PORTFOLIO · 2026

Software Engineer

Novel Bafagih

Three projects, told by what changed: the problem, what I built, and what it did.

AgentFlow · multimodal.dev · TIA



novelbafagih09@gmail.com · www.novelbafagih.com

ABOUT ME

Hi, I'm Novel

Software engineer who builds products end to end, from design to backend. Founding frontend engineer on AgentFlow, an AI document platform for banks and credit unions, which I took from its first commit to more than a hundred client users a week. I use AI coding agents to move fast, and I test and own everything that ships.

Jakarta, Indonesia.

GitHub · LinkedIn · Upwork · www.novelbafagih.com

SKILLS

Languages

TypeScript · JavaScript · Python · Kotlin · Rust · SQL · Java · PHP

Frontend

React · Next.js · Tailwind CSS · Storybook · Vue · Angular · HTML · CSS

Backend and data

FastAPI · Supabase · PostgreSQL · SQLite · Django REST Framework · Sanity · Turso

Mobile and desktop

Android · Jetpack Compose · Tauri

AI

LLM product features · AI coding agents · MCP

Tools

Git · Playwright · PostHog · Figma · Vercel · Docker · Google Cloud

SELECTED WORK

Three projects

01

AgentFlow

AI platform · 2024 – 2026

The AI platform banks and credit unions use to take a document from upload to decision. I wrote its frontend's first commit when it had no users; two years on, more than a hundred client people use it every week.

0 → 100+

client people using it every week

02

multimodal.dev

Marketing site · 2026

I moved the company website off Webflow to Next.js and Sanity, matched it page for page, launched it, then made it faster on phones.

73 → 85

median mobile Lighthouse score

03

TIA

Web app · 2023 – 2024

A digital questionnaire that turns an orthopedic patient's answers into a picture a doctor can read before the consultation.

100+

questions with branching medical logic

CASE STUDY 01

AgentFlow

The AI platform banks and credit unions use to take a document from upload to decision. I wrote its frontend's first commit when it had no users; two years on, more than a hundred client people use it every week.

