SOLO APP FACTORY

나 혼자 앱 공장

컴퓨터 잘 몰라도 괜찮아요 — AI 딸깍으로 만들어 토스·구글·애플까지

몽상아틀리에

복붙이면 끝, 딸깍 문장 수록

AI에게 주는 족보 13종

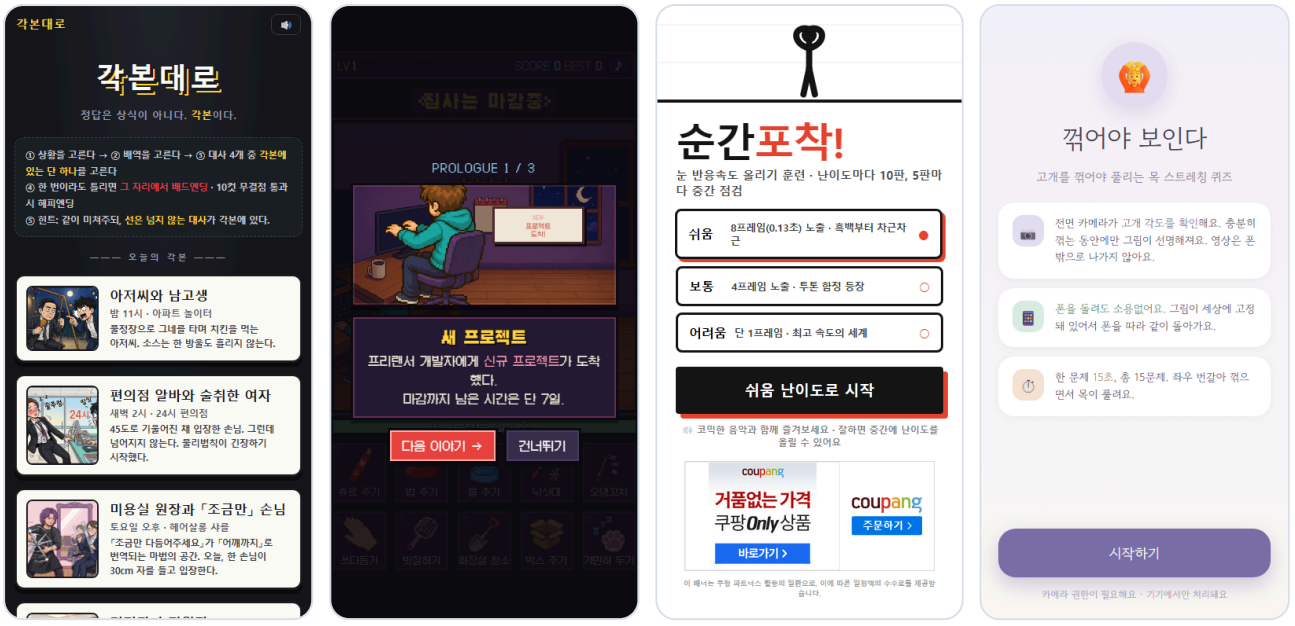
실제 출시로 검증

차례

- 들어가며 — 딸깍만 하면 정말 돼요
- 1장 설치는 딱 한 번만 고생해요 — ORCA와 AI 비서 들이기
- 2장 AI 부리는 기본기 — 그리고 돈 안 새는 3원칙
- 3장 첫 앱 만들기 딸깍
- 4장 웹에 올리기 딸깍 — 전 세계에 주소가 생겨요
- 5장 그림 뽑기 딸깍 — 추가 요금 없이 구독 한도로
- 6장 토스에 올리기 — 첫 스토어는 여기가 제일 쉬워요
- 7장 구글 플레이에 올리기
- 8장 아이폰 앱스토어에 올리기 — 맥 없어도 돼요
- 9장 광고·구독으로 용돈 벌기
- 쉬어가기 — 여기서부터는 AI에게 주는 「족보」예요
- 족보 ①~⑫ — 환경·그림·제작·웹·토스·구글·iOS 빌드·앱스토어 커넥트·심사·구독·광고·자동화
- 족보 ⑬ — 출시 체크리스트 · 함정 사전 40선 · 명령 모음

들어가며 — 딸깍만 하면 정말 돼요

먼저 고백부터 할게요. 이 책에 나오는 앱들은 전부 **코딩을 직접 하지 않고** 만들었어요. 윈도우 컴퓨터 한 대에 AI 비서를 들여놓고, 한국어로 부탁하고, 나온 걸 보고 "여긴 재미없어, 바꿔줘"라고 말한 게 전부예요. 그렇게 만든 게임과 앱이 토스, 구글 플레이, 애플 앱스토어에 실제로 올라가 있답니다.



이 책의 방법으로 만들어 실제 서비스 중인 앱들 — 병맛 대화 게임, 픽셀아트 아케이드, 반응속도 게임, 목 스트레칭 퀴즈

그래서 이 책은 두 부분으로 나눴어요.

- **1부 (1장~9장)**: 사람이 읽는 부분이에요. 설치를 딱 한 번 고생해서 끝내고, 그다음부터는 **복사해서 붙여넣는 "딸깍 문장"**으로 앱을 만들고 스토어에 올려요. 어려운 말은 최대한 뺐어요.
- **2부 (족보 ①~⑬)**: 사람이 읽는 부분이 **아니에요.** 실제 출시 과정에서 얻은 세세한 기술 노하우를 모아둔 "족보"인데, 여러분이 읽는 게 아니라 **AI에게 복사해서 건네주는 자료**예요. AI가 이걸 읽으면 제가 밟았던 함정을 건너뛰고 한 번에 해내요.

"딸깍"이 뭐냐면요

이 책에서 딸깍은 이런 뜻이에요: **회색 상자 안의 문장을 그대로 복사해서 AI에게 붙여넣고 엔터.** 그게 끝이에요. AI가 일하는 동안 여러분은 커피를 마시면 되고, 가끔 AI가 "이건 직접 로그인해 주셔야 해요"라고 부르면 그것만 해주면 돼요.

사람이 꼭 해야 하는 일은 생각보다 적어요. 정리하면 딱 세 가지예요.

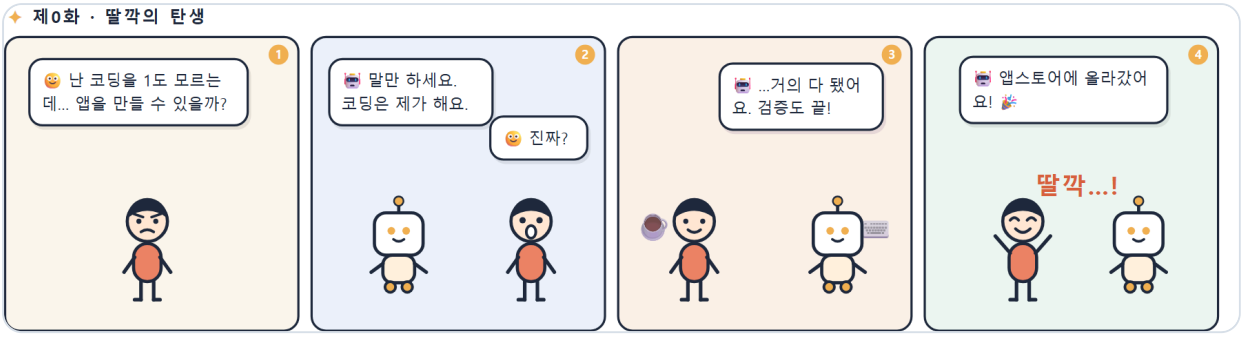
1. 각종 **로그인** — 이것도 「로그인 창 띄워줘, 내가 로그인할게. 그다음은 네가 해」라고 하면 AI가 그 화면 앞까지 데려다줘요. 여러분은 아이디와 비밀번호만.
2. **법적 동의·서명·최종 제출 버튼** — 책임지는 일이니 직접 읽고 눌러요.
3. **"재밌다/재미없다" 판단** — 이건 세상에서 여러분만 할 수 있어요.

가입, 콘솔 입력, 스크린샷, 광고 연동, 구독 상품 만들기... "이건 어렵겠지" 싶은 것도 일단 시켜보세요. 실제로 이 책의 앱들은 구독 상품 개설(전 세계 175개 지역 가격 설정까지)도 AI가 했어요.

돈 얘기도 미리 솔직하게

- 필요한 것: AI 구독 하나(예: 클로드 유료 구독), 그리고 스토어 등록비(구글 $25 한 번, 애플 연 $99 — 애플은 낼 때만).
- 필요 없는 것: 맥북(없어도 아이폰 앱 나와요), 종량제 API 요금(안 새게 막는 법을 2장에서 알려드려요), 외주비.
- 이 책이 약속하는 것: 여기 적힌 대로 하면 **앱이 스토어에 올라가요.** 그 앱이 돈을 얼마나 버는지는 앱의 재미가 정해요. 수익 보장은 못 해요. 대신 광고와 구독을 붙이는 법(9장)까지는 다 알려드릴게요.

> 일러두기: 화면과 정책은 계속 바뀌어요. 책과 화면이 다르면 당황하지 말고 AI에게 "화면이 이렇게 생겼는데 어떻게 해?"라고 물어보세요. 그게 이 책의 기본 자세이기도 해요 — 모르면 AI에게 물어본다, 끝.



1장 설치는 딱 한 번만 고생해요 — ORCA와 AI 비서 들이기

이 장만 넘기면 그다음부터는 전부 딸깍이에요. 순서대로 하나씩, 구구절절 적어둘게요. 30분이면 끝나요.

1-1. 준비물 확인

- 윈도우 컴퓨터 (맥이어도 거의 같아요)
- 클로드(Claude) 유료 구독 계정 — claude.ai에서 가입해요. AI 비서의 몸값이에요
- (선택) ChatGPT 유료 구독 — 5장에서 그림 뽑을 때 있으면 좋아요

1-2. Node.js 설치 — 도구들이 올라탈 받침대

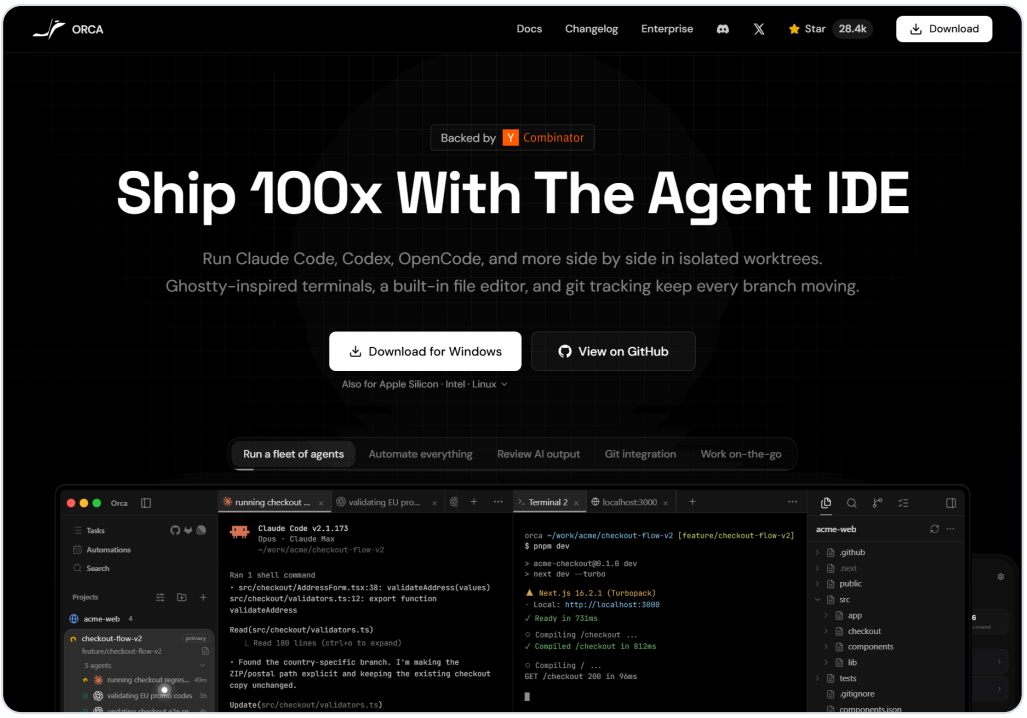
AI 도구들이 이 프로그램 위에서 돌아가요. 한 번 깔면 잊어버려도 돼요.

1. 인터넷에서 **nodejs.org** 에 들어가요.
2. 초록색 다운로드 버튼(LTS라고 적힌 것)을 눌러 받아요.
3. 받은 파일을 더블클릭하고, 계속 "다음"만 눌러요. 바꿀 것 없어요.
4. 확인: 시작 메뉴에서 **PowerShell**을 열고 `node -v` 라고 입력하고 엔터. `v24` 같은 글자가 나오면 성공이에요.

> 회사 컴퓨터라 설치 권한이 없다면: ZIP 버전을 받아서 `내 PC의 LOCALAPPDATA\Programs\nodejs` 폴더에 풀어도 돼요. 이 방법이 헷갈리면 나중에 AI에게 "노드를 관리자 권한 없이 설치하고 싶어"라고 부탁하면 돼요(족보 ①에 방법이 있어요).

1-3. ORCA 설치 — AI 비서들의 사무실

ORCA는 깃허브에 공개된 **무료 오픈소스 프로그램**이에요. 클로드 코드 같은 AI 코딩 비서를 여러 명 나란히 앉혀놓고 부리는 사무실이라고 생각하면 돼요. 비서마다 방(작업 공간)을 따로 줘서 서로 안 부딪히게 해주고, 터미널·브라우저·파일 보기가 한 화면에 있어요.



ORCA 공식 홈(onorca.dev). "Download for Windows" 버튼 하나면 설치돼요. 클로드 코드·코덱스 등을 나란히 돌리는 화면이 보이죠 — 5장의 "그림은 옆 탭 GPT에게" 같은 협업이 이 구조에서 나와요

1. **onorca.dev** 에 들어가요 (깃허브에서 Orca를 검색해도 돼요).
2. **Download** 버튼 → **Windows** 를 골라요.
3. 받은 설치 파일(.exe)을 더블클릭해요. 무료라서 결제 화면 같은 건 없어요.
4. 실행하면 프로젝트(작업 폴더)를 추가하라고 해요 — 아직 폴더가 없으니 일단 그대로 두세요. 3장에서 만들 거예요.

1-4. 클로드 코드 설치 — 우리의 메인 비서

1. PowerShell(또는 ORCA 안의 터미널)에 아래를 붙여넣고 엔터:

```
npm install -g @anthropic-ai/claude-code
```

1. 설치가 끝나면 `claude` 라고 입력하고 엔터.
2. 처음이라 로그인하라고 해요. 화면 안내를 따라가면 **브라우저가 열려요** → 클로드 구독 계정으로 로그인 → "허용"을 눌러요.
3. 터미널로 돌아오면 끝! 이제 한국어로 말을 걸 수 있어요. "안녕, 너 뭐 할 수 있어?"라고 인사해 보세요.

여기서 아주 중요한 것 하나만요. 로그인할 때 **API 키를 입력하라는 선택지가 보여도 무시**하세요. 우리는 무조건 **구독 계정 로그인**이에요. API 키를 넣으면 쓸 때마다 돈이 나가는 다른 지갑이 열려요(2장에서 자세히).

1-5. 보조 비서들도 들일까요? (선택)

없어도 돼요. 있으면 가끔 유용해요. 방법은 똑같아요 — 설치 한 줄, 구독 로그인 한 번.

- **Codex (ChatGPT)**: `npm i -g @openai/codex` → 실행 후 ChatGPT 구독 계정으로 로그인

- **Grok**: `npm i -g @xai-official/grok` → `grok login` → SuperGrok 또는 X 프리미엄 구독으로 로그인

ORCA를 열면 설치된 비서들이 자동으로 목록에 떠요. 프로젝트를 추가하고 비서를 고르면 대화가 시작돼요.

1-6. 자주 막히는 곳 세 군데

- **"node를 찾을 수 없다"고 해요** → 열려 있던 터미널·ORCA를 껐다 켜세요. 설치 전에 열린 창은 새 프로그램을 몰라봐요.
- **로그인이 풀렸어요** → 몇 시간~며칠 지나면 풀리는 게 정상이에요. 다시 로그인해 주면 돼요. 이건 AI가 대신 못 해줘요.
- **뭔가 이상한 영어 오류가 떠요** → 그 오류를 그대로 복사해서 AI에게 붙여넣고 "이거 왜 이래?"라고 물어보세요. 진심으로, 이게 정답이에요.

설치 끝! 이제부터는 진짜로 딸깍만 해요.

2장 AI 부리는 기본기 — 그리고 돈 안 새는 3원칙

2-1. 말 거는 법: 세 가지만 담으면 돼요

AI에게 일을 시킬 때 잘 되는 부탁에는 공통점이 있어요.

1. **뭘 원하는지** — "고양이 돌보는 픽셀 게임 만들어줘"
2. **어떤 느낌인지** — "귀엽지만 유치하지 않게, 폰 세로 화면으로"
3. **뭐가 되면 성공인지** — "다 되면 네가 직접 실행해서 에러 없는 걸 확인하고 보여줘"

3번이 핵심이에요. **"만들어줘"에서 멈추지 말고 "확인까지 해줘"라고 시키는 것.** 이 한 줄 차이로 결과물 품질이 완전히 달라져요. 이 책의 딸깍 문장에는 전부 이 마무리가 들어 있어요.

2-2. 이 문장들을 외워두세요 (만능 문장 6개)

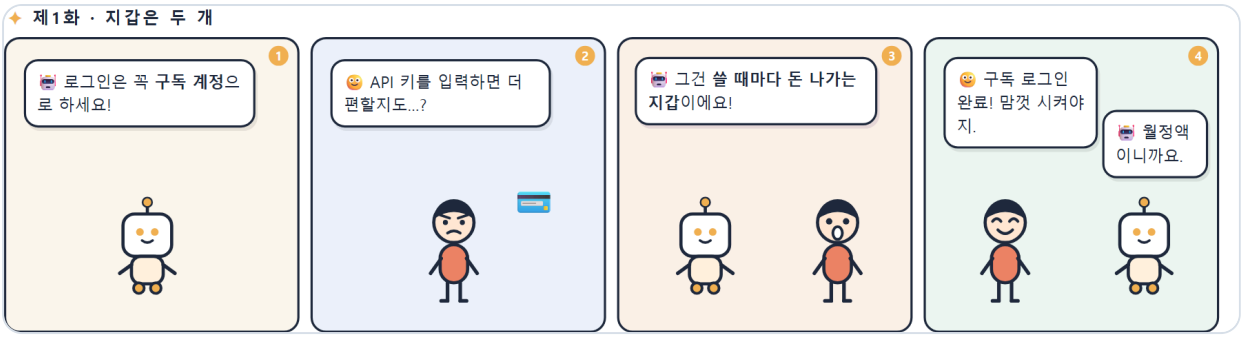
- 「**직접 실행해서 확인하고, 문제 있으면 고친 다음 다시 확인해줘**」— 기본 마무리
- 「**폰 화면 기준으로 스크린샷을 찍어서 보여줘**」— 눈으로 확인하고 싶을 때
- 「**서두르지 말고 원인부터 조사해줘**」— 뭔가 꼬였을 때
- 「**내가 직접 해야 하는 일이 나오면 멈추고 알려줘**」— 로그인·결제 같은 일
- 「**지금까지 알게 된 주의사항을 이 폴더의 CLAUDE.md 파일에 기록해둬**」— 다음에 또 안 헤매게
- 「**모르면 아는 척하지 말고 모른다고 해줘**」— 보험이에요
- 「**로그인 창 띄워줘, 내가 로그인할게. 그다음은 네가 해**」— ⭐ 이게 최강이에요. 애드몹 가입이든 스토어 콘솔이든, 이 한 문장이면 AI가 브라우저를 띄우고, 여러분이 로그인하면, 나머지 입력을 전부 이어받아요

그리고 뭐든 막히면: **오류 문구를 통째로 복사해서 붙여넣기.** 설명하려고 애쓰지 마세요. 원문이 최고의 설명이에요.

