서비스 설계 노트

이광복, AI 네이티브 프로덕트 엔지니어

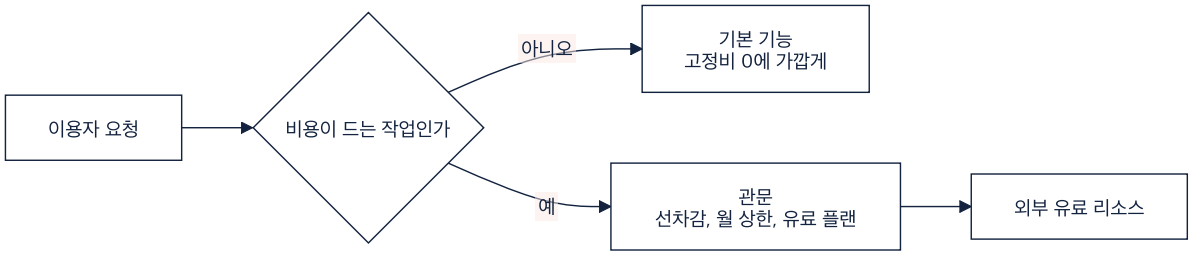
2026년 11월

설계 원칙

네 서비스 모두 개인이나 작은 팀이 운영합니다. 그래서 두 가지 기준을 먼저 정했습니다.

기준	내용
고정비	이용자가 없어도 나가는 비용은 0에 가깝게 둔다
변동비	이용자가 늘면서 커지는 비용이 매출보다 먼저 커지지 않게 막는다

이 두 가지를 지키면 이용자가 적을 때도 적자가 쌓이지 않고, 유료 이용자가 생기는 만큼 손익분기에 가까워집니다.



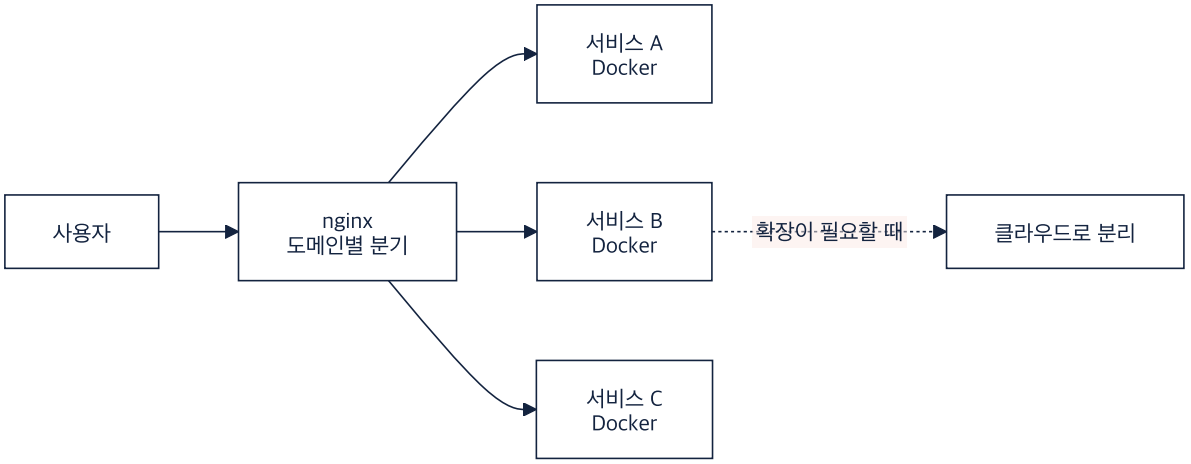
네 서비스에 공통으로 둔 비용 관문

한눈에 보기

서비스	고정비 구조	변동비 통제	수익 구조
LimeLink	요청이 있을 때만 과금되는 서버리스. 상시 서버, 캐시 서버 없음	쌓이는 데이터에 보관 기한. 외부 유료 리소스는 유료 플랜에만	Pro 프로젝트당 월 USD 10, 연 USD 100
LimeNote	자체 서버 한 대. 큐와 결제 원장을 DB 하나에	AI 호출 전에 크레딧 선차감	크레딧 구독 USD 9~99, 충전팩
띠롱	OCI 무료 제공 서버에 합류	AI 월 사용액 상한, 승인제 7일 제공	없음. 비용이 드는 기능만 승인제
독박스	서버 없음. 스토어 계정비만	이용자가 늘어도 늘어나는 운영 비용 없음	전면 광고, 광고 제거 구독 월 1,100원