CLIENT

Multimodal

ROLE

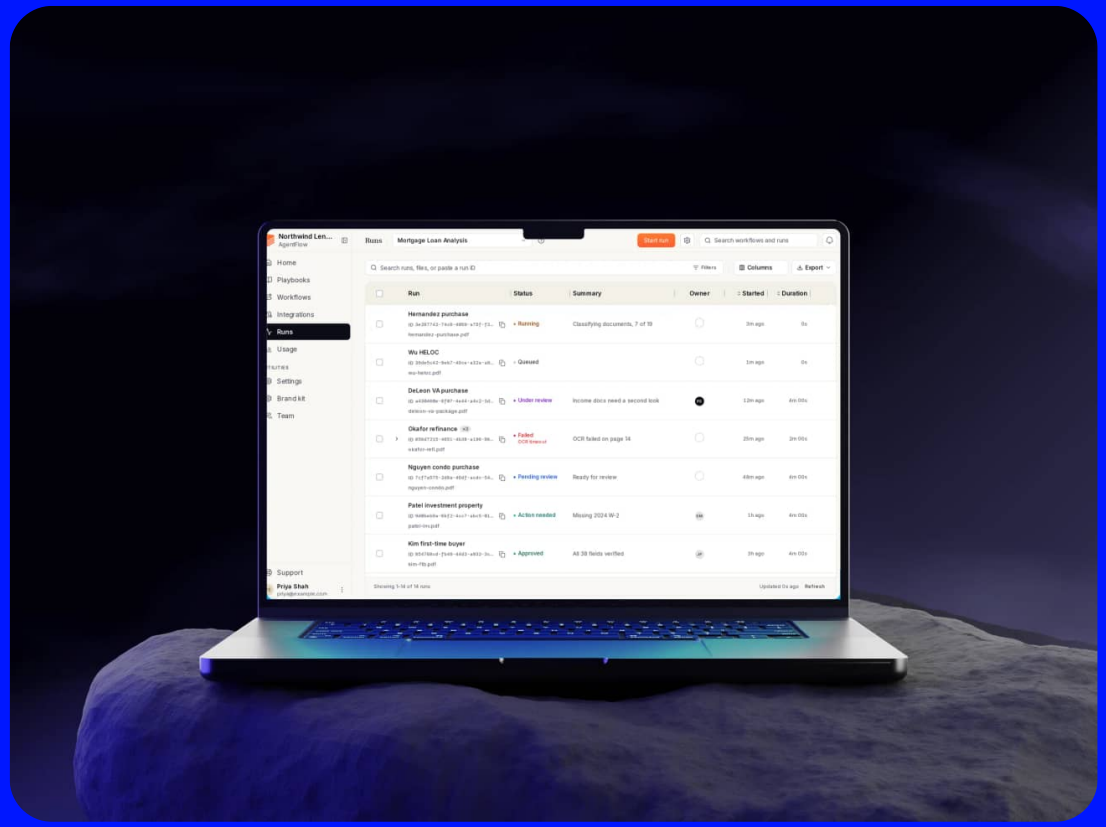
Founding frontend engineer

WHEN

Aug 2024 – Oct 2026

WHAT

AI platform, web



0 → 100+

client people using it every week

Read the full case study on the site ›

The project

AgentFlow takes a business document, a loan application, a balance sheet, an invoice, through AI extraction, human review and a decision. Banks, credit unions and other financial firms run their daily work on it. Its largest client is a bank that rolled it out to its lending teams.

Reviewers live in two screens: the Runs table, where they find the document they need, and the review page, where they check what the AI read. Admins watch usage and manage their team.

BUILT WITH

Next.js · React · TypeScript · Tailwind · Storybook · Python / FastAPI · PostHog · Playwright

My part

I wrote the first commit of AgentFlow's frontend in August 2024: the Next.js app, sign-in, the component kit and the data layer. Then I built its screens one by one as the product found its shape: review, configure, monitor, ingest, and a view for each kind of AI agent.

Over two years I built the screens those people work in every day: workflows, runs, document review, usage, team management, and the design system that ties them together. When a feature needed backend work I wrote that too, and I was one of the team's main frontend reviewers.

> HOW IT WAS BUILT
>
> Most of this ran as parallel AI coding sessions that I directed: one worktree and one agent per lane, each held to written rules, tests and a review before anything merged. AI made the work fast; I read, tested and own what shipped.

The full case study ›

From zero to a hundred people a week

THE PROBLEM

In August 2024 AgentFlow had no frontend and no users. Whoever came would judge it by the screens they worked in all day.

WHAT I DID

- Built those screens: the Runs table people find their documents in, the review page where they check the AI, Home, Usage and team settings.
- Kept them quick and clear as use grew: one filter bar, search that answers as you type, one design system on every page.
- Set up the product analytics, then used what people actually did to decide what to build next.

RESULT

In the summer of 2026 a bank rolled AgentFlow out to its business lending teams, and since then more than a hundred client people have used it every week.

One design system

THE PROBLEM

Home, Runs and Usage looked like three different products. A count of the code found 1,689 hand-typed colours (348 distinct), 33 separate status-to-colour maps and 417 font sizes typed as pixels.