2-3. 돈 안 새는 3원칙 💸

AI 요금은 지갑이 두 개예요. **구독**(월정액, 얼마를 쓰든 정해진 돈)과 **API**(쓴 만큼 계속 나가는 돈). 같은 회사 것이어도 서로 다른 지갑이에요. 원칙 세 개만 지키면 요금 폭탄은 없어요.

1. **비서 로그인은 무조건 구독 계정으로.** API 키를 입력하라는 화면은 전부 무시해요. 이미 구독료를 냈으니 그걸로 다 돼요.
2. **그림은 챗지피티·제미나이 '웹 화면'에서 뽑아요.** 코드로 그림 API를 부르면 장당 돈이 나가요. 웹 화면에서 뽑으면 이미 내는 구독료 안이에요 — 공짜라는 뜻은 아니고, 구독 한도 내에서 추가 요금이 없다는 뜻이에요. (5장에서 딸깍으로 하는 법이 나와요.)
3. **AI에게 이렇게 선언해두세요**: 「돈이 나가는 API는 절대 쓰지 마. 필요해 보이면 먼저 나한테 물어봐.」— 앱에 AI 기능을 넣고 싶어질 때도, 정말 필요한지부터 의심해요. 이 책의 앱들은 전부 API 호출 0건이에요(효과음도, 차트도, 문제 데이터도 다 앱 안에서 해결했어요).



2-4. ORCA에서의 하루 (감 잡기용)

1. ORCA를 열고 프로젝트 폴더를 골라요.
2. 클로드 코드 비서를 열고, 이 책의 딸깍 문장을 붙여넣어요.
3. AI가 일하는 동안 지켜봐요. 중간에 "이거 해도 될까요?"라고 물으면 읽고 허락해요.
4. 결과물을 보고 느낀 대로 말해요. "글씨가 너무 작아", "여기 심심해", "완벽해!"
5. 끝나면:「오늘 배운 것 CLAUDE.md에 기록해둬」

비서를 여러 명 부릴 수도 있어요(ORCA가 그걸 잘해요). 다만 처음에는 한 명이면 충분해요. 익숙해지면 "한 명은 게임 만들고, 한 명은 스토어 문구 쓰고"처럼 나눠 시켜보세요.

2-5. 비서가 다 해주는 일 (사실상 전부예요)

"이것도 시켜도 되나?" 싶을 때 보는 목록이에요. 전부 이 책에서 실제로 AI가 해낸 일들이에요.

- 앱 코딩·버그 수정·테스트·배포 (당연히)
- 디자인 적용, 그림 후처리, 아이콘·스플래시 생성
- 스토어 제출용 스크린샷·썸네일 제작 (규격 맞춰서)
- 앱 소개 문구, 키워드, 심사 메모 (글자 수 제한 지켜서)
- **각종 콘솔 가입·입력** — 애드몹 앱 등록, 플레이 콘솔 선언 9종, 토스 콘솔 앱 정보, 앱스토어 커넥트 전체
- **광고 연동** — 광고 단위 발급부터 코드 연결, 빈도 설계까지
- **구독·인앱결제 상품 개설** — 상품 생성, 175개 지역 가격, 무료 체험, 심사 제출까지
- 개인정보처리방침 페이지, app-ads.txt 같은 잡다한 준비물
- 심사 반려 메일 분석 → 수정 → 재제출
- 출시 후 확인 — "지금 라이브 버전에서 잘 돌아가는지 접속해서 봐줘"

여러분 몫으로 남는 건 로그인, 법적 서명·최종 제출, 결제 정보, 그리고 재미 판단. 정말 이게 다예요.

2-6. 마음가짐 하나

AI가 한 번에 완벽하게 해내는 날도 있고, 세 번 고쳐야 하는 날도 있어요. 정상이에요. 여러분은 감독이고, 감독의 일은 코드를 읽는 게 아니라 **결과물을 보고 방향을 말해주는 것**이에요. "왜 안 되지?"에서 멈추지 말고 "안 되는 걸 AI에게 보여주기"까지만 하면, 나머지는 알아서 돌아가요.

3장 첫 앱 만들기 딸깍

드디어 만들어요. 이 장에서 사람이 할 일은 **아이디어 정하기**와 **재미 판정**뿐이에요.

3-1. 아이디어가 아직 없다면, 이것부터 딸깍

```
심심풀이 모바일 웹 게임 아이디어가 필요해.
- 혼자 만들 수 있는 작은 규모로 10개 제안해줘
- 각각 한 줄 설명 + 예상 재미 포인트 + 만들기 난이도(상/중/하)를 붙여줘
- 참신한 것 5개, 검증된 장르 비틀기 5개로 섞어줘
```

나온 것 중에 마음에 드는 걸 골라요. 없으면 "3번이랑 7번을 합쳐봐" 같은 주문도 잘 먹혀요.

3-2. 만들기 딸깍 (이 책의 심장이에요)

아래 상자에서 ○○만 채워서 붙여넣으세요.

```
새 앱을 만들자. 주제: ooo (예: 회전된 그림을 고개를 꺾어야 맞출 수 있는 퀴즈)
분위기: ooo (예: 전문적인데 살짝 귀엽게. 유치한 건 싫어)

지켜줄 것:
- C:\apps\앱이름 폴더를 새로 만들어서 그 안에서만 작업해줘
- 폰 세로 화면 기준. 인터넷 연결이 없어도 완전히 도는 HTML 한 장짜리로 만들어줘
- 효과음이나 그림이 필요하면 일단 코드로 대신 만들고, 진짜 그림은 나중에 넣을 수 있게 자리를 비워둬
- 이 문서(족보 ③)의 제작 기준을 그대로 따라줘: [족보 ③을 여기 붙여넣기]

다 만들면:
- 네가 직접 실행해서 모든 기능을 눌러보고, 에러가 0개인 걸 확인해줘
- 작은 폰 화면(320px)에서도 안 깨지는지 확인해줘
- 폰 화면 스크린샷을 몇 장 찍어서 보여줘
```

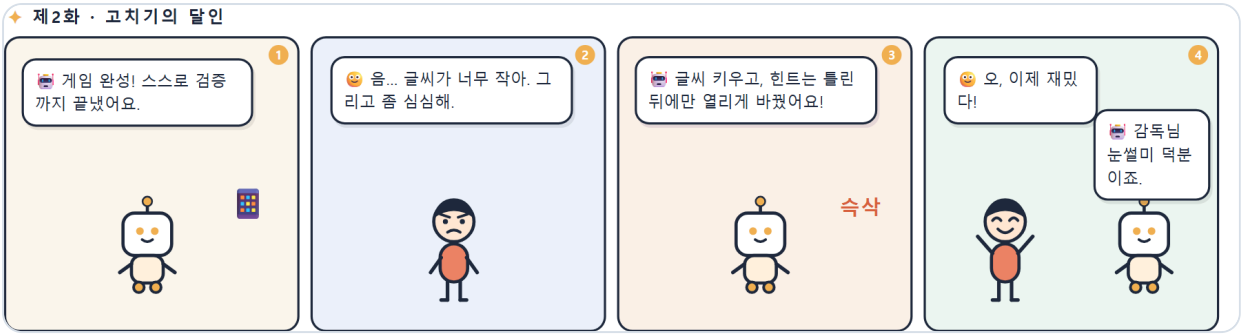
"[족보 ③을 여기 붙여넣기]" 부분은 이 책 2부의 **족보 ③(앱 제작 상세)**을 통째로 복사해서 넣는 거예요. 안 넣어도 앱은 나오지만, 넣으면 웹뷰에서 안 깨지는 구조·자동 검증 같은 실전 노하우가 처음부터 반영돼요.

3-3. 고치기 딸깍 — 여기서 앱이 완성돼요

첫 결과물은 보통 70점이에요. 나머지 30점은 여러분의 입에서 나와요. 실제로 제가 썼던 주문들이에요.

- 「글씨가 너무 작아. 특히 내가 적는 글은 시원시원하게 키워줘」
- 「시작하자마자 재미 포인트가 나와야 해. 설명 화면 줄여줘」
- 「너무 쉬워서 긴장감이 없어. 힌트는 한 번 틀린 다음에만 열어줘」
- 「이 색은 유치해 보여. 전문적인데 절제된 귀여움으로 바꿔줘」
- 「기다리면 공짜로 정답이 보이는 꼼수가 있네? 막아줘」

한 번에 하나씩 말하는 게 좋아요. 그리고 고칠 때마다 마법의 마무리: 「고친 다음 직접 확인하고 보여줘」.



3-4. 디자인을 확 바꾸고 싶다면 — 클로드 디자인 연동

말로 "예쁘게 해줘"를 반복하는 것보다 좋은 방법이 있어요. 클로드 웹사이트의 **디자인 도구 (claude.ai/design)**에서 시안을 먼저 만들고, 그 시안을 AI 비서에게 넘겨서 앱에 입히는 거예요. 실제로 이 책의 앱 두 개가 이 방법으로 옷을 갈아입었어요 — 연습일지 앱은 "클래식(골드+세리프)"으로, 목 스트레칭 퀴즈는 "부드러운 파스텔 웰니스"로요(들어가며의 네 번째 스크린샷이 바로 그 결과예요).

순서는 이래요.

1. **claude.ai/design** 에서 새 프로젝트를 만들고, 내 앱 스크린샷 몇 장(또는 HTML 파일)을 올려요.
2. 원하는 느낌을 말해요: 「이 앱을 전문적인데 살짝 귀엽게, 라벤더 파스텔 웰니스 느낌으로 리디자인해줘」
3. 시안이 나오면 **대화로 다듬어요**. "버튼은 더 둥글게", "이 초록은 좀 촌스러워" — 코드 걱정 없이 그림만 보고 말하면 돼요. 여기서 충분히 다듬는 게 요령이에요.
4. 마음에 들면 시안 파일을 **내려받아서** 프로젝트 안 `_redesign` 폴더에 넣어요.
5. 비서에게 딸깍:

```
클로드 디자인에서 만든 시안을 이 앱에 적용하자. 시안 파일은 _redesign 폴더에 있어.
- 원본 코드는 최대한 건드리지 말고, 위에 덧씌우는 스타일 파일로 적용해줘
  (마음에 안 들면 한 줄로 되돌릴 수 있게)
- 시안에서 색·폰트·둥글기 값을 정확히 뽑아서 그대로 써줘. 눈대중 금지
- 적용 후 주요 화면 3개를 폰 크기로 스크린샷 찍어서 시안과 나란히 비교해 보여줘
```

"덧씌우는 스타일 파일" 방식이 포인트예요. 원본을 안 건드리니까 실험이 공짜가 되고, 시안 A/B를 갈아 끼우며 비교할 수도 있어요.

3-5. 폰에서 직접 해보고 싶다면

```
지금 상태를 내 폰에서 해볼 수 있게 해줘.
같은 와이파이에서 접속할 주소를 알려주고, QR코드도 만들어줘.
```

폰으로 해보면 컴퓨터에서 안 보이던 게 보여요(버튼이 손가락보다 작다든지). 느낀 점을 그대로 말해주면 돼요.

3-6. 이 장의 사람 체크리스트

- ☐ 아이디어 정했어요
- ☐ 만들기 딸깍 붙여넣었어요 (족보 ③ 포함해서)
- ☐ 폰으로 해봤어요
- ☐ "재밌다"는 느낌이 들 때까지 고치기 딸깍을 반복했어요

마지막 항목이 제일 중요해요. **재미없는 앱은 스토어에 올려도 아무 일도 일어나지 않아요.** 스토어는 4장부터니까, 서두르지 말고 여기서 충분히 놀아보세요.

4장 웹에 올리기 딸깍 — 전 세계에 주소가 생겨요

앱을 스토어에 내기 전에 먼저 웹에 올려요. 이유는 간단해요. ①친구에게 링크로 자랑할 수 있고 ②스토어 등록할 때 필요한 준비물(아이콘 주소, 개인정보처리방침 페이지)이 생기고 ③웹 자체가 나중에 광고 수익 통로가 돼요. 전부 무료예요.

4-1. 사람이 할 일 (클릭 두 번이에요)

고백하자면, 가입조차 AI가 해줘요. 실제로 제 Cloudflare 계정도 AI가 배포를 준비하다가 브라우저를 띄워줬고, 저는 **계정 선택하고 "허용" 누른 게 전부**예요 — 계정이 없으면 그 과정에서 자동으로 만들어지거든요. 여러분 몫은 딱 두 번의 클릭이에요.

1. AI가 띄운 브라우저 창에서 **로그인(또는 구글 계정 선택)하고 "허용" 누르기**
2. 메일함에 온 **Cloudflare 인증 메일 클릭** — 요건 꼭 하세요. 안 누르면 이후 작업이 전부 막혀요.

> 왜 Cloudflare냐면요: 무료인데 트래픽 제한이 사실상 없고, **광고를 붙여도 되기 때문**이에요. 유명한 Vercel의 무료 플랜은 광고 게재가 금지라서, 나중에 애드센스를 달 계획이라면 처음부터 Cloudflare로 가는 게 맞아요.

4-2. 올리기 딸깍

```
이 앱을 Cloudflare Pages에 무료로 배포해줘. 프로젝트 이름은 ooo.
- 나 Cloudflare 계정 없어. 가입부터 진행하고, 로그인·허용 창이 뜨면 나를 불러줘
- 가입됐으면 인증 메일 눌러야 한다고 나한테 꼭 알려줘
- 배포 주소가 나오면 알려주고, 네가 직접 그 주소에 접속해서
  앱이 정상으로 뜨는지, 폰 화면에서도 잘 나오는지 확인해줘
- 개인정보처리방침 페이지(privacy)도 간단히 만들어서 같이 올려줘
- 세부 주의사항은 이 문서를 따라줘: [족보 ④를 여기 붙여넣기]
```

중간에 브라우저가 열리면서 Cloudflare 허용 버튼을 누르라고 할 수 있어요. 한 번만 눌러주면 다음부터는 안 물어봐요.

몇 분 뒤 `https://앱이름.pages.dev` 같은 주소를 받으면, 그 링크를 폰 카톡에 보내서 직접 열어보세요. 이 순간이 꽤 감동이에요 — 내가 만든 게 인터넷에 있어요!

4-3. 고칠 때마다 다시 올리는 딸깍

```
방금 고친 버전을 다시 배포해줘. 배포 후에 실제 주소에서 바뀐 게 반영됐는지 확인해줘.
(반영이 안 보이면 캐시 때문일 수 있으니 그것까지 확인해줘)
```

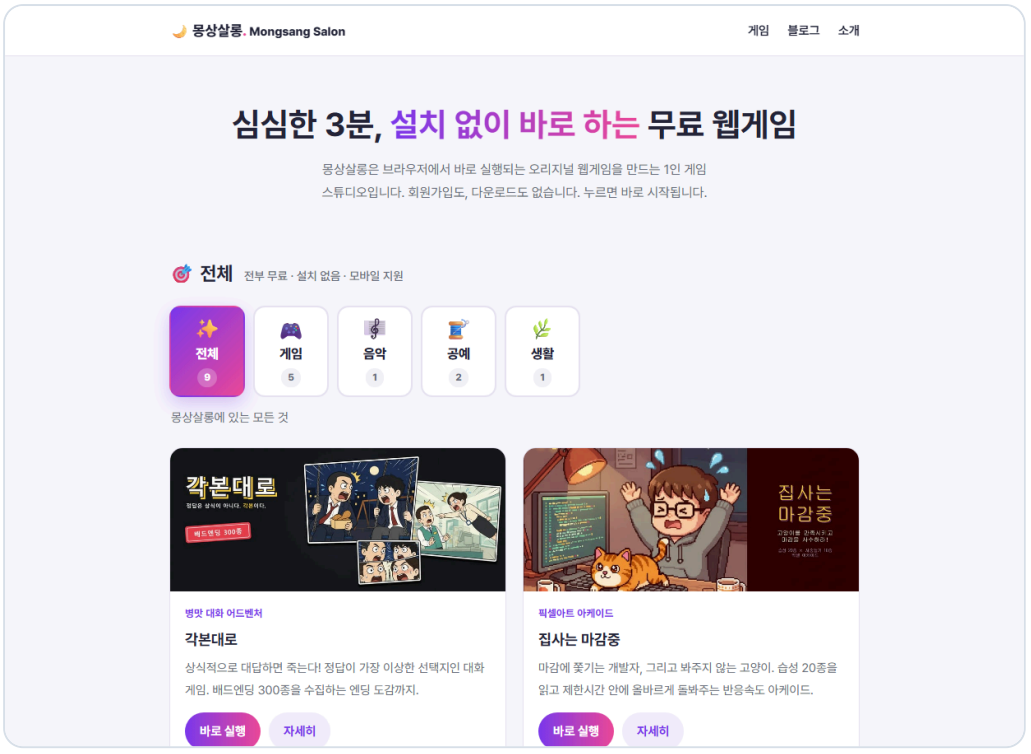
배포 직후엔 예전 화면이 잠깐 보일 수 있어요(전 세계 서버에 퍼지는 시간이에요). AI가 알아서 확인해주니 기다리면 돼요.

4-4. 여기까지 하면 생기는 것들

- 앱 주소: `https://앱이름.pages.dev`
- 아이콘 주소: `https://앱이름.pages.dev/icon.png` (6장 토스 등록 때 필요해요)
- 개인정보처리방침 주소 (7장 구글 등록 때 필요해요)

4-5. (나중에) 진짜 도메인과 포털 만들기

앱이 여러 개 쌓이면, 도메인을 하나 사서(1년에 만원쯤) 앱들을 한곳에 모은 "포털 사이트"를 만드는 단계가 와요. 광고 심사에 유리한 구조인데, 이건 9장에서 다시 만나요.



앱들을 한 도메인에 모은 포털 사이트 예시. 이런 구조가 나중에 광고 심사에 유리해요 (9장에서 자세히)

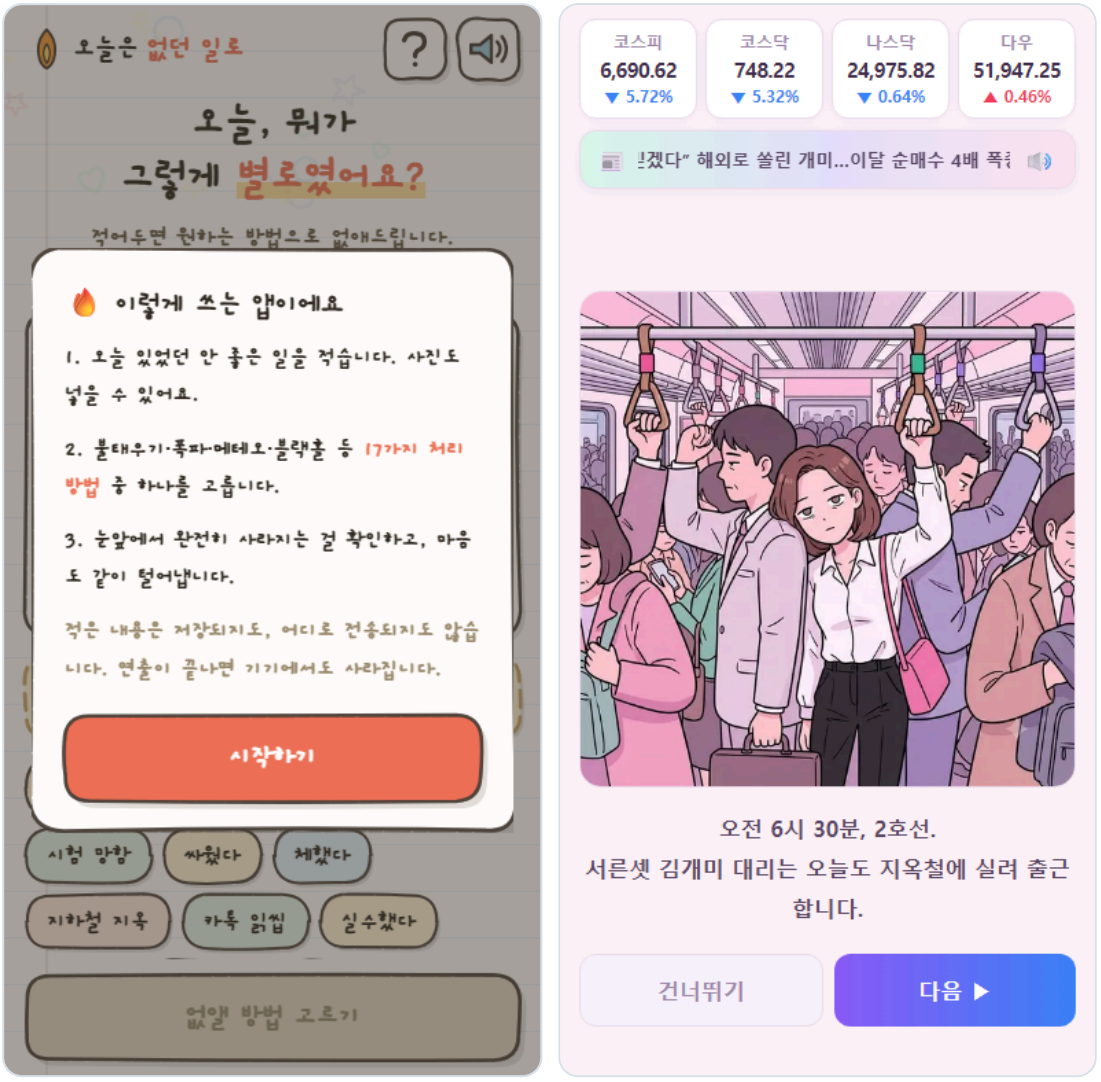
도메인 살 때 딱 하나만 조심하세요: **철자를 꼭 확인하고 결제 버튼을 누르세요.** 오타 도메인은 환불이 안 돼요. (네, 겪어본 사람이 하는 말이에요.)

