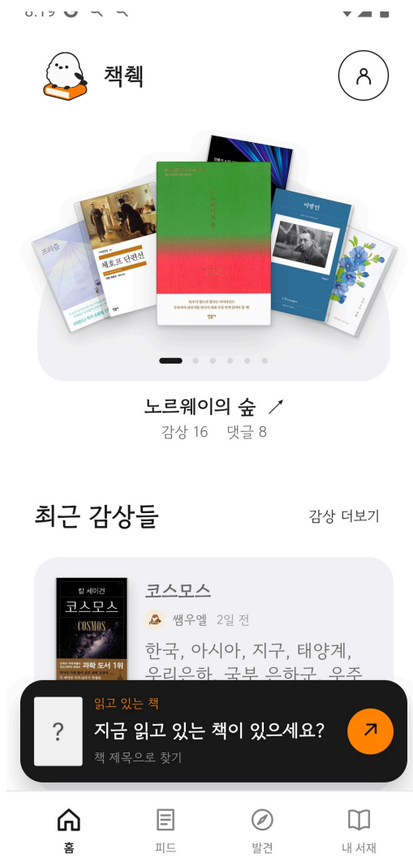
책췍 github.com/woowacourse-teams/2026-chaekchaek

네 번의 실험으로 감상에서 추천까지

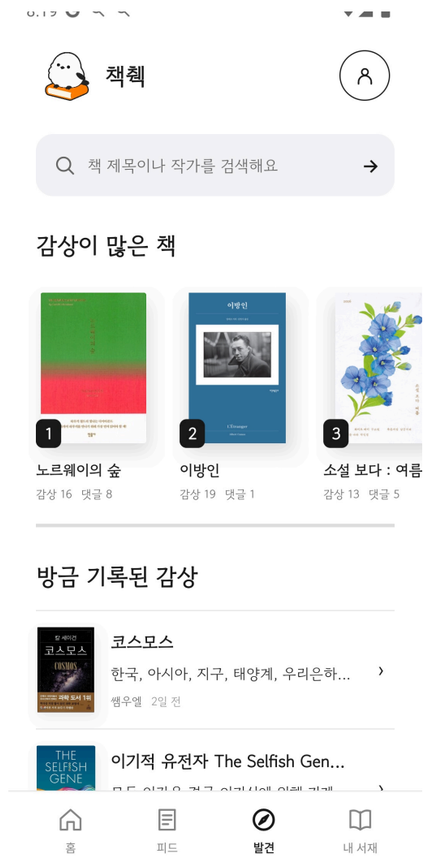
우아한테크코스 팀 프로젝트 / 2026.08 - 현재
실험 설계, AI 활용 구현, 홍보, 인터뷰, 결과 해석

책췍은 읽은 책의 감상을 익명 또는 닉네임으로 남기고 다른 사람과 나누는 서비스입니다.

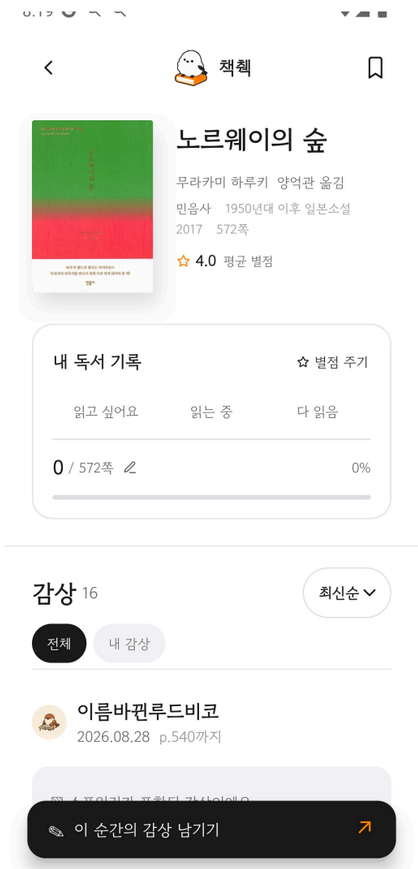
사람들이 어떤 조건에서 감상을 남기는지 확인하기 위해 실험을 설계하고 Codex로 웹 페이지를 구현했습니다. SNS와 커뮤니티에 직접 홍보하고, 참여 기록과 인터뷰를 바탕으로 다음 실험의 질문과 참여 방식을 정했습니다.



홈 화면



발견 화면



도서 상세 화면



도서 상세 화면 - 감상

1차 실험

장르별 반응 탐색, 책의 친숙함이 영향

여섯 장르 중 반응이 활발한 분야를 찾으려 했지만, 반응은 팀이 고른 책 가운데 유명하거나 익숙한 책에 몰렸습니다. 모르는 책이 많아 아는 책에만 감상을 남겼다는 답을 듣고 미독서자에게 맥락을 주기로 했습니다.

실험 페이지: chaekchaek.vercel.app

2차 실험

맥락 제공, 미독서자 제출은 여전히 적음

발췌문 6쪽과 기존 감상을 제공하고 대조군 없이 읽음 여부별 반응을 비교했습니다. 미독서자 제출은 이방인 32개 ID 중 0개, 사랑의 단상 23개 ID 중 1개였고, 작성 부담을 직접 들어 입력을 줄여 보기로 했습니다.

실험 페이지: 이방인 · 사랑의 단상

3차 실험

입력 부담 줄이기, 한 줄 감상은 남았지만 교류는 적음

이모지와 투표, 질문과 선택 사항인 한 줄 감상을 시도했습니다. 한 줄 제출은 21개 ID였지만 좋아요 또는 답글은 3개 ID에 그쳐, 질문 방식은 앱에 도입하고 교류 대신 추천 수요를 확인하기로 했습니다.

