MeiuxMeiux

SOFTWARE · LOGISTICS · CONSULTING

REQUEST FOR PROPOSAL · CONTINUATION TRACK

MEIUXMEIUX · ENGINEERING PARTNERSHIP

Complete this repository

An invitation to bid on finishing Nah — a credit-only, white-label electronic-signature platform that is already partly built and running in production.

PROJECT **Nah**

CONTENTS

What's inside this document

Read this top to bottom. Every section is short and written in plain language. Each one ends with what it means for you. Technical words are explained the first time they appear and again in the glossary at the end. Do not worry if English is not your first language — this document was written to be easy to translate.

How to read this document

> **This is a request for a proposal, not a contract**
> We are looking for one or more developers to help us **finish** a software project called **Nah**. Part of it is already built and running. We want you to read this, understand the work, and then send us a written offer: what you will do, when, in what order, and how you want to be paid. We will compare offers and reply.

There are **two** versions of this document.

- **This one — "Complete this repository."** You take the existing code we already wrote, and you finish it.
- **A second one — "Greenfield this repository."** You ignore our code and build the same product fresh, your way, against the same requirements.

You may bid on either path. Pick the one you are most confident in. Both paths end with the **same finished product** and the **same rules** about security, ownership, and how the work is hosted.

> **One thing we will not discuss in this document**
> We do not state any prices, rates, or budgets here. Money is decided after we see your offer. Section 10 explains the *shapes* of payment we are open to. Please put your numbers in your reply, not your assumptions.

What a good reply looks like

A short written proposal that covers: the work you will do, the order you will do it in, how long each part takes, when and how you expect to be paid, the people doing the work, and any questions. Section 11 lists this in full. You do not need slides or a long company profile. We value clarity over polish.

What this section means for you: read everything once, then decide which of the two paths you want to bid on, then write us a plain offer.

SECTION 02

Nah in one minute

Nah is an **electronic-signature platform**. It lets a company send a document, have people sign it legally online, and keep a tamper-proof record of who signed and when.

Nah does not build the cryptographic signing itself. It is a **white-label wrapper** around a signing service called **Firma.dev**. "White-label" means the end customer sees *their own* brand, not ours and not Firma's. Nah owns the customer relationship, the look, the pricing, and the extra tools around the signature.

The selling point is the **pricing model**. Most signing products charge **per seat** — every employee who might ever sign pays a monthly fee. Nah charges **per document** instead, from a prepaid **credit** balance. A business buys credits, spends one credit per document, and pays nothing when nobody is signing. This is friendlier for the many firms whose signing volume is uneven.

1 repo
Four products ship from a single codebase

Credit
Prepaid, pay-per-document — no per-seat fees

BYO brand
Each reseller client keeps their own branding

Nah is also **multi-tenant**. We resell it to many client firms. Each client gets their own branded space, their own credit wallet, their own users, and — through Firma's API — their own isolated signing workspace. One running system serves all of them, kept apart by software.

What this section means for you: the product is a re-brandable, credit-billed signing portal with extra workflow and AI tools layered on top of Firma.dev.

SECTION 03

The four parts of every version

Whichever path you bid on, the **finished** product must contain **four parts**, and they must all live in **one repository** (one codebase).

PART 1

The public website

The marketing landing page and the place where the desktop app is downloaded. Fast, simple, brandable. This part is already live.

PART 2

The web app

The signed-in product in a browser: upload a document, let AI place the fields, choose signers, send, track, and download the finished signed file.

PART 3

The desktop app

The *same* web app, wrapped as one installable program for **Windows 11 and macOS**. It must update itself when a new version exists.

PART 4

The admin panel

Our private control room: manage client firms, credits, sales, free-credit grants, run promotions, see document history, set branding.

