HOW I BUILT AN IPHONE APP WITH AI

Thirty days. One person.

From a clickable mockup to the App Store.

01	02	03	04	05
C	L	E	A	R

Mikołaj Salecki

Digital consultant

salecki

v1.0

JULY 2026 · 24 PP

Twenty-four pages, *one build.*

§ Foreword — why thirty days 03

I THE PRODUCT

01 A coach for how you land 04
02 The product, on the phone 05

II THE RESEARCH

03 Before the first line 06
04 The landscape — seven rivals 07
05 The gaps, and the position 08
06 What to charge, and why 09
07 The name, the line, the score 10
08 Where it gets found 11

III THE METHOD

09 Two days before the code 12
10 The ten rules 13

IV THE BUILD

11 Thirty days, three phases 15
12 Day six — delete everything 16

V THE MACHINERY

13 Fifteen services, seven languages 17
14 Self-hosted, on purpose 18
15 Three modes, one hostile room 19

VI THE FAILURES

16 Eight things that broke 20

VII SHIPPING

17 Shipping, and what it leaves 21

VIII THE RECORD

18 The month, in figures 22
19 Closing — what it cost, what it taught 23

FOREWORD

Why *thirty days.*

A deadline is not a schedule. It is a filter that decides what never gets built.

I am not a developer. I run a consultancy, I work with AI every day, and I have the same problem as most people in my position: a head full of ideas and a long record of talking about them instead of finishing them.

So I set a constraint instead of a plan. Thirty days, from the first mockup to a submission on Apple's review queue. Alone. Alongside a full-time job. No team, no budget, no second attempt.

The constraint did the work a roadmap never does. When you have thirty days, you do not debate architecture for a week—you pick one and find out. You do not build the settings screen nobody asked for. And when the backend you spent six days on turns out to be the wrong shape, you delete it in a single commit instead of defending it for a month.

What this document is

This is a build log, not a victory lap. It contains the pivot that saved the project, the two production failures I found after launch, and the numbers behind all of it—including the unflattering ones. Everything here is checked against git history, App Store Connect, or the running stack.

It is written for people who work with AI and want to know where the real line sits: what it genuinely accelerates, where it quietly costs you time, and how much of the work is still yours no matter what model you point at the problem.

The short answer, which the rest of this document earns: the bottleneck was never writing code.

For the ideas that never got past the coffee.

Mikołaj Salecki

DIGITAL CONSULTANT · JULY 2026

A coach for *how you land.*

Most speaking tools grade the transcript. Tymbre reads the delivery—and hands back one physical thing to change.

01 / 04	02 / 04	03 / 04	04 / 04
C	**3**	**Q**	**EU**
CLEAR	**Modes**	**Q&A**	**Private**
Five dimensions of presence. My own framework, not an off-the-shelf metric.	*Audio for voice. Frame for face and eye contact. Stage for the whole body.*	*A simulator that plays a hostile room and asks the questions you fear.*	*No trackers, no advertising identifiers, hosted in the European Union.*

Tymbre is an iOS presentation coach. You record a short run—a talk, a pitch, a difficult conversation—and it reads not only what you said but **how you said it and how you stood**: pace, range, conviction, body language.

The output is deliberately not a dashboard. Most tools in this space hand you a number and leave you to interpret it. Tymbre scores you across five dimensions of presence and then gives you **one physical cue for the next take**: a breath before the opening line, a planted stance through the close. Concrete, repeatable, tied to what you just did.

The name

Tymbre is pronounced **TAMBER**, after *timbre*—the quality that makes one voice distinguishable from another singing the same note. The logo is a set of coral bars: a waveform, five of them, one per dimension.

CLEAR

Cadence, Lift, Expression, Authority, Resonance. Five axes, plotted as a pentagon, scored per take. It is my framework rather than a borrowed one, which means it is also mine to defend. The bet is that presence is coachable and that the coachable parts are physical, not psychological. You cannot instruct someone to be more confident. You can tell them to slow down before the third sentence.

Modes

Three ways to record, graded by how much you are willing to show. **Audio** listens to the voice alone. **Frame** adds the face and eye contact. **Stage** takes the whole body, which is where most of the tells live. The subscription tiers climb the same ladder—Solo, Speaker, Master—because the product's ambition and the customer's are meant to point the same way.

