

PRODUCT & TECHNICAL AUDIT

PAGEVOICE

Independent review of the existing platform and mobile migration plan – verdict, risks, findings and a costed path to iOS and Android.

Sample report based on a real engagement. Client name and identifying details have been changed.

CLIENT

Lumen Labs B.V.

Netherlands · non-technical founder, small in-house team

PRODUCT

PageVoice

Printed text to audio for people with low vision and older adults

PREPARED BY

Ihor Klymchuk

Fractional CTO · 2026

01 EXECUTIVE SUMMARY

The platform is worth keeping. The mobile apps are worth building — after five fixes.

VERDICT

REUSE THE BACKEND, MIGRATE TO REACT NATIVE

— fix 5 issues first

One codebase for iOS and Android, the existing Python API kept, and the web app left untouched.

WHAT'S SOLID

- **Good backend foundation** — clean REST structure, worth reusing as is for mobile.
- **Security baseline is solid** — with one high-risk item (token storage).
- **Partial test coverage** — a base to extend, not a blank page.

RECOMMENDED PATH

Start with a short discovery, then fix the five findings in the backend — they cost money and block mobile either way. Migrate the frontend to React Native task by task with AI agents under engineer review. Build the native pieces the product actually needs: a real audio player and in-app purchases. Finish with accessibility, privacy and store submission.

TOP RISKS — BUSINESS CONSEQUENCE FIRST

- 01 **You are billed twice for the same job** — duplicate paid provider calls.
- 02 **One website bug could hand over user accounts** — tokens stored in the browser.
- 03 **Large documents and audiobooks fail to upload** — server limit far below what the app accepts.
- 04 **Listening breaks after 15 minutes** — media links expire mid-session.
- 05 **The App Store will reject the app as it bills today** — Apple requires in-app purchase.

ESTIMATE

212–308 h

Total, all phases

196–284 h

Without the optional offline cache

Plus **1–2 weeks of store review** calendar time, which no amount of engineering can shorten.

02 RISK OVERVIEW

Seven items, ordered by what they cost you. Every one is fixed inside the plan – nothing here is a reason not to proceed.

RISK	SEVERITY	BUSINESS IMPACT	WHERE IT'S FIXED
Billing not App Store compliant	Critical Release blocker	The app will be rejected without Apple IAP / Google Play Billing – no launch at all	Native work
Quota race condition on paid provider calls	High	Direct money loss – the same document is billed twice by the AI provider	Backend fixes
Tokens in browser localStorage	High	Any cross-site scripting bug becomes full account takeover – documents contain personal data	Backend / frontend fixes
Upload size mismatch – 6 MB server cap vs 200 MB audio / ~100 MB scans	High	The core feature fails for long documents and audiobooks – the user sees an error with no explanation	Backend fixes
Synchronous processing – no background jobs	Medium	Slow responses and timeouts – and those timeouts are what trigger the double billing above	Backend fixes
Signed media URLs expire after 15 minutes	Medium	Paused listening resumes into a dead link – and it is a hard blocker for the mobile player	Backend fixes + player
Email links point to the web app	Medium	Verification and password reset don't open the mobile app – new users get stuck at sign-up	Store & release

SEVERITY SCALE

Critical – blocks release

High – money or account safety

Medium – degrades the experience

Low – worth fixing when nearby

— Risk-by-risk effort and sequencing are itemised in the [task-level estimate sheet](#) .

03 SCOPE, METHOD & CURRENT ARCHITECTURE

WHAT WAS REVIEWED

Backend code – REST API, services, jobs

Frontend code – web app and admin panel

Configuration – nginx, storage, provider setup

Delivery process – branching, CI, deployment, tests

LIMITATION WORTH KNOWING

Performance was assessed from the code and architecture. No production metrics were available, so timing claims describe the shape of the problem, not measured numbers.

CURRENT STACK

React + Vite web app

Separate admin panel

Python REST API

nginx

Paid OCR / TTS provider

CURRENT ARCHITECTURE – WITH PROBLEM SPOTS MARKED

● High ● Medium

CLIENTS

Web app – Vite / React

- Access & refresh tokens kept in localStorage

Admin panel

Curates the public library

EDGE

- nginx ● Request body capped at 6 MB – below what the app accepts

API

Python REST API

- OCR and speech generation run inside the HTTP request – no job queue
- Quota consumed after the paid call, not atomically

DATA

Database

Users, titles, quotas, library metadata

File storage

- Signed media URLs valid 15 min

PAID

OCR + text-to-speech provider Billed per call

Figure 1 – PageVoice as it runs today. Coloured dots mark the findings detailed on the next two pages.