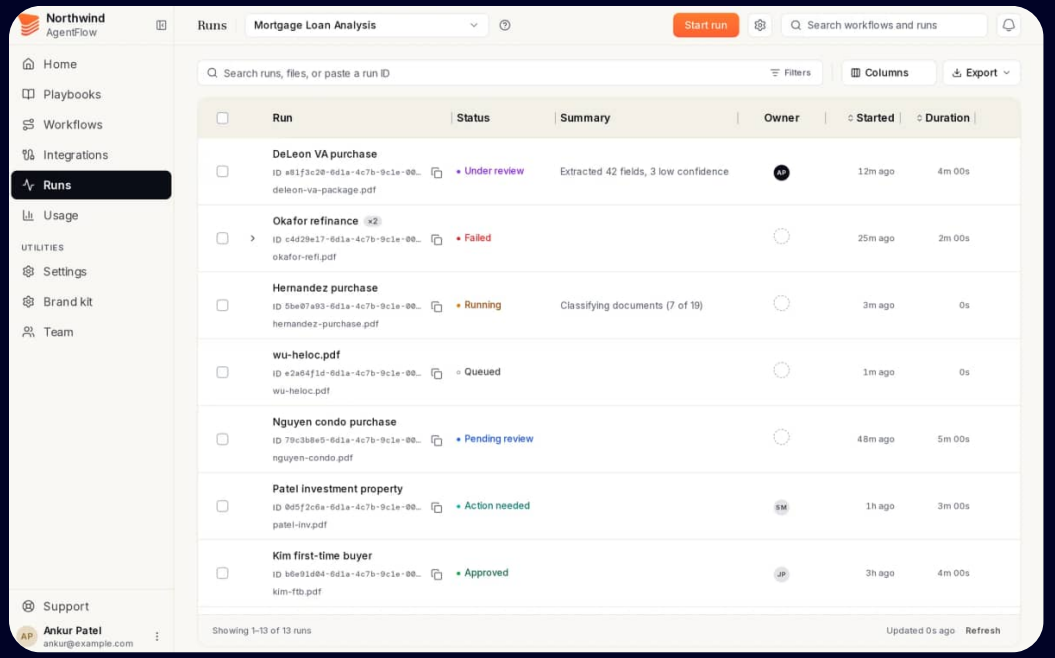
WHAT I DID

- Audited every v3 screen with a Playwright capture rig: 166 verified findings, 141 more checked and thrown out.
- Wrote the spec and made the calls: one token set, one status pill and one set of tones, one type ramp. Kept the brand orange even where it costs contrast, on purpose.
- Shipped it as a stack of nine pull requests, with a pixel-diff check proving the colour codemod moved nothing on screen.
- Added a lint rule, no-off-canon-color, so the next one-off colour fails the build.

> ## RESULT
>
> The whole stack merged on 22 September 2026. Every page now reads as one product, and new screens start from shared parts instead of copies.

33 → 1

status-to-colour maps



The Runs page on the new tokens. Demo data.

Finding work, fast

THE PROBLEM

Reviewers search all day, and the tables fought them: slow search, sort controls that only showed on hover, statuses that did not match between the table and its filter.

WHAT I DID

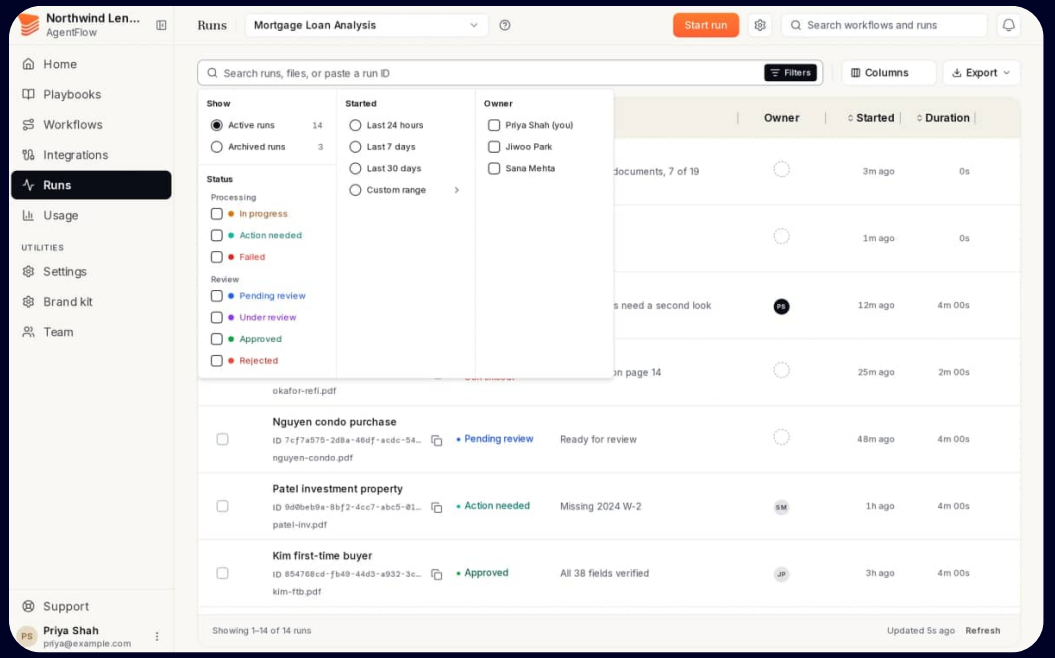
- One filter bar shared by every table: chips for what is set, one search field.
- A status filter that can tick a Processing and a Review status together, which took a backend change as well.
- Sort arrows visible at rest on every table, and one status, one colour, everywhere.
- Workflow search that filters in the browser instead of asking the server on every key.