> *"Most tools hand you a number. A number is not a coaching instruction."*

Tymbre

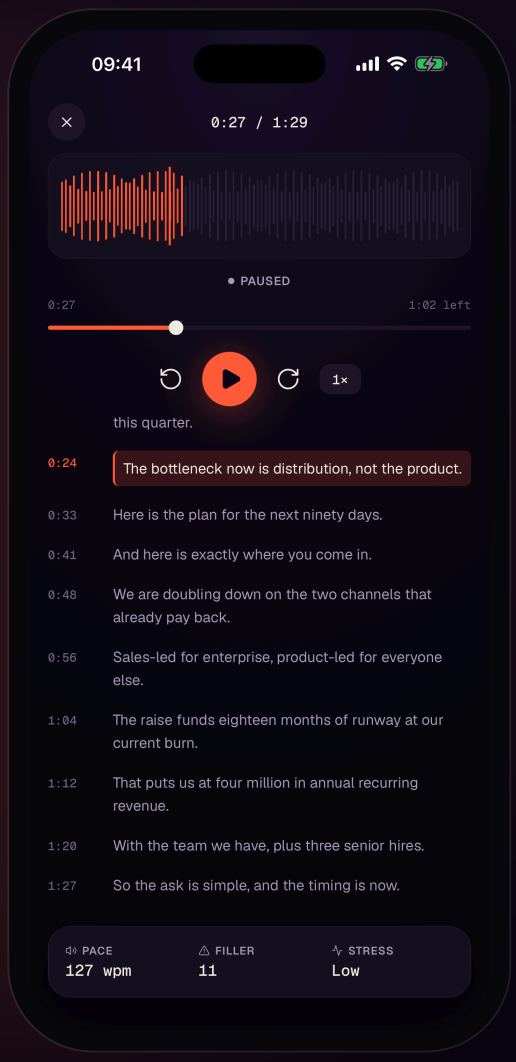
This is the *thing itself.*

Two screens from the shipped build. Not concepts, not renders—the app that is in the store, running.

01

The CLEAR score

Five axes, one pentagon, scored per take. C, L, E, A, R around the edge.



02

Playback with the transcript

Every line timestamped. Pace, fillers, and stress measured against what you said.

The pentagon is the product thesis drawn as a shape. **A single number tells you that you were an 84 and leaves you there.** Five named axes tell you which one to fix on Tuesday — and the app hands back a physical cue rather than a metric.

Before the *first line.*

Three assistants, three jobs, one week of reading.
The build was not the first thing that happened.

01 / 04	02 / 04	03 / 04	04 / 04
PX	**GM**	**CL**	**7**
Perplexity	**Gemini**	**Claude**	**Rivals**
Market sweep. Who exists, what they charge, what their users complain about.	*Synthesis, the long comparisons, and the shape of the category.*	*Design, decisions, and every line of the build itself.*	*Named, priced, and read down to their one-star reviews.*

A month is not enough time to discover the market while building for it. So the market came first, and it came from three assistants doing three different jobs.

Perplexity swept. Who is already in this category, on which platform, at what price, with how many ratings. It is the tool that cites, which matters when the output will decide what you charge.

Gemini synthesized. Long comparisons, pricing tables, the shape of the category. Fast on big documents, and good at holding a wide comparison in one place.

Claude decided and built. Mockups, architecture, the rules, the code. The research fed it; it did not do the research.

The rule I set

No claim entered the plan without a source I could open. **Ratings, prices, and review counts are quoted, not remembered.** The competitive picture in this chapter is dated June 2026 and was re-verified during the pre-launch audit—because a price you read once in May is a rumor by June.

What it cost

Roughly a week of evenings, spread across the first days and revisited before launch. It is the cheapest week in the project. Every hour spent reading a rival's one-star reviews saved a day of building the wrong thing.

What none of them could do

No assistant decided what the product should be. They compressed a week of reading into an evening and priced the competition. They did not decide that presence is physical rather than psychological, or that the app should be the one you open the week before a talk that matters. That part has no shortcut, and it is exactly the part people mean when they say the machine cannot do their job.

> *"A competitor's price is only true on the day you read it. Everything else is memory."*

The people already *in the room.*

Seven rivals, priced and read in June 2026. The weakness column is what their own users say, not what I wish were true.

RIVAL	PLATFORM	PRICE	EDGE	WHAT USERS SAY
Yoodli	Web + desktop. No iOS app.	Free 5 roleplays lifetime · $8/mo annual · $20/mo	AI roleplay with personas, live Zoom and Meet feedback. G2 4.7.	Pivoting to enterprise. *Corrections sound very automated.* Dropped Toastmasters, August 2025.
Speeko	iOS, Watch, Mac. 4.73★ from 4,500.	$24.99–29.99/mo or ~$89.99/yr	Real-time alerts, warm-ups, unlimited free basic recordings.	Expensive monthly. **Zero video or body language.** No deck awareness, no Q&A.
Vocal Image	iOS. 4.52★ from **9,900** — the largest base in the niche.	~$12.99–15/mo, ~$80/yr	Voice identity, structured lessons.	Trial billing traps. *Robotic AI feedback.* Slow support.
Orai	iOS + Android. 4.62★ from 3,600.	~$9.99–12.99/mo, $40–50/yr	Courses with personalization, facial expression.	*Lost connection* in roughly half of sessions. Paid Insights reported broken.
Poised	macOS, Windows. Owned by Deepgram.	Free + $19/mo ($13 annual)	Real-time coaching inside live meetings.	Not mobile. A different job: meetings, not rehearsal.
Wellspoken	iOS + Android. 4.75★ from 1,600. New in 2025.	Subscription	Mock interviews, roleplay, a daily habit loop, its own Index score.	Young. Shallower analysis.
VirtualSpeech	VR and web.	~$45/mo	50+ avatar scenarios, enterprise reach.	Expensive and course-shaped. Not a daily habit.

