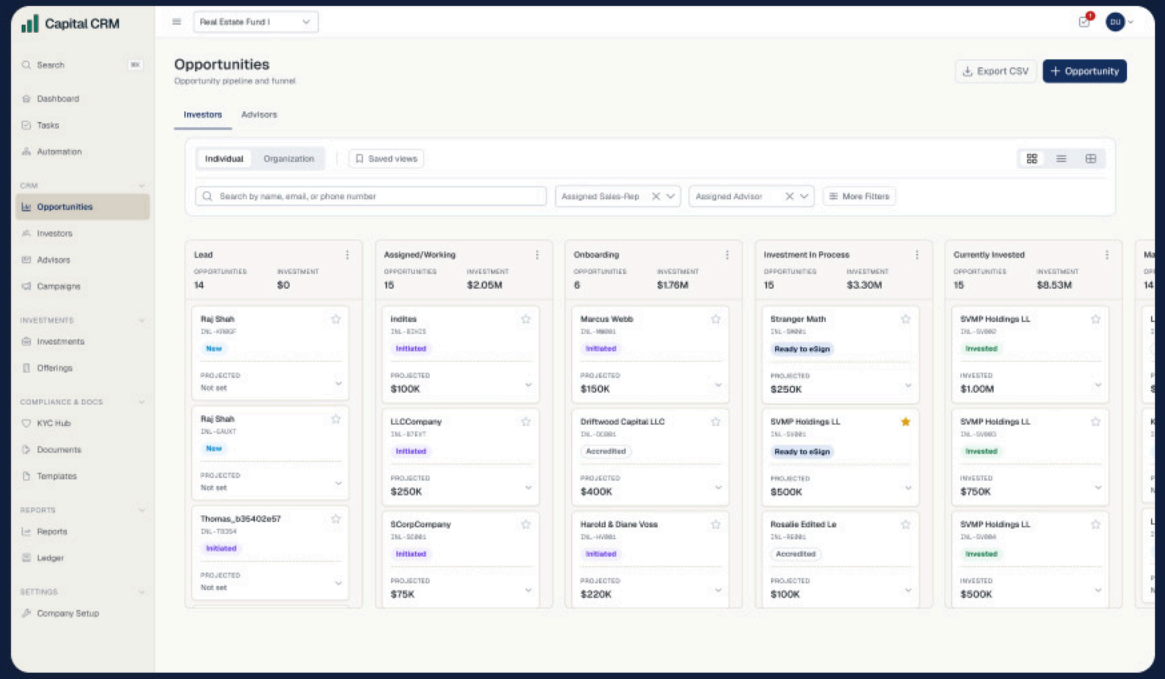
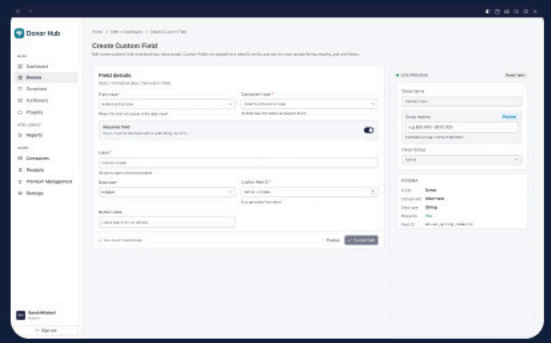
Fintech subsidiary of a lending and investment firm

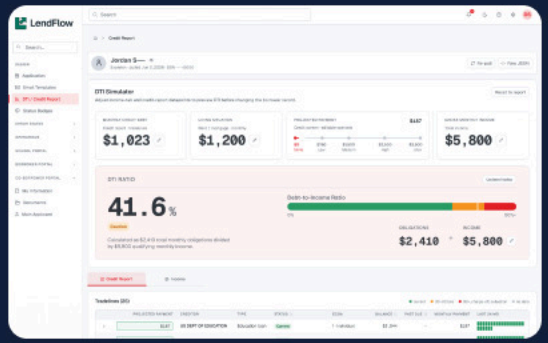
One foundation. Four brand themes.

How I chose and built one PrimeNG design system, then got five product teams onto it. Two of them said no first.

Raj Shah · Senior Product Designer · Apr–July 2026







The results, before the story.

~1 hr

to generate a brand theme, down from ~2 months

5 of 5

product teams on the system, both holdouts included

4

brand themes on one PrimeNG foundation

40+

engineers building on it across five teams

White-labeling used to mean detaching and restyling Angular Material by hand, about two months per request. Now Claude Code makes the token edits, the theme generator outputs a preset, the front-end lead and I review it, and a CI lint rule keeps hard-coded values out.

Theme time covers generation only, not QA or deployment. Sources and caveats for every number are on slide 15.

Two designers. Five product teams. 40+ engineers

The brief from engineering leadership was easy to say and hard to do: the same components in every product, with each product's brand on top.