RESULT

Typing “forum cu” used to send 10 requests and leave the list unfiltered; now it sends none and matches in 23 ms. Filter use went from about one in ten users to about seven in ten.

~10% → ~70%

of users filter their runs



The shared filter panel on Runs. Demo data.

CASE STUDY 02

multimodal.dev

I moved the company website off Webflow to Next.js and Sanity, matched it page for page, launched it, then made it faster on phones.

CLIENT

Multimodal

ROLE

Frontend engineer, migration and speed

WHEN

Sep – Oct 2026

WHAT

Marketing site, web



73 → 85

median mobile Lighthouse score

Read the full case study on the site ›

The project

multimodal.dev is where banks and credit unions first meet Multimodal: solution pages, the product tour, the newsletter, the demo form. It had grown to 485 pages on Webflow, and marketing needed an engineer for every change.

The new site runs on Next.js with content in Sanity, so the team edits pages themselves.

BUILT WITH

Next.js · Sanity · Vercel · GSAP · Lighthouse · Beehiiv

My part

The first build of the new site matched none of the old pages. I ran the effort that made it match, fixed the gaps inside the existing code, took it through launch, and then ran the speed work.

I also wrote the site's design system from the pages that shipped, and the handbook marketing runs it from.

> HOW IT WAS BUILT
>
> Parity and speed both ran as rounds of parallel AI agents: read-only auditors first, then small build lanes, each change measured before it merged. I set the rules, read the diffs and owned the result.