THE TWO ORPHANS

Yoodli ended its Toastmasters partnership in August 2025, leaving roughly 280,000 members without a sanctioned practice tool. Microsoft retired Teams Speaker Coach the same month. Two audiences lost their app and nobody has claimed them.

WHAT THE TABLE DECIDED

Every mobile rival is audio-only. Every rival with video is desktop-bound. That single line in the grid is the whole reason Stage mode exists—and the reason the app is iOS-first rather than web-first.

The gaps, and *the position.*

The honest half of a competitive map, and the room it pointed to. Both written before launch, not discovered after it.

01 / 04

PDF

Deck-aware

Upload the slides. Coverage scored against what you actually said.

02 / 04

75

Minutes

A real 45-minute keynote fits. No per-session cap.

03 / 04

EU

Hosted

Self-hosted, transparent retention, no card for Solo.

04 / 04

5

Axes

CLEAR is brandable. An Index is a number.

What Tymbre does not have

Real-time feedback while you speak. Yoodli, Poised, and Speeko interrupt you mid-sentence; Tymbre is post-hoc only, and that is the biggest gap in the table. **Conversational roleplay:** Yoodli's core loop is a machine that talks back; Tymbre's Q&A writes the questions but does not hold the conversation. **Live meetings, web, desktop, Android, a lesson ladder, an annual plan** — all absent. Android is out of scope on purpose; the rest is a resourcing fact, not a philosophy.

Not missing: eye contact and body language, which no mobile rival has at all. Vocal variety. Q&A with personas. Progress trends. PDF and share. And deck awareness, which nobody on iOS has in any form.

The room nobody was standing in

Slides. The only mobile app that reads your deck: upload a PDF, get coverage against what you actually said, see the current slide picture-in-picture while recording.

Somatic suggestions with timestamps. Physical actions, not metrics — the direct answer to the loudest complaint about every rival in the table: *robotic feedback.*

Stage mode. The whole body, on a phone. Yoodli and Poised are webcams from the waist up; every mobile rival is audio-only. The table had an empty column and Stage stands in it.

Privacy as a sales position, in a category whose loudest complaints are billing traps. And **CLEAR**: a score you can explain is a score someone repeats.

The line that became the strategy

Yoodli went enterprise. Poised went to meetings. The mobile rivals are drill apps. Nobody was building **the app you open the week before something that matters** — deck-aware, full run-throughs, Q&A from your own slides, Stage as a dress rehearsal. That gap was not a feature idea. It was the product.

What to charge, and *why.*

The number was not chosen by feel. It was placed against seven other numbers.

01 / 04

9.99

Speaker

Under Speeko and Vocal Image monthly. Level with Orai.

02 / 04

19.99

Master

Matches Yoodli Advanced and Poised. Justified by video minutes.

03 / 04

30

Free

Minutes a month on Solo. No card, no trial, no countdown.

04 / 04

0

Annual

Every rival has one. The pricing call I am least sure of.

Pricing is where solo builders guess, and the guess is usually a shrug at $4.99. The table made the decision instead.

Where the money sits

Speaker at $9.99 a month lands in the consumer sweet spot: under Speeko and Vocal Image monthly, level with Orai. **Master at $19.99** matches Yoodli Advanced and Poised, and is justified by video minutes—the expensive thing nobody else on mobile does at all.

The two risks I shipped anyway

No annual plan. The whole category converts on annual: $40 to $90 a year. Against Speeko's $89.99-a-year anchor, monthly-only Tymbre reads two to three times more expensive than it is. This was a deliberate beta decision and it is the one I would revisit first.

Solo is 30 minutes a month against Speeko's *unlimited free recordings.* Optically it loses. The counter-argument is depth: Solo gets the full CLEAR analysis and three somatic suggestions, where unlimited-but-shallow gets you a number.

Why Poland pays 99.99 zloty

Because they are the same price. Apple quotes US tiers **before tax** and every foreign tier **inclusive of local VAT**. Poland's 23 percent is already inside the zloty figure. The App Store also rounds every currency to a local price point—charm-priced, not converted. Two numbers that look 20 percent apart are one number wearing two tax regimes.

Why Solo exists

No card, no trial, no countdown. The loudest complaints in this category are billing traps, so the free tier is a position rather than a funnel: the full CLEAR analysis, three somatic suggestions, thirty minutes a month. **If the free analysis is not good enough to sell the paid tier on its own, the paid tier should not exist.**

> *"Pricing is where solo builders shrug at $4.99. The table decides better than the shrug."*

The name, the line, *the score.*

Three pieces of language, each of which had to survive a search, a store listing, and being said out loud.

01 / 04

Y

The letter

The real word was gone. That single y is what the name cost.

02 / 04

M

Capital M

Sentence case, no period, no prefix, never an exclamation.

03 / 04

5

Axes

Cadence, Lift, Expression, Authority, Resonance.

04 / 04

1

Index

What the rivals offer instead. One number, nothing you can repeat to yourself.

Tymbre

Pronounced **TAMBER**, after *timbre*—the quality that makes one voice distinguishable from another singing the same note. It is exactly what the product measures, which is rare luck in naming. The y is there because the real word was gone, and because a name you cannot spell after hearing it is a name you cannot search for. That trade is real: every listing carries the pronunciation.