실험 페이지: chaekchaek-ex.vercel.app

4차 실험

추천 수요 확인, 추천 요청 글의 반응이 더 많음

독서 커뮤니티 글 5,526건을 반응 수를 가린 채 분류하고 같은 책으로 감상 글과 추천 요청 글을 짝지어 올렸습니다. 고유 반응자 중앙값은 추천 요청 4명, 일반 질문 3명, 감상 2명이었고 추천 방향을 다음 프로토타입으로 검증하기로 했습니다.

사용 기술 Kotlin / Jetpack Compose / Compose Multiplatform

책췍 chaekchaek.vercel.app/experiments/stranger?book=stranger

맥락을 제공해도, 감상 작성은 적었다

2차 실험

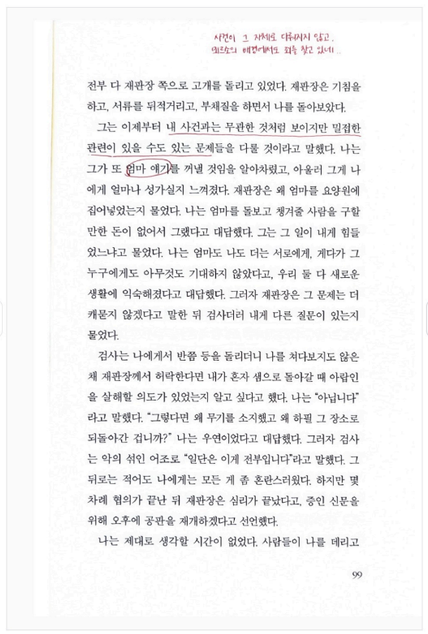
가설: 책을 읽지 않아도 발췌문과 다른 사람의 감상을 보면 자기 생각을 남길 수 있을 것이다.

배경을 이해하도록: 이방인의 앞선 줄거리

앞선 줄거리

> 뫼르소는 범죄 사건에 연루되어, 피의자 신분으로 법정 재판에 참석하게 된다. 재판이 시작되었지만, 뫼르소의 '범죄 행위' 그 자체보다는 지난 어머니 장례식에서의 태도와 평소 행실을 끌어와 그를 비난하기 시작한다.

내용을 접하도록: 발췌문 6쪽



전부 다 재판장 쪽으로 고개를 돌리고 있었다. 재판장은 기침을 하고, 서류를 뒤적거리고, 부채질을 하면서 나를 돌아보았다.

그는 이제부터 내 사건과는 무관한 것처럼 보이지만 밀접한 관련이 있을 수도 있는 문제들을 다룰 것이라고 말했다. 나는 그가 또 엄마 얘기를 꺼낼 것임을 알아차렸고, 아울러 그게 나에게 얼마나 성가실지 느껴졌다. 재판장은 왜 엄마를 요양원에 집어넣었는지 물었다. 나는 엄마를 돌보고 챙겨줄 사람을 구할 만한 돈이 없어서 그랬다고 대답했다. 그는 그 일이 내게 힘들었느냐고 물었다. 나는 엄마도 나도 더는 서로에게, 게다가 그 누구에게도 아무것도 기대하지 않았다고, 우리 둘 다 새로운 생활에 익숙해졌다고 대답했다. 그러자 재판장은 그 문제는 더 캐묻지 않겠다고 말한 뒤 검사더러 내게 다른 질문이 있는지 물었다.

검사는 나에게서 반쯤 등을 돌리더니 나를 쳐다보지도 않은 채 재판장에게 허락한다면 내가 혼자 샘으로 돌아갈 때 아랍인을 살해할 의도가 있었는지 알고 싶다고 했다. 나는 "아닙니다"라고 말했다. "그렇다면 왜 무기를 소지했고 왜 하필 그 장소로 되돌아간 겁니까?" 나는 우연이었다고 대답했다. 그러자 검사는 악의 섞인 어조로 "일단은 이게 전부입니다"라고 말했다. 그 뒤로는 적어도 나에게는 모든 게 좀 혼란스러웠다. 하지만 몇 차례 협의가 끝난 뒤 재판장은 심리가 끝났다고, 증인 신문을 위해 오후에 공판을 재개하겠다고 선언했다.

나는 제대로 생각할 시간이 없었다. 사람들이 나를 데리고

99

1 / 6

다른 생각을 보고 자기 감상을 남기도록

함께 나눈 감상 2

이 책, 어떻게 읽었나요? 짧게 남겨도 좋아요.

남기기

가끔 삶이 거대한 어항 같다는 생각이 듭니다. 인간이 만든 법과 도덕 안에서 살고, 감정까지 정해진 방식으로 보여줘야 한다는 점에서 우리는 어항 속 금붕어와 크게 다르지 않은지도 모릅니다. 세상에 정해진 의미가 없다면 허무할 수도 있지만 무의미에서 시작되는 의미만큼은 누구도 강요할 수 없는 나만의 것이기도 합니다. 이방인은 당연하다고 믿어온 삶의 규칙과 감정이 정말 내 것인지 묻게 만드는 책입니다.