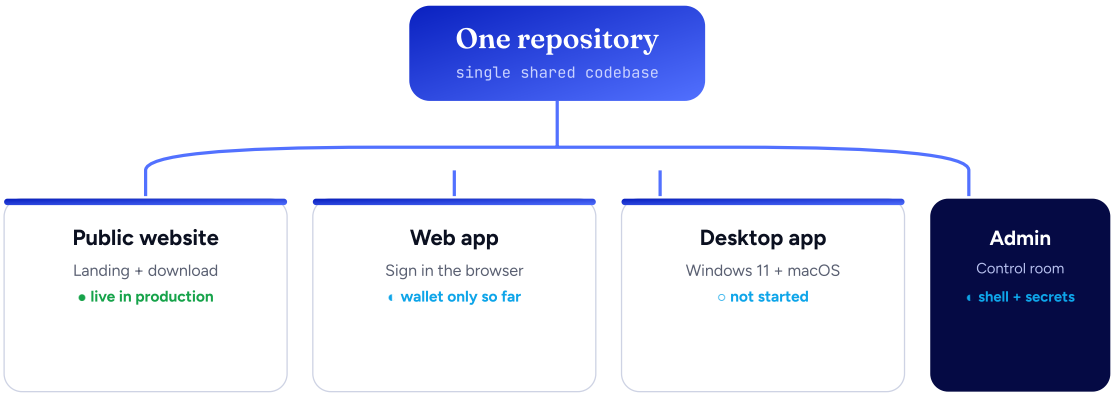

Figure 1. Four products, one codebase. The desktop app is the same web app in a native shell — it must never be a second, separate program.

The web app and the desktop app **share the same screens**. The desktop app is the web app placed inside a small native wrapper so it can open and save files on the user's computer. Building the screen twice is not allowed.

What this section means for you: your bid must deliver all four parts from one repository, and parts 2 and 3 must share one set of screens.

SECTION 04

How the system fits together

There is one **backend** (the server program) and one **frontend** (the screens). The frontend runs in the browser and inside the desktop app. The backend is the only part that is allowed to hold secret keys and talk to outside services.

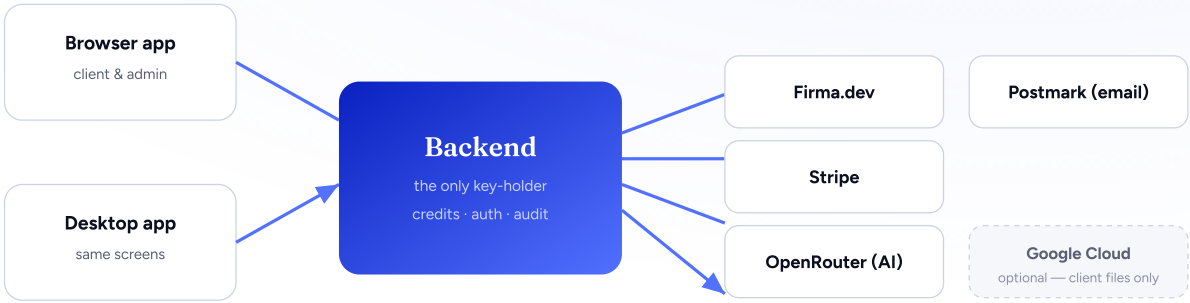

Figure 2. Outside services are only ever reached **through the backend**. The browser and desktop app never hold a Firma, Stripe, AI, or email key.

Three rules drive this picture, and they are not negotiable:

RULE 1

The frontend never holds a privileged key

Only the backend talks to Firma, Stripe, the AI provider, and email. A leaked browser bundle must never expose a usable secret.

RULE 2

Secrets are entered at runtime, never stored in the repository

Every key is typed into the admin panel after launch and stored encrypted in the database. The codebase carries zero real secrets — ever.

RULE 3

No cloud services, with one allowed exception

The system runs on a plain server. The only permitted cloud dependency is **optional** Google Cloud Storage, and only for client-owned files (their templates, documents, uploaded images).

What this section means for you: design every feature so that the only program holding a real secret is the backend, and the only place a human enters a secret is the admin panel.

SECTION 05

Where the repository is today

This is the honest picture. The repository is **real, deployed, and serving traffic in production today**. The foundation is solid; the revenue and signing features are not built yet.