04 FINDINGS – MONEY & THROUGHPUT

One problem seen from two sides: long synchronous calls create the timeout that causes the double billing. Fix 02 first, then 01.

Finding 01

● High

Quota race condition on paid provider calls

WHY IT MATTERS

You pay the AI provider twice for one document, and the user's quota is wrong either way. The loss grows with traffic and is invisible until the invoice arrives.

WHAT WE FOUND

Quota is consumed **after** the paid call, and not atomically. The frontend disables the button during the request, but when a long call times out the button re-enables while the original request is still running on the server. The retry creates a new document with a new idempotency key – so the provider is called, and billed, a second time.

RECOMMENDATION

Make quota consumption atomic and record it **before** the paid call, keyed on the source document rather than the request. Retries then resolve to the same job instead of buying a new one.

Finding 02

● Medium

No background jobs – processing blocks the request

WHY IT MATTERS

This is the "slow processing" the team reports. The user waits on a spinner with no progress and no estimate, and on mobile the operating system may suspend the app before the call returns – which loses the result the user already paid for.

WHAT WE FOUND

OCR and speech generation run inside the HTTP request – in the job service and the library API. Request duration therefore tracks document length, and long documents reliably hit the timeout that feeds finding 01.

RECOMMENDATION

Move processing to asynchronous background jobs with status polling. The app returns immediately, shows real progress, and survives being backgrounded – and the timeout that causes double billing disappears.

05 FINDINGS – UPLOADS, ACCOUNTS & PLAYBACK

Finding 03

● High

Upload limits contradict each other

6 MB nginx cap **200 MB** app accepts

WHY IT MATTERS

A core promise – photograph a whole book chapter, publish an audiobook – fails for exactly the largest files. The user sees a generic error and blames the app.

WHAT WE FOUND

nginx caps request bodies at 6 MB, while the application accepts audio up to 200 MB and multi-page scans of roughly 100 MB. The request is rejected before it reaches the API.

RECOMMENDATION

Align the limits across nginx and the app, and consider direct-to-storage uploads so large files bypass the API entirely.

Finding 04

● High

Login tokens stored in the browser

WHY IT MATTERS

One cross-site scripting bug hands an attacker full access to accounts. Users upload letters and bills, so a breach is a personal-data incident under EU rules, not just a support ticket.

WHAT WE FOUND

Access and refresh tokens are persisted in localStorage via the store's persist layer, which any JavaScript on the page can read.

RECOMMENDATION

On web: short-lived access token in memory, refresh token in an httpOnly secure cookie. On mobile: secure storage – iOS Keychain, Android Keystore. Do this before the migration, so the mobile app starts on the correct pattern.

Finding 05

● Medium

Media links expire mid-listen

WHY IT MATTERS

Your users listen to long documents in short sittings. Pausing for twenty minutes and pressing play should not fail. It is also a hard blocker for the mobile player, which must resume from the lock screen.

WHAT WE FOUND

Audio and image URLs carry a 15-minute signed grant, and the current web player has no handling for expiry – it simply plays into a dead URL.

RECOMMENDATION

Add a refresh-on-expiry pattern to the player on both web and mobile: request a fresh signed URL and continue from the same position, invisibly to the user.

06 TEAM & DELIVERY PROCESS

Three low-effort changes your own team can make alongside the migration. They are not included in the estimate.

01 Code reaches production without review

Work is pushed straight to the main branch. A single mistake in a billing or upload path can go live unseen – and the two costliest findings in this report live in exactly those paths.

RECOMMENDATION

Require pull-request review on main. One rule, one afternoon of setup.

02 Tests run automatically, deployment does not

CI runs the test suite, but releases are performed by hand. Manual steps are where releases slip and differ between environments – and mobile releases add store submissions on top.

RECOMMENDATION

Add a scripted deploy pipeline with a staging step before production.

03 Test coverage is partial and uneven

Some areas are well covered, others not at all. Coverage is a base to build on, but the gaps sit where errors cost real money.

RECOMMENDATION

Cover the billing and upload flows first; leave the rest to grow with the work.

IN PLAIN TERMS

The team is competent; the process is simply younger than the product. None of this needs new hires – it needs three rules written down and enforced by tooling rather than memory.

07 MOBILE OPTIONS COMPARED

PageVoice is a listening app. Background audio, lock-screen controls and store billing decide this, not code style.