개인 서버 구성

항목	내용
클라우드	OCI 무료 제공 서버 1대. 4 vCPU, 메모리 24GB, 디스크 180GB
온프레미스	개인 홈서버
실행 방식	서비스마다 Docker 컨테이너로 띄우고, nginx가 도메인에 따라 요청을 나눈다
운영 비용	백엔드 서버 운영 비용 0원에 가까움
확장	서비스별로 나눠 두어, 필요해지면 그 서비스만 클라우드로 분리해 올린다
변경 이력	처음에는 학습 목적으로 k3s와 ArgoCD를 구성했고, 지금은 모두 Docker 기반으로 바꿨다



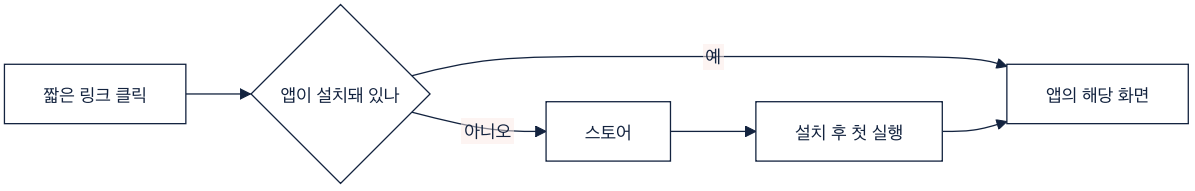
개인 서버의 요청 흐름

LimeLink

항목	내용
서비스	앱 설치 전후로 원하는 화면에 보내 주는 딥링크 서비스
역할	백엔드 리드. 재설계, 이관 계획, 기술 결정
운영	Hellovelop Inc., 취미로 모인 팀, limelink.org
수익	Free, Pro(프로젝트당 월 USD 10 또는 연 USD 100)
이용	글로벌 가입자 1,000명

무엇을 하려고 했나

항목	내용
목표	앱 팀이 딥링크 서버를 직접 운영하지 않아도 되게 한다
기존 문제	FastAPI와 DynamoDB로 운영하던 첫 버전은 개발 환경을 따로 띄울 수 없었고, 데이터 규칙을 코드에서만 지켰다
조건	운영 중인 웹, iOS, Android 연동과 결제를 끊지 않는다
결과	NestJS와 PostgreSQL로 다시 만들었다