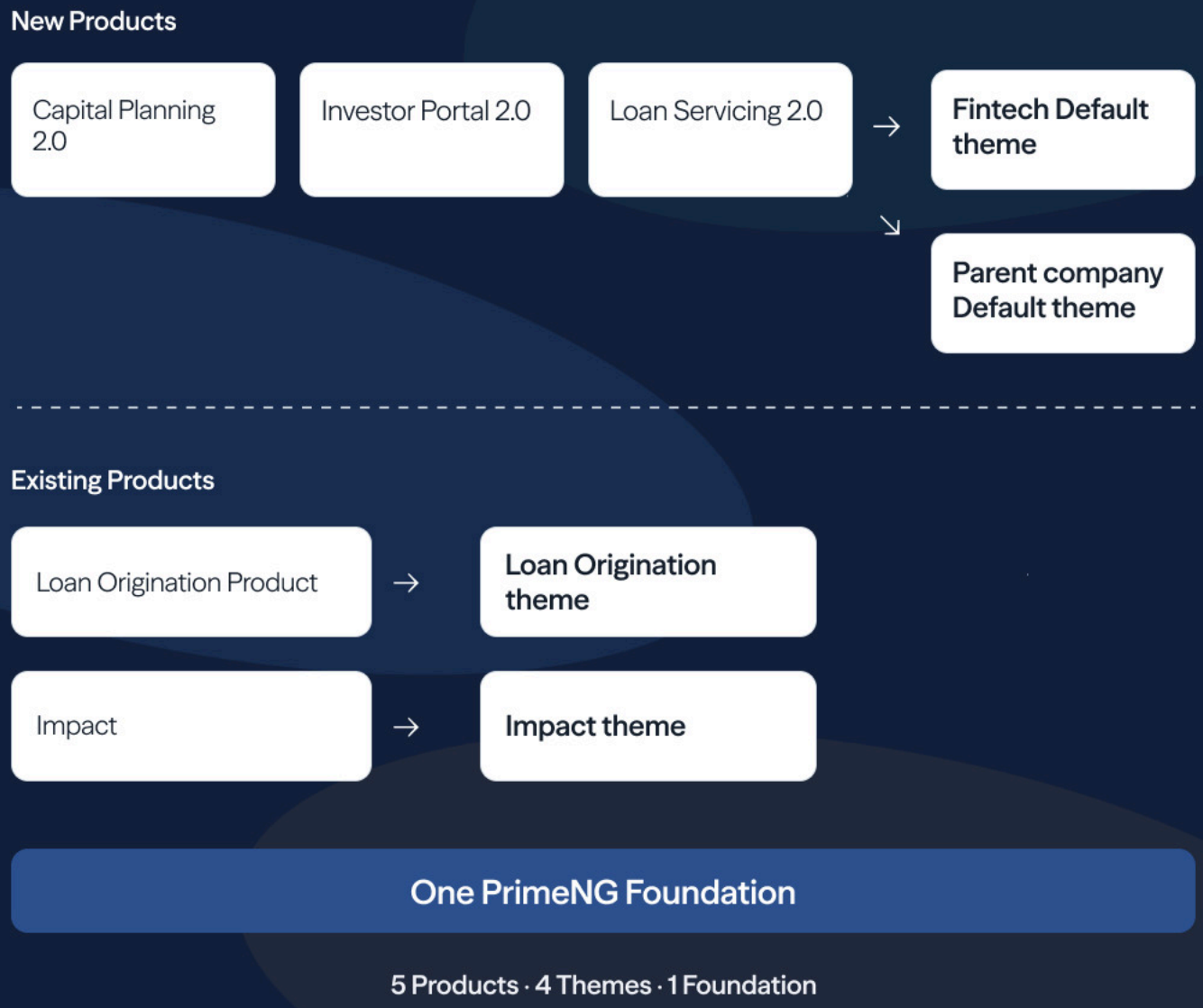
Roles

Me (Senior Designer):
System strategy and design, engineering partnership, and adoption across all five teams

Mid-level designer:
Product design for 2 of 5 product teams

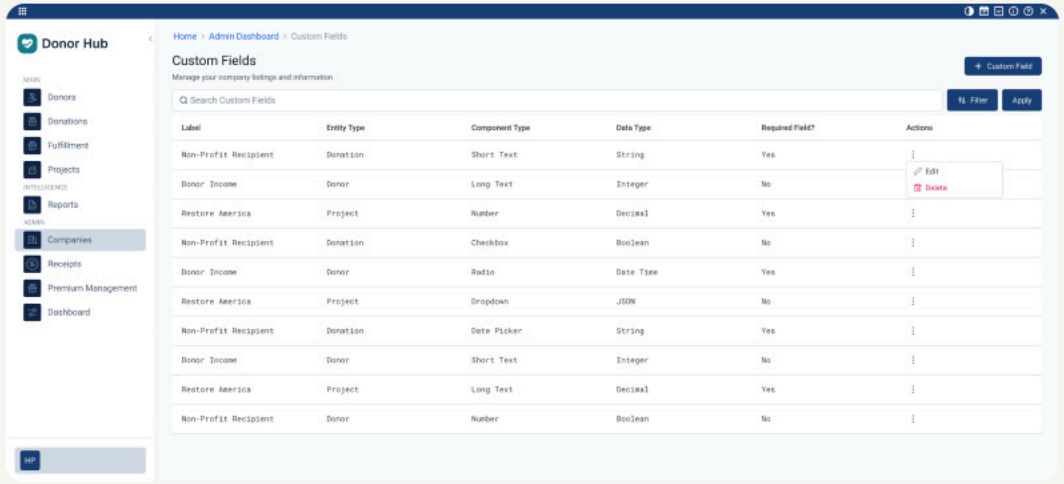
Front-end lead:
Storybook and theme integration