Master the room

Sentence case, no period, capital M only. Not *master your body*, not an exclamation mark, never a prefix. The rule is written down because a tagline that drifts is not a tagline. **Room** does the work: it is the talk, the pitch, the interview, and the difficult conversation, without naming any of them.

CLEAR

An acronym is a constraint wearing a name. Five letters mean five axes and no sixth, and every letter has to earn its place instead of being bent to fit the word. **CLEAR** survived because it is also what the app does to your delivery, and because you can say it out loud without checking the spelling.

Wellspoken has an Index. Tymbre has five named things a person can repeat to themselves in the ninety seconds before they walk on. That is the entire difference, and it is not a technical one.

What the name costs

A made-up word buys you a trademark and a domain, and charges you the discovery you would have had for free from *speech coach*. **Tymbre ranks for nothing on its own.** Every listing has to carry the pronunciation, and the search you did not inherit has to be bought instead. That bill does not arrive in the design review. It arrives in the ad account.

Where it gets *found.*

A store listing is a search product. The research picked the category, the words, and the two audiences already looking.

01 / 04

EDU

Category

Where the rivals actually rank. Productivity buried Orai.

02 / 04

7

Keywords

Chosen against on-device rivals, not against the whole internet.

03 / 04

280k

Orphaned

Toastmasters members, left without a sanctioned tool.

04 / 04

175

Markets

Where the app shipped. Where the ads did not.

The category argument

Education, not Productivity. Not preference—evidence. Speeko, Vocal Image, and Wellspoken live and rank in Education. Orai positioned into Productivity and it reads, from the outside, like a burial. The category is the single highest-leverage string in a listing and it is chosen once.

The words

Public speaking coach. Presentation practice. Speech coach. Filler words. Interview practice. Rehearse presentation. Pitch practice. Chosen against the apps a person actually has on their phone, not against the category in the abstract. The finding worth the whole sweep: **nobody in the store owns "presentation" plus "slides"**—which is the one thing Tymbre does that no rival does.

The two orphans

August 2025 emptied two rooms. Yoodli ended its Toastmasters partnership, leaving roughly **280,000 members** without a sanctioned practice app. Microsoft retired Teams Speaker Coach. Both audiences are organized, vocal, and currently unserved. Neither has been claimed.

The listing is the product

Subtitle, keywords, category, and ten screenshots: for most people the store page *is* the app. So it got a pipeline of its own—images rendered and uploaded straight through the API. **A listing you cannot regenerate with one command is a listing you will never fix.**

And then the ads

Apple Search Ads went live after launch, in the EU, the UK, the US, and Australia. The first campaign served **zero impressions**—a $2 target cost-per-acquisition against a keyword nobody bids two dollars on. Deleted, rebuilt manually. The store listing was research. The ad account was tuition.

Two days *before the code.*

Day zero was mockups. Day one was a rulebook with no features in it. Neither shipped anything, and together they decided the month.

Day zero brought no repository, no stack decision and no code. Clickable mockups in Claude Design, and a research pass on what the app was actually for.

Why screens beat documents

Hand a model a written specification and you hand it a document full of holes you cannot see. “Show the user their progress” contains at least four unstated decisions: over what period, compared to what, in what form, and what happens when there is no data yet. The model fills every hole. Plausibly. Wrong. You find out when you are looking at the screen.

A mockup forecloses that. The empty state is drawn or it is not. Every ambiguity is resolved by the act of drawing. It cost one day and shipped nothing — against discovering on day nineteen that the results screen I described and the one I imagined were different objects. Those screens became **39 custom UI primitives, 4,263 lines.** No component library.

The first commit had no features

It landed on day one, late, and shipped a single file — `CLAUDE.md`: **1,089 lines** of rules for how the model was permitted to behave here.

It compiles, it reads well, and it quietly solves a problem adjacent to the one you had. Review is exactly where plausibility wins, so the rules are process constraints rather than aspirations: *state your assumptions before implementing. If two attempts with the same strategy failed, stop and change approach.*

Then 1,089 became 115

The file never grew; it was distilled. An unenforceable rule is noise, and noise in an instruction file is worse than silence. Every rule that survived had caught a real failure. The ones that went sounded wise.

Short and enforced beats long and ignored.

> *“The failure mode of AI is not bad code. It is plausible code.”*

Rules one *through five.*

What survived a month of evidence. Each rule exists because it caught something real, not because it sounded wise.

01 Think before coding

ASSUMPTIONS ALOUD

State assumptions explicitly. If there are two readings of the request, *surface both instead of silently picking one.* Confusion admitted early is cheap. Confusion discovered in review is not.

02 Simplicity first

NOTHING SPECULATIVE

The minimum code that solves the problem. No abstractions for single-use code, no configurability nobody asked for, *no error handling for impossible states.*

03 Surgical changes

NARROW DIFFS ONLY

Do not improve adjacent code. Do not refactor what is not broken. *Every changed line should trace to the request*—anything else is unreviewable noise.

04 Goal-driven execution

DEFINE DONE FIRST

Turn tasks into verifiable goals. Not *add validation* but *write tests for invalid inputs, then make them pass.* Strong criteria let the model run alone; weak ones need a babysitter.