좋아요 0 답글 0

의욕이 없고 권태를 느끼는 주인공
소시오패스와 다를 바 없지만 사실 누구나 가져봤을 만한 감정과 생각들
결국 사회적 규범이라는 가면 아래에 있는 나의 모습이었다
감정과 사건의 본질보다 도덕성에 집착하는 등장인물들은 현실과 다를 바 없어보였고
사제 앞에서 속마음을 쏟아내는 마지막 장면은 한마디 한마디가 인상적이었다
모두가 가면을 쓴 세상에서 민낯의 주인공은 이방인이었다

좋아요 0 답글 0

관찰 결과

안 읽었어요를 선택한 방문자의 감상 제출

이방인

0 / 32

감상 제출 ID / 미독서 선택 ID

사랑의 단상

1 / 23

감상 제출 ID / 미독서 선택 ID

설계의 한계

모두에게 맥락 제공, 별도 대조군 없음, 사전 성공 기준 없음

읽음 여부별 반응을 비교했으며, 맥락 제공 자체의 효과는 판단할 수 없었습니다. 감상을 남기기 편해졌다는 의견과 앱의 책 읽기 기능을 요청하는 반응도 들었습니다.

사용자 피드백

앱과 실험 사용자에게서 직접 들은 작성 부담

책과 휴대폰을 번갈아 봐야 함, 인용구의 쪽수 기억하기, 글쓰기 자체의 피로

다음 실험

작성 부담 완화

이해할 맥락과 함께, 생각을 떠올리고 입력하는 부담도 줄여보자.

책췍 chaekchaek-ex.vercel.app/

한 줄 감상은 남겼지만, 타인과의 교류는 적었다

3차 실험

두 가지 접근: 입력 부담을 줄이기 + 질문과 상황으로 생각할 단서 제공하기

01 이모지와 투표



입력 부담을 줄여 가볍게 참여하도록

02 질문과 한 줄 감상



생각할 단서를 주고, 원하면 150자 이내로 작성

관찰 결과

책 페이지 방문 60개 ID

2026.10.06 - 2026.10.11 / 행동별 고유 방문자 ID

이모지 또는 투표	16
한 줄 감상 제출	**21**
타인 감상 노출	37
좋아요 또는 답글	**3**

항목 간 ID 중복 있음. 실제 사람 수나 순차 전환 단계로 해석하지 않음.

도입 판단

한 줄 감상 제출 21개 ID

텍스트 제출이 예상보다 많았습니다. 질문과 상황을 주는 방식을 앱에 도입하기로 했고, 현재 준비 중입니다.

수요 재검토

좋아요 또는 답글 3개 ID

감상을 쓰는 것과 타인과 교류하는 것은 별도로 살펴봐야 한다고 판단했습니다.

해석의 한계

이모지 도입 보류와 실험 간 비교 한계

이모지는 기대보다 반응이 낮아 도입하지 않기로 했습니다. 실험별 참여자와 조건이 달라 참여 증가의 원인을 특정하지 않았습니다.

다음 검증

감상 교류보다, 자신에게 맞는 책 추천에 관심이 더 클까?

추천에 대한 관심을 4차 실험에서 독서 커뮤니티 데이터로 확인했습니다. 추천 요청 글이 감상 글보다 반응을 많이 받아, 추천 방향을 다음 프로토타입의 검증 대상으로 정했습니다.

책췍

추천 요청 글은 감상 글보다 반응이 많았다

4차 실험

질문: 독서 커뮤니티에서 책 추천 요청은 감상 공유보다 더 많은 반응을 얻는가?

① 기존 글 관찰

- 독서 커뮤니티 28일치 글 5,526건
- 반응 수를 가린 채 추천 요청·감상·기타로 분류
- 일반 질문 글 200건을 대조군으로 함께 집계
- 1차 지표: 글쓴이를 뺀 고유 댓글 작성자 수

② 짝지은 게시

- 실제로 읽은 같은 책으로 감상 글과 추천 요청 글을 한 쌍씩
- 노인과 바다, 1984 두 쌍을 비슷한 시간대에 게시
- 링크와 서비스 소개 없이 반응만 비교

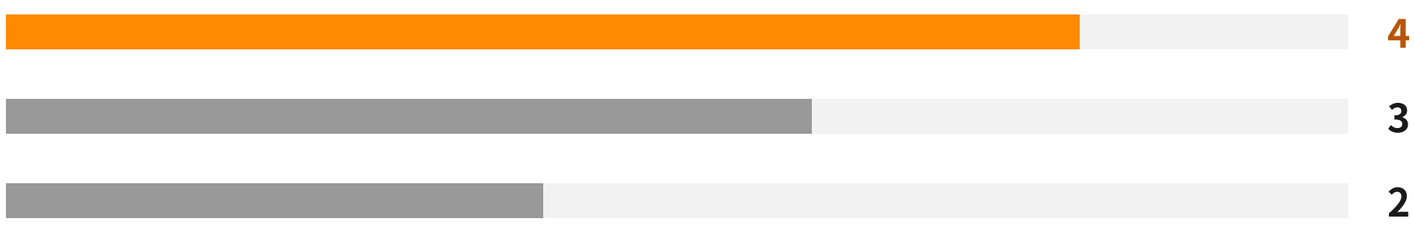
2026.10.17 - 10.18 게시 / 10.19 중간 집계

2026.09.15 - 2026.10.12 / 새 글을 올리지 않고 관찰

관찰 결과

글 유형별 고유 반응자 중앙값

독서 커뮤니티 28일 / 글쓴이를 뺀 고유 댓글 작성자 수



분류 신뢰도

세 분류자가 따로 분류한 150건의 일치도 확인

추천 요청·감상·기타에서 50건씩 뽑은 150건을 본인, Claude, 가설을 모르는 다른 AI가 따로 분류했습니다. Fleiss kappa는 0.80이었고, 추천 요청 분류는 모든 쌍에서 kappa 0.82 이상이었습니다.