LimeLink 링크 흐름

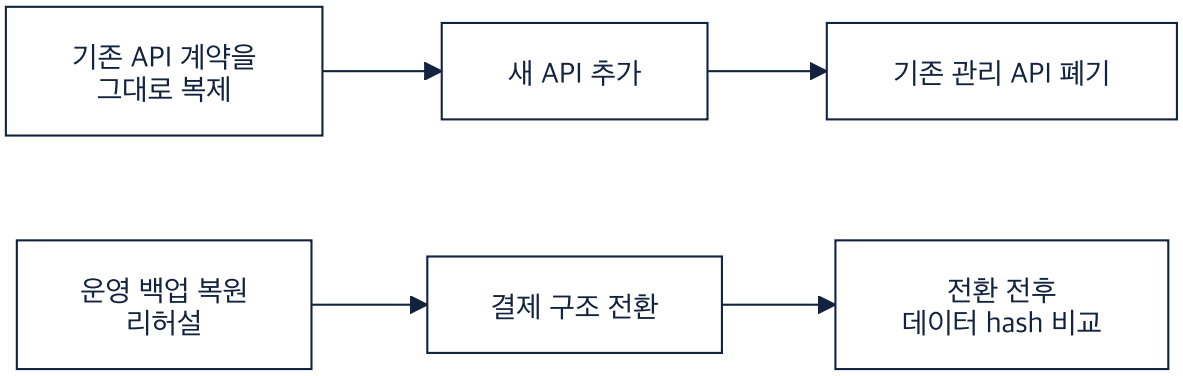
비용 구조

결정	이유
요청이 있을 때만 도는 서버리스 유지	트래픽이 없으면 비용도 없습니다. 함수마다 메모리를 따로 잡았고, 링크 이동 함수는 무거운 프레임워크 없이 DB 연결을 재사용합니다.
Redis 없이 PostgreSQL 하나	세션, 작업 잠금, 결제 알림 중복 제거를 PostgreSQL 테이블로 처리해 캐시 서버가 필요 없습니다.
DB 접속용 네트워크 장비 없음	전용 연결 서비스와 고정 IP 프록시는 둘 다 추가 인프라가 필요해 고르지 않았습니다.
통계는 카운터만 증가	하루 한 번 로그를 조회해 집계하던 배치를 없애고, 이벤트가 올 때 링크별 숫자만 올립니다.
만료 데이터는 예약 작업이 정리	DB 확장 기능은 호스팅 환경에 따라 못 쓸 수 있어, 어디서든 도는 예약 작업으로 정리합니다.

손익분기와 수익 구조

- 요금제는 Pro 하나, 프로젝트 단위 과금입니다. 연간 요금이 월간 12개월보다 USD 20 쌉니다.
- 무료 이용자가 비용을 키우지 못하게 막았습니다.
  - 커스텀 도메인은 구독 중인 프로젝트만 만들 수 있습니다.
  - 매칭 신호와 코드는 24시간, 서버 로그는 90일 뒤 정리합니다.
  - 7일 안에 활성화되지 않은 커스텀 도메인은 외부 리소스까지 지웁니다.
- 구독이 끝나도 기존 커스텀 도메인 링크는 마지막 유료 기간 뒤 1년간 동작하고, 새로 만드는 것만 막습니다.
- 결제와 청구서 발급은 Paddle이 대행합니다.

리스크를 줄인 방식



서비스를 멈추지 않은 이관 순서

상황	한 일
API 교체	기존 API를 복제한 버전을 먼저 띄우고, 새 API를 추가한 뒤 기존 관리 API를 폐기했습니다. 경로, 응답 형식, 쿠키 규격 등 호환 조건 12개를 먼저 정했고 결제 알림 주소는 그대로 뒀습니다.
데이터 이관	덤프와 적재를 스크립트 하나로 묶고 이미 있는 행은 건너뛰게 해, 여러 번 돌려도 안전합니다. 기존 ID는 그대로 옮겼습니다.
스키마 변경	추가, 백필, 검증, 제약 전환의 4단계로 나누고 데이터를 잃는 되돌리기는 막았습니다.
인증 교체	기존 토큰과 새 토큰을 모두 받는 과도기 릴리스를 거쳐 AWS Cognito를 걷어냈습니다.
결제 구조 전환	운영 DB 백업을 로컬에 복원해 리허설한 뒤 적용했고, 전환 전후 결제 데이터 hash를 비교했습니다.
결제 알림	서명을 확인한 원문을 처리 전에 먼저 저장해, 처리 중 문제가 생겨도 다시 처리할 수 있습니다.