05 Done means confirmed

SHIPPED IS NOT DONE

Certainty that the change was pushed is not evidence that it works. *Until it has been seen running, it is not done.* This rule alone caught more false completions than any other.

Rules six *through ten.*

The second half. These are the ones about knowing when to stop—the hardest class of rule to write and the most expensive to omit.

06 Look harder, not faster

NEW DATA, NOT NEW GUESSES

When a fix fails, the next move is to learn more—inspect state, add visibility, read the source. *A second hypothesis without new information is the same dice with a new label.*

07 Two failures, change approach

THE THIRD TRY IS BANNED

After two attempts with one strategy, stop. Say what was tried, what will be different, and only then continue. *Persistence without a change of method is just volume.*

08 Destructive acts need forethought

KNOW THE COST, KEEP A WAY BACK

Before deleting, resetting, or overwriting: know exactly what disappears, get consent if it touches shared state, *keep a path back.* Asking costs less than regret.

09 The report is a symptom

NOT A DIAGNOSIS

A bug report describes what someone saw, not what caused it. *Verify the cause against the actual state* before building a fix around the reporter's theory.

10 Secrets stay secret

EVERY KEY IS PRODUCTION

Never paste, log, or commit credentials—*not even while debugging.* If a task seems to need a key that is not there, stop and ask rather than invent one.

IV · THE BUILD

Thirty days, *three phases.*

One deadline, no slack. The phases were not planned —they are what the git history turned out to contain.

DAYS 00–06 · PHASE 01

Decide.

Mockups, first commit, and the discovery that the entire backend was the wrong shape. The most consequential week, and the one with the least to show for it.

- → **Day 00.** Research and clickable mockups. No repository yet.
- → **Day 01.** First commit: `CLAUDE.md`, 1,089 lines. Zero features.
- → **Day 03.** Apple Developer Program enrollment—started early because the paperwork has its own clock.
- → **Days 01–06.** A custom backend: Hono, BullMQ, a Python sidecar, Drizzle, migrations, Docker.
- → **Day 06.** All of it deleted in one commit and moved to Appwrite.
- → Two applications, `apps/api` and `apps/analysis`, **lived five days** and were gone.

Enrolling in the Developer Program on day three mattered more than any code written that week. Apple's clock runs independently of yours: agreements, tax forms, and the DSA declaration all have review queues. Start them before you need them, or the deadline you miss will be an administrative one.

DAYS 07–24 · PHASE 02

Build.

The long middle. Screens, backend functions, payments, notifications, the analysis pipeline. This is where the 413 fixes happened.

- → **33 screens** and 58 components, 39 of them custom design-system primitives.
- → **18 backend functions** in Python, wired to Appwrite events, schedules, and HTTP.
- → Payments, subscription tiers, and **server-side quotas**—never trusted to the client.
- → Push notifications rebuilt natively after the first approach proved unreliable.
- → **Day 16.** The record: 99 commits in one day.
- → **Day 24.** Apple verifies the DSA declaration. The administrative clock lands.

The middle is where a deadline is actually won or lost. Nothing here was dramatic. It was 413 fixes, one at a time, in a stack I had chosen six days earlier and could no longer renegotiate. The pivot bought this phase its speed; the phase itself was just work.

DAYS 25–30 · PHASE 03

Ship.

Audits, the build pipeline that had to be rebuilt from scratch, and a launch-blocking bug found hours before submission.

- → **Three AI audits**: 30 agents, 1.65 million tokens, roughly 70 findings.
- → Every critical finding fixed before submission. The rest were documented and deferred.
- → The cloud build quota ran out—for the second time. **The pipeline was rebuilt from zero**, down to one command.
- → A **launch-blocking bug** surfaced in session delete and rename. Found and fixed the same day.
- → **49 commits in 24 hours.** The busiest day of the month.
- → Build 27, version 2.3.0, submitted for review. Apple acknowledged receipt the same morning.

The final day is a stress test of everything you deferred. Losing the build pipeline hours before submission was not bad luck—it was a dependency I had never owned, failing at the moment it mattered. Rebuilding it took a day I did not have and produced a one-command build I still use.

Thirty days end to end. Apple approved on June 27, eleven days after submission. The code has not changed since June 16—everything after launch has been operations.

Day six: *delete everything.*

Six days of backend work, removed in a single commit. It is the reason thirty days was enough.

The project did not start on Appwrite. It started on a backend I designed myself: **Hono** for the API, **BullMQ** for job queues, a **Python sidecar** for the analysis work, **Drizzle** for the schema, migrations, a Dockerfile, and two separate applications in the monorepo— `apps/api` and `apps/analysis`.

It was a reasonable architecture. It was also, for one person on a thirty-day clock, a second product. Queues need monitoring. Migrations need a strategy. A sidecar needs a deployment story. None of it is visible to a user, and all of it competes for the same hours as the app.

On **May 23, day six**, one commit removed it:

```
feat: pivot backend from Hono/BullMQ/Python sidecar to Appwrite
```

Two applications that had existed for five days were gone. The queue infrastructure, the migration tooling, the container config—all of it deleted in favor of a self-hosted Appwrite instance with functions, a database, and auth already solved.