5장 그림 뽑기 딸깍 — 추가 요금 없이, 구독 한도 안에서

앱에 그림이 들어가면 때깔이 확 달라져요. 캐릭터, 소품, 아이콘… 수십 장이 필요한데, 이걸 이미지 생성 API로 뽑으면 **장당** 돈이 나가요. 우리는 그 대신 **챗지피티(또는 제미나이) 유료 구독의 웹 화면**에서 뽑아요.

솔직하게 정리하면 이래요. **공짜는 아니에요 — 구독료를 이미 내고 있으니까요.** 다만 ①장당 추가 과금이 없고 ②구독 등급별로 정해진 **생성 한도 안에서는 제대로 나온다**는 뜻이에요. 한도에 걸리면 "잠시 후 다시" 안내가 뜨는데, 좀 쉬었다 하거나 다른 서비스(제미나이↔챗지피티)로 갈아타면 돼요. 앱 하나 분량(수십 장)은 며칠에 나눠 뽑으면 구독 한도로 충분했어요.

문제는 "수십 장을 어떻게 체계적으로 뽑느냐"인데, 답이 바로 **지시서 딸깍**이에요.



왼쪽: 챗지피티 웹에서 받은 크레용 손그림과 아이콘 24종을 입힌 앱. 오른쪽: 제미나이 웹에서 받은 캐릭터 일러스트 30장으로 만든 게임 오프닝. 둘 다 구독 한도 안에서, 추가 요금 없이 해결했어요

5-1. 흐름을 먼저 그려볼게요

1. 클로드 비서가 → **"그림 지시서"를 파일로 만들어** 작업 폴더(art)에 넣어둬요 (어떤 그림이 몇 장 필요한지, 그림체는 뭔지, 파일 이름은 뭘로 저장할지가 적힌 문서)
2. GPT 비서가 → 그 지시서를 읽고 **그림을 생성해 art 폴더에 저장**해요

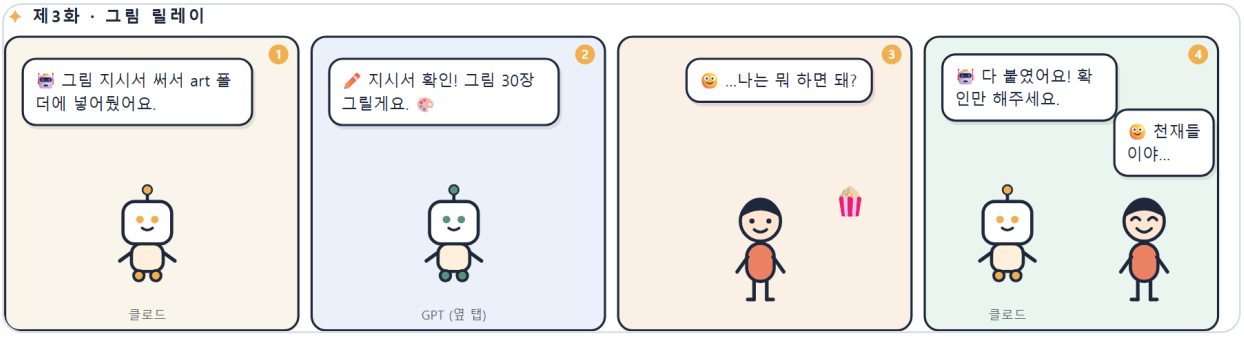
3. 클로드 비서가 → 배경 제거, 크기 조절, 앱에 넣기까지 **자동으로** 해요

눈치채셨나요? **사람 차례가 거의 없어요.** ORCA의 진가가 여기서 나와요 — **같은 작업 폴더에 탭을 하나 더 열고 GPT(코덱스) 비서를 띄운 다음**, 이 한 줄만 던지면 돼요:

```
art 폴더의 그림 지시서(ART_BRIEF) 파일을 읽고, 지시서대로 그림을 전부 생성해서
지시서에 적힌 파일 이름 그대로 art 폴더에 저장해줘. 다 되면 몇 장 만들었는지 알려줘.
```

두 비서가 같은 폴더를 보고 있으니 파일로 자연스럽게 협업이 돼요. 클로드는 지시서를 써두고, GPT는 그림을 채워 넣고, 클로드가 다시 그걸 주워서 앱에 반영하는 릴레이예요.

만약 GPT 비서 쪽에서 이미지 생성이 안 되는 환경이거나 한도에 걸리면, 예비 방법은 고전적인 손 릴레이예요: 지시서를 챗지피티 웹 화면에 직접 붙여넣고, 나온 그림을 art 폴더에 저장하기. 어느 쪽이든 그다음은 다시 클로드 비서 몫이에요.



5-2. 지시서 만들기 딸깍

```
이 앱에 그림을 입히고 싶어.
- 필요한 그림 목록을 뽑아줘 (본문 그림 + 아이콘, 우선순위 표시)
- '그림 지시서'를 한국어로 만들어서 art/ART_BRIEF.md 파일로 저장해줘. 지시서에는
  ① 모든 그림에 공통으로 들어갈 그림체 설명 (그래야 30장이 한 사람 그림처럼 나와)
  ② 그림마다: 프롬프트 + 저장할 파일 이름 + "이렇게 나오면 다시 요청하세요" 기준
  ③ 배경은 제거하기 좋게 요청하는 방법
  ④ 글씨나 테두리는 그리지 말라는 당부 (그건 네가 코드로 처리하니까)
  를 담아줘
- art 폴더에 그림 파일이 들어오면 배경 제거→크기 조절→앱 반영까지
  자동으로 되는 스크립트를 만들어줘
- 그림이 아직 없어도 앱이 정상으로 돌게 해줘 (그림이 오면 한 장씩 살아나는 구조)
- 세부 노하우는 이 문서를 참고해줘: [족보 ②를 여기 붙여넣기]
```

5-3. 직접 붙여넣을 때의 요령 (예비 방법)

GPT 비서 대신 챗지피티 웹 화면에 지시서를 직접 붙여넣는 경우의 요령이에요.

- **마음에 안 들면 미련 없이 "다시"를 외치세요.** 추가 요금이 없으니까요(생성 한도만 신경 쓰면 돼요). 지시서에 적힌 "다시 요청 기준"에 걸리면 무조건 다시예요.
- 그림체가 슬금슬금 변하면 지시서의 공통 스타일 문단을 다시 붙여넣으면 돌아와요.

- 저장할 때 **파일 이름을 지시서대로** 하는 것만 신경 쓰세요. 이름이 곧 주소라서, 이름만 맞으면 나머지는 자동이에요.

다 넣고 나면 비서에게: 「art 폴더에 그림 넣었어. 처리해서 앱에 반영하고, 이상한 그림 없는지 한 장에 모아서 보여줘」

5-4. 알아두면 좋은 것

- 아이콘이나 로고처럼 **글자가 들어가는 그림은 이미지로 만들지 않는 게 좋아요.** 글씨는 폰트로 처리하는 게 훨씬 선명하고, 나중에 문구를 바꿀 수도 있어요. AI가 알아서 그렇게 하겠지만, 혹시 그림에 글씨를 넣으려고 하면 말려주세요.
- 스토어에 낼 스크린샷·썸네일도 전부 AI가 만들어요(다음 장들에서 딸깍에 포함돼 있어요). 여러분이 포토샵을 켤 일은 없어요.

6장 토스에 올리기 — 첫 스토어는 여기가 제일 쉬워요

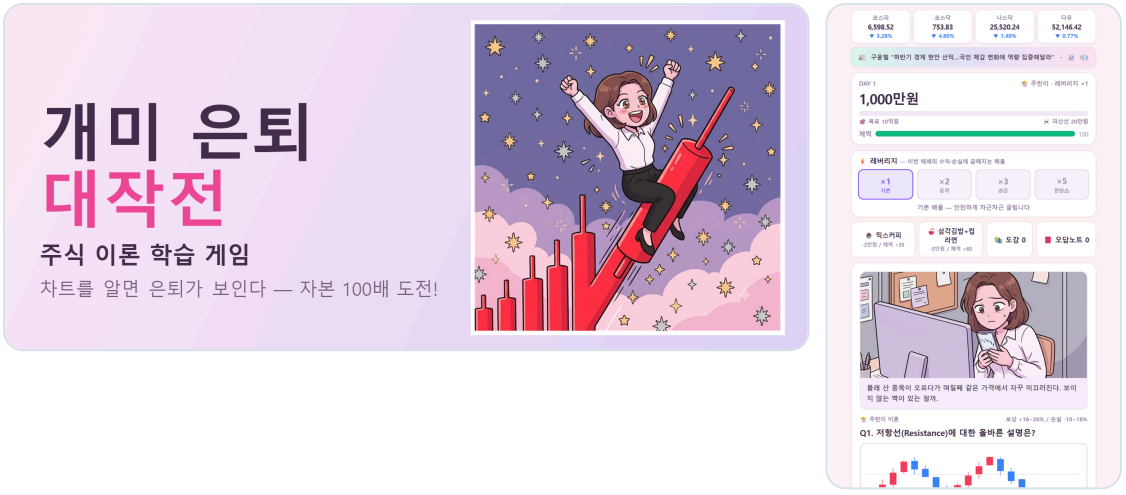
토스 앱 안에서 실행되는 미니앱, "앱인토스"예요. 세 스토어 중 진입장벽이 제일 낮고(심사 1~2일), 등록비도 없어요. 첫 출시 경험으로 딱이에요.

6-1. 사람이 할 일

1. **앱인토스 개발자 콘솔에 가입**해요 (apps-in-toss 개발자 페이지). 로그인이 **본인 명의 토스 앱 QR 인증**이라 이건 꼭 직접 해야 해요.
2. 나중에 등록 마지막 단계의 **법적 확인 체크박스들과 "검토 요청" 버튼** — 이것도 직접 읽고 눌러요. (인허가·책임에 대한 약속이라 AI가 대신 체크하면 안 되는 항목이에요.)
3. 나머지는 전부 딸깍이에요.

6-2. 토스용으로 포장하기 딸깍

```
이 앱을 앱인토스(토스 미니앱)에 올릴 준비를 해줘.
- 토스용 번들(.ait 파일)을 만들어줘
- 웹 버전 배포와 폴더가 꼬이지 않게 조심해줘
- 실수로 당겨서 새로고침되거나 뒤로 스와이프로 진행이 날아가지 않게 웹뷰 설정도 챙겨줘
- 스토어 제출용 자산도 규격대로 만들어줘:
  로고 600×600(불투명 배경, 다크모드용도 한 장 더), 썸네일 1932×828, 세로 스크린샷 636×1048 4장
- 콘솔에 넣을 문구도 써줘: 앱 이름(10자), 영문명(15자, 명사형), 부제(20자), 상세 설명(500자), 키워드
- 함정이 많은 곳이니 이 문서를 먼저 읽고 시작해줘: [족보 ⑤를 여기 붙여넣기]
```



AI가 만들어주는 제출용 자산 예시 — 썸네일(왼쪽)과 세로 스크린샷(오른쪽). 실제 앱 화면을 자동으로 찍어서 만들어요

6-3. 콘솔에 입력하기

콘솔 입력도 AI가 브라우저를 조작해서 대신할 수 있어요(로그인만 여러분이 해주면요).

```
앱인토스 콘솔에 앱 정보를 등록하자.
- 크롬을 원격 조작 모드로 띄워줘. 내가 QR로 로그인할게
- 로그인되면 앱 만들기부터 이미지 업로드, 문구 입력까지 채워줘
- 단, 법적 확인 체크박스와 최종 '검토 요청' 버튼은 절대 누르지 말고 나한테 넘겨줘
```

다 채워지면 여러분이 화면을 한 번 훑어보고, 체크박스 읽고, 검토 요청을 눌러요. 보통 1~2일이면 결과가 와요.

한 가지 조심할 것: 이 콘솔은 **"임시저장" 버튼이 실제로는 검토 요청처럼 처리된 사례**가 있어요. 그러니 처음부터 "제출해도 되는 상태"로 채운다고 생각하는 게 마음 편해요.

6-4. 카테고리 꿀팁

게임을 "게임" 카테고리로 등록하면 게임 등급분류라는 서류 절차가 따라와요. 그런데 퀴즈·학습·콘텐츠성 앱은 **"생활" 같은 비게임 카테고리**로 등록하면 그 절차가 통째로 사라져요. 실제로 게임처럼 보이는 앱이 심사에서 "생활>콘텐츠"로 분류된 경우도 있었어요. 애매하면 비게임 쪽이 편해요.

6-5. 출시 후에 광고 붙이기 (선택, 9장 미리보기)

토스에는 자체 인앱 광고가 있어요. 콘솔의 「**수익화**」 **메뉴에서 광고 그룹을 만들고, AI에게 그룹 ID를 알려주면** 연동해줘요. 여기서 하나만 기억하세요 — 인앱 광고는 **돈을 버는 쪽**이에요. 콘솔에서 돈이 나가는 건 푸시 발송·포인트 프로모션 같은 마케팅 기능뿐이니, "광고 붙였는데 돈 나가는 거 아니야?" 걱정은 안 해도 돼요.

새 버전을 올렸는데 뭔가 반영이 안 된 것 같으면, AI에게 이렇게: 「콘솔에서 지금 출시된 버전이 며칠자 빌드인지 확인해줘」 — 옛 버전이 그대로 출시 중인 게 원인인 경우가 많아요.

7장 구글 플레이에 올리기

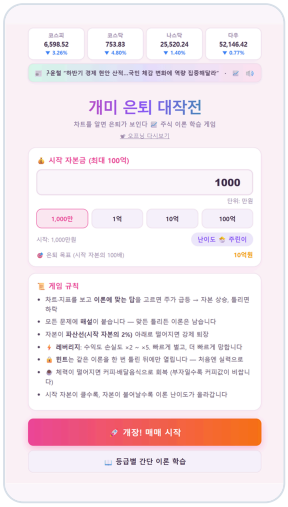
안드로이드 폰에 깔리는 진짜 앱이 되는 순간이에요. 등록비는 평생 한 번 $25.

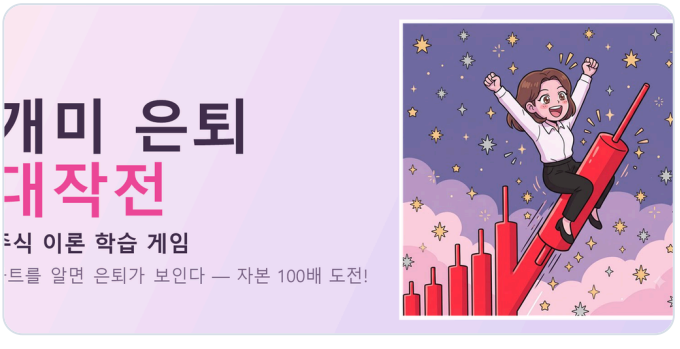
7-1. 사람이 할 일

1. **Google Play Console 개발자 계정 가입** ($25, 본인 인증 필요 — 신분증 확인이 있어서 하루쯤 걸릴 수 있어요)
2. **서명 열쇠(키스토어) 백업** — 아래 7-2에서 AI가 만들어주는데, 이 파일과 비밀번호는 **앱의 평생 신분증**이에요. 잃어버리면 앱 업데이트를 영영 못 올려요. 클라우드와 USB, 두 군데에 보관하세요.
3. **콘텐츠 등급 설문의 마지막 "다음" 버튼** — 신기하게도 이 버튼 하나만은 자동화가 안 돼요(구글이 사람 확인용으로 막아둔 걸로 보여요). AI가 설문을 다 채워놓고 부르면, 그 창에서 버튼만 눌러주세요.
4. 최종 심사 제출 버튼.

7-2. 안드로이드 앱으로 포장하기 딸깍

```
이 웹앱을 구글 플레이에 올릴 안드로이드 앱으로 만들어줘.
- 원본 폴더는 건드리지 말고 별도 래퍼 폴더에서 작업해줘
- 서명용 키스토어를 만들고, 파일 위치와 비밀번호를 나한테 알려줘 (내가 백업할게)
- 출시용 AAB 파일을 빌드하고 서명까지 확인해줘
- 스토어 등록에 필요한 것도 다 만들어줘:
  아이콘 512, 그래픽 이미지 1024×500, 폰 스크린샷(9:16으로 변환),
  간단한 설명(80자)과 자세한 설명
- 개인정보처리방침 주소는 4장에서 만든 웹 페이지를 쓰면 돼
- 세부 절차와 함정은 이 문서대로: [족보 ⑥을 여기 붙여넣기]
```





AI가 만들어주는 구글 등록 자산 예 — 9:16으로 변환된 폰 스크린샷 2장과 그래픽 이미지(1024×500). 원본 화면 비율이 달라도 배경을 붙여 규격을 맞춰줘요

7-3. 콘솔 입력 딸깍

```
Play Console 입력을 도와줘.
- 자동화용 크롬을 띄워줘. 로그인은 내가 할게
- 스토어 등록정보, 앱 콘텐츠 선언(데이터 보안, 타겟층, 광고 여부 등)을 채워줘
- 콘텐츠 등급 설문은 답을 다 채운 다음, 마지막 '다음' 버튼에서 멈추고 나를 불러줘
- 최종 제출 버튼도 누르지 말고 나한테 넘겨줘
```

구글 콘솔은 입력할 게 유난히 많은데(선언만 9종류), 그래서 자동화 가치가 제일 큰 곳이기도 해요. AI가 채우는 걸 구경하다가, 불려가면 버튼 두 개(등급 설문 다음 버튼, 최종 제출)만 눌러주세요.

7-4. 뒷얘기 — 신규 계정의 "테스터 12명" 관문

솔직한 뒷얘기 하나 할게요. 2023년 11월 13일 이후에 만든 **개인** 개발자 계정은, 정식 출시 신청 전에 **비공개 테스트에 테스터 12명을 모아 14일 연속으로 유지**해야 하는 관문이 있어요(원래 20명이었다가 12명으로 완화됐어요). 12명이 실제 기기에 설치하고 옵트인 상태를 끊김 없이 유지해야 해서, 혼자 하는 사람에게는 이게 심사보다 어렵다는 말이 나와요.

그래서 시중에는 이 요건이 없는 **"옛날 계정"(2023년 11월 이전 생성)을 사고파는 거래**까지 생겼어요. "예전 계정 구합니다, 대신 올려드립니다" 같은 글이 그거예요. 하지만 계정 매매·대리 등록은 구글 정책 위반이라 걸리면 앱과 계정이 통째로 날아가고, 연결된 계정까지 정지될 수 있어요. **추천하지 않아요.**

정공법은 이 셋 중 하나예요.