판정

판정 규칙대로 추천 방향을 다음 단계로

추천 요청 중앙값이 감상의 1.5배 이상, 추천 요청 글 수가 감상의 1/3 이상, 일반 질문 대조군 기준 충족, 두 쌍 모두 추천 요청 반응이 많음을 모두 충족했습니다. 서비스 방향을 바꾼 것이 아니라 다음 프로토타입의 검증 대상으로 정했습니다.

해석의 한계

관계는 확인했지만 인과는 말하지 않음

무작위 배정이 아니라 글 길이와 작성 시각 차이가 섞입니다. 추천 요청은 일반 질문보다도 반응이 많았지만 그 차이는 감상과의 차이보다 작습니다. 짝지은 게시는 2쌍뿐이고 작성자 답글 등 계획 이탈이 있어 반대 신호가 없었다는 데까지만 해석했습니다.

다음 검증

추천을 받은 사람이 실제로 쓰는가

커뮤니티 반응은 앱 사용 의향이 아니므로, 사람이 직접 책을 골라 보내는 컨시어지와 가벼운 웹 프로토타입으로 추천을 실제로 읽는지 확인하는 실험을 팀에 제안하려 준비하고 있습니다.

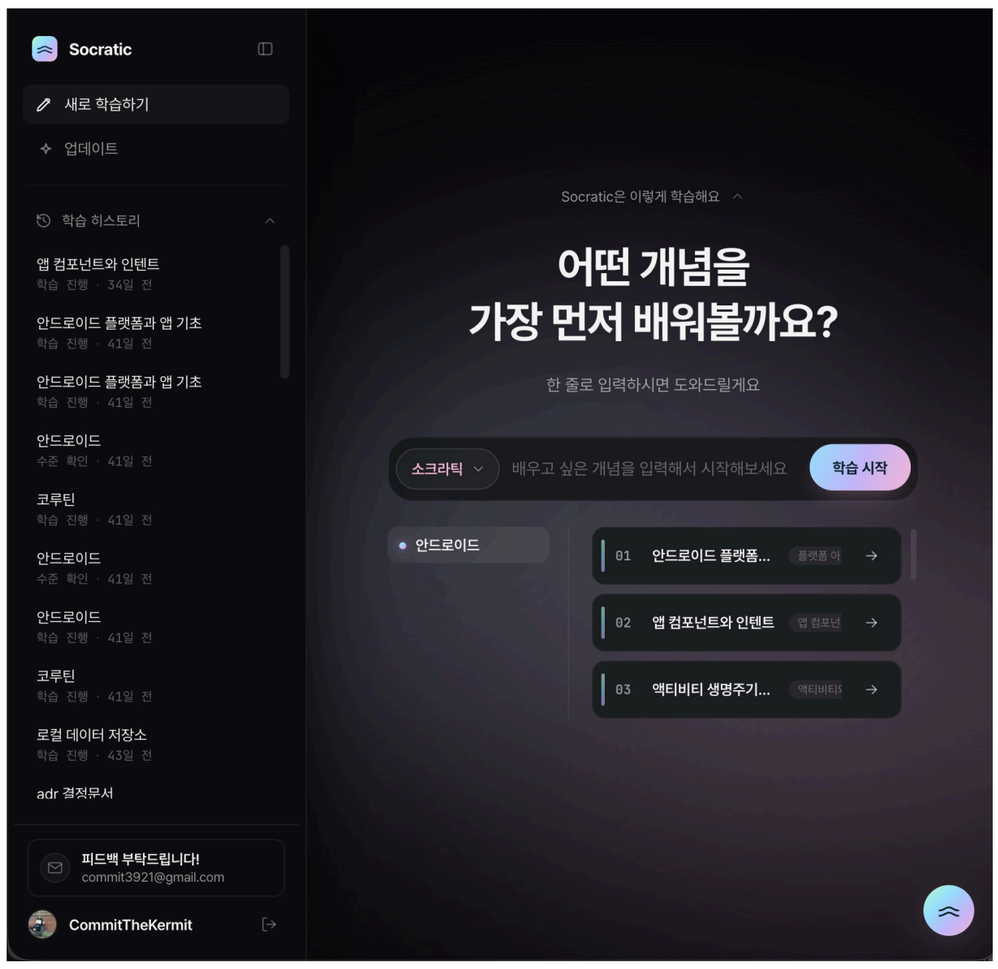