Why it counts as the best decision

Not because Appwrite is better. Because it was **already finished**. The pivot did not buy me a superior queue; it bought me the six weeks I would have spent maintaining my own.

This is a large share of why the churn figure is what it is: **255,699 lines added, 188,620 removed**, against 50,662 standing today. The project was written about five times over. None of it was waste. It was the cost of finding out.

The uncomfortable part

I could have made this call on day two with a couple of hours of honest reading. I did not, because building the backend was more fun than evaluating one. The five days were tuition.

> *"It did not buy a better queue. It bought the six weeks I would have spent maintaining mine."*

Fifteen services, *seven languages.*

A full product is not one codebase. It is a dozen accounts, each with its own clock, quota, and way of failing.

01 / 04

iOS

Mobile

React Native and Expo. 169 files, 34,286 lines of TypeScript.

02 / 04

PY

Backend

18 Appwrite functions, 7,975 lines of Python. Events, schedules, HTTP.

03 / 04

AI

Analysis

Several models behind the scoring and the Q&A simulator, each doing the part it is best at.

04 / 04

17

Tooling

Custom scripts and Expo plugins built because nothing existed.

The visible product is an iOS app. Behind it: a self-hosted backend, a marketing site, a payments layer, push infrastructure, an AI pipeline, object storage, error monitoring, transactional email, and the App Store machinery itself. **More than 15 external services**, each with an account, a key, a quota, and a distinct failure mode.

Seven languages

TypeScript for the app and the site. Python for the backend functions. Astro for the landing page. JavaScript for the build plugins. **PHP** for one maintenance script—because that was what the tool it patched happened to be written in. Shell for the pipelines. HTML and CSS throughout.

The accounts

Fifteen services is not a boast—it is a liability. Each one is a login, a key to rotate, a quota to watch, and a status page that starts mattering at two in the morning. Two of the month's eight failures came from a dependency I did not own: nothing to debug, only to wait out or replace.

Nobody chooses seven languages. You accumulate them, one pragmatic decision at a time, and then you own all seven.

Seventeen tools I had to build

Fourteen scripts and three Expo plugins, roughly 1,380 lines. A screenshot pipeline that renders App Store images and uploads them through the API. A demo-data seeding toolkit for the reviewer's account. A local iOS build pipeline—one command from prebuild to upload. A garbage collector for build artifacts, because the backend platform does not ship one.

None of this is product. All of it was necessary. **This is the part that surprised me most**: a meaningful slice of a solo build is not building the thing, it is building the machine that builds the thing.

Self-hosted, *on purpose.*

The easy path was a managed backend and an analytics SDK.
Both doors were closed early, because neither can be closed late.

The backend runs on **my own server**: a Hetzner instance in Germany, orchestrated with Coolify, running a self-hosted Appwrite stack of roughly two dozen containers. In front of it, **Cloudflare** — DNS, CDN, WAF, certificates. Recordings live in **R2**, errors go to Sentry. Every piece is a thing I chose and now own.

Why take the harder road

A managed backend is faster to start and slower to leave. The two properties that mattered — **data residency in the EU**, and recordings under my own key rather than a vendor's — cannot be retrofitted once you have shipped. Privacy claims that depend on someone else's terms of service are not claims. They are hopes.

Money is the smallest part: tens of dollars a month. The real cost is that **you are the operations team.** When a scheduler silently stops firing, nobody pages you. Both of those happened, and both are in the failures chapter.

What the app refuses

Tymbre records people rehearsing — voice, face, sometimes the whole body. Close to the most intimate data an app can hold, kept for the least defensible reason: practice. So the constraints came before the features.

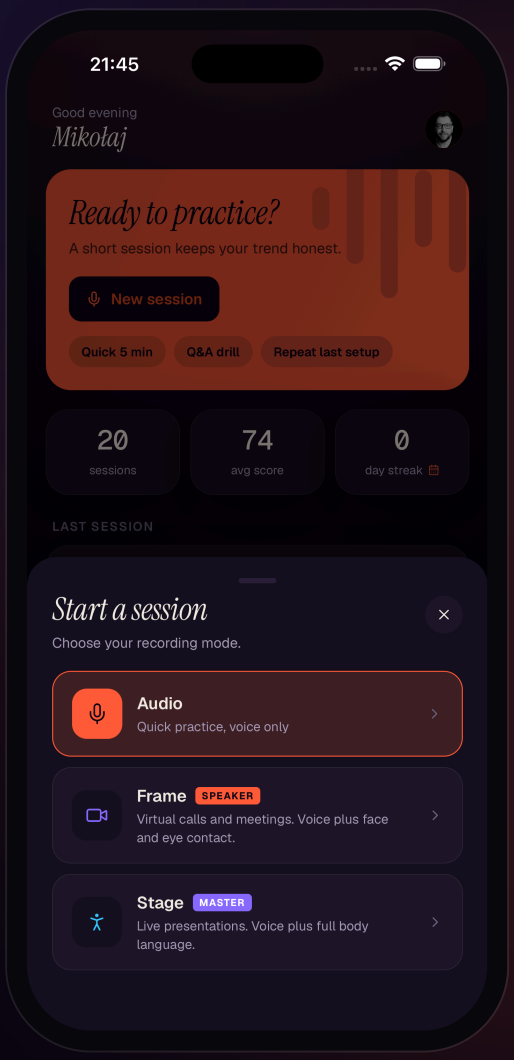
No trackers. None; there is no analytics SDK in the app. **No advertising identifier** — no IDFA request, no tracking prompt, because there is nothing to ask permission for, and the claim is verifiable in the binary. **PII scrubbed before monitoring:** a crash report should tell me the stack, not who was speaking. **EU-hosted end to end. Compliant with the EU AI Act**, because the app makes automated assessments of a person's delivery.