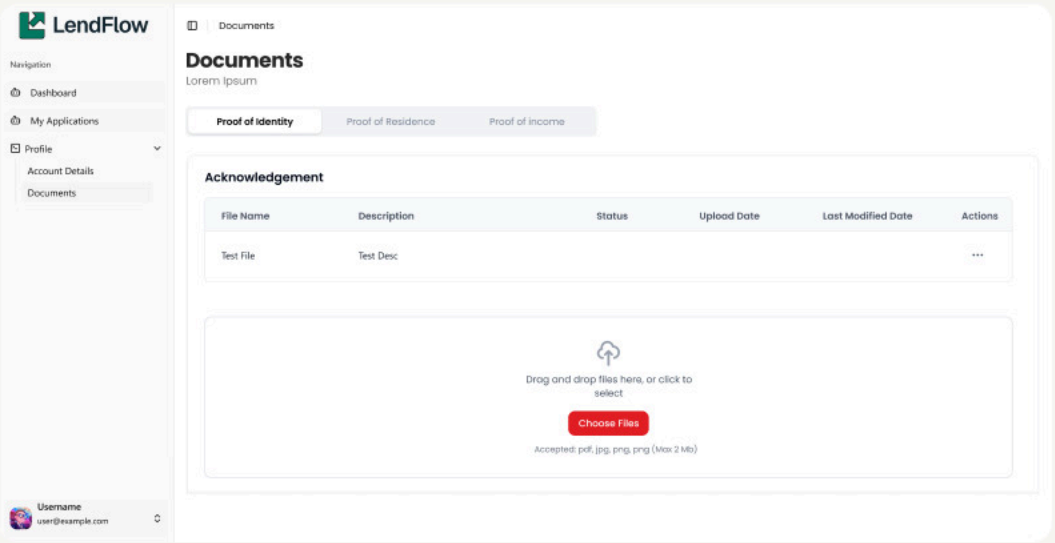
CTO and Engineering Manager:
Sponsors



Every new brand restarted the UI work.

Each product ran its own Angular Material build. White-labeling one meant detaching and restyling base styles by hand, roughly two months per request, squeezed in around feature work.





WHO WAS ASKING

Prospective customers wanting white-labeled instances

Parent company, wanting its own brand on lending products

Products moving from pilots to market

Shared controls drifted.
Product workflows were supposed to differ.

Engineering leaders already knew the UI was inconsistent. I audited behavior, states, and styling to find out how far it went, and which differences were bugs versus product decisions.

Control	LoanOrg Product Notes	Impact Notes	VERDICT
Button	Corner radii of 8 and 10px	Corner radii of 4 and 6px	Inconsistent with each other and within product
Link	14px Poppins medium, both blue and primary color links	16px Roboto Regular, only blue colored link	Inconsistent with each other and within product
Input	Corner radii of 8 and 10px, placeholder text 14 and 16px	Corner radii of 6 and 8px, placeholder text 13 and 16px	Inconsistent with each other and within product
Switchers	Full component width	Auto-full width of container	Inconsistent with each other
Pagination	Controls on both sides of the data table	Controls on only right side of the data table	Completely different components
Data table	Striped header row with semibold text	Plain header row with medium/ regular text	Inconsistent with each other
Dialogs	Only one width	Multiple widths	Inconsistent with each other
Avatars	Used in product	Not used in product	Inconsistent with each other

What stayed Product-specific

Workflows and domain features stayed with each product.

Controls and their states went into the system, where they could stop drifting.

Re-theming Angular Material was the default. I argued against it.

Criteria: white-label effort per brand, component coverage including data viz, and migration cost.

Option	What it meant	Verdict
Angular Material (re-theme)	No migration. Also no escape from undoing base styles by hand for every brand	Rejected
NG-Zorro	Same problem in a different design language: base styles to undo for each brand	Rejected
Taiga UI	Too few components, and thinnest exactly where we needed depth: data visualization	Rejected
Spartan (headless)	No base styles to undo, and no styles at all. Every brand needs a visual layer coded from scratch	Rejected
PrimeNG	Token-based theme presets, broad coverage including charts, plus a Figma kit and theme generator	**Chosen**

Chose

PrimeNG as the shared foundation for all five products.