소크라틱 런 socratic-learn-web.web.app

로그인 전에 도망가는 사람들, 학습 질문만 필터

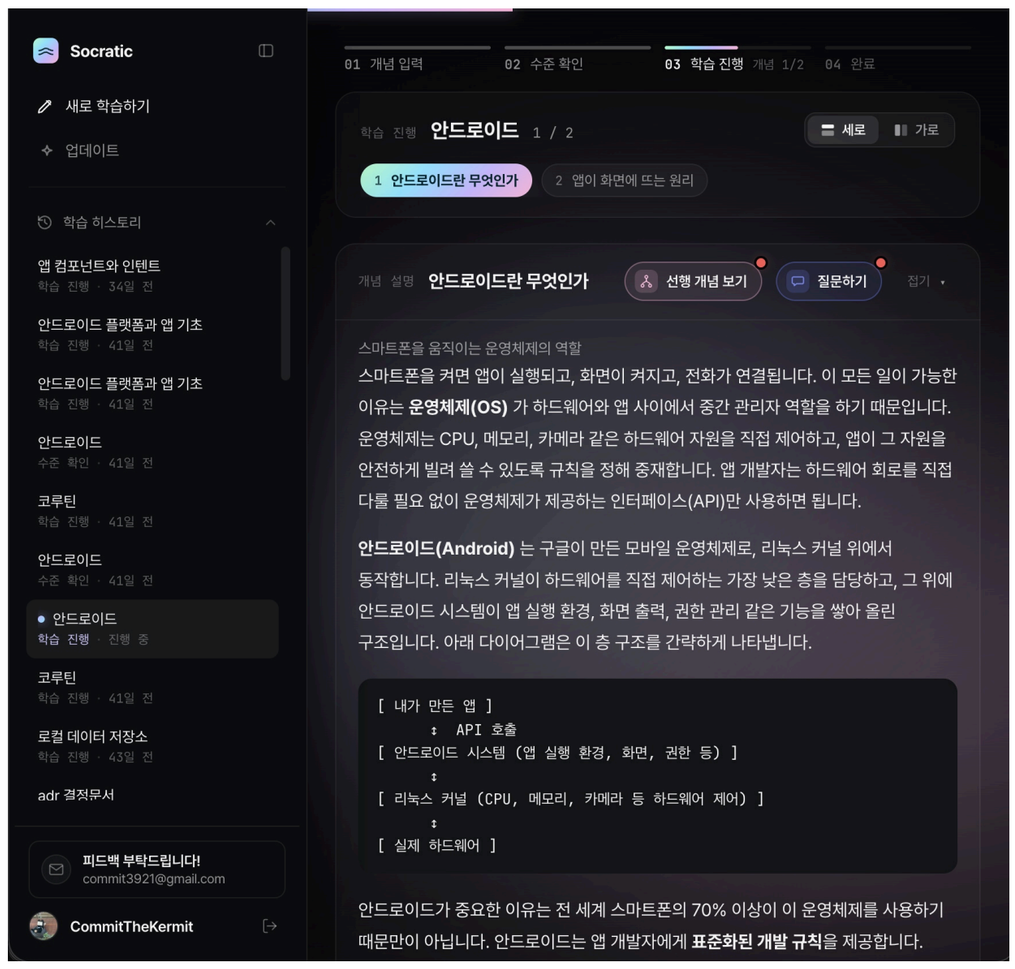
4명 해커톤 팀 개발 후 개인 운영 / 2026.05 - 현재
웹 서비스 운영, 피드백 기반 개선

배우고 싶은 주제를 입력하면 설명을 쏟아내는 대신 질문부터 던집니다. 진단 문항으로 아는 것과 모르는 것의 경계를 찾고, 주제를 작은 개념으로 나눠 배울 순서를 그린 다음, 한 개념씩 직접 답하며 이해를 확인합니다. 답변을 채점해 복습, 선행 학습, 다음 단계, 심화 중 지금 필요한 길을 추천합니다.

코루틴 사용 근거를 묻는 코드 리뷰를 계기로, 설명 뒤 확인 질문에 답하며 이해를 점검하는 Claude Code 스킬을 만들었습니다. 동료에게 공유한 뒤 스킬은 편리하지만 CLI에서 사용하기 불편하다는 반응을 받아 해커톤에서 웹 서비스를 제안했습니다. 4명 팀으로 개발한 뒤에는 혼자 운영하며 피드백과 오류를 개선하고 있습니다.



