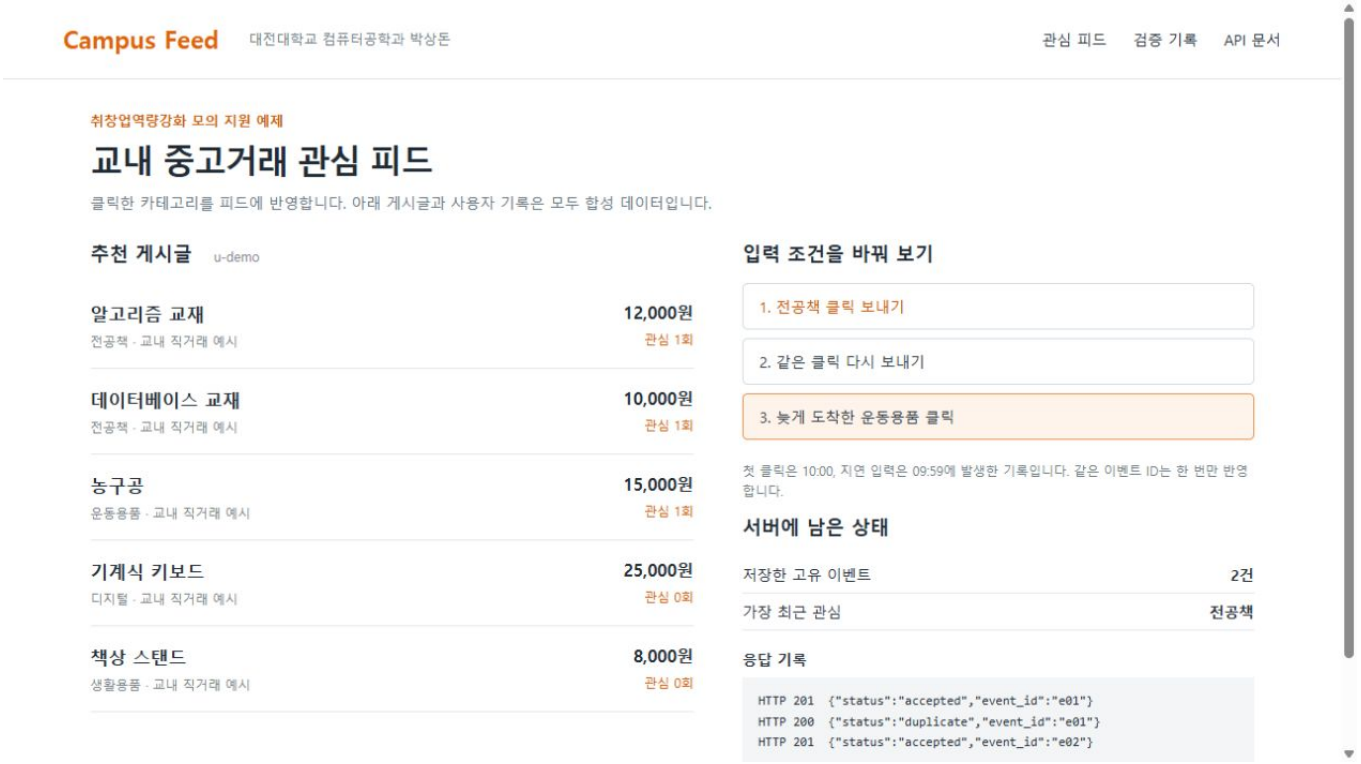
PORTFOLIO / PROJECT

교내 중고거래 관심 피드

당근 신입 백엔드 지원 준비 · Python / FastAPI / SQLite · 2026.09.06

클릭 기록을 피드에 반영하고,
중복 입력과 늦은 도착에도 집계를 일관되게 유지했습니다.

지원 직무: Software Engineer, Backend - 피드 (ML Data Platform, 신입)
사용 상황: 전공책을 본 학생에게 관련 게시글을 먼저 보여주는 교내 거래를 가정했습니다.



실제 실행 화면. 요청 3회 → 고유 이벤트 2건. 늦은 운동용품 기록 이후에도 최신 관심은 전공책입니다.

이번 예제에서 구현한 범위

구현·검증	확인한 결과
API와 데이터 처리	클릭 저장, 입력 검증, 카테고리 집계, 피드 정렬
실패 조건과 성능 실험	12개 테스트 통과 / SQL 조회 p95 5.304 → 0.087ms
담당 범위	이 수업 예제의 API·DB·화면·테스트·측정 코드 전체

합성 게시글 6개와 합성 이벤트로 검증했습니다. 실제 학생 대상 운영, 인증, 결제·채팅, 추천 모델 학습은 이번 범위에 포함하지 않았습니다.

한 번 저장하고, 함께 반영하기

중복 재전송 · 늦은 입력 · 중간 실패를 코드와 테스트로 확인

같은 요청이 다시 와도 집계는 한 번

event_id를 고유 키로 두고 이미 저장된 본문과 비교합니다. 같은 내용이면 duplicate(200), 같은 ID에 다른 내용이면 충돌(409)로 응답합니다. 이벤트 저장과 사용자 요약 갱신은 하나의 BEGIN IMMEDIATE 트랜잭션으로 묶었습니다.