1. **지인 12명 모으기** — 가족·친구 단톡방이면 생각보다 금방 모여요. 링크 눌러 설치하고 2주만 지우지 말아 달라고 부탁하면 끝이에요.
2. **테스터 품앗이 커뮤니티** — 개발자들끼리 서로 테스터가 되어 주는 커뮤니티가 여럿 있어요. AI에게 "구글 플레이 테스터 품앗이 커뮤니티 찾아줘"라고 하면 돼요.
3. **사업자가 있다면 조직(법인/사업자) 계정으로 가입** — 조직 계정은 이 요건이 아예 면제예요.

7-5. 알아두면 좋은 것

- 심사는 보통 며칠 걸려요. 첫 앱은 신규 개발자 검토가 붙어서 더 걸릴 수 있어요.
- 폰 스크린샷은 9:16 비율이 강제라서, 다른 비율 화면은 AI가 배경을 붙여 변환해줘요(딸깍에 이미 포함).
- 토스에 게임 카테고리로 내고 싶어질 때, "구글에 먼저 출시 → 등급 정보 획득 → 토스에 입력" 순서가 되기도 해요. 구글 등록을 해두면 여기저기 쓸모가 생겨요.

8장 아이폰 앱스토어에 올리기 — 맥 없어도 돼요

"아이폰 앱은 맥이 있어야 만든다"는 말, 이제 옛말이에요. 빌드는 **깃허브(GitHub)가 빌려주는 클라우드 맥**에서 돌리고, 등록·심사 제출은 애플의 공식 API로 AI가 해요. 이 방법으로 윈도우 PC에서 앱 여러 개가 실제로 심사까지 갔어요.

8-1. 사람이 할 일 (여기가 이 책에서 제일 손이 많이 가는 곳이에요, 그래봤자 몇 번의 클릭이지만요)

1. **애플 개발자 프로그램 가입** — developer.apple.com, 연 $99. 승인까지 하루 이틀 걸려요.
2. **GitHub 무료 가입** — github.com. 클라우드 맥을 빌리기 위해서예요.
3. **애플 API 열쇠 만들기** — App Store Connect 사이트에서 "관리자(Admin)" 권한의 API 키를 하나 만들어 내려받아요. AI가 화면을 같이 보면서 어디를 누를지 알려줄 수 있어요. 딱 하나 기억하세요: **권한은 꼭 Admin으로** (낮은 권한이면 클라우드 서명이 안 돼요).
4. 심사 결과 **메일 확인** — 보통 1~3일 뒤에 와요.

8-2. 아이폰 앱 만들어 올리기 딸깍

```
이 앱을 아이폰 앱스토어에 올리자. 나는 맥이 없어.
GitHub Actions 클라우드 빌드 방식으로 가자.

- 원본은 건드리지 말고 iOS 래퍼 폴더를 따로 만들어줘
- 아이콘과 시작 화면은 꼭 우리 앱 그림으로 바꿔줘 (기본 로고로 내면 100% 반려래)
- 햅틱(터치 진동) 같은 네이티브 기능을 한두 개 넣어줘
- GitHub 저장소 만들고, 클라우드 빌드 설정하고, 테스트플라이트 업로드까지 해줘
- 앱스토어 등록도 해줘: 앱 생성, 설명·키워드, 스크린샷(아이폰+아이패드 둘 다!),
  연령 등급, 개인정보 라벨, 심사 메모(한/영)까지
- ⚠ 판매 지역(다운로드 가능한 나라) 설정을 절대 빼먹지 마.
  이거 빼먹으면 출시돼도 아무도 다운로드를 못 받아
- 심사 제출은 '승인되면 자동 출시'로 해줘
- 순서와 함정은 이 문서 3개를 그대로 따라줘:
  [족보 ⑦(맥 없이 빌드), ⑧(등록 자동화), ⑨(반려 예방)를 여기 붙여넣기]
- 내가 직접 해야 하는 단계(가입, 키 만들기, 로그인)가 나오면 그때그때 멈추고 알려줘
```

이 딸깍은 좀 길게 돌아가요. 중간에 AI가 몇 번 여러분을 부를 거예요("API 키 만들 차례예요", "여기 로그인해 주세요"). 부를 때만 가서 해주면 돼요.

8-3. 심사, 무서워하지 않아도 돼요

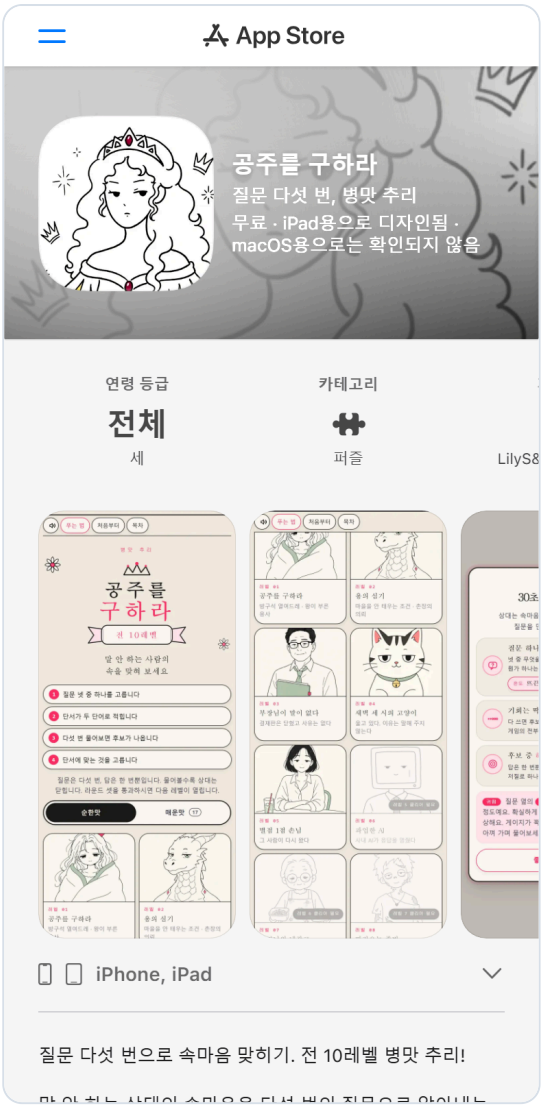
애플 심사가 제일 깐깐한 건 사실이에요. 그런데 반려 사유는 놀랄 만큼 뻔해서, 미리 다 막을 수 있어요(그게 족보 ⑨예요). 핵심만 맛보기로:

- 기본 아이콘 그대로 내기 = 무조건 반려 → 딸깍에 이미 "바꿔줘"가 들어 있어요
- "이거 그냥 웹사이트 아니야?"라는 의심 → 오프라인 완전 동작 + 햅틱 + 심사 메모로 방어
- 쓰지도 않는 푸시 알림 권한 넣기 → 오히려 감점이에요. 우린 안 넣어요

혹시 반려 메일이 와도 괜찮아요. 메일 내용을 통째로 복사해서 AI에게 주세요: 「반려됐어. 이거 읽고 고쳐서 재제출까지 해줘」 — 재심사는 처음보다 빨라요.

8-4. 승인되면

"승인되면 자동 출시"로 제출했으니, 심사 통과 메일이 오는 순간 앱스토어에 여러분의 앱이 올라가 있어요. 검색해서 스크린샷 찍고 자랑하세요. 여기까지 온 사람, 정말 많지 않아요.



실제로 이 파이프라인으로 출시된 앱의 앱스토어 페이지예요. 윈도우 PC에서, 맥 없이, 이 책의 딸깍 순서 그대로 만들어졌어요

9장 광고·구독으로 용돈 벌기

앱이 스토어에 올라갔으니, 이제 수익 장치를 달아볼게요. 과장 없이 말할게요 — 처음엔 커피값이에요. 하지만 구조를 만들어두면 앱이 늘 때마다 수익 통로도 늘어요.

9-1. 뭐가 어디에 붙는지 지도부터

- 아이폰·안드로이드 앱 → **애드몹(AdMob)** 광고
- 토스 미니앱 → **토스 인앱 광고** (6장에서 살짝 봤죠)
- 웹사이트 → **애드센스, 카카오 애드핏** (심사 필요), **쿠팡 파트너스** (심사 없음, 바로 가능)
- 유료 기능 → **광고 제거 구독** 같은 인앱 결제

9-2. 사람이 할 일 — 여기도 "가입하러 가세요"가 아니에요

돈 관련이라 어려울 것 같죠? 여기도 시작은 똑같아요: 「**로그인 창 띄워줘, 내가 로그인할게. 그다음은 네가 해**」.

1. **애드몹** — 가입 화면 열기, 앱 등록, 광고 단위 발급까지 AI가 브라우저로 진행해요. 여러분은 구글 로그인만. (계정 승인에 하루쯤 걸려요)
2. **구독 상품 개설** — 이것도 딸깍이에요. 실제로 상품 만들기, 175개 지역 가격 설정, 무료 체험, 심사 제출까지 전부 AI가 콘솔과 API로 해냈어요. 여러분은 로그인 몇 번.
3. **유료 앱 계약**(구독 팔 때 필수) — 화면 열기와 항목 안내까지는 AI가 해줘요. 다만 **계약 서명, 계좌번호, 세금 정보** 같은 민감한 확정 입력은 계정 주인이 직접 하는 게 원칙이에요. 구독 계획이 있다면 미리 해두세요.
4. 그리고 제일 중요한 약속: **내 광고는 내가 누르지 않기.** 가족 동원도 안 돼요. 걸리면 수익 회수에 계정 정지까지 가요. 진짜예요.

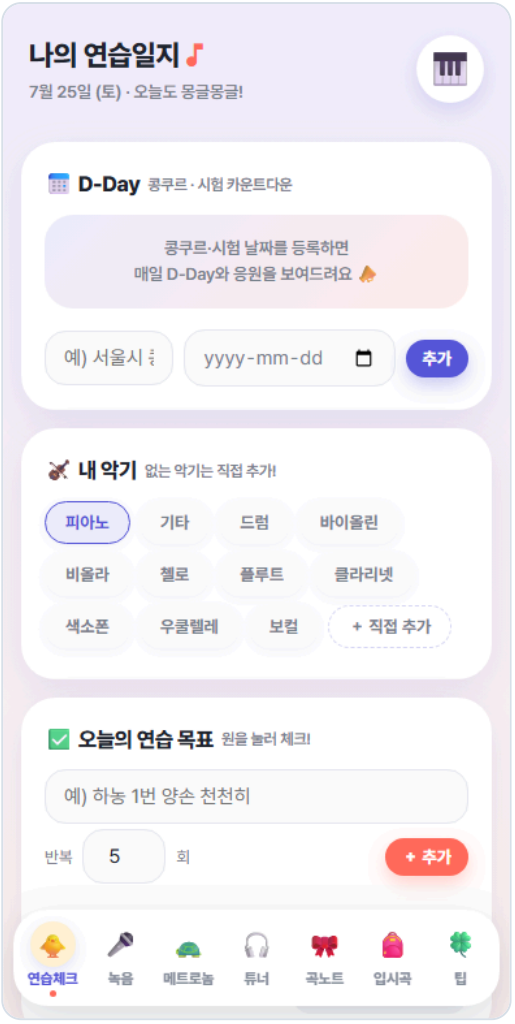
9-3. 앱에 광고 붙이기 딸깍

```
이 앱에 애드몹 광고를 붙이자.
- 배너(하단 상시)와 전면 광고(자연스러운 쉬는 타이밍)로
- 전면 광고는 절대 과하지 않게: 횟수 조건에 더해 최소 시간 간격도 걸어줘
- 광고가 안 떠도 앱 진행이 절대 막히지 않게 해줘
- 애드몹 콘솔에서 앱 등록과 광고 단위 만들기도 도와줘 (브라우저 자동화로, 로그인은 내가)
- 출시 빌드에서 테스트 광고 모드가 꺼졌는지 꼭 확인해줘
- 세부 함정은 이 문서대로: [족보 ⑪을 여기 붙여넣기]
```

광고 빈도는 "많이 = 많이 번다"가 아니에요. 유저가 떠나면 0원이거든요. 결과 화면 먼저, 광고는 그다음 — 이 원칙만 지켜도 리뷰 테러는 안 당해요.

9-4. 광고 제거 구독 딸깍 (선택)

```
'광고 제거' 연 구독을 붙이자. 가격은 ooo원.
- 구매 버튼은 스토어에서 상품이 조회될 때만 나타나게 해줘
  (유료 앱 계약이 아직이어도 심사에 안전하게)
- 복원 버튼, 가격·자동갱신 안내 문구, 구독 관리 링크까지 심사 기준대로
- 애플 쪽 상품 개설은 순서가 까다로우니 이 문서대로 해줘: [족보 ⑩을 여기 붙여넣기]
- 첫 구독은 앱 버전과 함께 심사에 넣어야 한다니까 그 부분은 나랑 같이 하자
```



광고 제거 연 구독을 붙여 앱스토어에 낸 연습일지 앱. 웹 버전(사진)은 무료 그대로 두고, 앱에서만 광고와 구독이 돌아가요

9-5. 웹 광고는 조급해하지 않기

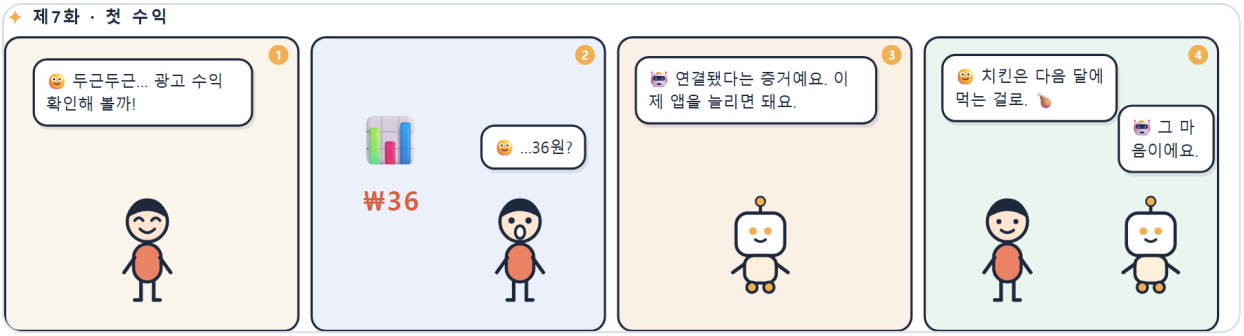
애드센스·애드핏은 사이트에 **읽을거리와 방문자**가 좀 쌓여야 심사를 통과해요. 앱 몇 개랑 소개 글만으로는 부족할 수 있어요. 순서를 추천하면:

1. 지금 당장: **쿠팡 파트너스** (심사 없이 시작, AI에게 "쿠팡 파트너스 배너 붙여줘. 고지 문구도 자동으로"라고 하면 돼요)
2. 앱이 3~4개 모이면: 도메인 사서 **포털 사이트** 만들기 (4장 끝에서 본 그 구조 — 「내 앱들을 모은 포털 사이트를 만들어줘. 앱마다 소개 페이지랑 블로그도」)
3. 글이 10편쯤 쌓이면: **애드핏 → 애드센스** 순서로 신청

하나 신기한 걸 알려드리면, 광고가 나오는지 AI가 화면으로 확인하는 건 불가능한 경우가 있어요(광고사가 자동화 브라우저에는 광고를 안 보여주거든요). **"광고 잘 나오나?"는 광고사 콘솔의 보고서 숫자로 확인**하는 거예요. AI가 "화면에 광고가 안 보여요"라고 해도 당황하지 마세요.

9-6. 현실적인 기대치

노출 수십 번에 몇십 원 찍히는 날부터 시작해요. 실망 금지! 그 숫자가 "연결은 됐다"는 증거예요. 그다음은 앱과 콘텐츠를 늘려서 노출을 키우는 게임이에요. 요금이 나갈까 봐 걱정되는 분들께 다시 한번: 광고는 **버는 쪽**이에요. 돈이 나가는 건 여러분이 직접 결제하는 것(개발자 등록비, 도메인)뿐이에요.



쉬어가기 — 여기서부터는 AI에게 주는 "족보"예요

축하해요, 사람이 읽는 부분은 여기까지예요. 🎉

지금부터 이어지는 **족보** ①~⑬은 제가 앱들을 실제로 출시하면서 밟았던 함정과 해결책을 전부 적어둔 기술 자료예요. 일부러 어려운 말을 안 풀었어요. 왜냐하면 **여러분이 읽을 글이 아니라, AI가 읽을 글**이거든요.

족보 사용법 (이게 전부예요)

1. 본편의 딸깍 문장에 `[족보 o를 여기 붙여넣기]`가 나오면 → 해당 족보를 **처음부터 끝까지 복사**해서 그 자리에 붙여넣어요.
2. 뭔가 막혔을 때 → 관련 족보를 통째로 복사해서 AI에게 주면서: 「이 자료 읽고, 지금 문제를 다시 봐줘」
3. 그게 다예요. 이해하려고 하지 마세요. 진심이에요.

족보 차림표

- 족보 ① 작업 환경 — 설치가 꼬였을 때, 요금 구조가 헷갈릴 때
- 족보 ② 그림 파이프라인 — 배경 제거가 지저분할 때, 그림이 이상한 위치에 붙을 때
- 족보 ③ 앱 제작 기준 — 새 앱 만들 때 (3장 딸깍의 단짝)
- 족보 ④ 웹 배포 — 배포가 안 되거나 옛 화면이 계속 보일 때
- 족보 ⑤ 앱인토스 — 토스 빌드·콘솔·인앱 광고 전 과정
- 족보 ⑥ 구글 플레이 — 안드로이드 패키징과 콘솔 자동화
- 족보 ⑦ 맥 없이 iOS 빌드 — 깃허브 클라우드 빌드 전 과정
- 족보 ⑧ 앱스토어 커넥트 자동화 — 등록·스크린샷·심사 제출 API
- 족보 ⑨ 반려 예방 — 애플 심사 통과 전략
- 족보 ⑩ 구독·인앱결제 — 상품 개설 순서와 심사 함정
- 족보 ⑪ 광고 수익화 — 애드몹·애드센스·애드핏·쿠팡 전부
- 족보 ⑫ 브라우저 자동화 — 콘솔 입력을 AI가 대신하는 기술
- 족보 ⑬ 최종 체크리스트 + 함정 사전 40선 + 명령 모음

그럼, 즐거운 딸깍 생활 되세요. 첫 앱이 스토어에 올라가면 꼭 어딘가에 자랑하시고요. 🚀

족보 ① 작업 환경 상세 — CLI 설치와 과금 구조

1-1. Node.js를 관리자 권한 없이 설치하기

거의 모든 도구(AI CLI, 빌드 스크립트, 배포 CLI)가 Node.js 위에서 돌아요. 회사 PC처럼 관리자 권한이 없어도 설치할 수 있어요.

1. nodejs.org에서 **ZIP 배포판**(Windows Binary)을 받으세요. 설치 프로그램(msi)이 아니라 ZIP이에요.
2. `%LOCALAPPDATA%\Programs\nodejs` 폴더에 압축을 푸세요.
3. 사용자 환경변수 PATH에 그 폴더를 추가하세요(시스템 설정 → "계정의 환경 변수 편집").

함정: 이미 열려 있는 프로그램은 새 PATH를 몰라요. 터미널·에디터를 껐다 켜거나, PowerShell이라면 명령 앞에 이 한 줄을 붙이면 즉시 해결돼요.

```
$env:Path = "$env:LOCALAPPDATA\Programs\nodejs;$env:Path"
```

AI 에이전트가 "node를 찾을 수 없다"고 할 때도 십중팔구 이 문제예요. 프로젝트 메모(CLAUDE.md 등)에 이 한 줄을 적어 두면 AI가 알아서 붙여요.

1-2. AI CLI 3종 설치

세 도구 모두 npm 한 줄이면 끝나요. 공통점은 **월정액 구독 계정으로 브라우저 OAuth 로그인**하면 API 키 없이 동작한다는 거예요.