메인 화면



학습 화면

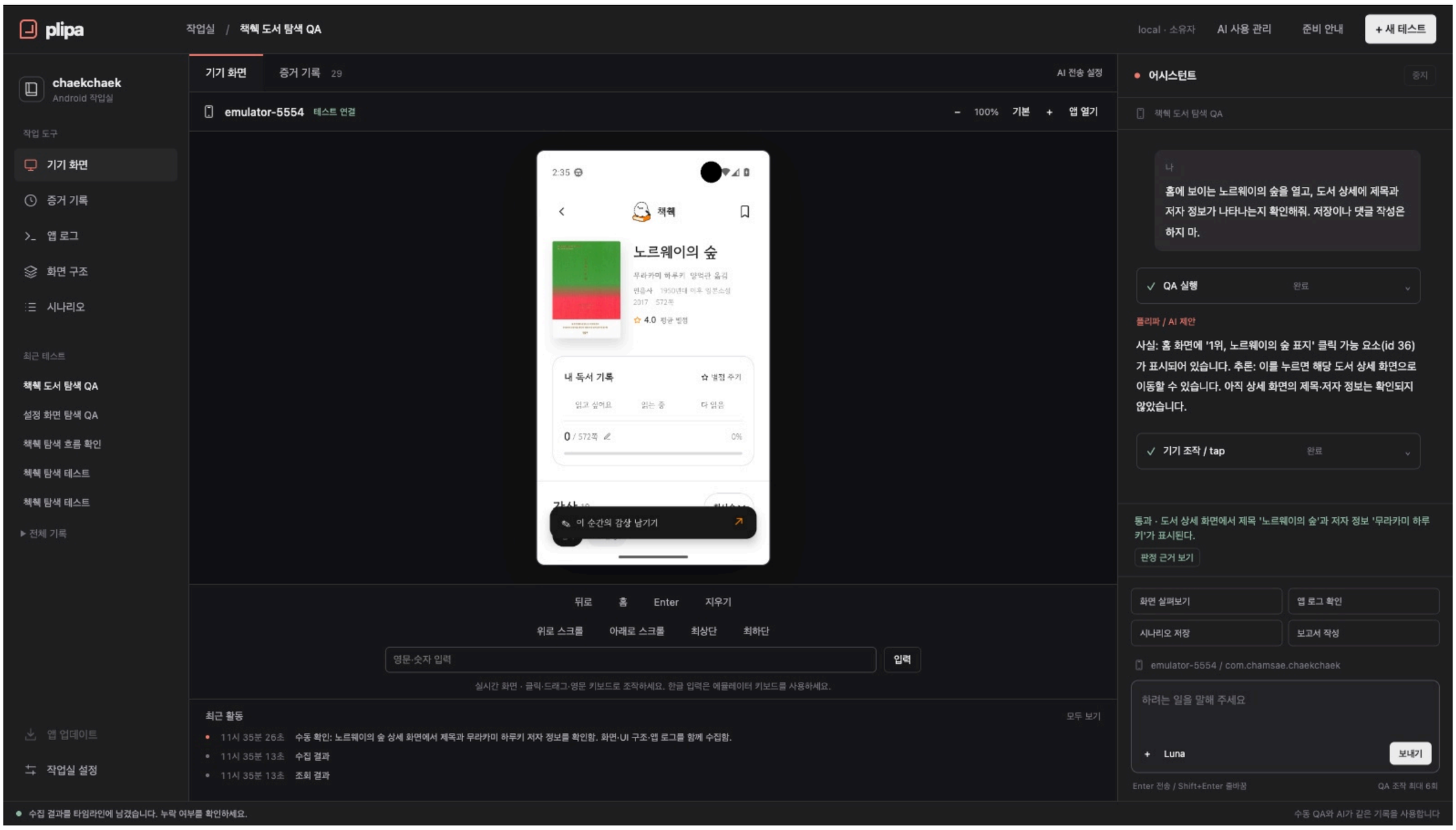
문제	**로그인 제한 기간의 학습 진입률 11.5% 확인** AI 호출 비용을 제가 부담해야 해서, 무제한 호출이 걱정돼 로그인한 사용자만 질문하게 막았습니다. 계산이 아니라 불안이 근거였습니다. GA4 퍼널로 17일을 보니 학습 진입은 11.5%였고, 로그인은 제가 세운 장벽이라 가장 먼저 의심할 지점이었습니다.
판단	**로그인 제한을 호출 상한으로 대체** uid 기준 일일 150회·분당 20회 상한을 Anthropic 호출에만 걸고, 카운터는 트랜잭션으로 올려 동시 호출에도 새지 않게 했습니다. 저장소 장애 시에는 통과시키고 최종 방어선은 API 결제 한도에 뒀습니다. 기록 저장은 익명 인증으로 옮기고, 로그인하면 linkWithPopup으로 같은 uid를 계정으로 승격합니다.
구현	**실제 로드맵을 기준으로 학습 선택지 구성** 진단 → 로드맵 → 학습 → 분기 → 완료 흐름을 만들었습니다. '다음 단계로 이동'은 LLM이 없는 로드맵을 지어내서, 프론트가 실제 로드맵으로 선택지를 만들게 했습니다. 자주 찾는 주제는 난이도별 로드맵을 미리 두었고(Android 로드맵은 직접 작성), 주요 흐름은 단위 테스트와 Playwright로 묶었습니다.
결과	**변경 전후 학습 진입률 11.5%, 26.7% 관측** 변경 후 학습 진입률은 26.7%로 관측됐지만 방문자는 15명이었습니다. 로그인 화면 이벤트가 없어 page_view에서 학습 시작까지를 대체 지표로 사용했습니다. 표본이 작고 로그인 단계의 이탈을 직접 측정하지 못해, 로그인 제한 해제의 효과로 단정하지 않았습니다.
운영 개선	**사용자 피드백 반영과 학습 진행 오류 수정** 공개 후 접수한 콘텐츠 중복, 혼란스러운 번호 체계, 수식 표시 문제에 대응해 중복 단계 생성을 억제하고 번호 표시를 정리했습니다. 미니 로드맵과 진행 바, KaTeX 수식 렌더링도 적용했습니다. 별도로 직접 사용하며 발견한 평가·분기 건너뛰기와 진행 교착을 수정하고, 학습 중 질문하기와 답변 재제출을 추가했습니다.
사용 기술	React / Vite / Firebase (Auth · Firestore · Functions · Hosting) / Claude API / Vitest / Playwright

플리파 github.com/CommitTheKermit/flippa

기기 없이도 Android 오류를 재현해 넘겨주는 작업실

개인 프로젝트 / 2026.09 - 현재
Android 에뮬레이터 QA 도구 설계와 구현

브라우저에서 Android 에뮬레이터를 조작하고 QA 증거를 공유하는 AI QA 작업실입니다. 사람이 직접 화면을 만지거나 Codex에게 테스트를 맡기면, 조작과 화면 구조, 앱 로그가 같은 타임라인에 쌓입니다. 고른 기록은 재현 순서와 기대 결과, 실제 결과를 붙여 QA 묶음으로 내보내 동료에게 전달합니다.



작업실 전체 화면

문제

기기 없는 팀원의 "안 된다" 보고

팀원 대부분이 Android 기기가 없어 오류 보고에서 재현 절차와 로그를 얻기 어려웠습니다. 당근 밋업에서 본 QA 도구 slipper에서 착안해, 브라우저만으로 에뮬레이터를 조작하고 증거를 묶어 넘기는 작업실을 만들었습니다.

구현

화면 스트림, AI 조작 승인, QA 증거 묶음

Emulator gRPC로 화면을 스트림하고 터치와 키 입력을 바로 전달합니다. Codex에게 테스트를 지시하면 전송·삭제처럼 확인이 필요한 조작은 승인을 받은 뒤 실행하고, 수동과 AI 조작을 화면 구조·로그와 함께 같은 타임라인에 남깁니다. 고른 기록은 재현 순서·기대 결과·실제 결과와 함께 QA ZIP으로 내보냅니다.