✓ **Already built and live**

You inherit a working skeleton, not a blank page. The hard groundwork — safe deploys, login, encrypted secrets, roles, and the public site — is done and proven in production.

DONE	**Project foundation & safe deployment** Repository, automated checks, production server, one-command releases that swap in instantly and can roll back in one step.
DONE	**Login & sessions** Secure passphrase login with strong password hashing and persistent sessions. (Passkey login is the next step — see Section 6.)
DONE	**Roles & permissions** Three roles from day one: platform admin (us), client admin, client member. Every screen is gated by role.
DONE	**Encrypted secrets, entered through the admin panel** A working admin screen where keys are pasted in, stored with strong encryption, never shown again, and testable with one click. This is the backbone of Rule 2.
PARTIAL	**Admin panel** The shell and the secrets screen exist. Client management, billing, sales, and document history screens are not built.
PARTIAL	**Client web portal** A read-only credit wallet (balance + history) exists. Nothing can add or spend credits yet, and there is no document workflow.
DONE	**Public website + a live status page** The marketing landing page is live, with a public technical "stack" page that shows project health.

In short: **the safe, boring, hard-to-get-right plumbing is finished.** What remains is the visible product that earns money.

What this section means for you: if you bid on this path, you can trust the foundation and spend your time on the features in Section 6.

SECTION 06

What is left to build

This is the scope of the job. We have grouped it into eight work areas. The order below is roughly the order we expect, but your bid may propose a better order.

AREA 1

Passkey login for clients and admins

Modern phone/laptop passkey sign-in, with the existing passphrase as a fallback and email-based recovery. No SMS.

AREA 2

Payments & credit management (Stripe)

Buy credit packs, an optional subscription, automatic top-ups, refunds. Admin tools to grant free credits, run sales and promotions, and adjust balances. Stripe is reached only through the backend.

AREA 3

Firma.dev signing client & per-client isolation

Connect to Firma. Create an isolated Firma workspace for each reseller client so their documents never mix. Send a document, spend one credit, record an audit entry. Decide and build the refund-on-failure rule.

AREA 4

Complex document workflows

Multiple signers in a chosen order, people who only receive a copy (CC), and the ability to change the participants and order while a workflow is in progress.

AREA 5

AI assistance & bring-your-own-key

AI document analysis paid for with credits, through the backend only. A client may plug in their own AI provider key to lower their own cost; that path is discounted, not free.

AREA 6

AI auto-input field mapping

The system reads an uploaded PDF, finds where signatures, dates, and fields belong, and the end client can also prompt the AI in their own words to assign the right signer or recipient to each field. A human confirms before sending.

AREA 7

Desktop app: one installer + auto-update

One bundled installer for Windows 11 and macOS, wrapping the same web screens. It checks for new versions and updates itself. It authenticates the user, shows credits, and can buy services — all through the backend.

AREA 8

Migration helper tools

Tools that help a new client move in from **PandaDoc** and **DocuSign**: read their existing templates with their own API key and show which ones the AI has already mapped. (Hands-on migration *service* is something we offer separately — you are not asked to perform migrations, only to build the tools.)

A reference exists for some of this

We have already built a document editor and an AI-analysis flow for an earlier, single-customer project. We can place a copy of that code into the repository for you to adapt. You do not have to invent the document editor or the AI analysis from nothing — you have to integrate, generalize, and connect them to the multi-client, credit-billed system.

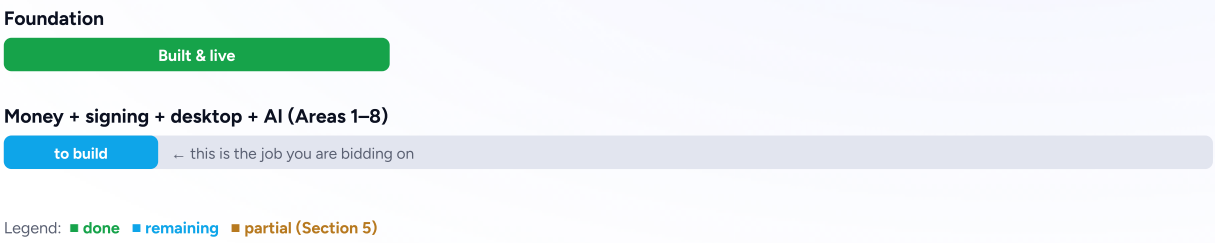

Figure 3. The unglamorous foundation is finished. Your work is the visible, revenue-earning product.