도구	설치	로그인
Claude Code	`npm i -g @anthropic-ai/claude-code`	`claude` 실행 → 구독 계정 OAuth
Codex CLI (ChatGPT)	`npm i -g @openai/codex`	ChatGPT 구독 계정 OAuth
Grok CLI	`npm i -g @xai-official/grok`	`grok login` → SuperGrok/X Premium+ OAuth

Grok CLI 관련 실측 메모 두 가지예요.

- npm 설치 때 postinstall 스크립트가 보안 설정으로 차단되어도 걱정할 것 없어요. 첫 실행 시 실행 파일이 `~/.grok/bin/`에 자체 배치되므로 그대로 동작해요.
- 인증이 만료되면 `grok login`을 **사람이 직접** 다시 실행해야 해요. 브라우저 OAuth라 AI가 대신 못 해요.

셋 중 주력은 하나면 돼요. 이 책의 모든 앱은 Claude Code 하나로 만들어졌고, 나머지는 보조(다른 시각의 검토, 특정 모델이 강한 작업)로 써요.

1-3. 가장 중요한 원칙: 구독과 API 크레딧은 다른 지갑이다

여기서 돈이 새요. 반드시 이해하고 넘어가세요.

- **구독(월정액)**: Claude Pro/Max, ChatGPT Plus, SuperGrok 등. CLI 대화, 웹 채팅, 웹 화면에서의 이미지 생성이 포함돼요. 아무리 써도 정해진 요금이에요.
- **API 크레딧(종량제)**: 코드가 `api.anthropic.com`, `api.openai.com`, `api.x.ai` 같은 주소를 직접 호출하면 **건당 과금**돼요. 구독을 아무리 비싼 걸 써도 API 요금은 별도예요.

예를 들어 SuperGrok 구독이 있어도 xAI API로 이미지를 생성하면 콘솔에 별도 결제가 쌓여요. "구독 있으니 공짜겠지"가 가장 흔한 착각이에요.

실전 규칙은 이래요.

1. 코딩·글쓰기·디버깅·배포 = **CLI(구독)로.** 여기가 작업량의 90%예요.
2. 이미지·대량 콘텐츠 생성 = **웹 구독 화면에서.** 방법은 족보 ②에서 상세히.
3. 앱 안에 AI 기능을 넣을 때만 API를 검토하되, 그 전에 "이 기능이 정말 서버 호출이 필요한가"를 의심하세요. 이 책의 앱들은 전부 API 호출 0건으로 만들어졌어요(사운드는 웹오디오 합성, 차트는 절차 생성, 문제는 내장 데이터).

1-4. 프로젝트 폴더 위생

AI 에이전트는 작업 폴더를 통째로 훑어요. `node_modules`와 브라우저 프로필 폴더가 들어 있으면 파일 58만 개를 뒤지느라 느려지고 경고가 떠요. 작업 루트에 `.gitignore`와 `.ignore`(ripgrep용) 파일을 만들어 다음을 제외하면 3천 개 수준으로 줄어들어요.

```
node_modules/
.*-profile/
dist/
*.ait
```

또 하나. 프로젝트마다 `CLAUDE.md`(또는 AGENTS.md)에 **그 프로젝트의 빌드·배포 명령과 함정**을 적어 두면, 다음 세션의 AI가 같은 삽질을 반복하지 않아요. 이 책 자체가 그 메모들의 집대성이에요.

족보 ② 그림 파이프라인 — "지시서 패턴" 상세

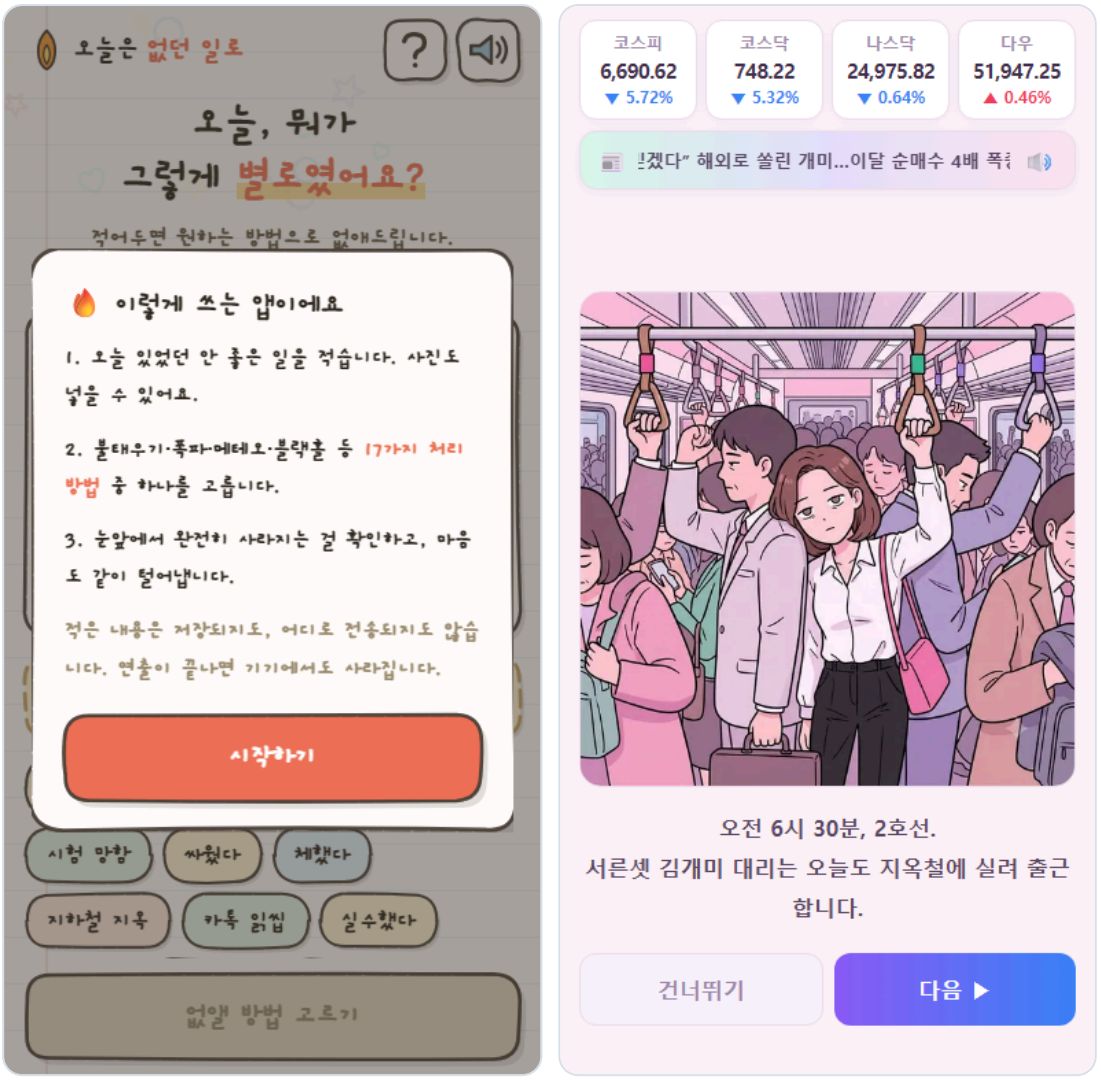
앱 하나에 들어가는 그림은 생각보다 많아요. 캐릭터 30장, 소품 14장, UI 아이콘 24장, 스토어용 스크린샷과 로고까지. 이걸 이미지 API로 뽑으면 장당 과금이 쌓이지만, **ChatGPT Plus나 Gemini 유료 구독의 웹 화면에서 생성하면 추가 과금 없이(구독 등급별 생성 한도 내에서)** 뽑을 수 있어요. 구독료 자체는 내는 것이므로 "공짜"가 아니라 "한계비용 0원"이 정확해요. 한도에 걸리면 시간을 두고 나눠 뽑거나 서비스를 갈아타세요. 문제는 웹 화면 생성을 "체계적으로" 시키는 방법인데, 답은 지시서예요.

2-1. 지시서(ART BRIEF) 패턴

AI 코딩 도구에게 그림을 직접 그리게 하지 말고, **그림 생성용 지시서 문서를 쓰게 하세요.** 지시서는 파일(예: `art/ART_BRIEF.md`)로 저장해 두는 게 좋아요 — 같은 작업 폴더에 띄운 GPT(Codex) 에이전트 탭에 "이 파일 읽고 그림 생성해서 art/에 지시서의 파일명대로 저장해줘"로 위임할 수 있거든요(ORCA 멀티탭 협업, 사람 개입 최소). 에이전트가 이미지 생성을 지원하지 않으면 사람이 ChatGPT/Gemini 웹 화면에 붙여넣고 결과를 저장하는 수동 릴레이로 폴백해요. 지시서에 반드시 들어가야 하는 요소:

1. **공통 스타일 블록** — 모든 프롬프트 맨 앞에 붙는 동일한 문단(그림체, 색, 선 굵기, 캐릭터 외형). 이게 있어야 30장이 한 사람 그림처럼 나와요. 예: "손그림 크레용/수채, 그림일기 톤" 또는 "양복 입은 개미 캐릭터, 라이트 파스텔".
2. **파일별 프롬프트 + 파일명 규약** — `bomb.png`, `icon-fire.png`처럼 결과물 파일명을 지시서에 못 박아요. 후처리 스크립트가 파일명으로 자동 분류해요(예: `icon-` 접두사는 자동으로 작게 리사이즈).
3. **캔버스 크기와 앵커** — 그림이 코드에서 어디에 물리는지(파쇄기 투입구, 용의 입 위치 등)를 비율로 지정.
4. **재요청 신호** — "이렇게 나오면 다시 요청하세요" 목록(잘림, 배경 섞임, 스타일 이탈).
5. **그리지 말 것 목록** — 테두리·배경 낙서·글씨는 코드가 처리하니 그리지 말라고 명시. 이 한 줄이 재생성 낭비를 크게 줄여요.

실전에서 이 패턴으로 소품 14장(1차)과 UI 아이콘 24장(2차)을 받았는데, 2차는 전량 지시서대로 도착해 수정 요청이 거의 없었어요.



지시서 패턴의 결과물 — 왼쪽: ChatGPT 웹에서 받은 크레용 손그림·아이콘을 입힌 앱, 오른쪽: Gemini 웹에서 받은 캐릭터 일러스트 30장을 쓴 게임 오프닝

2-2. 받은 그림 후처리는 전부 자동화

사람이 하는 일은 "그림을 `art/` 폴더에 넣기"까지예요. 그다음은 스크립트가 해요. 처리 파이프라인(sharp 라이브러리 기준):

- 배경 제거 → 여백 트림 → 최대 크기 리사이즈(본문용 720px, 아이콘 168px) → webp 변환 → 앱에 data URI로 인라인(외부 요청 0건 유지)

여기서 얻은 실측 교훈들:

- **배경색은 마젠타처럼 채도 높은 색으로 요청**하면 제거가 깨끗해요. 손가락 사이, 입 안 같은 "갇힌 구멍"까지 전역 제거할 수 있어요.
- **"가장자리에서 가장 많은 색 = 배경"이라는 가정은 깨져요.** 그림이 화면 가장자리까지 차 있으면 그림 자체가 지워져요. 후보 색마다 실제로 flood fill을 돌려 보고, 그림 한가운데를 지우는 후보는 탈락시키는 검증을 넣어야 안전해요.
- **트림하면 앵커 비율이 틀어져요.** 원본 크기와 트림 박스를 메타 파일(`_meta.json`)에 기록하고, 앱 코드가 보정하게 하세요.

- **지시서에 적은 앵커와 실제 그림 속 위치는 달라요.** 도착한 그림 위에 격자를 그려 주는 실측 도구를 만들어 값을 전부 다시 쟀어요(예: 파쇄기 투입구 0.22로 지시했지만 실물은 0.369). "지시서 값 믿지 말고 실측"이 원칙이에요.

2-3. 폴백 설계 — 그림이 없어도 앱은 돈다

코드에서 그림을 쓸 때는 항상 `art(이름)`이 null이면 도형/이모지로 그리는 폴백을 두세요. 그러면 그림을 **한 장씩 넣어 가며** 개발할 수 있고, 특정 그림이 마음에 안 들어 빼도 앱이 안 깨져요. 그림 34장을 이 방식으로 무중단 교체했어요.

2-4. 글씨는 이미지가 아니라 폰트로

"제목을 크레용 손글씨처럼, 이미지로라도"라는 요구가 있었는데, 정답은 이미지가 아니라 **폰트 임베드**였어요. 글자가 바뀌어도 되고, 선명하고, 용량도 작아요. 무료 폰트(SIL OFL 라이선스 = 상업·임베드 허용) 중 고르고, 서브셋 도구로 한글 상용 2,350자+앱에 쓰는 문자만 남기면 3.1MB TTF가 277KB woff2가 돼요. 주의: 굵은 웨이트(700)가 기본(400)보다 파일이 클 수 있으니 하나만 쓰고 브라우저 합성 볼드로 때워요. 그리고 `<button>`은 폰트를 상속하지 않으니 CSS에 `font-family: inherit`를 잊지 마세요.

2-5. 품질 검수도 스크립트로

- **컨택트 시트**: 연출·아이콘을 한 장의 격자 이미지로 모아 렌더링해서 한눈에 검토. 배경 제거 잔여물은 밝은 배경+어두운 배경 두 장으로 잡아요.
- **배치 확인**: 실제 화면에 그림을 얹은 스크린샷을 자동 생성해 깨진 이미지 수를 세어요.

이 장의 요지: **사람은 붙여넣기와 저장만, 판단은 시트 보고, 나머지는 전부 코드.** 이미지 API 비용은 이 구조에서 0원이에요.

족보 ③ 앱 제작 기준 — 단일 HTML 철학과 자가 검증

3-1. 왜 "자기완결형 HTML 한 장"인가

이 책의 앱 대부분은 최종 산출물이 **index.html 한 장**이에요. 프레임워크 빌드 체인 없이, 조립 스크립트(`build.mjs`)가 `src/*.js`, `src/*.css`를 순서대로 이어 붙여 한 파일로 만들어요. 이유는 명확해요.

- **웹뷰는 적대적 환경이에요.** 앱인토스 웹뷰는 `file://`로 열리기도 하는데, 이때 ESM 모듈 로딩과 외부 요청이 막혀요. 실제로 토스 테스트에서 JS가 통째로 실행되지 않은 사고가 있었고, 해결책은 전부 인라인(단일 파일화)이었어요. Vite를 쓰는 프로젝트라면 `vite-plugin-singlefile`이 같은 역할을 해요.
- **외부 요청 0건 원칙**: 폰트는 서브셋 임베드, 효과음은 웹오디오 합성, 이미지는 data URI. 네트워크가 없어도 완전 동작하는 앱은 iOS 심사(오프라인 동작)에서도 유리해요.

3-2. 부팅 진단 오버레이 — 원격 디버깅이 안 될 때

웹뷰 안에서 뭐가 죽었는지는 밖에서 안 보여요. HTML에 이런 안전장치를 심어요: 앱이 정상 부팅되면 `__GAME_BOOTED` 플래그를 세우고, **4초 안에 플래그가 없으면 화면에 빨간 박스로 오류 메시지·URL·UA를 표시**해요. 사용자(또는 심사자)가 그 화면을 스크린샷 찍어 주면 원인을 바로 특정할 수 있어요. 실기기를 못 만지는 원격 개발에서 이 장치가 여러 번 살렸어요.

3-3. 검증은 눈이 아니라 스크립트로

AI에게 만들게 했으면 검증도 AI가 스크립트로 하게 하세요. 표준 세트:

- **스모크 테스트**: Playwright로 전 기능을 실제 클릭해 보고 콘솔 에러 0건 확인(예: 파괴 연출 17종 전수 실행).
- **뷰포트 테스트**: 320~430px 전 구간에서 가로 넘침 0px, fps 실측. 특히 스크롤 컨테이너의 `scrollWidth` 넘침은 문서 전체만 봐서는 못 잡으니 컨테이너 단위로 검사.
- **컨택트 시트**: 애니메이션 연출은 6프레임 시트로 뽑아 한눈에 품질 검토. 캡처 시점은 시간(ms)이 아니라 **진행률 기준**으로 잡아야 매번 같은 그림이 나와요.
- **배포 후 라이브 체크**: 배포 URL에서 부팅 확인 + 외부 요청 0건 검증.

Playwright는 프로젝트마다 설치할 필요 없어요. 한 프로젝트에 설치해 두고 다른 프로젝트에서 `createRequire`로 빌려 쓰면 돼요.

3-4. 모바일 웹 특유의 함정들

전부 실기기에서 실제로 터진 것들이에요.

- **iOS 키보드가 화면을 가려요**: `body{position:fixed}` 레이아웃에서 iOS는 키보드가 떠도 레이아웃 높이를 줄이지 않아요. 하단 버튼이 키보드 뒤에 깔려요(실측 324~388px). 해결: `visualViewport.height`에 body 높이를 맞추는 스크립트.

- **iOS 무음 스위치**: 웹오디오가 무음 모드에서 죽어요. 무음 WAV를 `<audio>`로 루프시켜 오디오 세션을 미디어 모드로 전환 + 사용자 제스처 안에서 버퍼 재생으로 잠금 해제.
- **pointerup에서 파일 선택창 열지 마세요**: 터치에서 `input[type=file].click()`을 pointerup 핸들러에서 부르면 브라우저가 사용자 조작으로 인정하지 않아 창이 안 열려요. 평범한 click 핸들러를 쓰세요.
- **overflow-y:auto는 가로도 잘라요**: CSS 규칙상 overflow-x가 auto로 계산되어, 요소 밖으로 삐져나가는 장식(손그림 테두리 등)이 잘려요. 여백을 컨테이너 안팎으로 쪼개서 해결.
- **모달에서 드래그하다 밖에서 손을 떼면 닫혀요**: backdrop 클릭 판정은 pointerdown 시작점 기준으로 가드.

3-5. 원본과 래퍼를 분리하라

같은 앱을 웹·토스·iOS·안드로이드로 내다 보면 플랫폼별 코드가 섞이기 쉬워요. 원칙:

- **원본 게임 폴더는 건드리지 마세요.** 플랫폼 래퍼(iOS용, 안드로이드용)는 별도 폴더를 만들고, `sync-web.mjs` 같은 동기화 스크립트가 원본 빌드 결과를 복사해 오면서 플랫폼별 손질(웹 광고 제거, 네이티브 브리지 주입)을 해요.
- 원본이 부르는 인터페이스(예: 광고 함수 `window.AitAds`)를 래퍼가 **같은 이름으로 재구현(shim)** 하면 원본 무수정으로 iOS 광고가 나가요. 이 shim 패턴이 멀티 플랫폼 유지보수의 핵심이에요.
- 함정 하나: 웹용 광고 코드의 허용 호스트 목록에 `localhost`가 들어 있으면 **Capacitor 앱(내부적으로 localhost)에서 웹 광고가 떠 버려요.** 래퍼 동기화 때 반드시 무력화하세요.

3-6. 디자인 시안(claude.ai/design) 적용 패턴

- 시안 zip을 `_redesign/` 폴더에 두고 참조. **원본 무수정 오버라이드 CSS**(예: `classical.css`)를 뒤에 로드해 덮어요 — 한 줄 빼면 원상복구되므로 실험 비용이 0이에요.
- 시안에서 디자인 토큰(주조색·포인트색·폰트·radius·그림자)을 실측해 CSS 변수로 정리한 뒤 덮는 것이 효율적이에요. 눈대중 근사치는 금지예요.
- 배경 그라데이션처럼 원본에 강하게 박힌 스타일은 오버라이드 우선순위(선택자 구체성)로 눌러야 할 수 있어요.
- 폰트가 CDN 의존이면 웹뷰(토스 등)에서 안 뜰 수 있어요 — 서브셋 자체 호스팅으로 하드닝(위 3-4 폰트 절 참조).
- 적용 검증은 주요 화면 3개를 폰 뷰포트로 스크린샷 찍어 시안과 나란히 비교(compare 스크립트 패턴). 클로드 코드에 DesignSync 연동이 있으면 디자인 프로젝트와 직접 동기화도 가능(에셋 등록을 해야 카드가 보이는 점 주의).