이슈

서명이 달라 막힌 Google 로그인과 앱 업데이트

서버에서 빌드한 책헥 APK로는 Google 로그인이 실패하고 업데이트도 INSTALL_FAILED_UPDATE_INCOMPATIBLE로 거부됐습니다. 서버 디버그 키로 서명돼 OAuth 등록 인증서와 달랐던 탓으로, 등록 키로 재서명해 로그인을 확인했습니다. 이후 등록 키로 빌드하고 설치 전 인증서 지문을 검사하며, 서명이 다르면 데이터 삭제 확인을 거쳐서만 재설치하게 했습니다.

한계

실기기와 사용자별 분리는 다음 과제

에뮬레이터 전용이라 실기기에서만 나타나는 문제는 재현하지 못합니다. 운영 중인 홈서버는 에뮬레이터를 골라 쓰는 방식이라 사용자별 로그인 정보가 분리되지 않아, 사용자별 실행 환경 분리를 다음 과제로 두었습니다. 팀원의 실제 사용 결과는 아직 확인 중입니다.

사용 기술

Python / JavaScript / Android Emulator gRPC / ADB / Codex CLI

주식회사 네모반듯, 툭 [tuk.im](tuk.im)

너무 뜨거운 앱, 포인트를 퍼다주는 앱

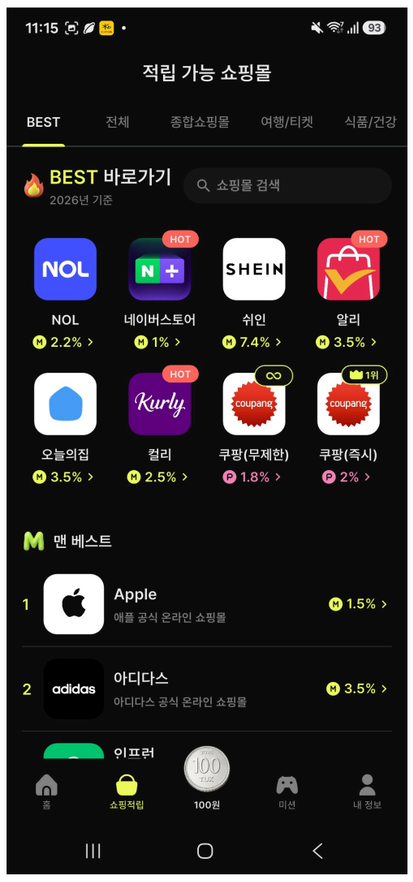
주식회사 네모반듯 / 2025.09 - 2026.02
단독 개발과 운영

툭은 제휴 쇼핑몰에서 구매하면 캐시백을, 미션과 참여형 콘텐츠로 포인트를 적립해 출금하는 리워드 커머스 앱입니다.

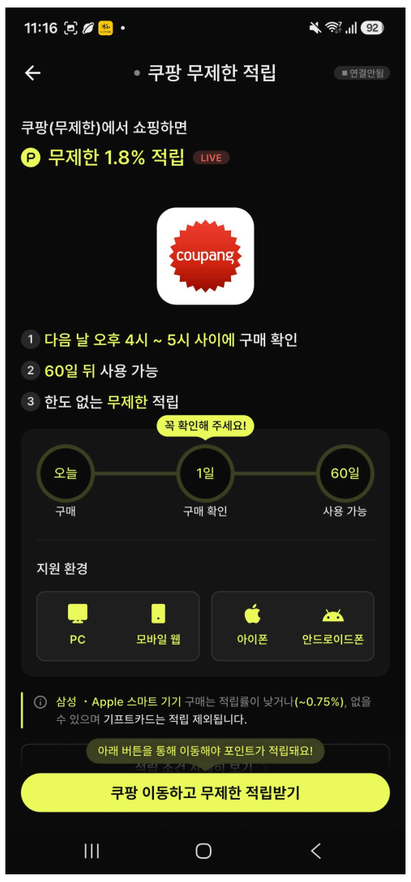
9월에 입사하자마자 외주 개발자에게 인수인계를 받아, 외주사가 6개월간 개발한 앱을 혼자 이어 개발하고 운영했습니다.



메인 화면



쇼핑몰 목록 화면



쇼핑몰 상세 안내

발열

손에 쥐기 뜨거운 앱, 화면의 부담을 하나씩 줄였습니다

앱을 켜 두면 손으로 느낄 만큼 발열이 심했습니다. 과도한 동적 요소부터 줄이고, 캐러셀 자동 순회, 높은 이미지 해상도, 동영상 용량과 화질, 자동 재생을 차례로 줄였습니다. 여러 요소의 부담을 낮춘 결과 체감 발열이 줄었습니다. 온도나 자원 사용량을 수치로 측정하지는 않아 특정 요소 하나를 원인으로 단정하지 않았습니다.

재가입 보상

가입 보상 반복 지급 차단과 지급분 환수

출납 모니터링 중 탈퇴와 재가입을 반복하면 가입 보상 500포인트가 계속 지급되는 문제를 발견했습니다. 본인인증 식별값이 유지되지 않아 같은 사람을 새 사용자로 처리하고 있었습니다. 식별값이 유지되도록 수정해 보상 반복 지급을 막고, 반복 지급된 포인트를 환수했습니다.

회고

기능 추가 시 발열 상태 확인

이 일을 겪은 뒤에는 기능이 동작하는 것뿐 아니라, 실행 중 발열이 어떻게 달라지는지도 살피며 개발했습니다. 특히 움직임이나 미디어를 추가할 때는 화면 효과와 함께 기기에 주는 부담을 확인하며 조심스럽게 적용했습니다.