가장 고민한 부분

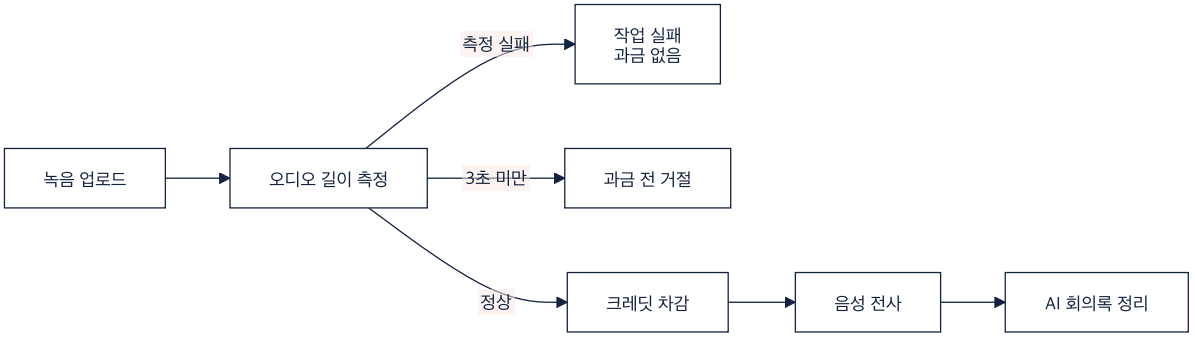
- 고정비가 생기는 리소스를 늘리지 않는 것. 캐시 서버와 네트워크 장비는 대안과 비교한 뒤 넣지 않았습니다.
- 운영 서버가 하나라 웹, iOS, Android, 결제 중 하나만 깨져도 장애였습니다. 스키마, 인증, 결제를 바꿀 때마다 과도기를 두고 한 단계씩 옮겼습니다.

LimeNote

항목	내용
서비스	회의 녹음을 받아써 고칠 수 있는 AI 회의록으로 만드는 웹 서비스
역할	백엔드 리드. 서비스 구조, 녹음에서 회의록까지의 처리 흐름, 결제
운영	Hellovelop Inc., 취미로 모인 팀, limenote.org
수익	무료 월 3크레딧, Starter USD 9, Pro USD 19, Business USD 99, 충전팩 USD 15
이용	마케팅 없이 회사 동료들이 사용

무엇을 하려고 했나

항목	내용
목표	회의가 끝난 뒤 녹음을 다시 듣고 회의록을 정리하는 시간을 줄인다
비용 문제	이용자가 늘수록 전사와 LLM 호출 비용도 같이 는다
원칙	결제한 양보다 AI를 더 쓸 수 없게 만든다



비싼 호출 앞에서 먼저 과금하는 순서

비용 구조

결정	이유
전사와 LLM은 사용량 과금 외부 API	GPU 서버가 없어 이용자가 없을 때 비용도 없습니다. 업체와 모델은 서버 설정만으로 바꿀 수 있고, 실제로 전사 업체를 한 번 바꿨습니다.
큐, 진행률, 크레딧 원장을 PostgreSQL 하나에	메시지 브로커 없이 몰리는 요청을 큐로 받습니다. 운영할 저장소가 DB와 파일 저장소 둘뿐입니다.
작업 처리 서버 분리는 병목이 생길 때	이미지 하나에 진입점 둘을 두어 설정만으로 나눌 수 있습니다. 한때 API 서버에 합쳤다가 운영에서 다시 분리했습니다.
오디오는 브라우저가 저장소로 직접 업로드	API 서버가 큰 파일을 들고 있지 않습니다. 업로드 상한 500MB로 메모리와 저장 비용을 함께 막습니다.
정기 크레딧은 조회 시점에 지급	스케줄러와 배치가 없습니다. 이중 지급은 DB 제약이 막습니다.
AI 실패는 싼 방법부터	일시적 오류는 같은 모델로 재시도하고, 그래도 실패하면 다른 회사 모델로 한 번만 넘깁니다. 처음부터 다른 모델로 다시 보내면 비싸고 결과 품질도 달라집니다.

손익분기와 수익 구조

- 비싼 호출 전에 과금합니다.
  - 길이를 잴 수 없으면 작업을 실패시킵니다.
  - 3초 미만 오디오는 과금 전에 거절합니다. 무음 녹음이 크레딧을 쓰고 빈 회의록을 만드는 일을 막습니다.
  - 같은 작업은 재시도해도 한 번만 과금합니다.
- 가격표는 업그레이드 경로로 짰습니다.
  - 플랜 차이는 크레딧 양뿐입니다. 1크레딧은 녹음 60분입니다.
  - Starter 지급량은 2주 만에 바닥나지 않으면서 상위 플랜으로 올라갈 이유가 남는 선으로 정했습니다.
  - 충전팩 단가를 Pro 업그레이드보다 비싸게 둬, 크레딧이 떨어지면 Pro가 더 유리합니다.
- 무료 크레딧이 새는 경로를 막았습니다.
  - 구독 크레딧은 이월되지 않습니다.
  - 취소 후 재가입으로 무료분을 덧받을 수 없습니다.
  - 환불은 첫 결제 후 14일 안, 1크레딧 이하 사용 시에만 됩니다.

