스트 검색 + instance_index로 정확한 위치 특정
이미지 (jpg, png)	OCR 좌표(bbox) 기반 직접 마스킹

표 5 . 지원 형식에 따른 마스킹 방식

2.1.6 프라이버시 보호 이력

모든 주요 활동들은 로그로 기록되어 프라이버시 보호 이력 페이지에서 확인할 수 있다.

기록 항목	내용
이벤트 타입	이메일 전송, 정책 업로드, 엔티티 생성, 사용자 권한 변경 등
사용자 정보	이메일, 역할(권한)
리소스 정보	리소스 타입, ID
상세 정보	변경 내용, 파라미터
IP 주소	접속 IP
결과	성공/실패 여부 및 에러 메시지
타임스탬프	KST 기준

표 6 . 로그 형식

3. 구현 환경

3.1 팀 소개 및 개발 환경

팀원	역할
정지윤	팀장, FastAPI 개발 및 연동
육은서	Vector 검색 기능 구현, ShadCN 활용 UI 개선
이시온	PII 탐지 엔진 개발, JWT 인증 및 RBAC 구현, MongoDB 관리

표 7 . 팀원 소개 및 역할

구분	항목	상세 내용
백엔드	언어	Python 3.10+
	프레임워크	FastAPI
	데이터베이스	MongoDB 6.0+, OpenAI Vector Store
	서버	Uvicorn ASGI Server
	인증	JWT (JSON Web Tokens)
프론트엔드	언어	TypeScript 5.0+
	프레임워크	React 18, Vite 5.0
	UI 라이브러리	shadcn/ui, Tailwind CSS
	상태 관리	React Hooks
AI	LLM	Openai GPT-4o
	NER	monologg/koelectra-base-v3-naver-ner
	OCR	Naver Clova OCR API, Openai Vision

표 8 . 개발 환경

3.2 독자적 개발

3.2.1 JWT 인증 기반 RBAC (Role-Based Access Control) 시스템

계정마다 접근 제어를 관리하는 것이 아닌 계정에 부여된 역할에 따라 API 엔드포인트 접근 권한을 제어한다. backend/app/auth/auth_utils.py에 주요 기능이 구현되어 있다.

FastAPI의 Dependency를 활용하여 API 요청이 들어왔을시, @app.get("/api", dependencies=[Depends(get_current_auditor)])로 어떤 계층인지를 확인 및 인증한다.

함수명	기능
get_current_user	사용자가 보낸 JWT 토큰이 유효한지 확인 및 해당 토큰의 메일

	정보로 DB 참조
get_current_root_admin	get_current_user 선행 이후, role이 ROOT_ADMIN인 경우만 요청 승인
get_current_policy_admin	get_current_user 선행 이후, role이 POLICY_ADMIN인 경우만 요청 승인
get_current_auditor	get_current_user 선행 이후, role이 AUDITOR인 경우만 요청 승인

표 9 . Depends에 구현한 함수명 및 기능

각 함수에서는 사용자의 JWT를 통해서 사용자의 Role을 참조한다. 이렇게 구현함으로써 신분 위조 공격을 막고 안전한 RBAC 체계를 구현할 수 있다.

3.2.2. 하이브리드 PII 탐지 엔진 및 정확도 개선을 위한 중복 엔티티 우선순위 로직 구현

기존 비교군의 한국 특화 PII가 부족한 점을 개선하기 위해 Regex(정규표현식) 기반 엔진과 NER(개체명 인식) 모델을 결합한 하이브리드 엔진을 개발하였다.-

탐지 엔티티	Regex
주민등록번호	r"\d{6}[\-]?\d{7}"
여권번호(구형)	r"([MSROD]\d{8})"
여권번호(신형)	r"([MSROD]\d{3}[a-zA-Z]\d{4})"
운전면허번호	r"\d{2}-\d{2}-\d{6}-\d{2}"
이메일	r"[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+"
전화번호(010-)	r"\b0(1[016789])[\-]?\d{3,4}[\-]?\d{4}\b"
전화번호(지역번호)	r"\b(02|0[3-6][1-4])[\-]?\d{3,4}[\-]?\d{4}\b"
신용카드 번호	r"(\d{4})([-. \s])\d{4}\2\d{4}\2\d{4}"
계좌 번호	r"\b\d{3}-\d{2,6}-\d{2,7}\b"
	r"\b\d{4}-\d{2,6}-\d{2,7}\b"
	r"\b\d{6}-\d{2}-\d{6}\b"
IP 주소	r"\b(?:25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(?:\.(?:25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3}\b"
	r"(([0-9a-fA-F]{1,4}:){7,7}[0-9a-fA-F]{1,4}|...)" *길어서 축약
MAC 주소	r"[0-9a-fA-F]{2}[:\-][0-9a-fA-F]{2}[:\-][0-9a-fA-F]{2}[:\-][0-9a-fA-F]{2}[:\-][0-9a-fA-F]{2}[:\-][0-9a-fA-F]{2}"
GPS 주소	r"\b([+-]?\d{1,2}\.\d+)[,\s]+([+-]?\d{1,3}\.\d+)\b"
	r"(?:위도|경도|latitude|longitude|lat|lon)\s*:\s*([+-]?\d{1,3}\.\d+)"