입력 순서	입력	실제 응답·상태
1	e01 / 전공책 / 10:00	201 accepted / 고유 이벤트 1건
2	e01 / 전공책 / 10:00	200 duplicate / 여전히 1건
3	e02 / 운동용품 / 09:59	201 accepted / 2건, 최신 관심 전공책

최신 관심은 도착 순서가 아닌 발생 시각으로 정합니다. 발생 시각이 같으면 event_id 순서를 사용해 결과가 일관되게 결정되도록 했습니다.

12개 테스트가 통과한 범위

검증 묶음	재현한 조건과 확인 결과
중복·충돌·재접속	동일 입력 재전송과 DB 재접속 후에도 1회 반영. 같은 ID의 다른 본문은 거부.
지연·동일 시각·사용자 분리	늦은 입력이 최신 관심을 덮어쓰지 않음. 동일 시각은 일관된 순서, 사용자 집계는 독립.
트랜잭션·동시성	저장 직후 오류 주입 시 원본과 요약을 함께 롤백. 8개 스레드에서 16회 재전송해 1회만 수락.
API·피드·인덱스	잘못된 입력 거부, 이력 없는 사용자는 최신순. 인덱스 전후 결과는 같고 실행 계획은 변경.

근거: store.py / test_app.py / evidence/verification.json / tests.txt
테스트 실패 0건. 의존성의 폐기 예정 기능 경고 2건은 원본 로그에 남겼습니다.

조회가 느린 이유를 확인한 뒤 인덱스 추가

동일 데이터·동일 SQL 비교 · 아래 시간은 HTTP 응답 시간이 아닙니다

통합 p95 5.304ms → 0.087ms

SQL 조회 p95 약 98.4% 감소. 모든 측정에서 조회 결과가 같고 오류는 없었습니다.

실험 조건

합성 이벤트 100,000건 / 사용자 1,000명 / seed 20260906
조건당 300회 × 3회차 = 900개 측정값, 회차·조건별 워밍업 30회.
한 프로세스·한 SQLite 연결에서 순차 실행. execute와 fetchall만 측정.
회차별 순서: 기준→인덱스 / 인덱스→기준 / 기준→인덱스.

회차	인덱스 없음 p95	인덱스 있음 p95
1	4.3541ms	0.0843ms
2	4.6296ms	0.0823ms
3	6.4508ms	0.0901ms
통합 900회	5.3043ms	0.0866ms

p95는 정렬한 측정값의 95% 지점입니다. 통합 값은 회차별 p95의 평균이 아니라 900개 원본 측정값으로 계산했습니다.

실행 계획으로 원인 확인

변경 전: SCAN events + 임시 B-tree 집계
변경 후: SEARCH events USING COVERING INDEX
인덱스: events(user_id, category, occurred_at)

사용자로 좁혀 읽을 수 있게 되면서 전체 스캔이 사라졌습니다. 인덱스 공간은 약 3.20MiB 늘었습니다. 쓰기 비용과 동시 요청 성능은 별도로 측정해야 합니다.

환경: Windows 11 / Intel Core Ultra 9 285K / 논리 CPU 24개
Python 3.12.14 / SQLite 3.53.1 / 측정일 2026.09.06
원본: evidence/benchmark.json, latency.csv (1,800행)
범위: 로컬 캐시가 워밍업된 SQL 조회. 네트워크·HTTP·동시 부하·쓰기 지연 제외.

코드와 기록으로 다시 확인할 수 있게

실행 코드 묶음: campus-feed_실행예제.zip

파일을 열어 확인할 위치

파일	확인할 내용
app.py / store.py	요청 검증, 관심 집계, 중복 처리와 트랜잭션
test_app.py / verify.py	12개 시나리오와 테스트 기록 생성
benchmark.py / evidence/	데이터 생성, 3회차 측정, 실행 계획, 원본 CSV
README.md / web/	설치·재현 방법과 실제 화면

Python 3.12 가상환경에서 실행

```
python -m pip install -r requirements.txt
python -m uvicorn app:app --host 127.0.0.1 --port 8765
python verify.py
python benchmark.py
```

브라우저: http://127.0.0.1:8765 / API 문서: /docs
가상환경 생성부터의 명령은 README에 있습니다. 재측정하면 evidence 결과가 갱신됩니다.

공고의 요구와 연결한 근거

요구를 읽고 정한 준비 항목	이번 결과물
컴퓨터공학 기초·코드 품질	트랜잭션과 인덱스의 선택 이유, 입력 검증
완성한 개발 경험	API부터 화면·실행 방법까지 연결한 로컬 예제
측정 가능한 개선	같은 SQL의 전후 비교, 실행 계획과 원본 기록

다음 준비: 실제 사용자 상황 확인 → 인증·권한 → PostgreSQL 이식 → 배포와 HTTP 부하 실험. 현재 자료는 모의 지원용이며 실제 운영 경험이나 추천 정확도 향상을 주장하지 않습니다.

공식 자료

당근 채용공고 · 5823087003 (2026.09.06 확인)
SQLite 공식 문서 · Query Planning
AWS Architecture Blog · Karrot feature serving (2025.08.14)