Every one of those is decided before the code exists or not at all. **Nothing here is a feature.** All of it is a door closed early, which is why it was still shut at the end. That is the trade. **Not simpler — yours.**

> *"A privacy claim that depends on someone else's terms of service is not a claim. It is a hope."*

Three modes, *one hostile room.*

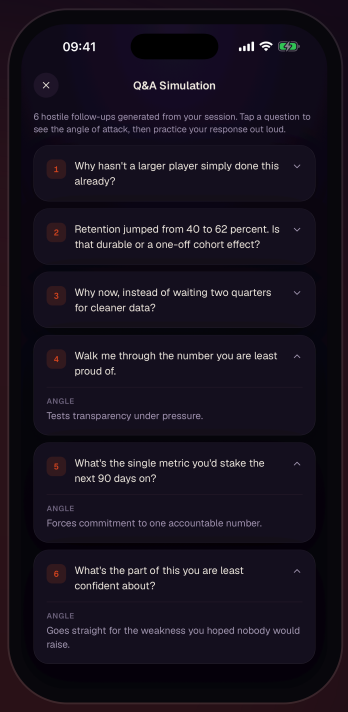
What the tiers actually buy, and the feature that came from asking what people are afraid of.



01

Choosing a mode

Three ways to record, and the whole tier ladder in one screen.



02

The Q&A simulator

Hostile follow-ups from what you just said, each labeled with the angle it attacks from.

- **Audio**
  Voice only. Quick practice, nothing to set up.
- **Frame** SPEAKER
  Voice plus face and eye contact. For calls and meetings.
- **Stage** MASTER
  Voice plus the whole body. For the room you walk into.

The mode picker is the pricing page in disguise. **Solo, Speaker, Master climb the same ladder as Audio, Frame, Stage** — you pay for how much of yourself you are willing to put in the frame. The Q&A simulator exists because the question that ends a pitch is never the one in your deck.

VI · THE FAILURES

Eight things *that broke.*

Named, dated, and mostly self-inflicted. The two that reached production are the ones worth your attention.

01 *Subscriptions sold in two countries out of 175.*

The app shipped to 175 territories. The subscriptions were configured for two. In 173 countries you could install it, reach the paywall, and find nothing to buy—while ads ran across the EU. Found and fixed nineteen days after launch.

02 *The scheduler that silently stopped.*

A delta-sync bug in the backend platform. After a restart it only loads schedules changed since it booted; ours were stamped at the last deploy. Five cron jobs—cleanup, retention, notifications—stood dead for two weeks. Nothing alerted. Nothing logged an error.

03 *Five days spent on a backend that shipped nothing.*

Hono, BullMQ, a Python sidecar, Drizzle, Docker. Deleted on day six. The right call, made four days late, because building it was more interesting than evaluating alternatives.

04 *The build quota, twice.*

The cloud build allowance ran out on day nine and again on day thirty—the second time hours before submission. A dependency I never owned, failing exactly when it mattered. The fix was to stop renting it.

05 *A launch-blocking bug found on the last day.*

Session delete and rename crashed against a cached list. Found hours before submission, in a flow used on every single session. Manual testing caught it. Nothing else would have.

06 *Monitoring that was never on.*

The error-reporting key was never set on the scheduled functions. Every heartbeat had been quietly doing nothing for weeks. An audit found it, not an alert—which is the whole problem with monitoring you have not verified.

07 *Fourteen reverts.*

Fourteen commits exist purely to undo other commits. Not a failure of process—the process working. But it is the honest shape of the month, and any log that shows none is hiding something.

08 *One file, edited 108 times.*

The profile screen was touched 108 times across 30 days. Not a bug, a signal: the place a design keeps getting rewritten is the place the design was never actually decided.

THE PATTERN

Six of these eight were invisible until something went looking. **The failures that hurt are not the ones that crash—they are the ones that quietly succeed at doing nothing. There is not one automated test in this project.** Instead: three AI audits before submission — **30 agents**, **1.65 million tokens**, roughly **70 findings**, every critical one fixed. That was the QA, and an argument for adversarial review as standing practice.

Shipping, and *what it leaves.*

Seventeen builds and a paperwork queue that ignores your deadline — then the half nobody writes about.

17 builds went to TestFlight, numbered 3 through 27. The gaps are failures: builds that would not sign, builds the upload tool rejected, builds that ran out of quota. Build 27 became version 2.3.0 and went to review on **day 30**. Approval came eleven days later.