리스크를 줄인 방식

상황	한 일
크레딧 정합성	잔액 컬럼 없이 기록만 쌓는 원장이라 이력과 잔액이 어긋나지 않습니다. 동시 차감은 계정 단위로 순서를 맞춥니다.
결제 확정	결제 확정은 결제사 알림으로만 합니다. 계정을 특정할 수 없는 알림은 붙이지 않고, 구독 중이면 새 구독을 막아 이중 청구를 피합니다.
작업 실패	재시도와 멈춘 작업 회수로 다시 처리하고, DB 제약으로 같은 작업이 두 번 돌지 않게 했습니다. 잘린 AI 응답은 실패로 처리합니다.
출시	허용 목록으로 가입을 제한해 운영하다가 열었습니다. 결제는 샌드박스 가격으로 검증한 뒤 운영 값으로 바꿨습니다.

가장 고민한 부분

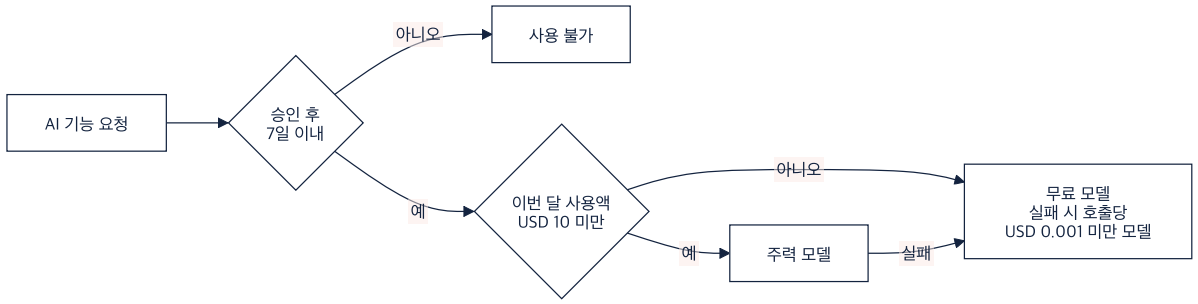
- 과금 순서. 길이 측정, 거절, 차감, 전사 순서를 지켜 어느 단계에서 실패해도 결제 없이 AI가 돌지 않습니다.
- 가격표. 크레딧이 떨어진 이용자가 상위 플랜을 고르게 짰고, 무료 크레딧이 새는 경로는 하나씩 막았습니다.

띠롱

항목	내용
서비스	완료한 일을 결과 메모로 남겨 주간보고와 고과 기록으로 모으는 웹 앱
역할	1인 개발, 기획부터 운영까지
운영	이광복, ddirong.com
수익	없음. 비용이 드는 기능만 승인 후 7일 무료 제공
이용	마케팅 없이 회사 동료들이 사용

무엇을 하려고 했나

항목	내용
문제	주간보고나 고과 시즌마다 지난 몇 달 동안 한 일을 다시 찾아야 했다
방식	할 일에 체크할 때 결과 메모를 받고, 기간별로 이어 붙여 보고서 텍스트로 만든다
전제	요금을 받지 않으므로 운영 비용이 거의 늘지 않아야 한다
결정	매일 쓰는 기능에는 외부 유료 호출을 넣지 않고, 돈이 드는 AI 기능만 따로 묶는다



