PROJECT **Multi-Agent 기반 에이전틱 워크플로우 자동화 플랫폼** · FlowIt

황대원 · PM / Backend

개인의 업무 노하우를, 팀 전체가 재사용하는 자산으로

" 자연어로 말하면 AI가 검증된 Skill을 조합해 실행하고, 하네스(작업지침)가 산출물의 품질을 보장하는 멀티에이전트 플랫폼을 5인 팀으로 구축 "

프로젝트 배경 · 기여

[프로젝트 개요]

자동화 도구는 늘어도 업무 노하우는 사람에게 갇혀 있다. **AI 도입 기업 88% — 스케일업 성공은 33%** (통합 복잡성·모니터링 부재가 실패 원인). 자연어로 말하면 AI가 Skill을 조합해 실행하는 플랫폼으로 해결.

[프로젝트 기여]

MY ROLE

- 전체 아키텍처 설계 — Clean Architecture · 모노레포
- 온톨로지 그래프 DB · GraphRAG 설계·구현
- 하네스 엔지니어링 — 품질 가드레일 코드화
- 인프라(IaC) — GCP · Modal · Neo4j · Terraform
- 멀티에이전트 구조 · SSE API 구현

[기술 스택]

Python · FastAPI · PostgreSQL 16 · pgvector · Neo4j · LangGraph · Modal · Next.js · Terraform · GCP

핵심 설계 · 기법

★ 온톨로지 GraphRAG (핵심 설계)

문제 top-k 벡터 검색만으론 노드 호환성·필수 연결을 표현 못 해 AI 생성 워크플로우의 23%가 실행 불가.

설계 Neo4j 온톨로지 — vector seed → 1-hop(CAN_FOLLOW) 확장 → ground truth 허용집합으로 노드 조합을 결정론적 제약.

하네스 엔지니어링

Clean Architecture 의존성 단방향 + common_schemas SSOT + PR 게이트 3종(drift·pytest·Ruff)으로 규칙을 코드로 강제.

인프라 · 멀티에이전트 · API

Cloud Run·Modal·Neo4j 3티어를 Terraform 7모듈로 코드화 · SA 최소권한 3계정 격리. 5 Modal 앱 독립 배포·장애 격리, /v1/ai/compose SSE 스트리밍 API.

결과 · 결론 · 한계점

[핵심 성과] 온톨로지 도입 전/후

MY ROLE

지표	Before 벡터만	After GraphRAG
끊긴 워크플로우율	23.2%	**0.0%**
QA pass	0.45	**0.75**
motif-correctness	75%	**100%**

+ **개인화 노이즈 4.83→0.83** (5.8배 ↓) · 94+ 자동화 테스트

[결론]

"측정 없는 고도화는 추측" — 온톨로지로 구조를 결정론적 제약하니 품질이 정량 개선. 구조는 코드가 강제하고 LLM은 파라미터만 채운다.

[한계점]

- 10가지 워크플로우 구조에 대한 유저 시나리오만 검증되어 확장 시 추가 검증 필요
- 개인화 게이트 헤드룸이 얇아 코퍼스 확장 시 재측정 필요

배운 점

soft 힌트만으론 작은 LLM의 구조를 못 바꾼다는 실패에서 **"구조는 코드로 강제, LLM은 파라미터만"**이라는 원칙을 세웠고, **아키텍처·인프라·품질 체계를 코드로 강제**해 5인 팀이 안전하게 협업하는 기반을 만든 경험을, 측정 기반으로 시스템을 개선하는 실무 역량으로 이어가고자 합니다.