What this section means for you: these eight areas, plus the rules in Sections 4 and 9, are the work. Estimate them in your bid.

SECTION 07

The signing workflow we need

This is the heart of the product. It must support real-world business documents, which are rarely "one person signs one box."

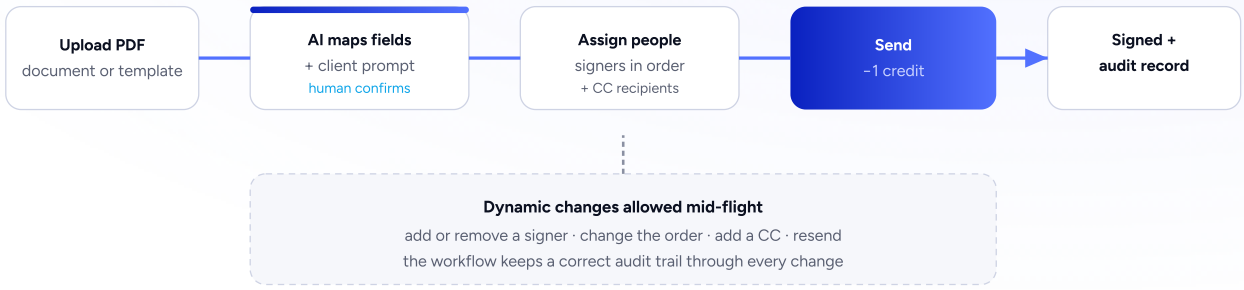

Figure 4. Upload → AI mapping (with a human check) → assign people → send (one credit) → signed file plus a tamper-proof audit record. Participants and order can change while the workflow is running.

The workflow must support, at minimum:

- **Multiple signers**, signed in a chosen order (or all at once).
- **CC recipients** who get a copy but do not sign.
- **Dynamic adjustments**: change who is involved, and in what order, after the workflow has started — without breaking the audit record.
- **AI auto-input**: the system proposes which field belongs to which signer. The end client can also type a plain-language instruction (for example, "the buyer signs page 3, the witness initials every page") and the AI assigns fields accordingly. A person always confirms before anything is sent.
- **Credit accounting**: exactly one credit per document send, with a clear, agreed rule for what happens if Firma fails after the credit is taken.

What this section means for you: plan for editable, multi-party workflows and an AI mapping step with a human confirmation — not a single-signer happy path.

SECTION 08

How you will work with us

This section protects **both sides**. It lets you work freely and fast, and it lets us keep client data and our production system safe. Please read it carefully and price it into your bid; it changes how you work day to day.

We use a **two-server model**.

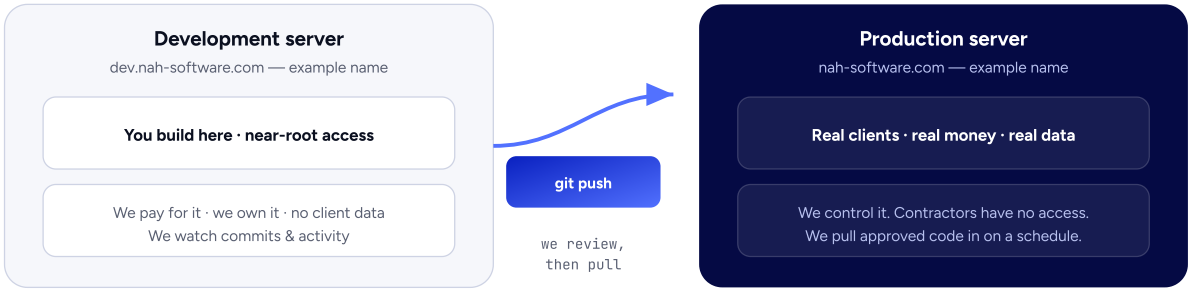

Figure 5. You work on a development server we own and pay for. We pull reviewed code into a separate production server you cannot reach.