족보 ④ 웹 배포 상세 — Cloudflare Pages가 정답인 이유

앱을 스토어에 올리기 전에 웹부터 배포해요. 이유: ①즉시 테스트 링크가 생기고 ②스토어 심사용 아이콘·개인정보처리방침 URL이 필요하며 ③웹 자체가 광고 수익 채널이 돼요.

4-1. 호스팅 선택: Vercel의 함정

Vercel 무료(Hobby) 플랜은 광고 게재를 명시적으로 금지해요. 공식 문서에 "Google AdSense를 포함한 광고 플랫폼"이 상업적 이용 예시로 적혀 있고, 위반 시 통보 없이 프로젝트가 중단될 수 있어요. 애드센스·쿠팡 배너를 달 사이트라면 처음부터 **Cloudflare Pages**로 가세요.

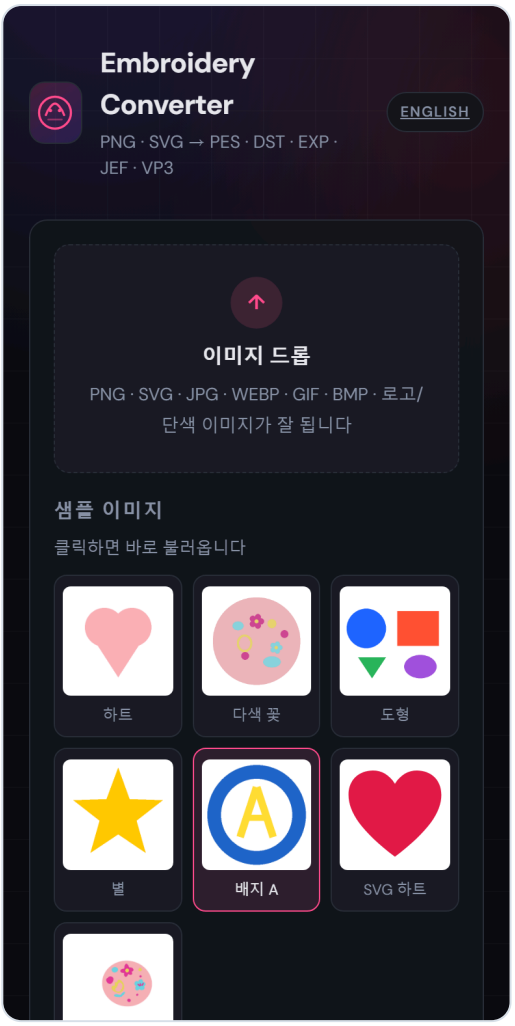
- Cloudflare Pages 무료: 대역폭 무제한, 상업적 이용 허용, 파일 2만 개·개당 25MiB·빌드 월 500회. 취미~중소 규모엔 사실상 무제한.
- 서버 기능이 필요하면 Pages Functions(`functions/api/*.js`)로 충분해요. RSS 프록시, 통계 수집, 라이선스 검증까지 전부 이걸로 처리했어요. DB는 D1(SQLite), 키-값은 KV.

계정이 없어도 `wrangler login` 흐름에서 브라우저 OAuth로 신규 계정이 자동 생성돼요(사용자는 허용 클릭만). ⚠ **신규 가입 직후 이메일 인증 전에는 API가 전부 거부돼요**(에러 8000077) — 인증 메일 클릭이 선행 조건이에요. 미인증 여부는 `api.cloudflare.com/client/v4/user`에서 `modified_on == created_on`으로 확인할 수 있어요.

배포는 한 줄이에요.

```
npx wrangler pages deploy dist --project-name 프로젝트명 --branch main
```

`--branch 이름`을 다르게 주면 `이름.프로젝트.pages.dev` 미리보기 URL이 생겨요. 배경색 시안 3종을 각각 브랜치로 올려 폰에서 직접 비교하는 식으로 쓰면 디자인 결정이 빨라져요.



Cloudflare Pages 무료 플랜으로 운영 중인 도구 앱 예 — PNG를 자수 도안으로 바꿔 주는 변환기. 이런 앱도 배포 명령 한 줄이에요

4-2. 정적 호스팅의 자잘한 함정들

- **트레일링 슬래시 308**: Cloudflare Pages는 `blog/index.html`을 `/blog → /blog/`로 리다이렉트해요 (Vercel은 정반대). 목록 페이지는 폴더가 아니라 **루트의** `blog.html`로 두면 양쪽 다 리다이렉트 없이 200이 나와요.
- 이 308 때문에 **구글·네이버의 "HTML 파일 업로드" 소유 확인이 실패해요.** 서치콘솔은 DNS TXT(도메인 속성) 방식, 네이버 서치어드바이저는 HTML "태그" 방식을 써야 해요.
- **엣지 캐시**: 배포 직후엔 옛 내용이 보여요. 검증은 `?cb=랜덤` 쿼리를 붙이거나 배포 고유 URL(해시.프로젝트.pages.dev)로 먼저 하고, 10~20초 뒤 재확인하세요.
- **앱인토스 빌드와 폴더 충돌**: `ait build`는 출력 폴더(dist) 루트를 RN 번들로 덮어쓰고 정적 파일을 `dist/web/`으로 옮겨요. 웹 배포 폴더를 아예 분리(`web/`)하거나 Cloudflare엔 `dist/web`을 올려야 해요. `ait build` 직후에 `wrangler pages deploy dist`를 하면 사이트 루트가 404가 돼요.

4-3. 도메인과 검색 등록

- 커스텀 도메인은 애드센스 신청의 사실상 전제조건이에요(*.pages.dev, *.vercel.app 서브도메인으로는 신청 불가). Cloudflare Registrar에서 사면 DNS까지 한곳에서 끝나요. 루트 도메인이 승인되면 서브도메인 전체가 커버되므로, 이후 새 앱은 서브도메인만 붙이면 돼요.

- 오타 도메인을 실수로 결제하면 **환불이 안 돼요**(실제 사고). 결제 전에 철자를 확인하세요.
- 검색 유입은 저절로 안 생겨요: 서치콘솔 등록+사이트맵 제출, 네이버 서치어드바이저(HTML 태그 방식, 마지막 제출 버튼과 캡차는 사람이), RSS 발행(네이버·다음이 받아요), robots.txt에 Yeti·Daum 허용까지가 기본 세트예요.

4-4. 자체 통계는 프라이버시 프리로

구글 애널리틱스 대신 D1 테이블 하나로 (날짜, 경로, 유입 도메인, 국가, 기기) 카운터만 집계하는 자체 비콘을 만들었어요. 쿠키·IP·사용자 ID를 저장하지 않으면 동의 배너도 필요 없어요. 두 가지 교훈:

- **신규 사이트 트래픽은 봇이 압도적이에요.** User-Agent 필터(bot|crawl|spider|headless|curl|python 등)를 넣기 전 첫날 조회수 대부분이 크롤러와 취약점 스캐너였어요. 봇 판별 신호: 존재하지 않는 경로, 한국어 사이트인데 해외 IP 비중, 방문당 페이지 1.0대, 리퍼러 없음.
- **집계 데이터를 임의로 지우지 마세요.** 테스트 흔적을 정리한다고 DELETE를 돌렸다가 실제 방문 기록까지 날린 사고가 있었어요. 지우기 전에 반드시 확인하고, D1은 Time Travel로 30일 내 복구가 가능하다는 것도 알아 두세요.

족보 ⑤ 앱인토스(토스 미니앱) 출시 상세

토스 앱 안에서 실행되는 미니앱이에요. 진입장벽이 세 스토어 중 가장 낮고(심사 영업일 1~2일), 웹 앱을 거의 그대로 올릴 수 있어요.

5-1. 빌드 파이프라인

- 공식 스캐폴드는 Granite+Vite예요. 설정은 `granite.config.ts` 하나예요. **`appName` 은 콘솔에 등록한 이름과 정확히 일치**해야 하고, 등록 후엔 수정 불가예요.
- `npm run build` (내부적으로 `ait build`) → `앱이름.ait` 번들 생성 → `npx ait deploy --api-key 키` 로 업로드하거나 콘솔에서 수동 업로드.
- 빌드 도구 없는 순수 HTML 앱도 올릴 수 있어요. 그 경우 SDK(`@apps-in-toss/web-bridge`)에서 필요한 함수만 esbuild로 IIFE 번들해 전역(`window.AIT_AD` 등)에 노출시키는 브리지 파일 하나만 만들면 돼요.

`ait build` 의 검증된 함정 두 가지:

1. **출력 폴더를 RN 번들로 덮어써요** — 웹 배포 폴더와 분리하세요(족보 ④ 참조).
2. `package.json` 에 `devDependencies` 가 아예 없으면 내부에서 `Object.keys(undefined)` 로 죽어요 — **빈 `"devDependencies": {}` 라도 넣어야 통과**해요.

`granite.config.ts` 의 `webViewProps` 는 꼭 손보세요. `pullToRefreshEnabled:false` , `bounces:false` , `overScrollMode:'never'` , 스와이프백 비활성 — 게임 중 실수로 새로고침되어 진행이 날아가는 사고를 막아요. 카메라를 쓰면 `allowsInlineMediaPlayback:true` 가 필수예요. `brand.icon` 은 URL이므로 웹에 icon.png를 호스팅해야 하며, **리다이렉트되는 도메인을 넣으면 아이콘이 깨져요.**

5-2. 콘솔 등록 — 문서에 없는 실측 규격

로그인은 본인 명의 토스 앱 QR 인증이라 **사람이 직접** 해야 해요(세션은 브라우저 프로필에 남아 재사용 가능). 공식 문서와 콘솔 실물이 다른 부분이 많았어요. 실측 기준:

항목	실측 제한
앱 이름	10자
영문명	15자, 명사형(명령문·동사구 금지)
부제	20자
상세 설명	500자
키워드	5~8개
로고	600×600 PNG, 투명 불가 + 다크모드용 별도 요구
스크린샷	세로 636×1048 최소 3장 (가로는 무시됨, 혼용 불가)
썸네일	1932×828

주의 세 가지예요.

- **「임시저장」 버튼이 실제로는 검토 요청으로 처리된 사례가 있어요.** 초안 저장으로 믿지 말고, 제출 직전 상태라고 생각하고 채우세요.
- 법적 진술 체크박스(인허가 완료, 손해배상 책임 등)는 대신 체크하지 말고 반드시 본인이 읽고 체크하세요.
- 카테고리: 게임으로 등록하면 게임 등급분류(IARC/게임위)가 요구돼요. 교육·콘텐츠성 앱은 **비게임(생활) 카테고리**로 등록하면 등급분류가 불필요해요. 토스 심사가 카테고리를 직권으로 바꿔 주기도 해요(게임 신청 → 생활>콘텐츠로 지정된 사례). 건강 관련 앱은 "건강>심리" 같은 분류가 인허가 문의를 부를 수 있어 "일상>취미"처럼 안전한 분류가 나아요.

5-3. 인앱 광고 2.0 연동

- 콘솔 사이드바 「**수익화**」에서 광고 그룹(배너/전면형/리워드)을 만들어요. 생성 직후 "구글 반영 중"(최대 2시간)이 뜨는데, **광고 그룹 상세 화면 URL의 `groupId` 값(`ait.v2.live.…`)이 곧 코드에 넣는 광고 그룹 ID예요.**
- SDK: `TossAds.initialize/attachBanner` , `loadFullScreenAd/showFullScreenAd` . **토스 앱 밖(일반 브라우저)에서는 `isSupported()` 호출 자체가 예외를 던져요** — 광고 초기화 전체를 try로 감싸지 않으면 웹 버전 부팅이 죽어요.
- 광고 빈도는 횟수만 세면 안 돼요. "3회마다"로 하면 한 판이 짧은 게임에서 광고가 18초 간격으로 떠요. **횟수 조건 + 최소 간격(60~120초)**을 병행하고, 결과 화면을 먼저 보여준 뒤 광고를 띄우세요. 토스 분석 탭이 "광고를 과하게 보여주는지"를 지표로 봐요.
- 광고 안 뜨거나 무응답이어도 진행이 막히지 않게 타임아웃(12초)을 걸어 두세요.
- 수수료: 광고 중개 수수료 15%(당시 면제 프로모션 중). 이건 수익에서 떼는 몫이지 청구가 아니에요. **콘솔에서 돈이 "나가는" 것은 유입·성장 메뉴의 스마트 발송(푸시)과 프로모션(포인트 지급)뿐**이에요 — 인앱 광고는 버는 쪽이에요. 이 구분을 헷갈리면 "광고비 집행한 적 없는데?"라는 오해가 생겨요.

- 광고 분석 탭의 "SDK를 업데이트하세요" 배너는 범용 안내라 무시해도 돼요(광고 그룹 탭에 노출·수익이 잡히면 연동 정상). 분석 데이터는 하루 한 번 오전에만 갱신돼요.

5-4. 스마트 발송(광고성 푸시) 문구 심사

문구 규칙이 은근히 까다로워요: "~요"로 끝나면 마침표 필수, 제목에 반말 질문형 금지, 본문 명령형 금지, 그리고 **"어떤 서비스인지 문구만으로 알 수 있어야" 통과**해요. AI가 채워 주는 기본 문구("토스에서 17가지 방법을 써봐요")는 범용적 표현이라 반려돼요. 신규유입·재방문 두 안이 **각각 판정**되므로 둘 다 구체적으로 고쳐야 전체가 통과해요. 이 등록 화면은 자동화가 끝까지 안 먹혀서 사람이 직접 하는 게 빨라요.

5-5. 출시 후

새 버전은 새 .ait 업로드로 등록해요. **콘솔의 출시 버전이 어느 날짜 빌드인지 꼭 확인**하세요 — "광고가 안 나온다"의 진짜 원인이 코드가 아니라, 광고 연동 이전의 옛 번들이 아직 출시 버전인 것이었던 사례가 있어요.

족보 ⑥ 구글 플레이 출시 상세

6-1. 패키징: Capacitor냐 TWA냐

웹 앱을 안드로이드 앱으로 만드는 길은 두 가지예요.

- **Capacitor 래퍼(권장)**: 웹 소스를 `www/`에 복사해 네이티브 앱으로 감싸요. iOS 래퍼와 구조가 같아서 한 번 익히면 양쪽에 써요. 플러그인(AdMob, 햅틱, IAP)도 붙어요.
- **TWA(Bubblewrap)**: 배포된 웹사이트를 그대로 앱으로 여는 방식이에요. 웹 배포가 곧 앱 업데이트라는 장점이 있지만, 사이트에 `.well-known/assetlinks.json`을 올려야 하고 광고·네이티브 기능 통합이 제한적이에요.

윈도우 로컬 빌드 환경: Android SDK는 `%LOCALAPPDATA%\Android\Sdk`, JDK는 Temurin(17 또는 21). Bubblewrap이 gradlew 호출에 실패하는 PC 설정(NoDefaultCurrentDirectoryInExePath)이라면 `cmd //c ".\gradlew.bat bundleRelease"`로 직접 돌리면 돼요.

6-2. 키스토어 — 절대 분실 금지

업로드 키스토어는 한 번 만들면 그 앱의 평생 신분증이에요.

```
keytool -genkeypair -v -keystore 앱이름-upload.keystore -alias 별칭 -keyalg RSA -keysize 2048 -
validity 10000
```

- 키스토어 파일과 비밀번호는 **저장소 밖(gitignore) + 별도 백업**. 분실하면 앱 업데이트를 영영 못 올려요(플레이 앱 서명을 쓰면 구제 절차는 있지만 험난해요).
- 산출물은 AAB(`bundleRelease`). 서명 검증은 jarsigner로.
- CI(GitHub Actions)는 ubuntu 러너+JDK 17이면 충분하고, 키스토어는 base64로 시크릿에 넣어요. `versionCode`는 러너 실행 번호를 쓰면 중복 걱정이 없어요.

6-3. 신규 개인 계정의 프로덕션 관문 (2026 상반기 기준 사실 확인됨)

- 2023-11-13 이후 생성된 **개인** 계정은 프로덕션 신청 전 **비공개 테스트: 테스터 최소 12명, 14일 연속 옵트인** 요건이 있어요(2023년 도입 시 20명 → 2024-12에 12명으로 완화, 14일은 불변). 12명은 실기기 설치 기준이며 에뮬레이터·중복 계정은 미집계, 중간 이탈로 12명 미만이 되면 14일 카운트가 깨져요.
- **조직(법인/사업자) 계정은 이 요건 면제예요.**
- 이 때문에 요건이 없는 구계정(2023-11 이전 생성) 매매·대리 등록 시장이 존재하나, 계정 매매는 정책 위반으로 계정·연관 계정 정지 사유예요. 정공법: 지인 12명 / 테스터 상호 품앗이 커뮤니티 / 조직 계정 전환.

6-4. 스토어 등록 준비물

- 스토어 등록정보: 간단한 설명(80자), 자세한 설명, 아이콘 512, 그래픽 이미지 1024×500, 스크린샷 최소 2장. **폰 스크린샷은 9:16 비율 강제**예요 — 다른 비율 이미지는 배경 패딩을 붙여 1080×1920으로 변환하는 스크립트를 만들어 두면 편해요.
- **개인정보처리방침 URL은 필수예요.** 카메라·마이크를 쓰는 앱은 특히요. 웹에 privacy 페이지 한 장 올리면 돼요.
- 앱 콘텐츠 선언 9종(콘텐츠 등급, 데이터 보안, 타겟층, 광고 여부 등)은 법적 진술이므로 내용은 본인이 판단하고, 입력 자체는 자동화로 채워도 돼요.

6-5. Play Console 자동화 — 되는 법과 안 되는 것

콘솔 입력이 많아서 자동화 가치가 커요. 2026년 크롬 기준으로 **막히는 길**부터 볼게요:

- 평소 쓰는 기본 프로필에 원격 디버깅 포트를 붙이는 것 → 크롬 136+가 거부.
- 프로필을 복제해서 자동화 실행 → 쿠키가 앱 결합 암호화(크롬 127+)라 복제본에서는 로그아웃됨.
- Playwright가 띄운 브라우저에서 구글 로그인 → 자동화 감지로 로그인 자체가 거부됨.

되는 방법은 하나예요:

1. **빈 폴더를 user-data-dir로 지정해 일반 크롬을 직접 실행**해요: `chrome.exe --remote-debugging-port=9222 --user-data-dir="빈폴더" URL`
2. 그 창에서 **사람이 직접 로그인**해요(일반 크롬이라 구글이 안 막고, 로그인은 그 폴더에 저장되어 다음에도 유지).
3. 스크립트는 `chromium.connectOverCDP('http://127.0.0.1:9222')`로 붙어서 조작해요. 이 방식은 자동화 플래그를 심지 않아 덜 감지되고, 브라우저가 스크립트와 독립적이라 붙었다 뗐다 하며 장시간 작업할 수 있어요.

콘솔 UI 조작 팁(전부 실전 검증):

- 닫힌 shadow DOM 입력칸은 locator가 못 뚫어요 → 래퍼의 boundingBox 중앙을 마우스 클릭 후 `keyboard.insertText`.
- 드롭다운(카테고리 등)은 가상 스크롤이라 JS 스크롤이 안 먹어요 → 열고 ArrowDown을 누르며 `aria-activedescendant`를 읽는 루프로 항목을 찾아 Enter.
- 한국어 라벨 주의: 게임 카테고리 "Trivia"는 콘솔에서 **"퀴즈"**예요.
- 파일 업로드는 업로드 버튼 클릭 시 `waitForEvent('filechooser')`로 잡아 `setFiles`.

자동화가 절대 안 되는 것: 콘텐츠 등급(IARC) 설문의 마지막 「다음」 버튼이에요. 14문항을 전부 정상적으로 채워도 자동화로는 버튼이 활성화되지 않아요(모든 문항 열람을 감지하는 게이트로 추정). **이 버튼 하나만 사람이 그 창에서 직접 클릭**하면 돼요. 단, 수정 화면으로 재진입하면 답이 초기화되니, 답이 채워진 바로 그 창에서 클릭하세요.