개발과 운영

앱과 백엔드 개발, 운영 환경 유지보수

AI 도구를 활용해 Flutter 앱과 NestJS 백엔드의 기능을 개발하고, 네이버 클라우드의 Kubernetes 운영 환경과 PostgreSQL을 유지보수했습니다.

사용 기술

Flutter / Dart / Riverpod / flutter_hooks / Firebase / NestJS / Kubernetes / PostgreSQL

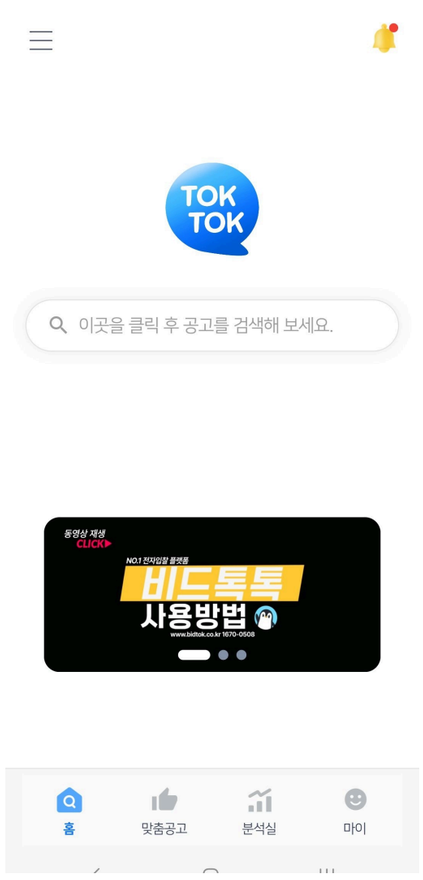
뉴지니(헤스티아, 비드톡톡) play.google.com/store/apps/details?id=kr.co.newjini.bidtoktok

하나의 코드베이스로 협력사 브랜드 앱 16개 배포

뉴지니 / 2024.01 - 2025.08
Flutter 앱 개발, 구조 설계, 빌드, 배포

비드톡톡 / 2024.01 - 2024.07
전자입찰 공고 탐색과 AI 분석을 제공하는 앱입니다. Flutter로 개발해 Android와 iOS 스토어에 출시했습니다.

헤스티아 / 2024.07 - 2025.08
전자입찰 공고 탐색과 컨설팅 분석을 제공하는 앱입니다. 기존 Kotlin/XML 앱을 Flutter로 포팅하고, 협력사 16곳의 브랜드 앱을 공통 코드와 브랜드별 빌드 설정으로 관리했습니다.



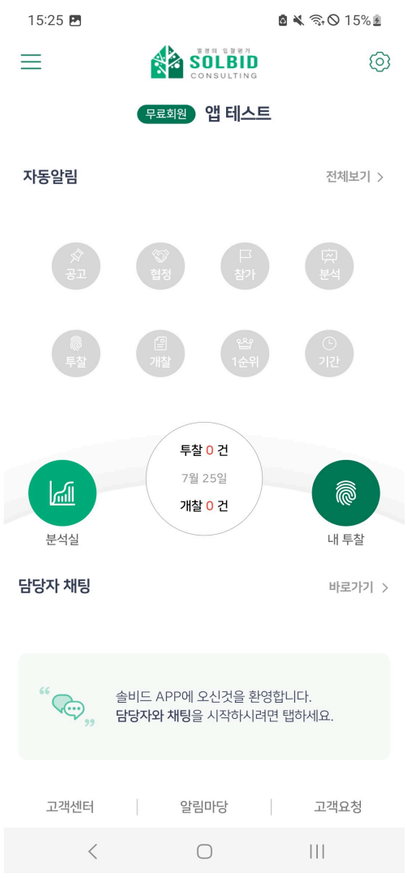
비드톡톡 메인화면



진비드 메인화면



비드윈 메인화면



솔비드 메인화면



비드플러스 메인화면

문제

협력사 증가에 따른 브랜드별 앱 배포 필요

브랜드마다 다른 것이 앱 이름과 아이콘만이 아니었습니다. 협력사가 각자 자기 도메인으로 운영해 API 서버 주소가 달랐고, 음성 안내 파일과 Firebase 프로젝트도 브랜드별로 따로였습니다. 코드를 복사해 앱을 따로 만들면 기능 하나를 고칠 때마다 같은 수정을 브랜드 수만큼 반복해야 했습니다.

구조

빌드 설정을 통한 브랜드별 앱 분리

공통 코드에 브랜드별 빌드 설정을 분리해 앱 이름, 패키지명, 서버 주소, Firebase 설정과 서명 설정을 관리했습니다. 각 앱에는 자기 브랜드 설정만 들어가고, 브랜드가 늘어도 코드베이스를 복사하지 않습니다. 브랜드 추가 절차는 문서로 정리하고 런처 아이콘은 스크립트로 한 번에 생성했습니다.

운영

약 2주 반 동안 브랜드 앱 14개 추가, 브랜드 앱 16개 운영

2024년 7월 공통 구조를 마련한 뒤, 2025년 2월에는 약 2주 반 동안 브랜드 앱 14개를 추가했습니다. 서명 키 3종과 브랜드별 Firebase 설정을 관리했고, FCM data 메시지로 받은 알림은 앱에서 구성해 종류별로 TTS 음성과 아이콘을 적용했습니다. 모바일 담당이 한 명으로 줄어든 뒤에도 같은 코드베이스의 브랜드 앱 16개와 비드톡톡, 관리자·인트라넷 앱의 배포와 스토어 심사를 운영했습니다.

사용 기술

Flutter / Dart / Firebase / Spring / MariaDB