Gave Up

Zero migration for Loan Origination and Impact, and independence from one vendor's roadmap (see decision 5).

Would have flipped if

No new products on the roadmap and white-label demand stayed theoretical. Then re-theming Angular Material is the cheaper, correct call.

Deep dive → Appendix A1

Brands own color and type. The system owns behavior.

Three token tiers kept brand above the component layer. A brand can pick its own blue. It can't pick its own focus state.

	Primitive	Semantic	Component	Rendered
Default	blue.500 #163060	primary.500 → {blue.500}	button.primary.background → {primary.color}	Save changes
Impact	blue.500 #1C213F	primary.500 → {blue.500}	button.primary.background → {primary.color}	Save changes
Loan Org	red.500 #e31f26	primary.500 → {red.500}	button.primary.background → {primary.color}	Save changes

Chose

Primitive → semantic → component tokens. Brands change only the first two tiers.

Gave Up

Component-level brand freedom. No brand gets its own focus ring or error state, however strongly it feels about it.

Pattern

Followed the standard three-tier model on purpose. It maps straight onto PrimeNG's presets, so there was no custom structure to learn.

Deep dive → Appendix A2, A3

Build-time themes: simpler now, a ceiling later.

Each brand ships as its own product. So each build carries one theme, not every theme just in case.

Option	What it meant	Verdict
Separate component sets per brand	Maximum freedom, which is how we got here in the first place	Rejected
Runtime theme switching	Useful if an app changes brands while someone is using it. None did. Every build would still carry, and need testing against, every theme	Rejected
One preset per product build	One theme per build, a smaller testing surface, and a new brand is a new preset	**Chosen**

Chose

Build-time presets.

Gave Up

Live or self-serve branding. Every new white-label customer means a new build.

Would have flipped if

White-label customers multiplied or wanted to brand the product themselves. At that volume, runtime theming earns its complexity.

Deep dive → Appendix A4

The fourth theme was the real test of the pipeline.

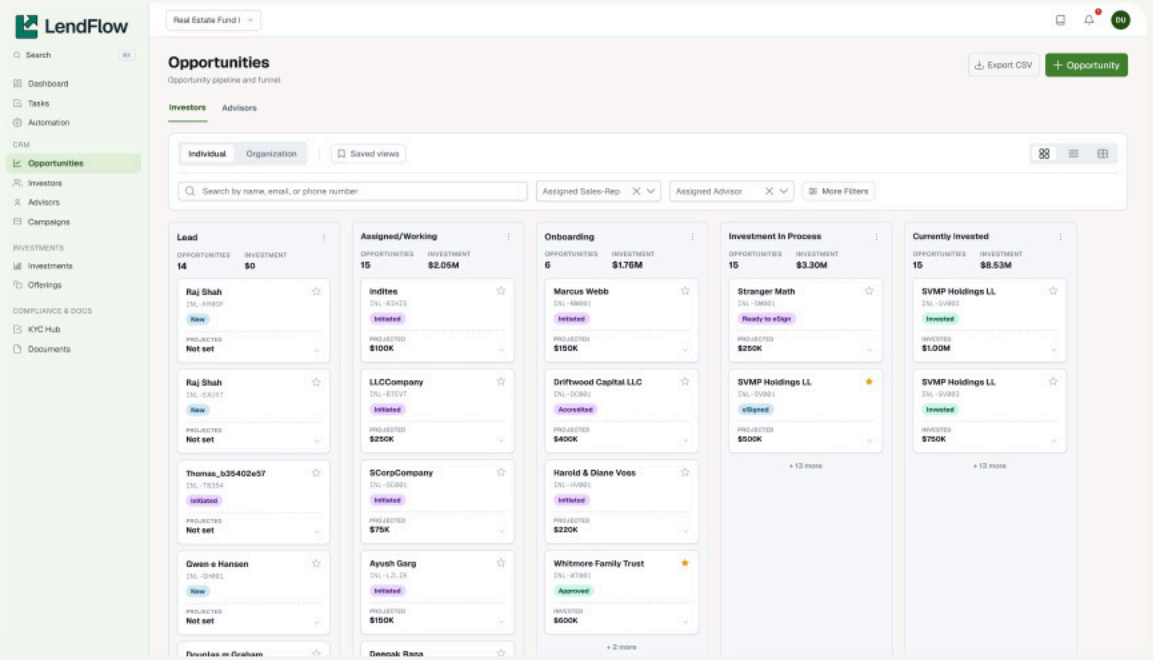
When the parent company wanted its own brand on Loan Servicing 2.0, the theme took about an hour. Not a sprint. An hour.

Figma Kit
Set brand primitives and type in the PrimeUI kit

Tokens → JSON
Design Tokens Manager and Claude Code apply the brand guide

Theme Generator
PrimeUI Theme Generator outputs a PrimeNG preset

Storybook
The front-end lead and I review every shared control



The hour covers generating the theme, not QA, assets, or deployment. Claude Code made the token edits. I chose the tokens and checked its work.