The development server — yours to use

- We provision a dedicated **development server (a VPS)** and **we pay for it**.
- You get a working account with enough access to install the tools you need and run the project. You can treat it as your workshop.
- It carries **no real client data** and **no production secrets**. If something goes wrong on it, nothing important is lost.
- One server is fine for development and testing together. Your domain for it could be, for example, `dev.nah-software.com` (illustration only).
- We keep a few protected, owner-only tools on the box (for example, automatic database backups) that your account cannot see or change.

The production server — ours to control

- The real, paying system runs on a **separate** server that **only we** control. Contractors do not have access to it.
- You commit and push your work to the repository. **We review it and pull it into production on our own schedule.** You never deploy to production.
- This means we never have to worry about what happens on the development box. You have freedom there; we keep the real system safe.

The repository — ours to own

- We create the repository on a company account and invite you as a collaborator on a **private** repository.
- This is true even if you bid on the greenfield path. We own and control the repository in every case.

- We track progress through commit history and activity on the development server. We will not wait silently and hope a finished version appears. We expect steady, visible progress.

> ✓ **Why this is good for you**
>
> You get near-total freedom on a powerful machine we pay for, with no fear of breaking anything real. You are never on call for production. The boundary is simple: build freely on dev, we carry the risk on prod.

The steps below are the working loop:

1. **We set up the dev server, domain, and repository**
   Before you start, we make the project easy to move between domains and environments, so your work is portable from day one.
2. **You build on the dev server and commit often**
   Small, frequent commits with clear messages. The project's task runner (see Section 9) is how you build, test, and deploy to the dev environment.
3. **You push to the private repository**
   We watch the commit history and the dev activity, so progress is always visible to both sides.
4. **We review and pull into production on our schedule**
   We may also contribute our own work on our own development copy. Production is always our responsibility, never yours.

What this section means for you: plan to work on a server we own, commit and push constantly, and never expect production access. Price the workflow, not just the features.

SECTION 09

Technology and ground rules

We want to keep the technology the **same as what we already use**, because we have to maintain this for years after you finish.

Backend

Go — one compact, fast server program. SQLite to start. Plain and dependency-light by choice.

Frontend

SvelteKit + TypeScript, with Tailwind CSS and a copy-in component set. One screen set shared by web and desktop.

Desktop

Electron, packaged with electron-builder. Windows 11 first, macOS to follow. Self-updating.

Outside services

Firma.dev (signing), Stripe (payments), an AI provider via OpenRouter, Postmark (email) — all backend-only.

Server

A plain Linux VPS behind the Apache web server. Pull-mode deploys that swap in atomically and roll back in one step.

Tooling

just / justfile for every build, test, and deploy command. Playwright + unit tests for quality.

You may propose a different stack — with a condition

If you are confident you can meet every requirement with a different technology, you may include that proposal in your bid. But we **reserve the right to reject** a stack we do not want to inherit and maintain. On this "complete" path the code already exists in our stack, so a stack change effectively becomes the greenfield path. Whatever the stack, **just / justfile must drive the build and deploy commands** so the project is easy for us to take over.

These rules cannot be broken, on either path:

RULE	**No real secrets in the repository, ever** Every key is entered at runtime in the admin panel and stored encrypted. Placeholders only in the codebase.
RULE	**Passkey authentication for clients and admins** Modern passkey sign-in is the target for everyone, with passphrase fallback and email recovery.
RULE	**One repository, four parts** Website, web app, desktop app, and admin all live and ship together from one codebase.

RULE

No cloud except optional Google Cloud Storage for client files

And only for client-owned content. Nothing else depends on a managed cloud service.