The queue you cannot sprint

Developer Program enrollment on day three. Paid Applications Agreement and the EU **DSA declaration** on day fourteen; Apple verified the DSA data on day twenty-four. None of it compresses by trying harder. **If the deadline is thirty days, the paperwork starts in week one or it becomes the reason you miss it.**

Certificates, provisioning profiles, entitlements, a privacy manifest, three Expo plugins to make the native build behave. None of it interesting, all of it mandatory, every piece failed at least once. When the cloud build quota ran out for the second time — on the final day — I rebuilt the pipeline locally. **One command.**

Then the code stopped

There have been **no commits since June 16**. The app went live on June 27; every hour since has gone into operations, not code.

It means noticing that subscriptions were sold in two countries while the app was live in 175: nothing broken, the configuration simply wrong where code does not look. It means five scheduled jobs dead for two weeks from a platform bug that produces **no error, no alert, no log line.** Just absence.

Building has a finish line. Operations has a heartbeat: it never finishes and it degrades quietly. Thirty days bought a product. **The product bought a permanent job**, and I did not price that in.

> *"The cron that does not run produces exactly the same log output as the cron that never existed."*

The month, *in figures.*

Pulled from git, App Store Connect, and the running stack. The unflattering ones are here too.

01 / 04

30

Days

Day zero mockups to Apple's review queue. One day off in the whole month.

02 / 04

1,116

Commits

One every 37 minutes on average. Record day: 99.

03 / 04

444k

Lines

Moved across the month. 50,662 standing in the repository today.

04 / 04

200

Hours

Estimated from session clustering. Alongside a full-time job.

Time

Thirty days, May 17 to June 16. **29 days with commits**—the only empty one was June 13. Longest unbroken run: **26 days**. Longest gap between any two commits in the entire month: **27 hours**. Roughly **200 hours** across **86 working sessions**, estimated by clustering commit timestamps, not self-reported.

Code

1,116 commits: 413 fixes, 164 features, 120 documentation, 83 chores, 29 refactors, **14 reverts**. Lines added **255,699**, removed **188,620**—about **444,300 moved** against **50,662 standing** today. The project was written roughly five times over. The largest single commit removed 45,832 lines in one cleanup. **967 distinct files** were touched at some point; one of them, the profile screen, 108 times.

Shape

33 screens. 58 components, of which **39** are custom design-system primitives (4,263 lines). **18 backend functions**, 7,975 lines of Python. **24 Astro files** for the marketing site. **347 files** in the repository across **25 file types**.

Tooling

7 languages in one repository. **15+ cloud services** wired together. **17 command-line tools** written along the way, because nothing off the shelf did the job. **17 builds** submitted, numbered 3 to 27. The rulebook shrank from **1,089 lines to 115**.

Rhythm

28 percent of commits landed between midnight and 5 a.m. The three busiest hours of the day were 23:00, 02:00, and 01:00. **17 percent** landed on a Saturday or Sunday. Zero meetings. Zero automated tests. Zero people other than me.

CLOSING

What it cost, *what it taught.*

Thirty days is not a productivity claim. It is a report on where the work actually went once the typing stopped being the hard part.

- **The constraint did the work.**

  Thirty days did not make me faster. It made me delete things—the custom backend, the settings nobody asked for, the second opinion I wanted on the architecture. A deadline is a filter, and most of what it filters out is the work you would have enjoyed most.

- **The bottleneck moved, it did not disappear.**

  The model wrote most of the lines. It did not decide anything that mattered: not the pivot, not the privacy posture, not the day spent on mockups. Typing got cheap. Judgment did not, and judgment is the part that does not parallelize.

- **Plausible is the enemy, not broken.**

  Broken code announces itself. Plausible code compiles, reads well, and solves a problem adjacent to yours. Every rule in the ten that survived exists to catch plausibility before it reaches review, because review is exactly where plausibility wins.

- **Operations is the other half, and it is unpriced.**

  The code stopped on June 16. The work did not. Two of the worst failures in this document happened after launch, produced no errors, and would never have been caught by a test—because nothing was broken. Configuration was simply wrong somewhere code does not look.

- **The paperwork does not negotiate.**

  Agreements, tax forms, the DSA declaration, the review queue. None of it compresses under effort. If the deadline is thirty days, the administrative clock starts in week one or it becomes the reason you miss.

- **What it does not prove.**

  One person, one app, one month. It does not prove this scales, that it survives a second engineer, or that skipping tests is wise beyond a solo sprint. It proves one thing—the distance between an idea and a store listing is now made almost entirely of decisions.

01.

Draw it first. A mockup is a specification that cannot hide its own ambiguity. One day, before any repository exists.

02.

Write the rules before the features. Short and enforced beats long and ignored. Distill them as evidence arrives.

03.

Verify the boring parts. Territories, prices, schedules, monitoring. The failures that hurt are the ones that quietly succeed at nothing.

The distance between an idea and a store listing is now made almost entirely of decisions.

— CLOSING, PAGE 23

CONTACT

Mikołaj Salecki
Digital consultant

hi@salecki.digital
salecki.digital

salecki