표 10 . PII 엔티티와 Regex

contact@naver.com 이라는 텍스트는 Email, URI로 동시 탐지되는 중복 문제가 존재하였다. 우리는 이 부분을 Priority 로직을 세워 개선하였다. 로직 플로우는 다음과 같다.

1) 탐지된 텍스트의 길이가 더 긴 엔티티를 우선한다.
2) 길이가 동일할 경우, 엔티티 인스턴스로 있는 신뢰도 점수가 더 높은 엔티티를 우선한다.

3.2.3. 멀티모달 정책 관리

1) 텍스트 추출 : 비정형 데이터인 정책을 추가하면 Openai Vision으로 텍스트를 추출한다.
2) 텍스트에서 가이드라인 추출 : 추출한 텍스트에서 VectorDB에 저장할 수 있는 포맷으로 LLM이 가공하여 저장한다.

위와 같이 정책 문서 업로드 단계를 따른다. 백그라운드에서 LLM이 정책 문서를 분석하여 실무 가이드라인을 생성한다. 생성한 가이드라인은 MongoDB와 Openai Vector Store에 저장된다. MongoDB는 사용자에게 저장된 정책 목록을 보여주고 사용자가 커스터마이징을 편하게 할 수 있도록 UI 구성을 위해 저장한다. Vector Store에는 추후 RAG 검색을 할 때 벡터 기반으로 인용 가이드라인을 찾을 때 사용한다. 이는 추후 이메일 마스킹 판단 시 근거로 사용된다.

4. 구현 결과

[시나리오 1: 사용자 역할 변경]
1. 제일 먼저 회원가입한 사용자는 System admin 역할 부여
2. 사용자 관리 페이지 이동
3. 이후 여러 계정 회원 가입 진행 후, 사용자들 시스템 관리자, 정책 관리자, 감사자, 일반 사용자 중 하나로 역할 부여

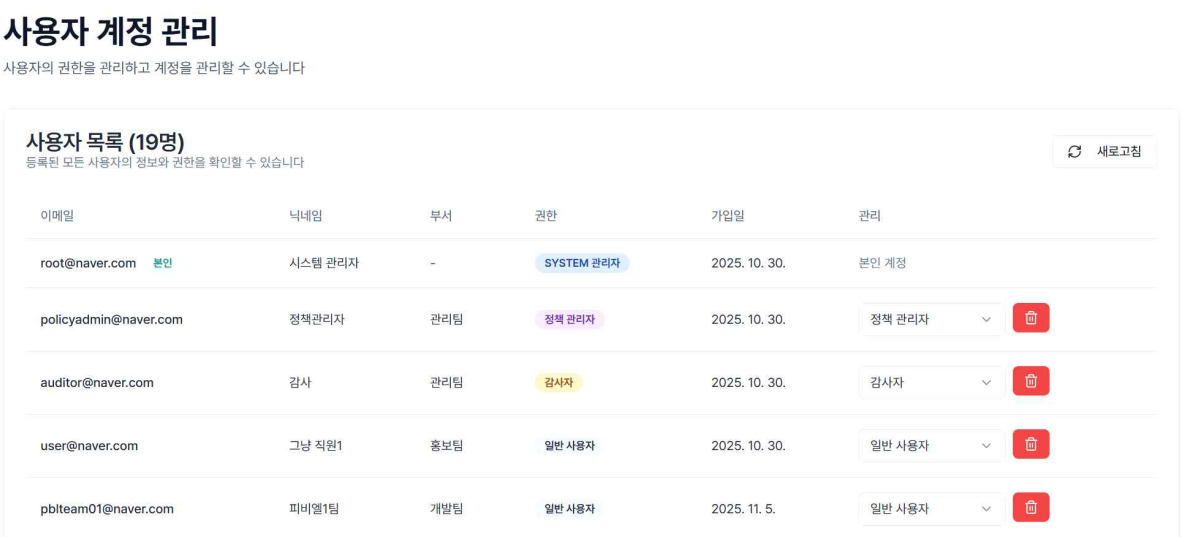

그림 2 사용자 계정 관리 페이지

[시나리오 2: 이메일 작성 및 전송]