6-6. 참고: 토스 게임 등급분류와의 관계

토스에 게임 카테고리로 내려면 등급분류 정보가 필요한데, 미출시 신규 게임은 "구글 플레이에 먼저 출시 → IARC 등급 획득 → 그 정보로 토스에 입력"하는 순서가 돼요. 즉 구글 등록이 토스 게임 등록의 선행 조건이 되기도 해요. (비게임 카테고리로 가면 이 문제가 통째로 사라져요 — 족보 ⑤ 참조.)

족보 ⑦ 맥 없이 iOS 빌드 — GitHub Actions 클라우드 빌드

아이폰 앱을 만들려면 맥이 필요하다는 통념은 이제 틀렸어요. **GitHub Actions의 macOS 러너**(무료 계정도 월 상당량 무료)에서 빌드·서명·TestFlight 업로드까지 전부 돌릴 수 있어요. 이 장의 파이프라인으로 앱 4종이 실제 심사 제출까지 갔어요.

7-1. 사전 준비물

1. **애플 개발자 계정**(연 $99·개인 또는 법인). 법인은 D-U-N-S 등 절차가 길어요. 개인 팀이 시작엔 간단해요.
2. **App Store Connect API 키**: ASC → 사용자 및 액세스 → 통합(API). 처음이면 「액세스 요청」부터(즉시 승인돼요). ⚠ **클라우드 서명에는 관리자(Admin) 권한 키가 필요해요.** "앱 관리" 권한 키로는 `Cloud signing permission error`가 나요. 키 ID, Issuer ID, .p8 파일 세 가지를 확보하세요. .p8는 재다운로드가 안 되니 안전하게 보관하세요.
3. GitHub 비공개 저장소 + 시크릿 4개: `TEAM_ID`, `ASC_KEY_ID`, `ASC_ISSUER_ID`, `ASC_KEY_P8_BASE64`.

시크릿 입력의 개행 함정: PowerShell에서 파이프로 `gh secret set`에 값을 넣으면 `\r`이 섞여 들어가 빌드가 알 수 없는 이유로 깨져요. `gh secret set 이름 --body "값"` 방식으로 넣고, 워크플로 쪽에서도 `tr -d '\r\n '`로 방어하는 이중 안전장치를 권해요.

7-2. Capacitor 프로젝트 구조

- Capacitor 7 이상은 CocoaPods가 아니라 **SPM**(Swift Package Manager)을 써요. CI에서 `pod install`이 필요 없고, xcodebuild에는 `-project ios/App/App.xcodeproj`를 넘겨요(xcworkspace가 없어요).
- Info.plist에 미리 넣어 두세요: 세로 고정, `UIRequiresFullScreen`(iPad 멀티태스킹 요구 회피), `ITSAppUsesNonExemptEncryption=false`(수출 규정 — 이걸 plist에 넣으면 빌드마다 API로 선언할 필요가 없어요), 마이크·추적(ATT) 등 권한 문구, AdMob 앱 ID.
- 웹 소스는 원본 프로젝트에서 `sync-web.mjs`로 복사(족보 ③ 분리 원칙). 원본 빌드가 만든 `.gitignore`가 www에 섞여 들어와 **에셋 전체가 커밋에서 빠지는 사고**가 있었어요 — 동기화 때 지우도록 스크립트에 포함하세요.

7-3. 워크플로 핵심 단계

`workflow_dispatch`(수동 실행) 트리거로:

1. `npx cap sync ios`
2. `xcodebuild archive` — `-allowProvisioningUpdates` + ASC API 키 인증 옵션(클라우드 자동 서명)
3. `xcodebuild -exportArchive` — export 옵션 `destination: upload`로 **아카이브에서 곧장 TestFlight 업로드**까지.

실전에서 걸렸던 것들:

- **애플이 최신 iOS SDK를 강제**해요(제출 시점 기준). 러너를 최신(`runs-on: macos-26` 등)으로 올리고 xcode-select로 최신 Xcode를 지정해야 업로드가 통과돼요.
- 워크플로 yml을 푸시해도 Actions 탭에 안 나타날 때가 있어요 — 파일을 한 번 수정하는 커밋을 넣으면 등록돼요.
- **기기가 0대인 계정(새 개인 팀)은 아카이브가 "no devices"로 실패**해요. 자동 서명이 개발 프로파일을 요구하기 때문이에요. 해결: API로 더미 기기를 하나 등록하면 돼요(`POST /v1/devices`, UDID는 `00008120-…` 형식이면 통과). 서명 관련 다른 우회(CODE_SIGNING_ALLOWED=NO 등)는 전부 실패했어요.
- **첫 빌드 업로드 전에 ASC에 앱 레코드부터 만들어야 해요.** 없으면 "Error Downloading App Information"으로 실패해요. 클라우드 서명이 번들 ID를 자동 등록해 주지도 않으니, 번들 ID 등록 → ASC 앱 생성 → 빌드 순서를 지키세요.

7-4. 빌드 확인 폴링의 함정

업로드된 빌드를 API로 확인할 때 `/v1/apps/{id}/builds`를 쓰면 안 돼요 — **이 엔드포인트는 정렬을 지원하지 않고 최신순도 아니라서** limit=1이 옛 빌드를 돌려줘요(이것 때문에 30분을 헛돈 적이 있어요). 반드시 이쪽을 쓰세요:

```
GET /v1/builds?filter[app]={appId}&sort=-uploadedDate
```

업로드 직후엔 builds 목록에 늦게 등록되므로, 몇 분 폴링할 각오를 하고, 잡힌 빌드가 정말 방금 올린 버전 번호인지 확인한 뒤 제출에 연결하세요.

족보 ⑧ 앱스토어 커넥트 자동화 — 메타데이터부터 심사 제출까지

앱스토어 커넥트(ASC)의 거의 모든 작업은 공식 REST API로 돼요. 인증은 API 키로 ES256 JWT를 서명해 Authorization 헤더에 넣는 방식이에요(서명은 ieee-p1363 포맷 — 라이브러리 기본값과 다를 수 있으니 주의하세요). 스크립트 하나(`asc.mjs` 류)를 만들어 두면 앱을 낼 때마다 재사용해요.

8-1. API로 되는 것 전부

- 번들 ID 등록, 앱 생성(이름·SKU·기본 언어)
- 버전 메타데이터: 설명, 키워드, 부제, 카테고리, 저작권, 심사 연락처, 가격(무료), contentRights
- 연령 등급 설문(2026년 신설 항목 — ageAssurance, advertising, lootBox 등 — 포함)
- 스크린샷 업로드: 슬롯 예약 → 청크 PUT → md5 커밋의 3단계
- 수출 규정(usesNonExemptEncryption) — 단, Info.plist에 넣는 편이 나아요(족보 ⑦)
- TestFlight 내부 그룹 생성과 테스터 추가
- 판매 지역(appAvailabilities), 심사 제출(reviewSubmissions)

API로 안 되는 것: App Privacy(개인정보 보호) 라벨. 이것만은 웹 콘솔을 브라우저 자동화(족보 ⑫)로 조작해 게시해요. 앱 생성 모달처럼 네이티브 select가 섞인 화면은 value setter+change 이벤트 디스패치로 뚫어요.

8-2. 스크린샷 — 만들기도 자동, 올리기도 자동

앱이 웹 기반이면 스크린샷도 Playwright로 찍어요. 로컬 서버로 www를 서빙하고 원하는 장면을 연출해 캡처하면 끝이에요. 필수 규격:

- iPhone 6.7인치: 1290×2796 (430×932@3x) — 최소 필수
- **iPad 12.9인치: 2048×2732 (1024×1366@2x) — 앱이 iPad를 지원하면 이것도 필수예요.** 아이폰만 준비했다가 제출 단계에서 걸려요.

8-3. ⚠ 최대 함정: 판매 지역 미설정

API로 만든 앱은 판매 지역(appAvailabilities)이 미설정 상태예요. 가격만 넣으면 심사는 통과하는데, **출시 후 어느 나라에서도 다운로드가 안 돼요**("해당 국가에서 사용할 수 없음"). 실제로 앱 하나가 이렇게 출시됐다가 발견했어요. 해결:

```
POST /v2/appAvailabilities
  availableInNewTerritories: true
  + territoryAvailabilities 175개를 included로 (임시 id ${t1}... 방식)
```

새 앱은 메타데이터 넣는 단계에서 반드시 판매 지역까지 함께 설정하는 것을 체크리스트에 박아 두세요. 반영은 전파에 수십 분 걸릴 수 있어요.

8-4. 심사 제출 흐름과 타이밍 함정

1. 빌드 선택(족보 ⑦의 7-4 정렬 함정 주의 — 방금 올린 빌드가 맞는지 버전 번호로 확인)
2. reviewSubmission 생성 → 항목 추가 → 제출. 승인 시 자동 출시(AFTER_APPROVAL)로 설정해 두면 심사 통과가 곧 출시예요.
3. 제출을 회수(canceled)한 직후 항목을 다시 추가하면 `ITEM_PART_OF_ANOTHER_SUBMISSION` 409가 나요 — **10초 이상 기다렸다 재시도**하면 돼요. 회수된 제출은 state가 COMPLETE로 표시되니 놀라지 마세요.
4. ASC API는 간헐적으로 401을 던져요(키가 멀쩡해도). **401도 재시도 대상**에 넣으세요.

8-5. 셸 환경 함정

Git Bash(MSYS)에서 `node asc.mjs GET "/v1/apps"`처럼 경로형 인자를 넘기면 MSYS가 경로 변환을 해서 호스트가 깨져요(물음표가 있으면 살아남는 랜덤성까지 있어요). **`MSYS_NO_PATHCONV=1`을 붙이거나 PowerShell에서 실행**하는 것으로 원천 차단하세요.

8-6. 심사 준비 세트

- 심사 연락처(이름·전화·이메일)는 한 번 넣으면 다음 앱에서 API로 조회해 재사용할 수 있어요.
- 심사 메모에 앱 특성을 한/영으로 명시하세요(족보 ⑨에서 상세).
- App Privacy 라벨의 전형적 구성(광고 앱 기준): 기기 ID·광고 데이터 = 타사 광고+추적O / 제품 상호작용 = 분석 / 충돌·성능 = 앱 기능, 전부 "신원과 연결 안 됨".
- 심사 기간은 보통 1~3일이에요. 결과는 계정 메일로 와요.

족보 ⑨ 반려 예방 — 웹 래퍼 앱의 생존 전략

웹 앱을 감싼 iOS 앱의 최대 리스크는 가이드라인 4.2(최소 기능 — "웹사이트를 앱으로 포장한 것")예요. 제출 전에 아래를 전부 챙기면 반려 확률이 크게 떨어져요.

9-1. 확정 반려 사유부터 제거

- **기본 아이콘·스플래시(Capacitor 로고 등)를 그대로 두는 것 = 확실한 반려예요.** 제출 직전 체크리스트 1번이에요. 아이콘 원화 한 장에서 아이콘·스플래시를 생성하는 스크립트(sharp)를 만들어 두면 앱마다 재사용돼요. 아이콘은 여백·자체 라운드 없이 풀블리드로 만드세요(iOS가 알아서 깎아요).
- 스크린샷 규격 미달(iPad 누락 — 족보 ⑧), 개인정보 라벨 미게시, 수출 규정 미선언도 기계적으로 걸려요.

9-2. 4.2(웹 래퍼) 방어

심사자가 "이건 그냥 웹사이트"라고 느끼지 않게 하는 실전 조합:

1. **완전 오프라인 동작** — 전 콘텐츠 내장, 외부 요청 0건(족보 ③의 단일 HTML 철학이 여기서 빛나요).
2. **네이티브 통합 최소 1~2개** — 햅틱 터치 피드백(터치마다 가벼운 진동, 스로틀 포함)이 가성비 최고예요. 메트로놈 박자 햅틱처럼 앱 성격과 맞물리면 더 좋아요.
3. **앱 전용 경험** — 웹에 없는 오프닝 연출(만화 컷 인트로 등)을 래퍼에서 주입.
4. **심사 메모에 한/영으로 명시**: "완전 오프라인 동작, 전 콘텐츠 내장, 웹사이트 래퍼가 아님".

9-3. 넣지 말아야 할 것

- **푸시 알림을 습관적으로 넣지 마세요.** 쓰지도 않는 권한 요청은 오히려 4.5.4 반려 사유가 돼요. "기능이 많아 보이면 유리하겠지"는 역효과예요.
- 앱 미리보기 영상도 필수가 아니에요. 없어서 반려되지 않아요.
- 무의미한 로그인, 위치 권한 등도 마찬가지예요 — 권한은 실제 기능이 쓰는 것만 넣으세요.

9-4. 광고 앱의 추가 체크

- ATT(추적 투명성) 동의창은 광고 SDK 초기화와 순서를 맞추세요.
- 심사 시점에 광고 계정이 승인 전이면 실광고가 안 나오는데, 그 상태(배너 자리 비어 있음)로 제출해도 문제없어요. **테스트 광고 모드(isTesting)로 출시하는 실수만 피하면 돼요** — 출시 빌드 전 isTesting:false 확인을 체크리스트에 넣으세요.
- 광고 때문에 연령 등급 설문의 advertising 항목을 정직하게 체크하세요.

9-5. 반려됐다면

반려는 사형 선고가 아니라 대화예요. 지적된 항목만 고쳐 재제출하면 되고, 재심사는 처음보다 빠른 경우가 많아요. 반려 사유가 모호하면 Resolution Center에서 질문할 수도 있어요. 중요한 것은 **제출 전에 위 목록으로 자가 심사를 마쳐서, 반려 사이클 자체를 줄이는 것**이에요.

족보 ⑩ 구독·인앱결제 개설 — 유료 앱 계약부터 심사까지

광고 제거 구독(연 단위)을 실제로 두 앱에 개설하며 검증한 절차예요. 애플 쪽이 서류·API 모두 가장 복잡하므로 애플 중심으로 설명할게요.

10-1. 선행 조건: 유료 앱 계약

상품을 만들기 전에 ASC의 **유료 앱 계약(Paid Apps Agreement)이 활성**이어야 해요. 법인/개인 정보 입력 → 계약 서명 → 은행 계좌·세금 양식 등록. AI가 브라우저로 화면을 열고 항목을 채우는 것까지는 도울 수 있지만, **계약 서명과 은행·세금 확정 입력은 계정 소유자가 직접** 해야 해요. 계약 전에 상품 API를 불러도 생성이 안 되니, **개발 일정에서 이 서류를 가장 먼저** 처리해 두세요. (계약만 되면 이후 상품 개설~심사 제출은 전부 자동화돼요 — 아래 절차 전체를 실제로 AI가 수행했어요.)

10-2. 구독 상품 개설 API 순서 (검증본)

순서가 중요해요. 하나라도 건너뛰면 뒤 단계가 409로 막혀요.

1. `subscriptionGroups` 생성(그룹)
2. `subscriptions` 생성 — 기간 ONE_YEAR, groupLevel 1
3. 현지화: `subscriptionLocalizations`(상품명·설명) + `subscriptionGroupLocalizations`(그룹 표시명)
4. **`subscriptionAvailabilities` — 전 지역 175개 관계로 POST.** 이걸 안 하면 다음 단계 가격 설정이 409로 거부돼요.
5. 가격: 한국(KOR)의 pricePoint를 찾은 뒤, **equalizations 엔드포인트로 174개 지역의 균등 가격을 받아 지역별로 각각 POST**해요. KOR 하나만 넣으면 콘솔 UI에 "가격 추가 필요"가 떠서 제출이 막혀요.
6. 무료 체험: `subscriptionIntroductoryOffers`는 **territory 관계가 필수 = 지역별로 175회 POST**.
7. 심사용 스크린샷 업로드(예약→청크 PUT→md5 — 앱 스크린샷과 같은 3단계).

10-3. 첫 구독은 앱 버전과 함께 심사받는다

여기가 가장 헤매는 지점이에요. **첫 구독 상품은 반드시 앱 버전과 같은 제출에 묶여야 하는데, reviewSubmissionItems API가 subscription 타입을 받지 않아요(UNKNOWN 에러).** 방법은 콘솔 UI뿐이에요:

- 구독 상품 페이지의 **「심사에 추가 v」 드롭다운 → "iOS 제출 초안" 선택**. 제출이 이미 심사 대기(WAITING) 상태면 회수한 뒤 초안 상태에서 추가하세요.
- 그래도 `SUBSCRIPTION_SUBMISSION_REQUIRES_GROUP_VERSION` 에러가 나면 **구독 그룹도 심사에 추가**해야 해요(그룹 페이지에서 같은 드롭다운). 최종적으로 제출 항목이 3개(앱 버전+그룹+구독)가 되면 성공이에요.

10-4. 클라이언트 구현 — 계약이 늦어져도 심사에 안전한 패턴

서류(10-1)가 늦어져 상품 없이 앱만 먼저 심사받는 상황이 실제로 생겨요. 이때 결제 버튼이 화면에 있는데 상품 조회가 실패하면 심사에서 걸려요. 해결 패턴:

- **스토어에서 상품이 조회될 때만 구매 버튼을 노출**해요(getProducts 성공 시 등장). 상품이 없으면 버튼 자체가 없으니 심사가 안전하고, 나중에 상품이 생기면 앱 수정 없이 버튼이 나타나요.
- 구독 상태는 앱 시작마다 현재 자격(entitlements)을 동기화하고 만료 시각을 로컬에 캐시하세요.
- 심사 필수 UI: 가격·기간·자동갱신 조건 고지 문구, **복원(Restore) 버튼**, 구독 관리 화면으로 가는 링크(manageSubscriptions).
- 광고 제거 플래그는 광고 스크립트보다 먼저 로드해서, 구독자에게 광고가 번쩍했다 사라지는 일이 없게 하세요.

플러그인은 가벼운 네이티브 IAP 래퍼(예: @capgo/native-purchases)로 충분했어요. 유료 서드파티 구독 관리 서비스는 앱 하나 규모에선 과해요.

10-5. 비소모성 vs 구독

"광고 제거 ₩2,900 한 번 결제"(비소모성)로 시작했다가 연 구독으로 바꾼 경험상: 비소모성은 복원 로직이 단순하지만 매출이 1회로 끝나고, 구독은 서류(그룹·지역·갱신 고지)가 많은 대신 갱신 매출이 남아요. 어느 쪽이든 **복원 기능은 필수**예요(비소모성 복원은 재구매 시도 시 무과금 처리되는 구조를 활용해도 돼요).

족보 ⑪ 광고 수익화 — AdMob·AdSense·애드핏·쿠팡

11-1. 전체 지도: 어떤 광고를 어디에

채널	지면	비고
AdMob	앱(iOS·안드로이드)	애플·구글 스토어 앱용
토스 인앱 광고	앱인토스 미니앱	족보 ⑤ 참조, 미디에이션이 AdMob
AdSense	웹사이트	심사 문턱 높음, 커스텀 도메인 필수
카카오 애드핏	웹사이트(국내)	트래픽도 심사함
쿠팡 파트너스	웹 어디든	광고가 아니라 제휴 링크, 심사 없음

계정 관계를 먼저 정리해야 해요. **AdMob(앱)과 AdSense(웹)는 다른 콘솔이지만 결제 프로필로 연결**되며, AdMob 계정을 만들면 연결된 호스팅 AdSense 계정이 생겨요. **같은 명의로 AdSense 2계정은 중복 거절 사유**이므로, 웹 수익화도 AdMob 연결 계정 하나로 통일하는 것이 안전해요.