Deep dive → Appendix A5

Three teams adopted on day one. Two said no.

The new products had nothing to migrate, so yes was easy. Loan Origination and Impact had live features, committed roadmaps, and enough autonomy to make no a realistic answer.

Product	Starting Point	Theme	Response
Capital Planning 2.0	New	Default	Adopted at launch
Investor Portal 2.0	New	Default	Adopted at launch
Loan Servicing 2.0	New	Default	Adopted at launch
Loan Origination	Existing	Loan Origination	Pushed back
Impact	Existing	Impact	Pushed back

Legacy Team Concerns

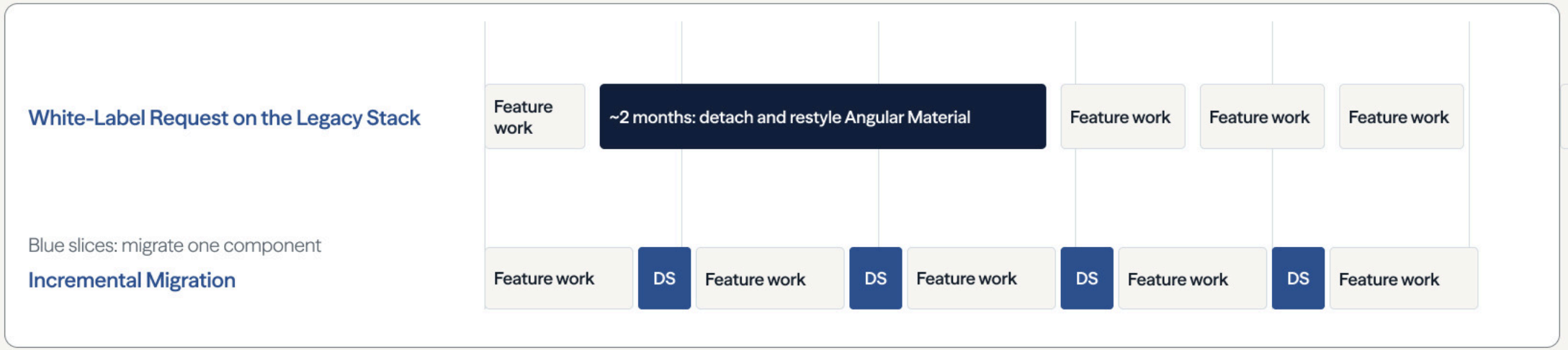
Will the new components handle our data the same way?

Will migrating slow down features we've already committed to?

The CTO mandated the system. These teams could have ignored it and gotten away with it.

Staying on Angular Material had a cost too.

My pitch to both teams' tech leads and engineers was arithmetic: one white-label request on the old stack would eat more roadmap than migrating in small slices.



My other arguments

The original brief: the same components in every product

White-label demand from other and future products

A modern look for products moving from pilots to market

Later, Loan Servicing 2.0 got white-label requests from state education departments and third-party loan servicers, and the parent company got its own theme.

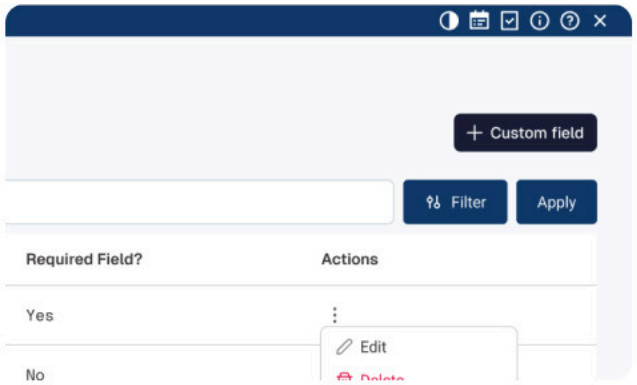
One component on one page, then the rest.

Their concerns changed the rollout, not the system. We migrated in order of how much data a component touches, so trust was earned before anything risky moved.

One component on one page → Full Pages

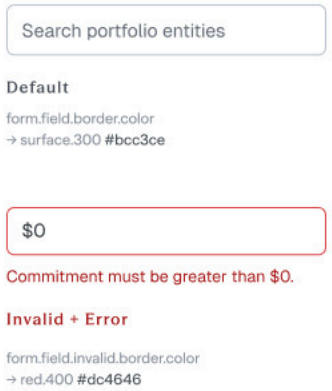
Buttons and Links

No data. Proves styling and build integration with nothing to lose.