The full case study ›

Visit multimodal.dev ›

Match it, page for page

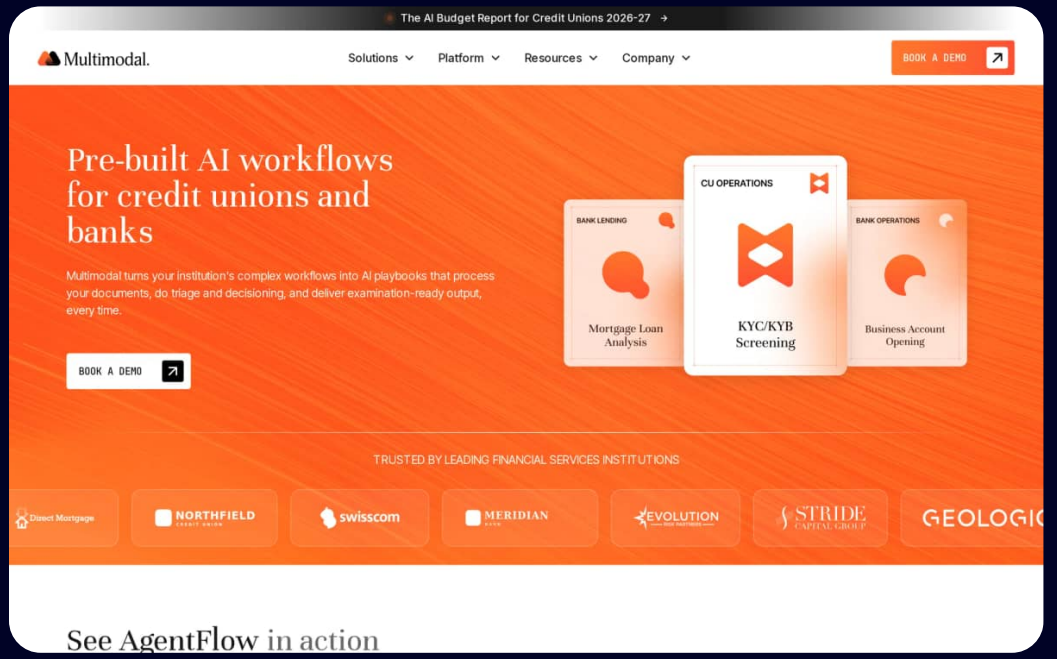
THE PROBLEM

The new site had missing text, rough animation, wrong button colours and broken blog layouts. On a comparison rig it matched 0 of 20 page widths; the closest still differed by 19.9% of its pixels.

WHAT I DID

- Built a parity rig that compares each page against the live Webflow site at several widths, using Webflow's own export of 189 pages.
- Fixed in rounds: GSAP animation, collection pages, 23 broken differences, and old security copy that 8 of 9 solution pages still carried.
- Moved newsletter sign-up straight to Beehiiv, so nothing is lost when the old Zapier link is switched off.

> **RESULT**
>
> The site launched matching the old one, with no lost pages and no broken sign-ups.



The home page as launched.

Launch day

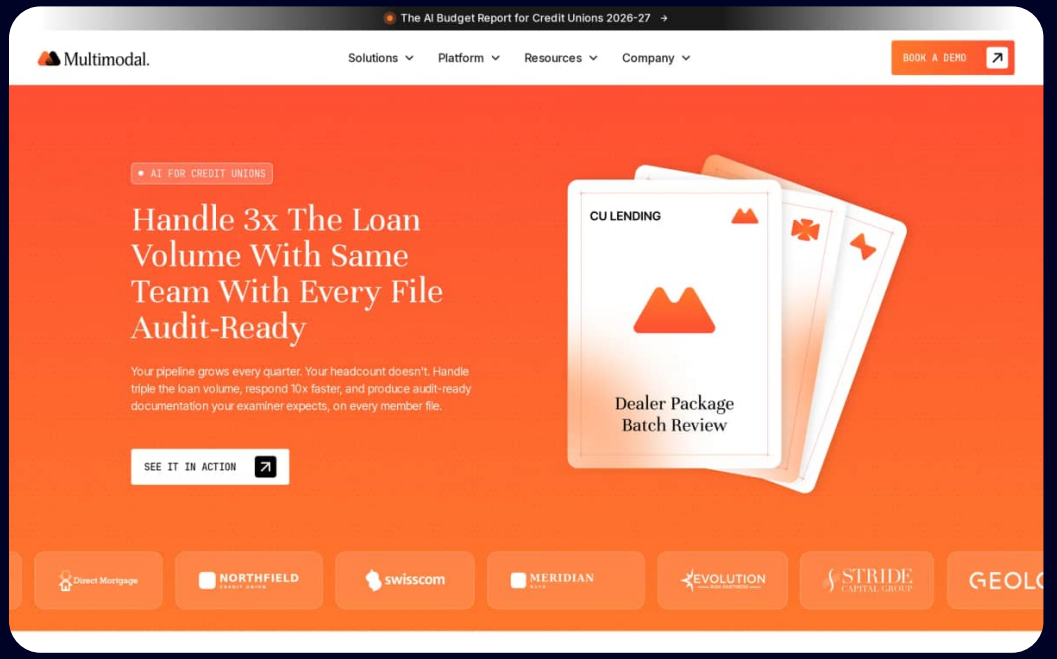
THE PROBLEM

Switching a 485-page site in one afternoon, without losing analytics or a single demo request.