1. 로그인 → 마이페이지 이동 후 본인의 개인 메일 SMTP 서버 설정
2. 메일 쓰기 페이지 이동
3. 수신자, 제목, 본문 입력
4. 첨부파일 드래그 앤 드롭 (PDF/이미지)
5. 마스킹 진행 버튼 클릭 → 마스킹 검토 페이지 이동
6. 수신자 유형 선택 → AI 분석 시작 버튼 클릭 후 자동 탐지 결과 확인
7. 마스킹 결정 확인 → RAG 근거 검토
8. 원하는 마스킹 엔티티 체크
9. 선택된 PII 마스킹 버튼 클릭 → 마스킹된 메일 본문, 첨부파일 화면 출력
10. 이메일 전송 클릭 → SMTP 발송
11. 실제 메일함 (네이버 메일, gmail 등)수신 확인

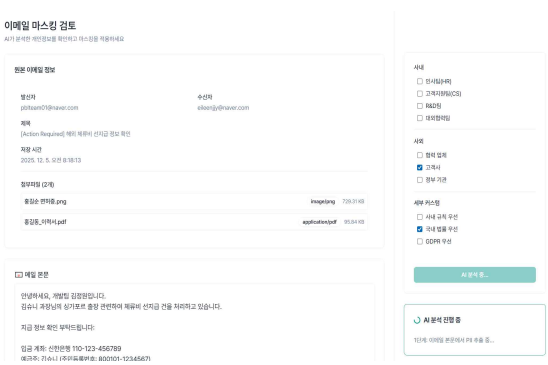

그림 3 마스킹 페이지 (좌측:커스텀 패널)

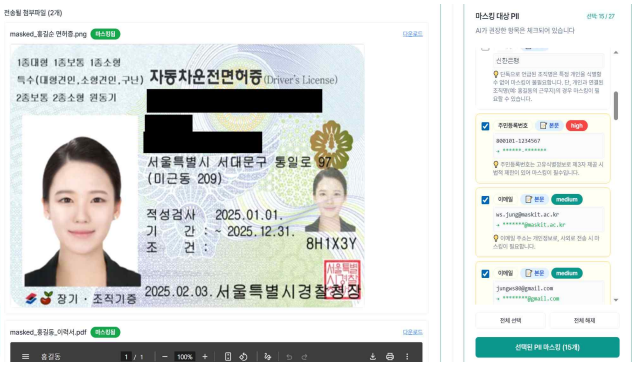

그림 4 마스킹 미리보기 (좌측:AI 분석 결과)

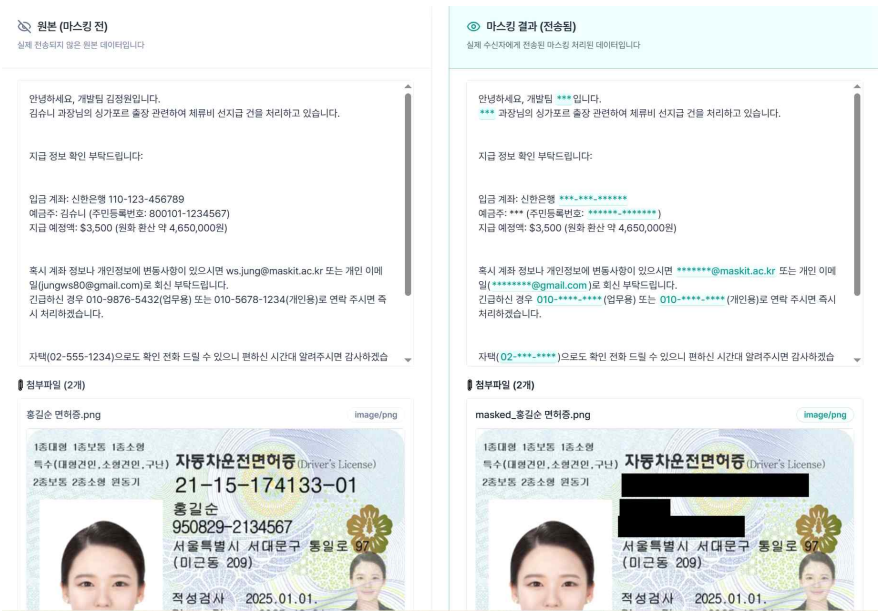

그림 5 보낸 메일함

[시나리오 3: 정책 업로드]