AI 기능에 건 세 겹의 비용 제한

비용 구조

결정	이유
OCI 무료 서버에 합류	다른 서비스와 같은 서버에 Docker로 올리고, 전용 DB 대신 공용 PostgreSQL 안의 데이터베이스 하나를 씁니다. 한 서비스가 자원을 다 쓰지 못하게 컨테이너마다 메모리와 로그 상한을 걸었습니다.
매일 쓰는 기능에는 AI 없음	보고서 텍스트는 기록 문장을 정해진 형식으로 이어 붙입니다. LLM은 면접 보조 기능에서만 씁니다.
AI 사용액 월 상한	주력 모델 사용액이 월 USD 10에 닿으면 무료 모델과 호출당 USD 0.001 미만 모델로 넘깁니다. 무료 모델만 쓰는 구성은 제공 중단과 잦은 사용량 제한으로 버렸습니다.
필요 없는 추론 끄기	이력서 항목 추출에서 추론을 끄자 55초 타임아웃이 나던 요청이 15.7초에 끝났습니다.
비용이 드는 기능은 승인 후 7일	PM 모드와 면접 보조 모드는 운영자 승인 뒤 7일간 무료입니다. 사용량이 예상보다 크게 늘지 않습니다.
유료 라이선스 대신 MIT	Office 미리보기에 사용자당 월 EUR 3인 Collabora 대신 MIT 라이선스인 Gotenberg를 썼고, 필요할 때만 켜는 선택 구성으로 뺐습니다.
서버가 안 해도 되는 일 줄이기	30초마다 페이지를 렌더하던 헬스체크(하루 2,880회)를 정적 응답으로 바꿨습니다. 빌드 캐시 56.8GB를 정리해 디스크 사용률을 37%에서 9%로 낮췄습니다.

손익분기와 수익 구조

- 요금제와 광고가 없습니다.
- 고정비는 무료 서버라 사실상 0이고, 변동비는 AI 기능에서만 생깁니다.
- 변동비에는 승인제, 7일 기한, 월 사용액 상한을 겹쳐 걸었습니다.
- 보존 기한: 이전 제출본 원본 90일, 알림 180일, 휴지통 30일.

리스크를 줄인 방식

상황	한 일
AI 공급자 장애	주력 모델이 실패하면 다른 공급자로, 그 안에서도 폴백 모델로 넘어갑니다. 폴백 모델의 입력 한도에 맞춰 입력을 잘라 둡니다.
조용한 실패 방지	AI가 실패하면 템플릿 질문으로 몰래 대체하지 않고 화면에 알립니다.
공유 서버	컨테이너별 메모리, 로그 상한과 빌드 캐시 정리를 두고, 새 서비스를 올릴 때 쓰는 체크리스트를 만들었습니다.
확장 순서	인스턴스를 늘리면 예약 작업이 중복 실행되므로, 수평 확장 전에 예약 작업 단일 실행부터 보장하도록 정했습니다. 문서 변환 큐는 실측값(변환 0.2~0.6초)을 근거로 미뤘습니다.

가장 고민한 부분

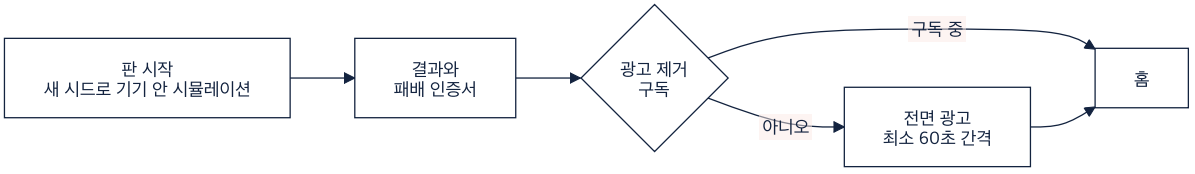
- 비용이 생기는 곳을 AI 기능 하나로 모으고, 금액 상한, 공급자 폴백, 승인 기한을 겹쳐 걸었습니다.
- 이미지 레지스트리 전환, 변환 큐, 수평 확장은 미루고, 다시 꺼낼 조건을 실측값과 함께 문서에 남겼습니다.