WHAT I DID

- Planned the switch so rollback was one DNS edit.
- Once DNS moved on 21 October 2026, set the Tag Manager and analytics settings on Vercel and checked in a browser that page views fired.

> ## RESULT
>
> The site switched at about 14:30 UTC and stayed up. Marketing now edits in Sanity without an engineer.



One of the solution pages.

Faster on phones

THE PROBLEM

Scores were uneven across the site, and the hero pages were heavy on mobile.

WHAT I DID

- Ran seven read-only audit agents, then four build lanes, with one rule: no visible design change without the designer's sign-off.
- Measured through an HTTP/2 front, after finding the local server's HTTP/1.1 made good changes look worse.

RESULT

Across 27 page types, measured over three alternating runs: largest paint 4.2 s to 3.2 s, pages 961 KB to 578 KB, and pages scoring 90 or more went from 3 to 8.



CASE STUDY 03

TIA

The Intelligent Anamnesis

A digital questionnaire that turns an orthopedic patient's answers into a picture a doctor can read before the consultation.

CLIENT

TUM Venture Labs × TUM Sports Orthopedics

ROLE

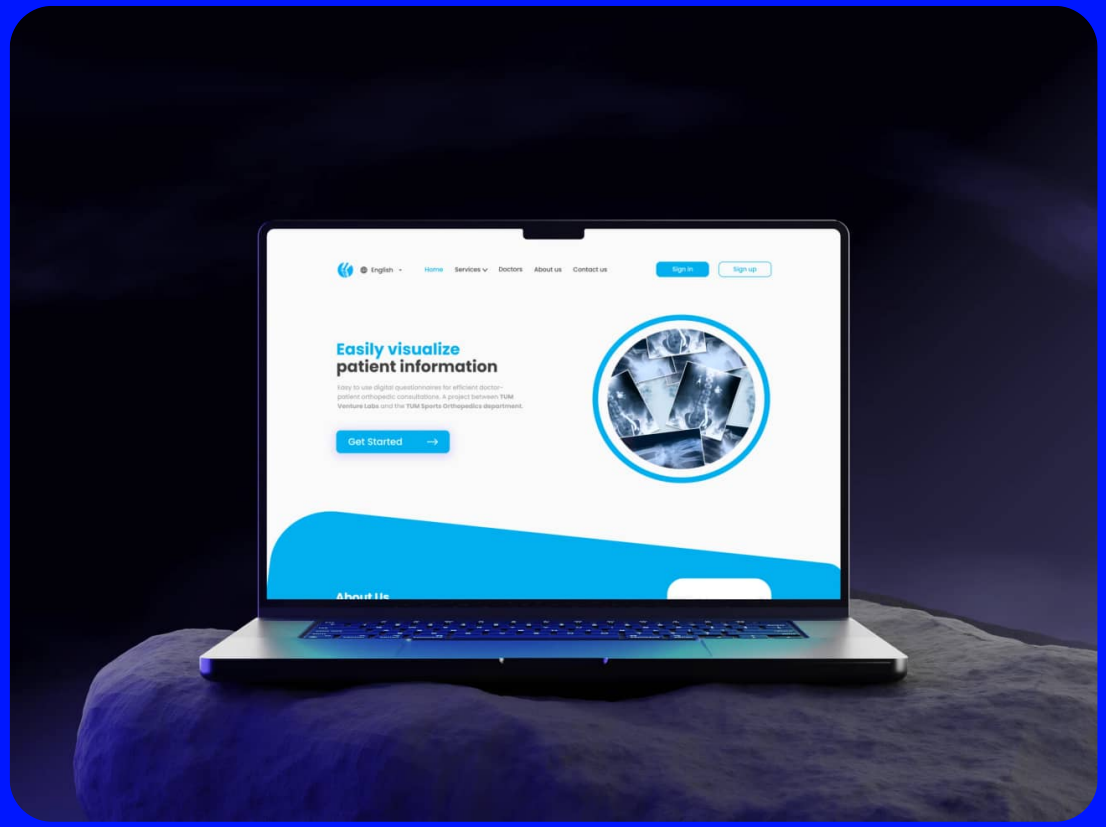
Web developer and product designer

WHEN

2023 – 2024

WHAT

Web app, web



100+

questions with branching medical logic

Read the full case study on the site ›

The project

Before an orthopedic consultation, doctors need the patient's history: where it hurts, since when, what has been tried. TIA collects that through a smart questionnaire and lays it out so the doctor can take it in at a glance.

It was a pilot between TUM Venture Labs and the Sports Orthopedics department at the Technical University of Munich, built for 500+ patients and their doctors.

BUILT WITH

React · Material UI · Supabase · PostgreSQL · Figma

My part

I was the only developer and also designed the product, working with the Master's and PhD students who led the research.

I built the questionnaire's medical logic, more than 100 questions and answers that branch on what the patient says, and the backend that stores it.

The full case study ›

One question at a time

THE PROBLEM

How do you collect a full orthopedic history from patients without a form that scares them off, and still hand the doctor something they can use in the room?

WHAT I DID

- Broke the history into single questions with a clear next step, so a patient only ever sees one thing to answer.
- Encoded the medical logic: follow-up questions appear only when an earlier answer calls for them.

> ## RESULT
>
> More than 100 questions and answers, with the branching the doctors asked for, in a flow a patient can finish on their own.

100+

questions with branching medical logic



One question per step.