OPTION	CODE REUSE	NATIVE CAPABILITIES	STACK IMPACT	VERDICT
React Native with Expo	~15–20% of the current frontend reusable – and the same language and ecosystem your team already works in	Full native support: background audio, lock-screen controls, camera, store billing. One codebase serves iOS and Android	Minimal – no new language, backend untouched	 Recommended
PWA	Reuse almost everything	Unreliable on iOS and a poor fit for a listening app – background audio and lock-screen controls are the product, and they are exactly what a PWA cannot guarantee	No change	 Not recommended
Flutter	No reuse	Good native support	Adds Dart. With the web app and admin panel kept, the stack becomes more complex and harder to hire for	 Not recommended
Native Swift + Kotlin	No reuse	Full	Two separate codebases to build and maintain, with no meaningful benefit for an app of this type	 Not recommended

The ~15–20% reuse figure is honest and low on purpose: screens must be rebuilt for touch. The real saving is that business logic, API contracts and the whole backend carry over unchanged – and your team keeps working in one language.

08 TARGET ARCHITECTURE

One new client, one new worker, two new billing integrations. Everything else you already own.

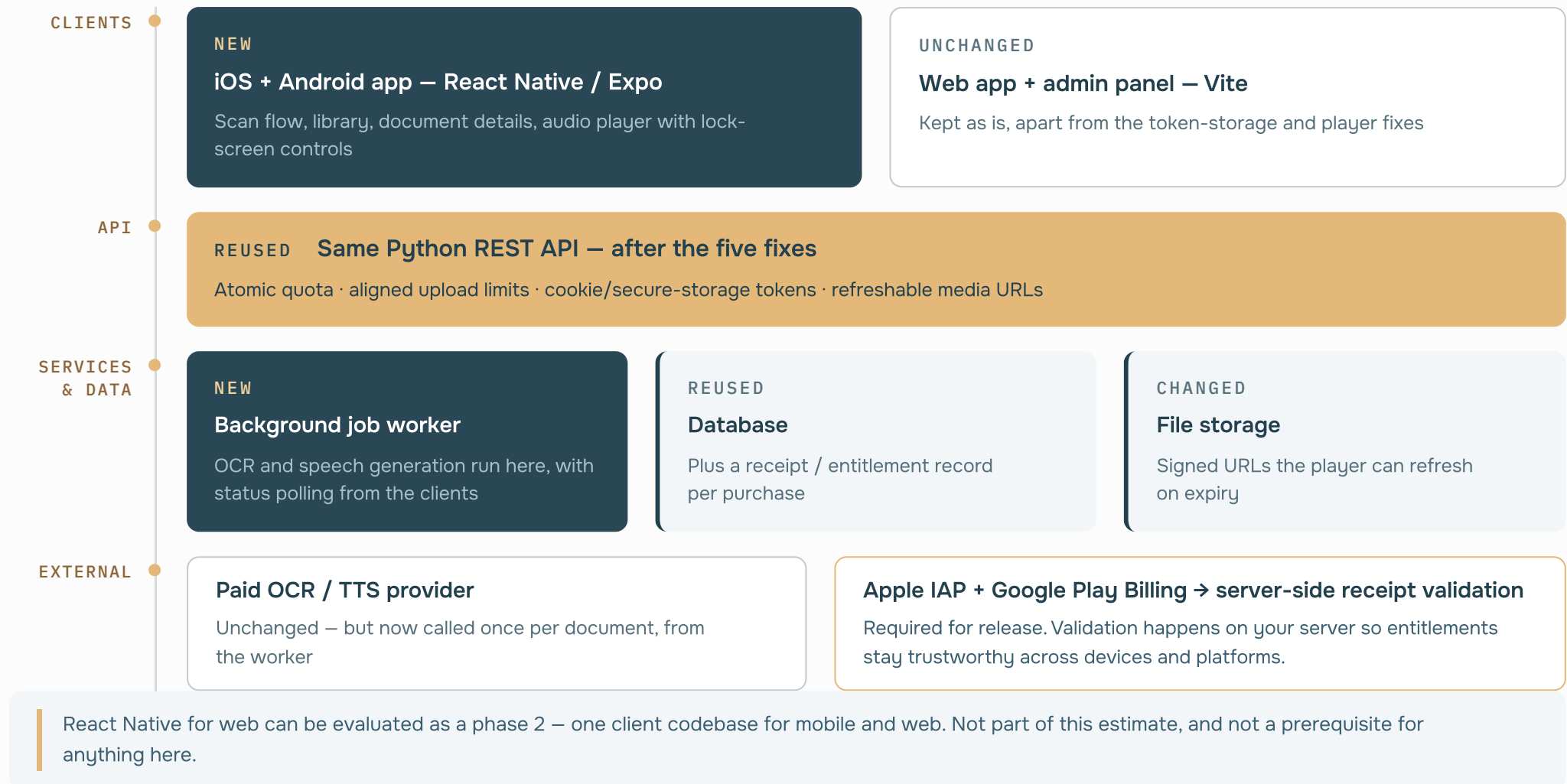


Figure 2 — Target architecture. Sand marks reused pieces, dark marks new work.

