◇ SELECT CASE STUDY / LIVE MARKETPLACE

From a million lines to 30,000.

A working rebuild of a beta marketplace that had become difficult to change.

Company identity withheld under NDA.

Over 1M → ~30k

Lines of code, original application to rebuild

350+ → 1

tRPC endpoints consolidated into one REST endpoint

The marketplace had beta customers and more than a million lines of code. A spike team had built to hundreds of specifications. Similar features followed different logic; state could change throughout the app. Tests were scattered, written rapidly with AI to different standards. Owners worried about reliability, and engineers were struggling to follow the system.

The challenge was to keep developing rapidly without agent costs spiraling or losing control of the software. A complete rewrite was judged necessary.

Finding what the product actually needed

I traced repeated behavior to decide which implementations could share a common core. Similar-looking screens were not enough reason to merge them. The redesign preserved meaningful product differences while giving shared records and decisions defined owners.

What changed beneath the interface

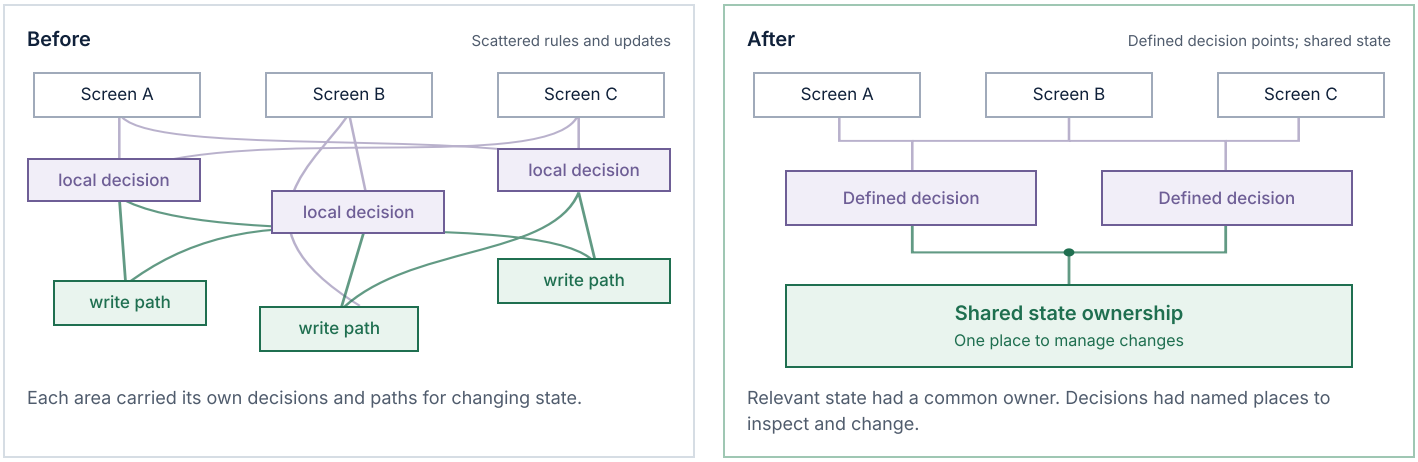

Illustration of the architectural change.

I consolidated 350+ tRPC endpoints into a single REST endpoint with shared authentication, validation and standardized SQL. Repeated views, components and styles became a Storybook library.

A working rebuild of their product

The founders reviewed a roughly 30,000-line implementation with a shared API, reusable interface components and documented architectural decisions. The proposed structure was there to examine in their own product.

◇ HOW I WORK WITH TEAMS TODAY

What could your team build this quarter?

Bring the project that keeps slipping, the change nobody wants to touch, or the AI rollout that has not made delivery easier.

I work with the people who know the product to find what keeps holding the work up. Then I lead an agreed piece of implementation with your engineers and agents. You see the proposed change in your own software as we build.

The distinctions I use to find the structure

System	Product	State	Judgment
What is true.	What matters.	What is written down.	Deciding what to write.

I trace what the application records, where those records change, and which decisions govern the changes. Product owners help establish what must remain different. An unused capability may belong to their next release; it is not automatically waste because it is quiet today.

Blind agent testing

My current process includes fresh agents given a task and access to the application, without the builder's explanation. They work through user flows and record what happens. I preserve those observations, repeat the tasks against the changed implementation, and investigate differences that could signal missing behavior or damage from standardization.

Use the application
Record what a user can do.

→

Preserve the record
Keep the steps and observations.

→

Repeat after changes
Look for changed behavior.

→

Investigate findings
Decide what needs correction.

The observations sit alongside requirements, integration tests and engineering review. A fresh agent can miss things; I check its findings rather than treating its verdict as proof.

Give the team something useful early

Where the interface needs consolidation, I deliver the component library early. Engineers can improve the building blocks and styling while the rest of the work continues. I document the shared decisions so the team can discuss and change them afterward.

Make the first commitment clear

Before we begin, we agree on the scope, fee, team involvement and checks for the first phase. I lead the implementation. Your engineers and product owners join the decisions that need them.

Which problem would you show me first?

In thirty minutes, we can work through what you want to build, where it keeps getting stuck, and what is worth investigating. A simple tool or a targeted fix may be enough. We will also establish whether I am the right person to lead the work.

Talk through your next move ↗

THE THINKING BEHIND MY CURRENT APPROACH

Creating Clarity around Reliable AI in Software ↗

Daniel Ackerman
dan@0to1.dev