독박스

항목	내용
서비스	모임에서 누가 쏠지 미니게임 한 판으로 정하는 파티 게임 앱
역할	1인 개발, 기획부터 스토어 배포 준비까지
상태	iOS, Android 스토어 심사 중
수익	전면 광고, 광고 제거 월 구독 1,100원

무엇을 하려고 했나

항목	내용
문제	사다리나 손가락 뽑기 앱은 결과가 밋밋하고, 10명이 넘으면 한 명씩 조작하느라 늘어진다
방식	폰 한 대로 2~20명이 가입 없이 바로 하는 미니게임 13종
원칙	벌칙 게임이라 결과를 믿을 수 있어야 한다. 결과는 매판 새 난수와 물리 시뮬레이션이 정하고, 연출은 결과를 바꾸지 않는다



서버 없이 기기 안에서 끝나는 한 판

비용 구조

결정	이유
서버와 계정 없음	판정, 기록 저장, 구독 확인까지 기기 안에서 끝납니다. 이용자가 늘어도 운영 비용이 늘지 않고, 보관할 개인정보도 없습니다.
문구도 서버 없이 운영	놀림 문구와 벌칙 문구를 JSON 한 벌로 두고 두 앱이 빌드할 때 넣습니다. 문구는 앱 업데이트로 바꿉니다.
분석, 크래시 수집 SDK 없음	출시 직후 지표는 스토어 지표와 이용자 인터뷰로 봅니다. 외부로 나가는 데이터는 광고 SDK 하나뿐입니다.
구독 확인을 기기에서	StoreKit 2와 Play Billing으로 확인해 영수증 검증 서버가 없습니다.
테스트 배포와 CI 시간	테스트 배포는 무료인 Firebase App Distribution을 씁니다. macOS 빌드 머신이 Linux보다 약 10배 비싸 iOS 빌드를 한 작업으로 묶고, 바뀐 게 없는 날은 건너뜁니다.

손익분기와 수익 구조

- 고정비는 Apple 개발자 계정 연 USD 99와 Google Play 개발자 계정 1회 USD 25입니다.
- 이용자 수에 따라 늘어나는 운영 비용이 없어서, 넘어야 할 선은 스토어 계정비와 CI 사용분입니다.
- 광고는 한 판을 마치고 홈으로 갈 때 나오고, 최소 60초 간격을 둡니다. 결과가 나오는 순간을 끊지 않도록 결과 화면 직전에서 홈으로 가는 길로 옮겼습니다.
- 광고 제거 구독 중이면 광고 SDK, 동의 창, 추적 허용 질문을 모두 건너뜁니다.
- 유입은 광고비 대신 패배 인증서 공유로 설계했습니다. 걸린 사람은 억울해서, 나머지는 놀리려고 단톡방에 올린다는 가정입니다.

리스크를 줄인 방식

상황	한 일
결과 공정성	출시 전 게임마다 1,000판 이상 자동 시뮬레이션해, 한 자리의 당첨 비율이 균등값의 1.5배를 넘지 않는지 확인했습니다.
두 플랫폼 일치	판 시드를 용도별로 나눠 두 플랫폼이 같은 규칙으로 난수를 씁니다. 같은 시드와 참가자면 같은 당첨자가 나와야 테스트를 통과합니다. 물리 게임은 완전한 일치를 보장할 수 없어 보장 범위를 먼저 정했습니다.
스토어 심사	심사 메모에 계정과 서버가 없다는 점, 광고 시점, 구독 위치를 적었습니다. 술자리 문구를 바꿔 전체 이용가를 노렸고 13세 미만은 타깃에서 뺐습니다. 실제 광고 ID는 스토어 빌드에만 넣었습니다.
개인정보	운영자가 수집하는 항목이 없습니다. 권한은 인증서 사진 저장과 iOS 추적 허용뿐입니다.

가장 고민한 부분

- 저장, 판정, 문구, 구독 확인까지 모두 기기 안에서 끝내 서버 비용 0원을 지켰습니다.
- 혼자서 두 플랫폼을 맞추려고 iOS를 기준 구현으로 두고 Android를 기능 단위로 맞췄습니다. 기능 폴더 하나는 에이전트 하나만 고치는 규칙으로 AI 에이전트를 병렬로 돌렸고, 첫 커밋부터 1.1.1 버전까지 8일이 걸렸습니다.