— Hours per component are broken down in the [task-level estimate sheet](#) .

09 MIGRATION STRATEGY

Four steps, each with a reviewable result. Nothing is rewritten speculatively.

Step 01

Reuse the backend fully

Once the five findings are fixed, the existing Python API serves the mobile apps unchanged. No parallel backend, no data migration.

Step 02

Break the frontend into migration tasks

An AI agent produces the task breakdown from the existing code; the tech lead verifies it before any work starts. You get a visible, countable plan.

Step 03

Migrate task by task

AI agents implement one task at a time, with code review after each and tech-lead verification. Progress is measurable weekly, not at the end.

Step 04

Native work & store compliance

Audio player, in-app purchases, accessibility and privacy – then publish to the App Store and Google Play.

WHY AI-ASSISTED MIGRATION IS SAFE HERE

This is a migration between two closely related stacks – React to React Native. The work is mostly routine restructuring of components that already exist and already work, not invention. Engineers verify every task, and the same review gate applies whether a human or an agent wrote the code. The result is significantly faster delivery at the same quality.

What AI is **not** used for: the audio player, in-app purchases, accessibility and the store submission. Those are judgement-heavy and platform-specific, and they are estimated as senior engineering work.

WHAT STAYS CONSTANT

- Code review after every task, by a human
- The tech lead owns the breakdown and the definition of done
- The web app keeps working throughout – no freeze on existing users
- You can stop after any task with working software in hand

10 STORE & PLATFORM BLOCKERS

Four requirements that decide whether the app is accepted at all. All four are inside the estimate on the next page.

Billing

● Critical · release blocker

The current billing model is not accepted by the App Store. Digital content must be sold through **Apple IAP** on iOS and **Google Play Billing** on Android.

Commercial consequence: the platform fee has to be priced into the plans before launch, not after.

Email links

● Medium

Verification and password-reset emails link to the web app, so a new user never lands in the app they just installed – they get stuck at the first step.

Fix: universal / app links, or code-entry variants that work regardless of which client is open.

Accessibility

● Core to the product

Your users are people with low vision and older adults. VoiceOver and TalkBack support, Dynamic Type, contrast and large touch targets are the product working – not a compliance checkbox.

This is the one area where cutting scope would damage the product's reason to exist.

Privacy

● Medium

Uploaded documents often contain personal data – letters, bills, medical post. Both stores require accurate privacy labels (App Privacy, Play Data Safety), in-app account deletion and a stated retention policy for uploaded images.

Inaccurate labels are a common rejection reason, and a repeat submission costs another review cycle.

11 ESTIMATE

Rough estimate for current functionality only – no new features. Hours, one senior engineer with tech-lead review.

PHASE	HOURS
1 · Discovery – detailed code review, QA baseline, refined estimate	16
2 · Backend fixes – findings 1–5	24–32
3 · Migration to React Native	60–76
Expo environment, theming, EAS project	4
Shared core	8
Screens – auth, library, document details, upload	16–24
Scan flow – image picker / camera	up to 8
Review, bugfixes, QA	24–32

PHASE	HOURS
4 · Native work	72–112
Audio player with lock-screen controls	24–40
In-app purchases	32–48
Offline audio cache – optional, listen without a connection	16–24
5 · Store, compliance & release	40–72
Developer accounts, certificates, builds, TestFlight + Play internal track, OTA strategy	8–16
Accessibility and privacy review	16–24
Platform requirements, store assets, review cycles	16–32

Discovery is the only phase priced as a fixed number – it exists to turn the ranges on the right into a committed plan.

TOTAL

212–308 h

Without the optional offline cache 196–284 h

Store review adds 1–2 weeks of calendar time on top of these hours, and can repeat if a submission is rejected.

— More detailed estimates – per task, with assumptions – are in the task-level estimate sheet .

12 ROADMAP & NEXT STEPS

One senior engineer full-time, plus tech-lead review. Phases overlap where they safely can.

Discovery



Backend fixes



Migration



Native work



Store & release



Store review



W1

W2

W3

W4

W5

W6

W7

W8

W9

W10

W11

TOTAL TO A PUBLISHED APP**≈ 8–11 weeks** – including 1–2 weeks of store review you cannot compress

Next step 01

Approve discovery

16 hours that turn this estimate into a committed plan with a QA baseline.

Next step 02

Fix the billing race condition & token storage now

Regardless of the migration decision – one costs money today, the other risks accounts today.

Next step 03

Open Apple & Google developer accounts

Business verification takes time and sits on the critical path. Start it in week 1.

— Week-by-week tasks behind this roadmap are listed in the [task-level estimate sheet](#).

Sample report based on a real engagement. Client name and identifying details have been changed.