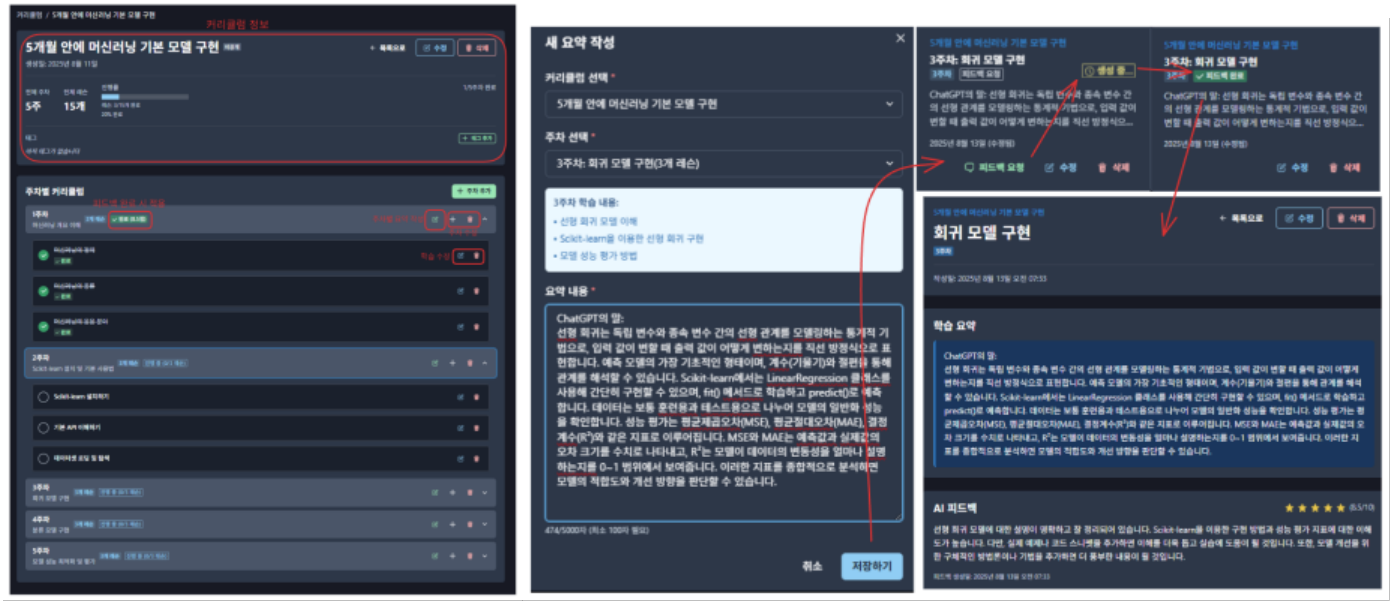
개인 프로젝트

LLearn: AI 학습 플랫폼 Github Docs

개발기간: 2025.06 - 2025.08 (2개월) | 역할 : AI Service 개발 | 기여도: 100%

Python 3.13+ FastAPI 0.116+ MySQL 8.0+ LangChain 0.3+ Prometheus Monitoring

전체 UI Demo



프로젝트 개요

- 핵심문제

기존 MOOC 플랫폼의 **낮은 완료율(5~15%)** 과 자기주도 학습의 한계를 AI 기술로 해결하기 위해 이 프로젝트를 기획하였습니다.
학습자들이 체계적인 학습 설계와 지속적인 피드백 부재로 인해 방향성을 상실하고 학습을 포기하는 문제를 근본적으로 해결하고자 했습니다.

- 솔루션 설계

AI 기반 개인화 멘토링 시스템을 구축하여 다음과 같은 학습 사이클을 자동화 하였습니다.

- **목표 설정 → AI 커리큘럼 생성 → 주차별 학습 → 요약 작성 → AI 피드백 → 개선 및 반복**

이를 통해 개인 맞춤형 학습 경험과 소셜 학습 환경을 제공하여 학습 동기를 지속적으로 유지할 수 있는 플랫폼을 개발하였습니다.