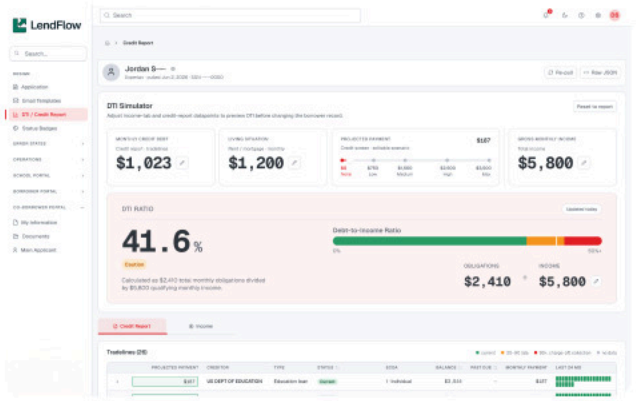
Inputs, Dropdowns, Paginations, and Selects

Collects data. Proves values and forms survive the swap.



Data Tables, Graphs, Form Blocks

Heaviest data. Moved last.



Impact: fully migrated. Loan Origination Product: everything except a few data-visualization pages. Data-handling problems reported: none.

Three months in, our foundation stopped being free.

New PrimeNG versions and LTS went paid. An Angular v22 upgrade was coming. Staying put was technically possible, which is different from being a good idea.

Option	What it meant	Verdict
Stay on PrimeNG v21	Last open-source version and works with Angular v22, but no LTS and growing compliance and performance risk	Rejected
License every engineer	Simplest, with a bill that starts at 40+ seats and grows with every hire	Rejected
Migrate off PrimeNG	Undoes three months of adoption and reopens the Loan Origination and Impact conversations	Rejected
Upgrade to v22 with a shared component repo	1–2 front-end leads manage a shared repo; 5 paid developer licenses	**Chosen**

Chose

Shared repo and 5 licenses, within the license terms. Approved by the CTO and Engineering Manager.

Gave Up

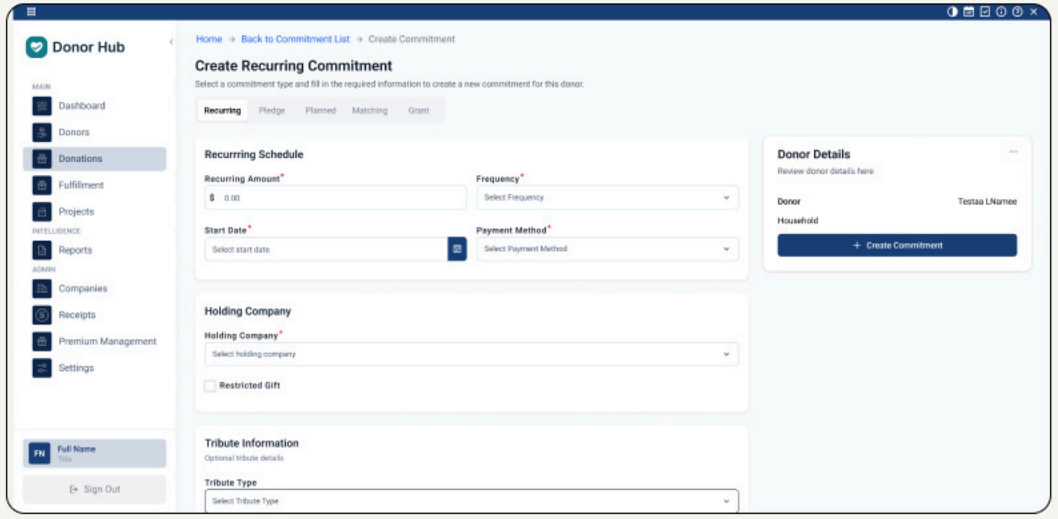
Contribution speed. Every component change now waits on one or two front-end leads.

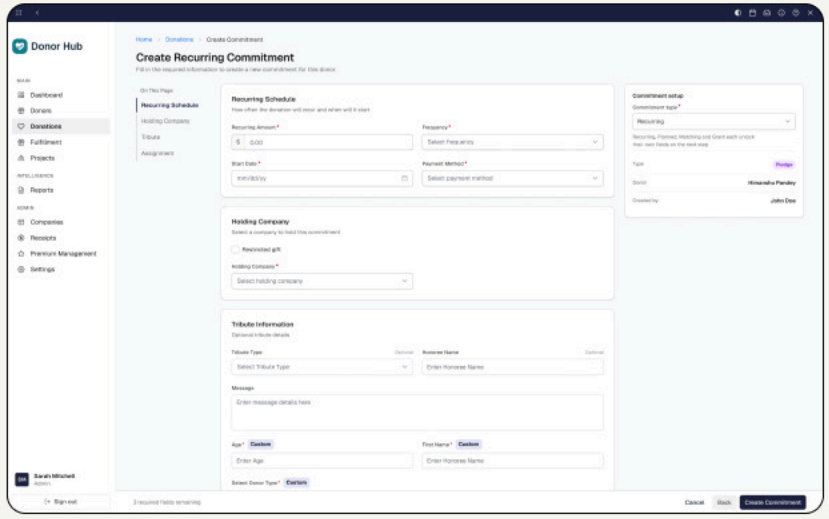
Would have flipped if

That queue started delaying feature work, or the license terms changed.