RULE

Safe by default

No destructive database actions, atomic deploys, backups before migrations, small and readable files and functions.

RULE

Portable

The product must move cleanly between domains and environments. No hard-coded hostnames or one-off setup.

What this section means for you: match our stack unless you make a strong case to change it, and design every feature to obey these rules from the first commit.

SECTION 10

How we can pay

We will not state numbers here. We will say clearly that we are **flexible** and open to several payment shapes. Tell us which one you want in your bid.

OPTION A

By the hour

You bill for time invested. Good when scope is uncertain or exploratory.

OPTION B

By feature / milestone

A fixed amount per agreed feature or work area, paid as each is delivered and accepted.

OPTION C

Whole project

One agreed amount for the entire finished product, paid across milestones.

We are happy to pay across **intervals tied to milestones** — for example, a portion when a work area is delivered and accepted. We can also mix these (for example, hourly for discovery, then per-feature for delivery).

We may also contribute

We can keep our own development copy and add our own time and skills to the project alongside you, on any of the four parts. You are a partner here, not a vendor we hand a wall to.

What this section means for you: choose the payment shape that fits how you want to work, and propose the milestone points where you expect to be paid.

SECTION 11

What we need back from you

Send a short, written proposal. Please cover every point below. Clarity matters more than length or design.

YOUR BID

The contents of a strong reply

1. **Which path** you are bidding on — "complete" (this document) or "greenfield".
2. **Your plan** — the work areas you will deliver and the order you will deliver them in.
3. **Milestones** — how you break the work into deliverable, acceptable pieces.
4. **Timeline** — a realistic estimate for each milestone and for the whole job.
5. **Payment** — which shape from Section 10, the amounts, and at which milestones you expect to be paid.
6. **Stack** — confirm you will use our stack, or propose a different one and explain why (we may reject it).
7. **Team** — who will do the work, and their relevant experience.
8. **Questions & assumptions** — anything unclear, and the assumptions your estimate depends on.

We will read every serious reply and respond. A counter-proposal that improves on this plan is welcome — that is the point of asking.

What this section means for you: answer these eight points and you have a complete bid.

SECTION 12

Glossary

Plain definitions of the terms used in this document.

Electronic signature `e-signature`	A legally valid way to sign a document online instead of on paper.
Firma.dev	The outside service that performs the actual cryptographic signing. Nah wraps it; it never touches our customers directly.
White-label	A product sold under someone else's brand. Each Nah client shows their own name and look, not ours.
Credit	A prepaid unit of value. One document send costs one credit. Clients buy credits in advance.
Multi-tenant	One running system that serves many client firms while keeping each one's data and branding separate.
Backend	The server program. It holds the secrets and talks to outside services. Users never see it directly.
Frontend	The screens a user sees and clicks, in the browser or the desktop app.
Passkey	A modern, password-free login using the security built into a phone or laptop.
VPS	Virtual Private Server — a rented computer in a data center that we control.
Repository `repo`	The place where all the project's code is stored and its history tracked.
Commit / push	Saving a unit of work to the project history, then sending it to the shared repository.
Atomic deploy	Releasing new code in a way that switches over instantly and can be undone in one step.
Pull-mode	The server fetches approved code itself. Nobody pushes files onto production by hand.
CC recipient	Someone who receives a copy of a document but does not need to sign it.
Audit record	A tamper-proof log of who did what and when — essential for legal signatures.
BYOK	"Bring your own key." A client supplies their own AI provider key to lower their own cost.
just / justfile	A simple task runner. One short command runs the right build, test, or deploy steps.
Greenfield	Building fresh from nothing, instead of continuing existing code.

PandaDoc / DocuSign	Competing signature products. New clients may want to move their templates over from them.

Ready to bid?

Send your written proposal covering the eight points in Section 11. We read every serious reply and respond. Honest questions and counter-proposals are welcome — a better plan than ours is exactly what we are hoping to receive.

PROJECT	REFERENCE	REACH US
Nah — Complete track	NAH-RFP-COMPLETE	meiuxmeiux.com/contact