1. POLICY_ADMIN 계정으로 로그인
2. 정책 추가 페이지
3. PDF 파일 업로드 (개인정보보호법, GDPR 등)
4. 제목, 발행 기관 입력
5. 백그라운드에서 GPT-4o가 가이드라인 자동 추출
6. VectorDB 동기화 → 즉시 RAG 검색에 활용
7. 엔티티 관리 페이지
8. 커스텀 엔티티 추가

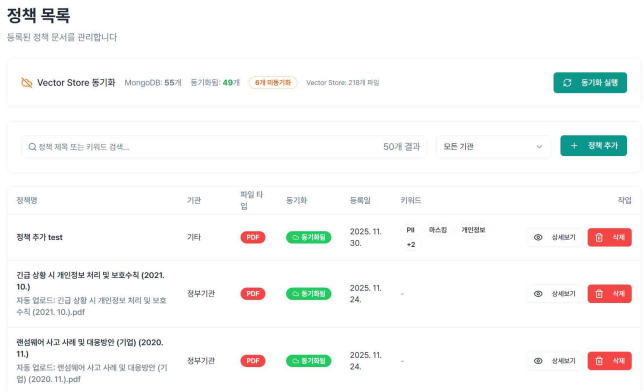

그림 6 정책 목록 페이지

그림 7 추출된 가이드라인

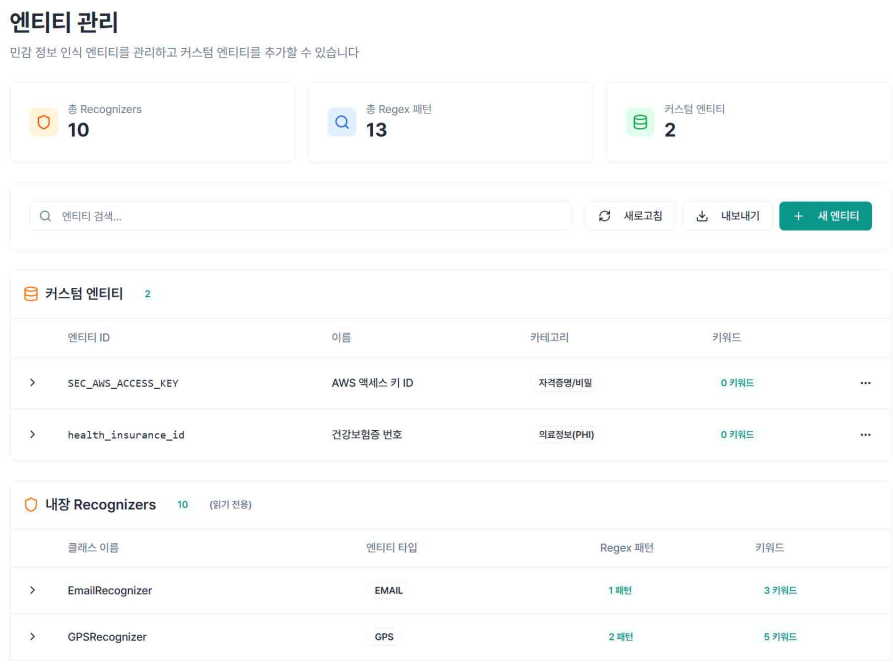

그림 8 엔티티 관리 페이지

[시나리오 4: 감사 로그 조회]
1. AUDITOR 계정으로 로그인
2. 프라이버시 전송 이력 페이지로 이동
3. 모든 사용자들의 이메일 전송 로그 확인
4. 정책 관리자의 정책, 엔티티 수정 관련 로그 확인
5. 시스템 관리자 역할 변경 로그 확인