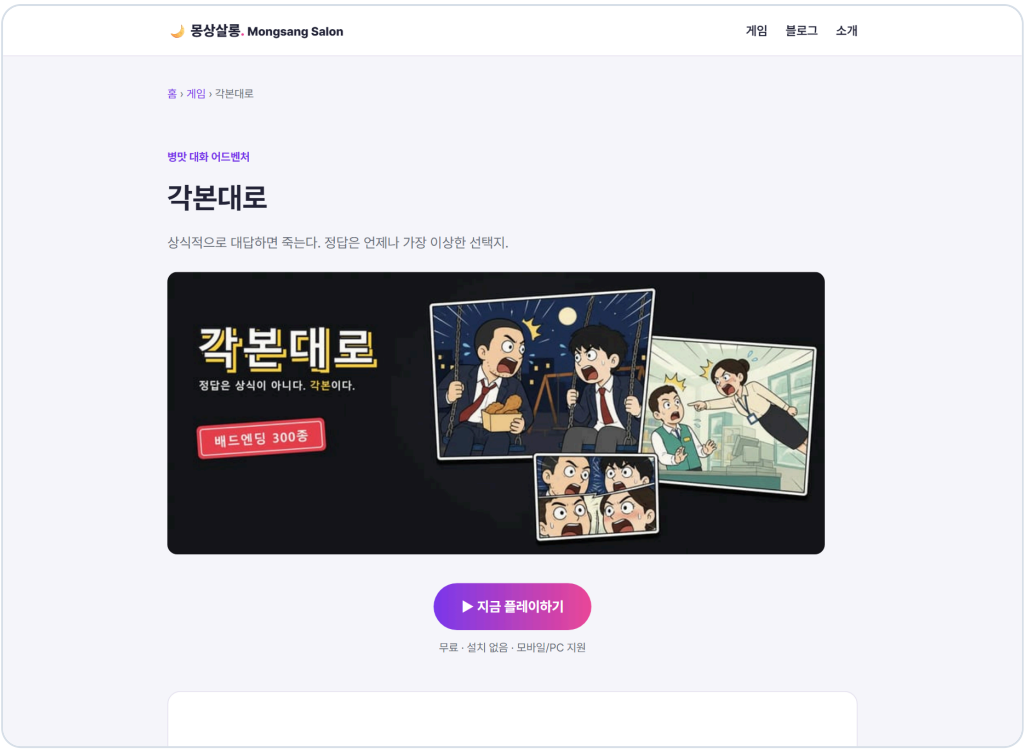
11-2. AdMob (앱 광고)

- 앱 등록 → 광고 단위 발급(배너·전면·리워드) → 앱 ID는 Info.plist(GADApplicationIdentifier), 단위 ID는 코드에.
- 신규 계정은 승인 대기(약 24시간)가 있고, **실광고는 계정 승인+앱 스토어 출시+스토어 연결 후**에야 나와요. 그 전까지 배너가 비어 보이는 것이 정상이에요.
- **app-ads.txt**: AdMob은 스토어 등록 페이지의 개발자 웹사이트(마케팅 URL) 도메인 루트에서 `app-ads.txt`를 크롤링해요. 없으면 404 페이지를 읽고 "정보 불일치" 오류를 내요. 내용은 한 줄이에요: `google.com, pub-계정번호, DIRECT, f08c47fec0942fa0`. 재크롤은 최대 24시간, 콘솔의 「업데이트 확인」으로 즉시 요청 가능해요.
- 전면 광고 빈도는 시간 게이트로: "화면 노출 누적 N초당 1회" 또는 "이벤트 M회+최소 간격" (족보 ⑤의 광고 빈도 원칙과 동일). 첫 지급은 잔액 $100 도달부터예요.

11-3. AdSense (웹) — 심사를 통과하는 구조

게임·앱 소개만 있는 사이트는 "가치 없는 콘텐츠"로 거절되기 쉬워요. 통과 전략:

- **포털 구조**: 게임들을 한 도메인에 모으고, 게임마다 장문 소개 페이지(조작법·공략·FAQ)+블로그(공략·상식 글)를 붙여 **읽을 텍스트가 있는 사이트**로 만들어요. 수만 자 규모면 승부가 돼요.
- 심사는 사이트(도메인) 단위예요. 서브도메인 쪼개기·티스토리 경유는 우회가 안 돼요.
- 게임은 소개 페이지에 iframe으로 임베드하면 ①체류 시간 ②텍스트 콘텐츠 ③광고 지면(iframe 밖)을 동시에 얻어요. iframe 안팎에 광고가 겹치지 않게 임베드 상태에선 게임 내부 광고를 꺼요(`window.self !== window.top` 게이트).



실제 운영 중인 게임 상세 페이지 — 대표 이미지·바로 플레이 버튼 아래로 장문 소개와 광고 슬롯이 이어지는 구조예요

11-4. 카카오 애드핏 — 진단 요령이 따로 있다

- 매체 심사가 콘텐츠 양+**트래픽 양**까지 봐요. 신규 도메인·글 7편으로 신청했다가 "콘텐츠 부족" 반려, 10편·4만 자로 보강해 통과했어요.
- **애드핏은 헤드리스 브라우저·데이터센터 IP에 광고를 송출하지 않아요.** Playwright로 보면 슬롯이 display:none인 게 정상이에요. "코드가 잘못됐나?"로 며칠 헤매지 마세요 — **송출 여부 판단은 오직 콘솔 보고서(Fill Rate·노출수)로만** 하세요. 실제로 "광고 요청 0건"이라고 오진했는데 콘솔에는 Fill Rate 95%가 찍혀 있었어요.
- 광고 단위는 매체(도메인)별 발급이에요. 승인된 도메인의 단위를 다른 도메인에 넣으면 정책 위반이에요. 같은 단위를 한 페이지에 두 번 넣으면 렌더링되지 않아요.
- **오클릭 유도 배치는 즉시 정지 사유예요.** 실행 버튼 바로 옆·아래에 광고를 두지 마세요(권장: 본문 흐름 중간, 버튼과 수천 px 이격). "안 누를 수 없게 해 달라"는 요구는 계정을 잃는 지름길이라 거절해야 해요. **본인이 자기 광고를 클릭하는 것도 적립금 회수·계정 정지 사유예요.**

11-5. 쿠팡 파트너스 — 심사 없는 시작점

- 광고 계정 심사를 기다리는 동안에도 수익 라인을 열 수 있어요. 위젯(캐러셀)을 발급받아 붙이면 끝이에요.
- 구현 주의: 쿠팡 스크립트는 document.write 기반이라 페이지에 직접 넣으면 깨져요 — **srcdoc iframe으로 격리**해서 삽입하세요.
- "이 포스팅은 쿠팡 파트너스 활동의 일환으로..." **의무 고지 문구를 배너마다 자동 삽입**하는 코드로 만들어 두면 잊을 일이 없어요.

- 호스트 게이팅: 웹 도메인에서만 노출하고 앱 웹뷰(토스·Capacitor)에서는 숨겨요. 스토어 앱 안에 제휴 링크가 보이면 스토어 정책 문제가 돼요.

11-6. 공통 원칙

1. 광고 설정은 코드 여기저기가 아니라 **설정 파일 한 곳**(ads-config)에 모으세요. 슬롯 켜고 끄기, 단위 교체가 한 줄이 돼요.
2. 심사 중인 채널이 있으면 광고 밀도를 최소로 유지하세요(페이지당 1개 등).
3. 플랫폼별 게이팅을 코드로 강제하세요: 웹 광고는 웹 호스트에서만, 앱 광고는 앱에서만, 임베드에선 끄기.
4. 수익 = 노출 × eCPM ÷ 1000. 노출 수십 회 단계의 CTR·수익 숫자는 통계적 잡음이니 의미를 부여하지 마세요. **병목은 거의 항상 코드가 아니라 트래픽이에요.**

족보 ⑫ 브라우저 자동화로 관리 콘솔 정복하기

앱 하나를 내려면 콘솔을 대여섯 개 만져요: AdMob, 플레이 콘솔, 앱스토어 커넥트, 애플 개발자 사이트, 토스 콘솔, 서치콘솔.... 폼 입력 수백 칸을 AI에게 시키는 기술이 이 장이에요. 족보 ⑥에서 소개한 패턴의 완전판이에요.

12-1. 표준 패턴: 사람이 로그인, 스크립트가 조작

```
1) chrome.exe --remote-debugging-port=9222 --user-data-dir="전용폴더" URL
   (반드시 "빈/전용" 폴더 – 기본 프로필은 크롬이 디버그 포트를 거부)
2) 사람이 그 창에서 로그인 (일반 크롬이라 사이트가 안 막음, 세션은 폴더에 저장)
3) 스크립트: chromium.connectOverCDP('http://127.0.0.1:9222') 로 접속해 조작
```

이 패턴의 장점: 자동화 플래그가 안 심어져 감지가 덜 되고, 브라우저가 스크립트와 독립이라 **여러 번 붙었다 떼며 몇 시간짜리 작업**을 할 수 있으며, 로그인 세션이 프로필 폴더에 남아 다음날 재사용돼요.

장시간 세션은 명령 파일 방식이 편해요: 컨트롤러 스크립트가 `cmd.jsonl`에 추가되는 JSON 명령을 읽어 실행하고 결과를 `out.jsonl`로 내보내는 구조예요. AI가 명령을 한 줄씩 밀어 넣으며 화면을 확인해요.

12-2. 철칙 목록

- **컨트롤러는 반드시 1개만.** 같은 프로필로 크롬을 여러 개 띄우면 서로 죽여서 "browser has been closed"가 랜덤으로 터져요.
- **자동화 프로필 크롬에서 구글 로그인을 시도하지 마세요.** "이 브라우저는 안전하지 않습니다"로 차단될 수 있어요. 구글 로그인이 필요하면 일반 크롬 창(위 1번 창)에서 사람이 해요.
- **애플 세션은 수 시간마다 만료**돼요. 만료되면 사람이 재로그인해야 하니, 긴 작업은 세션 상태 확인부터 하세요.
- 법적 진술 체크박스(스토어 등록의 인허가·책임 조항)는 **대신 체크하지 않아요.** 폼은 채우되 최종 제출은 사람이 검토 후 누르는 것이 원칙이에요.

12-3. UI가 말을 안 들을 때의 처방전

- **네이티브 select**(ASC 신규 앱 모달 등): 클릭으로 안 열려요 → JS로 value setter 호출 + change 이벤트 디스패치.
- **Angular Material 버튼**(AdMob): 텍스트로 클릭이 안 먹으면 구체적 셀렉터(`material-button.btn-yes`)로.
- **닫힌 shadow DOM**(플레이 콘솔): 좌표 클릭 + keyboard.insertText (족보 ⑥ 참조).
- **가상 스크롤 드롭다운**: ArrowDown + aria-activedescendant 루프 (족보 ⑥ 참조).
- **화면 좌표 함정**: 스크린샷은 devicePixelRatio(예: 1.25)가 곱해진 픽셀이라 **스크린샷 픽셀 좌표를 page.mouse에 그대로 쓰면 빗나가요**(÷1.25 필요). getBoundingClientRect 값은 CSS 좌표라 그대로 써요.

- **버튼이 anchor가 아니라 button인 사이드바**: href가 없으니 `el.closest('button').click()`으로.
- **파일 업로드**: filechooser 이벤트를 기다렸다 setFiles (족보 ⑥ 참조). 단, 토스 콘솔의 .ait 업로드처럼 file input이 무반응인 화면도 있어요 — 그런 건 CLI(`ait deploy`)나 사람 손으로 하세요.
- **라디오/라벨 클릭이 무시되는 프레임워크**(Vuetify 등): 클릭 후 **상태를 화면에서 다시 읽어 검증**하고 넘어가요. `check({force:true})`가 먹히기도 해요.

12-4. 자동화 불가 확정 목록 (사람 몫)

시도할 만큼 해 보고 안 되는 것으로 확정된 항목들이에요. 여기에 시간 쓰지 마세요.

1. 각종 최초 로그인: 토스 앱 QR 본인인증, 구글, 애플(만료 시 재로그인 포함)
2. 플레이 콘솔 IARC 설문의 마지막「다음」버튼 (족보 ⑥)
3. 네이버의 캡차 — 화면에 "자동 등록 방지"라고 명시되어 있어요. 우회하지 않아요
4. 유료 앱 계약·은행·세금 서류 (족보 ⑩)
5. 법적 진술 체크박스 최종 확인 (원칙상 사람)

12-5. 검증 관련 잔기술

- 네이버는 브라우저를 죽이면 로그인이 날아가요 — 한 프로세스가 창을 처음부터 끝까지 붙들고 작업하세요.
- PowerShell의 Invoke-RestMethod로 한글 폼을 테스트하면 본문이 ASCII로 가서 `???`로 깨져요. **폼 검증은 실제 브라우저(Playwright)로 하세요.**
- Playwright 기본 UA에는 HeadlessChrome이 들어 있어 자기 사이트의 봇 필터에 걸려요 — 실제 UA를 지정하세요.
- 사이트마다 "이 화면은 스크립트, 이 화면은 사람"을 기록해 두면 다음 앱에서 시행착오가 0이 돼요. 이 책의 부록이 그 기록이에요.

족보 ⑬ 부록 A 새 앱 출시 마스터 체크리스트

한 앱을 아이디어에서 3개 스토어까지 보내는 전체 순서예요. 괄호는 담당(AI=자동화 가능, 人=사람 필수)이에요.

0단계 제작

- ☐ 원본 앱 폴더 + 단일 HTML 빌드 (AI)
- ☐ 스모크·뷰포트·컨택트시트 검증 스크립트 (AI)
- ☐ 아트 지시서 파일 작성 (AI) → 그림 생성: 옆 탭 GPT 비서 위임 (AI, 폴백 시 人 붙여넣기) → 후처리 자동 (AI)
- ☐ 아이콘 600×600 + 스토어 자산 생성 스크립트 (AI)

1단계 웹

- ☐ Cloudflare Pages 배포, icon.png·privacy 페이지 호스팅 (AI)
- ☐ 라이브 체크(부팅·외부요청 0건) (AI)

2단계 앱인토스

- ☐ granite.config.ts (appName·webViewProps·brand.icon) (AI)
- ☐ .ait 빌드 — devDependencies:{} 함정, dist 덮어쓰기 함정 (AI)
- ☐ 콘솔 로그인 (人 QR) → 앱 정보·자산 입력 (AI) → 법적 체크+검토 요청 (人)
- ☐ 광고 그룹 생성 → groupId를 코드에 → 재빌드·재배포 (AI)

3단계 구글 플레이

- ☐ Capacitor 안드로이드 래퍼 + 키스토어 생성·백업 (AI 생성, 人 보관)
- ☐ AAB 빌드(로컬 또는 Actions) (AI)
- ☐ 콘솔: 등록정보·설정·선언 입력 (AI) → IARC 마지막 버튼 (人)
- ☐ 스크린샷 9:16 변환 (AI)

4단계 애플

- ☐ 번들 ID 등록 → ASC 앱 생성 (AI)
- ☐ **판매 지역 175개국 설정 — 앱 생성 직후 바로!** (AI)
- ☐ Capacitor iOS 래퍼: 아이콘/스플래시 교체, Info.plist(세로·FullScreen·수출규정·권한 문구) (AI)
- ☐ 햅틱 등 네이티브 통합 + 오프닝 (AI)
- ☐ GitHub repo+시크릿 4종(Admin 키!) → Actions 빌드 → TestFlight (AI)

- ☐ 메타·스크린샷(iPhone+iPad!)·연령등급·심사 연락처 (AI)
- ☐ App Privacy 라벨 — 브라우저 자동화 (AI)
- ☐ AdMob 실 ID 반영 + isTesting:false 확인 (AI)
- ☐ app-ads.txt 게시 (AI)
- ☐ 심사 제출(자동 출시 설정) (AI) → 결과 메일 확인 (人)

5단계 수익화 마무리

- ☐ (구독 앱) 유료 앱 계약 (人) → 상품 개설 API 시퀀스 → 버전과 함께 심사 추가(UI 드롭다운)
- ☐ AdMob 앱-스토어 연결(출시 후), 애드센스/애드핏은 웹 트래픽 보고 진행

부록 B 함정 사전 — 한 줄 요약 40선

환경·AI

- 구독 ≠ API 크레딧. 코드가 api.*를 부르면 별도 과금이에요
- 새 PATH는 이미 떠 있는 프로세스에 적용 안 돼요 — 세션 앞에 PATH 한 줄
- node_modules·브라우저 프로필은 .ignore로 제외 — AI가 폴더 읽는 속도가 달라져요
- CLI OAuth 재로그인은 사람만 할 수 있어요

이미지

- 지시서의 앵커 값과 실물은 달라요 — 도착 후 실측
- "가장자리 최다 색=배경" 가정은 깨져요 — flood fill 검증
- 트림하면 앵커가 틀어져요 — 원본 크기 메타 기록
- 굵은 폰트가 기본 폰트보다 파일이 클 수 있어요
- button은 폰트를 상속하지 않아요

웹·배포

- Vercel 무료 플랜은 광고 금지예요 — 광고 사이트는 Cloudflare Pages
- 폴더/index.html은 308 리다이렉트예요 — 루트 파일.html로
- HTML 파일 소유확인은 308 때문에 실패해요 — DNS TXT/메타태그로
- 배포 직후엔 엣지 캐시예요 — ?cb= 쿼리로 검증
- ait build는 dist를 RN 번들로 덮어써요 — 웹 폴더 분리
- 신규 사이트 트래픽은 대부분 봇이에요 — UA 필터 먼저

앱인토스

- appName은 콘솔 등록명과 일치, 등록 후 수정 불가예요
- package.json에 devDependencies:{} 없으면 ait build가 죽어요
- 토스 밖에서 광고 SDK isSupported() 호출 자체가 throw — try로 감싸기
- 「임시저장」이 검토 요청일 수 있어요
- 광고 그룹 ID는 콘솔 URL의 groupId 값이에요
- 광고 빈도는 횟수+최소 간격 병행
- brand.icon에 리다이렉트 도메인 금지

구글

- IARC 마지막「다음」버튼은 자동화 불가예요 — 사람이 그 창에서
- 기본 프로필 원격 디버깅은 크롬이 거부해요 — 빈 프로필+connectOverCDP
- 프로필 복제는 로그아웃돼요(쿠키 앱 결합 암호화)
- 게임 카테고리 Trivia = 한국어 콘솔에서 "퀴즈"
- 키스토어 분실 = 업데이트 불가 — 백업 2중으로

애플

- 클라우드 서명은 Admin API 키예요 — 앱 관리 키는 권한 에러예요
- 시크릿을 파이프로 넣으면 ₩r이 혼입돼요 — --body로 넣고 tr -d로 방어
- 기기 0대 팀은 아카이브가 실패해요 — 더미 기기 API 등록
- 빌드 폴링은 /v1/builds?sort=-uploadedDate — apps/{id}/builds는 최신순이 아니에요
- API로 만든 앱은 판매 지역 미설정 — 심사는 통과, 다운로드는 불가예요
- iPad 지원 앱은 iPad 스크린샷 필수예요
- 기본 아이콘 그대로 제출 = 반려
- 안 쓰는 푸시 권한 = 4.5.4 반려 사유
- 첫 구독은 버전과 함께, UI 드롭다운으로만 — 그룹도 심사에 추가
- 구독 가격은 175지역 가용성 → 균등가 174 POST 순서예요
- 제출 회수 직후 409 — 10초 대기 후 재시도
- Git Bash에서 API 경로 인자가 깨져요 — MSYS_NO_PATHCONV=1

광고

- 애드핏은 헤드리스에 광고를 안 줘요 — 판단은 콘솔 보고서로만
- 같은 명의 애드센스 2계정 = 중복 거절
- app-ads.txt 없으면 "정보 불일치" — 마케팅 URL 도메인 루트에 한 줄
- 본인 클릭·오클릭 유도 = 계정 정지의 지름길

부록 C 자주 쓰는 명령 모음

```
# Node PATH (PowerShell 세션 시작 시)
$env:Path = "$env:LOCALAPPDATA\Programs\nodejs;$env:Path"

# Cloudflare Pages 배포
npx wrangler pages deploy dist --project-name 이름 --branch main

# 앱인토스
npm run build            # → 이름.ait
npx ait deploy --api-key 키

# 안드로이드 AAB (래퍼 폴더에서)
cmd //c ".\gradlew.bat bundleRelease"

# GitHub 시크릿 (개행 안전)
gh secret set 이름 --body "값"

# 원격 디버깅 크롬 (자동화용)
chrome.exe --remote-debugging-port=9222 --user-data-dir="C:\work\프로필" URL
```

끝으로. 이 책의 노하우는 전부 "한 번 밟은 함정은 두 번 밟지 않는다"는 기록에서 나왔어요. 여러분도 앱을 낼 때마다 자신만의 함정 사전에 한 줄씩 보태 보세요 — 두 번째 앱부터는 놀랄 만큼 빨라져요.