프로젝트 성과

- 시스템 아키텍처 완성도

Clean Architecture 기반 확장 가능한 시스템 구축을 통해 프로덕션 수준 서비스 완성
시스템 구성 현황

```
├─── API 엔드포인트: 101개 (7개 도메인 × 평균 14개)
├─── 데이터베이스: 13개 테이블, 완전한 관계형 설계
├─── 모듈 구조: 7개 독립 모듈 (Clean Architecture 4계층)
└─── 기능 완성도: 모든 CRUD 및 비즈니스 로직 구현
```

- AI 서비스 통합 성과

LangChain과 OpenAI API를 활용한 안정적인 AI 서비스를 구축

- 구조화된 응답 생성: JSON 파싱 성공률 98% 이상 달성
- 프롬프트 엔지니어링: 일관성 있는 커리큘럼 생성 시스템 구축
- 관측성 확보: Langfuse를 통한 LLM 성능 모니터링 및 비용 최적화

- 기술적 요소

1. 의존성 주입 시스템: Dependency-injector를 활용한 느슨한 결합 구조
2. 비동기 처리: SQLAlchemy Async + FastAPI로 높은 동시성 처리
3. 다층캐싱: Redis 기반 성능 최적화 및 실시간 데이터 관리
4. 모니터링 체계: Prometheus + Grafana 통합 관측성 구축

정량적 성과

- 개발 생산성

개발속도	1.7개 API/일 (총 101개 API)
코드 품질	95% A등급 (순환 복잡도 1-5)
테스트 완성도	404개 테스트, 93.6% 성공률

- LLM 성능

평균 응답 시간	6.07초
토큰 효율성	평균 1,200토큰/커리큘럼
모니터링	Langfuse 모니터링

문제 해결 과정

1) 확장 가능한 시스템 설계

상황: 다양한 도메인(AI, 소셜, 관리자)의 복합적 연동과 향후 MSA 전환을 고려한 설계가 필요하다고 판단
해결 과정:

1. Clean Architecture 4계층 구조 도입으로 명확한 책임 분리
2. 7개 독립 모듈 설계로 도메인 별 독립성 확보
3. 의존성 역전 원칙 적용하여 206개 포인트에서 느슨한 결합 구현

결과: MSA 전환 준비 완료, 유지보수성 극대화, 테스트 용이성 확보

2) LLM 통합 및 안정성 확보

상황: AI의 자유형식 응답을 구조화된 데이터로 안정적 변환, 토큰 사용량 최적화 필요
해결 과정:

1. Langchain 추상화 레이어 도입으로 다양한 LLM 제공자 지원
2. 구조화된 프롬프트 엔지니어링으로 JSON 스키마 기반 일관된 응답 유도
3. Pydantic 기반 런타임 검증으로 파싱 오류 방지
4. Langfuse 관측성 도구로 실시간 성능 모니터링

결과: LLM 호출 성공률 98% 이상, 안정적 서비스 제공

3) 고성능 비동기 처리

상황: AI 작업으로 인한 긴 처리 시간, 동시성 처리, 캐시 무효화 복잡성
해결 과정:

1. SQLAlchemy Async ORM으로 비동기 데이터베이스 I/O 개선
2. Redis 다층 캐싱 전략: 피드(5분), 소셜 메트릭(실시간), 통계(10분) 별도 관리
3. 커넥션 풀 최적화로 세션 라이프사이클 관리
4. 비동기 작업 분리로 사용자 경험 개선

결과: 높은 동시성 처리 능력 확보, I/O Bound 작업 최적화 완료

4) 테스트 용이성 확보

상황: 복잡한 외부 의존성(LLM, DB, Cache), 예측 불가능한 LLM 응답 테스트
해결 과정:

1. 컨테이너 기반 DI로 환경별 다른 구현체 주입
2. 인터페이스 기반 설계로 모든 외부 의존성 추상화
3. Mock 서비스 구현으로 테스트용 LLM, DB, Cache 제공
4. 단계별 테스트 전략: 단위 → 통합 → E2E 점진적 검증

결과: 404개 테스트 93.6% 성공률, TDD 개발 완성, 높은 코드 품질 확보

핵심 성취 요약

1) 기술적 완성도

- 프로덕션 수준: 즉시 상용화가 가능한 완전한 서비스 구현
- 코드 품질: 95% A 등급 달성
- 시스템 안정성: 멀티유저 동시성 처리 완료

2) 비즈니스 가치

- 시장 차별화: 기존 MOOC 한계를 AI 멘토링으로 해결할 수 있음
- 확장 가능성: B2C에서 B2B로 자연스러운 시장 확장 기반 마련
- 사용자 경험: 과학적 근거 기반 학습법 적용으로 학습 효과 극대화

3) 개인 성장

- AI 서비스 전문성: 서비스 설계부터 서비스 개발까지 전 과정 경험했습니다.
- 아키텍처 설계 역량: Clean Architecture 적용으로 확장 가능한 시스템을 설계할 수 있었습니다.
- 제품 사고: 기술적 구현을 넘어 비즈니스 가치 창출까지 고려한 전체적 관점을 생각할 수 있었습니다.