Designed to ship

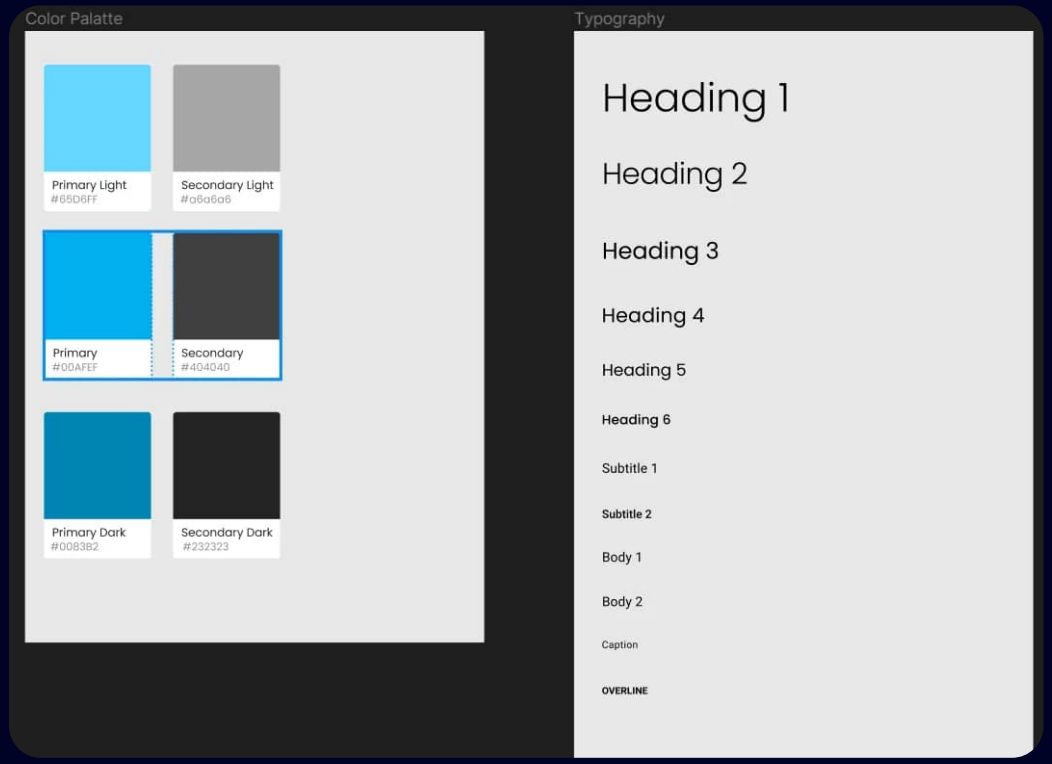
THE PROBLEM

A pilot lives or dies on getting in front of patients, not on a bespoke look.

WHAT I DID

- Designed in Figma on Material UI's components, with a small palette and type scale of our own.
- Kept the interface as plain as possible so the team could test with patients early.

> ## RESULT
>
> The design went from Figma to working screens fast, and stayed consistent because it was built from one component set.



The palette and type scale.

Built to be hosted by the hospital

THE PROBLEM

The client wanted to be able to run TIA on its own servers.

WHAT I DID

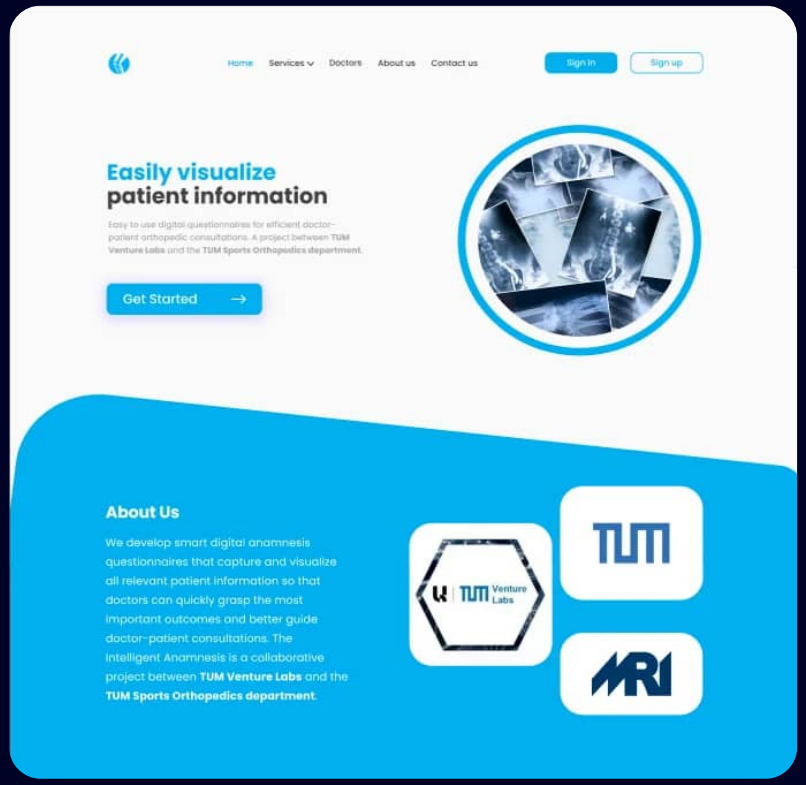
- Built the frontend in React with a clean architecture, so screens, logic and data stay apart.
- Chose Supabase on PostgreSQL for the backend because it is open source and can be self-hosted, and it kept development fast.

RESULT

A working pilot, built solo over seven months, ready for 500+ patients and their doctors.

500+

patients the pilot was built for



The landing page.

THANKS FOR READING

Let’s talk

Every project here is told in full on my site, with more work besides. If one of them looks like the problem you have, write to me.

EMAIL	novelbafagih09@gmail.com
WEBSITE	www.novelbafagih.com
GITHUB	github.com/kucingapi
LINKEDIN	linkedin.com/in/novel-bafagih
UPWORK	Upwork profile