Where each team landed by September 2026.

Product	Starting Point	Theme	How it adopted	Status as of Sept 2026
Capital Planning 2.0	New	Default	From day one	On the system
Investor Portal 2.0	New	Default	From day one	On the system
Loan Servicing 2.0	New	Default · Parent co theme added	From day one	On the system
Impact	Existing	Impact	Gradual migration	Complete
Loan Origination Product	Existing	Loan Origination Theme	Gradual migration	Complete except a few data-viz pages





Impact, the same page before and after migration.

What changed, what I measured, and what I can't claim.

Measure	Before	After	Source and Caveat
Producing a brand theme	~2 months of manual restyling per request	≈1 hour to generate a preset	Before: my estimate with the front-end lead. After: observed on the parent company theme. Generation only; excludes QA and deployment.
Product teams on the system	0 of 5	5 of 5	The Loan Origination product has a few data-viz pages left.
Brand themes from one foundation	0	4	Default, Loan Origination, Impact, Parent Company
Hardcoded colors or text values merged	Not tracked	0 in 3 months	Enforced by a CI stylelint rule. A guardrail, not an outcome.
Engineers building on the system	N/A	40+	Headcount across the five teams, still growing.
Data-handling problems during migration	N/A	None reported	To my knowledge; not formally tested.

Not claimed: Loan Origination signed new trade-school accounts after relaunching on the system.
The new look may have helped. I can't separate it from everything else that changed, so it stays off the scoreboard.

What I'd do differently.

01

Make vendor licensing a selection criterion from day one

PrimeNG's license change was survivable. It was also a risk I never scored. Now it sits on the list next to coverage and white-label cost.

02

Bring the teams likely to push back into the audit

Loan Origination and Impact raised fair concerns after rollout started. If they'd helped run the audit, the data-handling question would have been answered before anyone had a reason to say no.

Choosing a foundation we didn't build.

Two designers can't build and maintain a component library for five products. So the real question was which one to adopt, and what it would cost us later.

Criteria	Angular Material	NG-Zorro	Taiga UI	Spartan	PrimeNG
Base styling	Opinionated (Material)	Opinionated (Ant Design)	Token-based presets	Headless	Token-based presets
Work per new brand	Undo base styles by hand	Undo base styles by hand	Generate a preset	Code a new visual layer	Generate a preset
Coverage incl. data viz	Broad, including charts	Broad, including charts	Limited, especially data viz	Broad, including charts	Broad, including charts
Migration for existing products	None	Full	Full	Full	Full
Licensing and vendor risk	Not assessed	Not assessed	Not assessed	Not assessed	Not assessed, and it went paid 3 months in

Brand lives in two tiers. Behavior lives in the third.

Brand lives above the component layer. Change one primitive and every component in that theme follows, without anyone touching a component.

	Primitive	Semantic	Component	Rendered
Default	blue.500 #163060	primary.500 → {blue.500}	button.primary.background → {primary.color}	Save changes
Impact	blue.500 #1C213F	primary.500 → {blue.500}	button.primary.background → {primary.color}	Save changes
Loan Org	red.500 #e31f26	primary.500 → {red.500}	button.primary.background → {primary.color}	Save changes