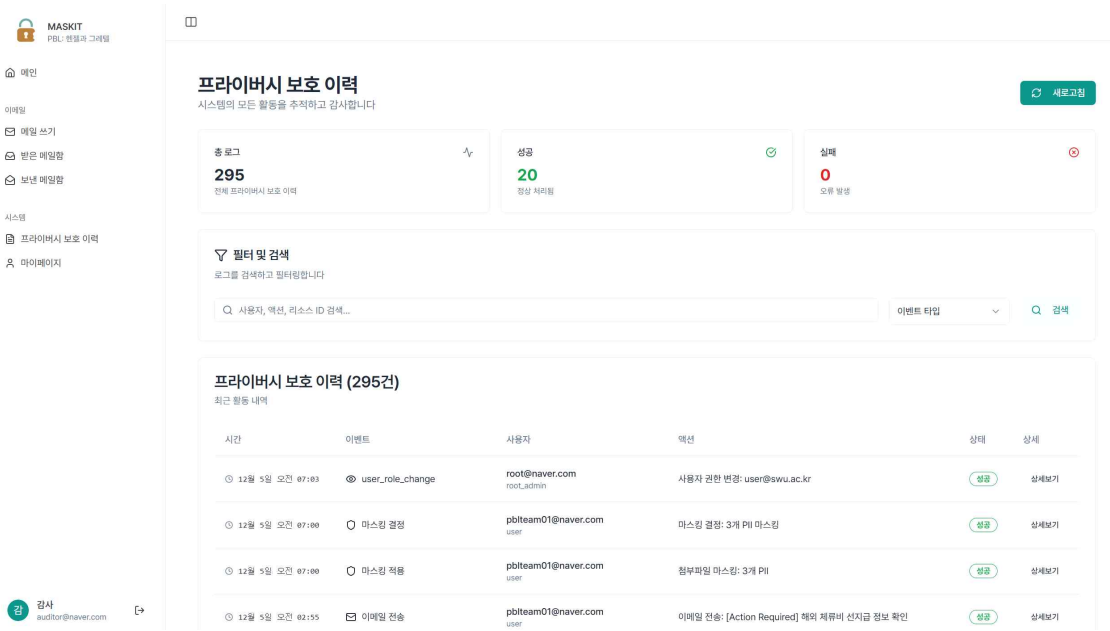

그림 9 프라이버시 보호 이력 페이지

5. 기대효과 및 향후 작업

5.1 기대효과

본 프로젝트인 MASKIT agent 도입을 통해 기대할 수 있는 효과는 크게 두 가지 측면으로 요약할 수 있다.

첫째, 반복적인 보안 점검 업무를 자동화하여 임직원의 업무 생산성을 대폭 향상시킬 수 있다. 보안 담당자와 일반 임직원이 일일이 문서를 열어 민감 정보를 확인하고 마스킹하던 기존의 비효율적인 프로세스를 제거함으로써, 업무 과중을 해소하고 본연의 핵심 업무에 집중할 수 있는 환경을 조성한다. 특히 대량의 문서 처리나 긴급한 업무 수행 시에도 AI 에이전트가 실시간으로 보안 가이드를 제공하고 마스킹을 수행하므로, 보안성과 신속성을 동시에 확보하여 전반적인 업무 효율성을 극대화할 수 있다.

둘째, 데이터 기반의 보안 감사 및 모니터링 체계를 확립하여 조직 내 보안 문화를 정착시킬 수 있다. 모든 이메일 발송 및 마스킹 이력이 투명하게 기록되고 관리되므로, 사후 감사 및 추적 관리가 용이해진다. 축적된 데이터를 바탕으로 부서별, 유형별 개인정보 취급 현황을 파악하고 맞춤형 보안 교육이나 정책 수립에 활용할 수 있다. 이는 단순히 기술적인 통제를 넘어, 조직 구성원들이 개인정보 보호의 중요성을 인식하고 자발적으로 보안 수칙을 준수하는 성숙한 보안 문화를 형성하는 토대가 될 것이다.

5.2 향후 작업

가이드라인 스키마 추출 최적화: 문서 인식 후 VectorDB에 저장될 가이드라인 스키마가 불완전하게 추출되는 문제를 해결하기 위해, 문서의 레이아웃과 맥락을 정밀하게 분석하여 유효한 정책 정보를 최대한 많이 확보하

는 추출 파이프라인을 고도화한다.

온프레미스(On-Premise) 환경 지원을 위한 AI 엔지니어링 : 데이터 보안에 극도로 민감한 공공기관이나 금융권 등 클라우드 서비스(SaaS) 이용이 제한적인 환경에서도 도입 가능하도록, 외부 네트워크 연결 없이 독립적으로 구동되는 온프레미스 AI 모델을 최적화한다.

로컬 LLM 최적화 (Ollama 활용): Llama 3, Mistral 등 고성능 오픈 소스 LLM은 OpenAI 모델 대비 성능 격차가 존재하므로, Ollama 환경에서 MASKIT 에이전트의 특화된 작업을 수행할 수 있도록 파인튜닝 및 프롬프트 엔지니어링을 통해 성능을 극대화한다.