Themeable per brand

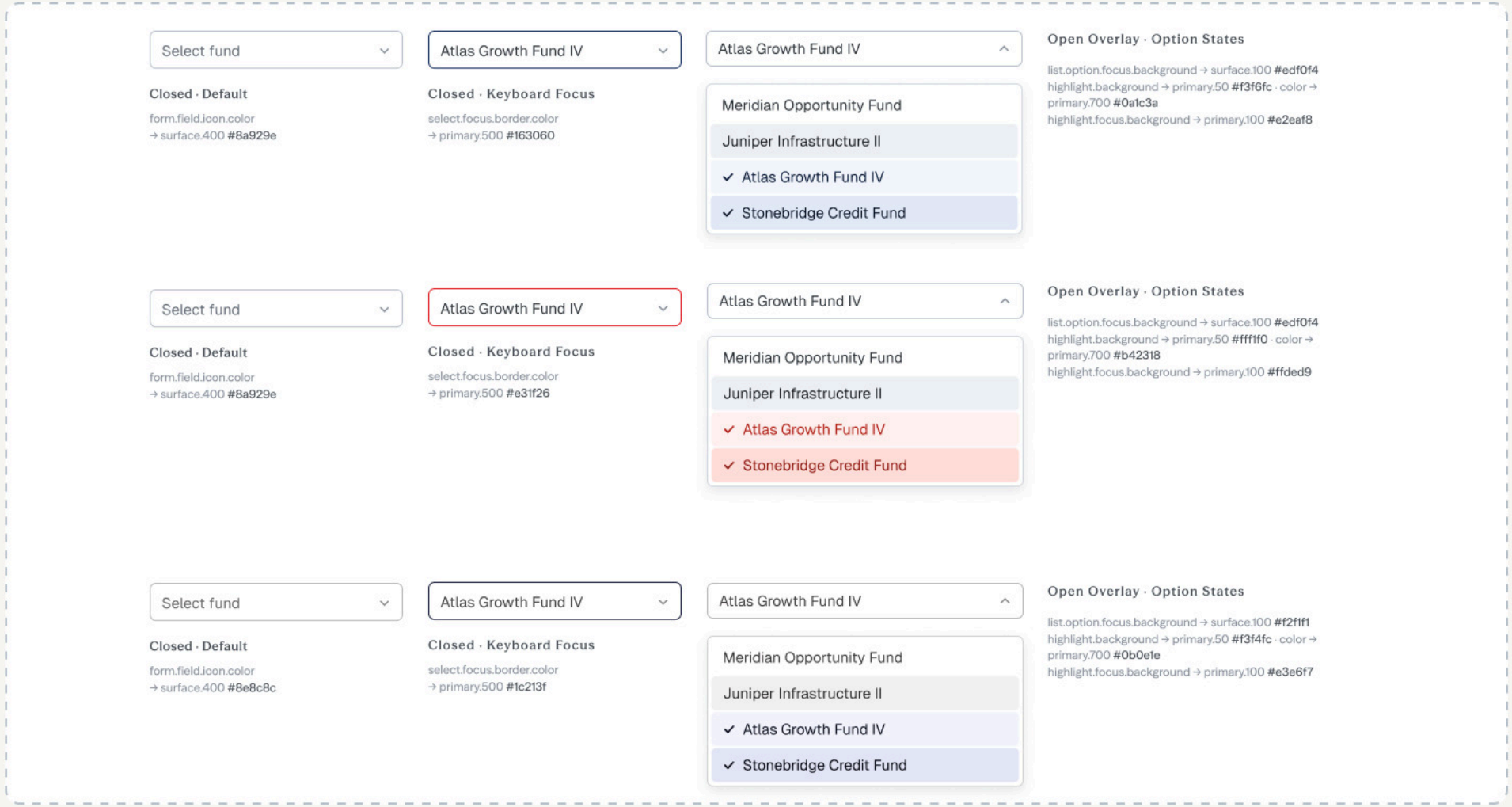
Primitive palette · semantic color roles · typography

LOCKED IN THE SYSTEM

Interaction states · focus behavior · component structure

States are a contract, not a style.

Every brand gets the same default, keyboard-focus, invalid, disabled, and selected behavior. Only the color tokens change.



Build-time themes: what we gave up, and where the line is.

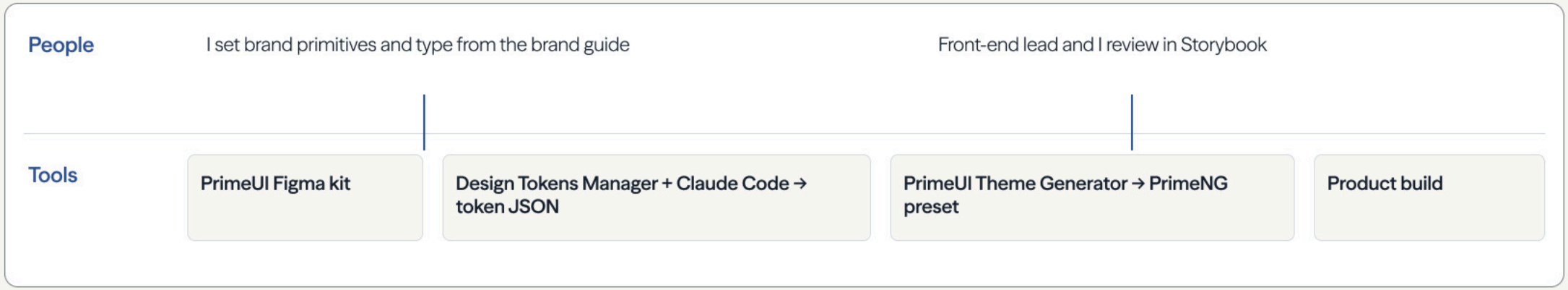
PrimeNG can switch presets at runtime. We chose not to need that.

Criteria	Build-time Preset	Runtime Switching
What ships	One preset per product build	Every preset in every build
What needs testing	Each product in its own theme	Each product in every theme
New white-label customer	Generate a preset (≈1 hr) plus a build	Add a preset; no new build
Live or self-serve branding	Not possible	Possible
Fits our situation (each brand ships as its own product)	Yes	More than we needed

Other Benefits Build-time Presets also mean smaller bundles, faster loads, and styling that renders the same way every time.

Automating themes, not judgment.

Claude Code did the repetitive token edits. The judgment calls, which brand inputs to use and whether the output was right, stayed with me and the front-end lead.



The hour was measured on the parent company theme and covers generation only. QA, assets, and deployment are extra.