CASE STUDY / 2026

Kairo

Screen-native AI guidance for practical software learning

A macOS application that combines voice and screen context to guide a learner to the next action, without taking control of their computer.

Role
Product and engineering owner

Evidence
100+ users; Sarvam Startup Program

Stack
Tauri, Rust, React, Fastify, PostgreSQL

The product problem

Practical software labs are highly contextual. A learner can be one click, command, or configuration change away from progress, but a conventional chat interface cannot see that state without a long explanation and repeated context switching.

Kairo makes a narrower promise: understand the learner's question and current screen, then point to the next action. It explains and guides, but does not click or type on the learner's behalf. That boundary preserves control while still making assistance useful in the moment.

How AI is used

The interaction begins with a voice question and, only when required, screen context. A text gate determines whether a frame is necessary; text-only requests discard it locally. When context is needed, the backend sends the request to speech and model providers and returns spoken guidance, a visual highlight, and a companion cursor.

The product work was not simply adding a model call. The important decisions were where to request context, when to refuse capture, how to make the result legible on screen, and how to maintain user trust while the system is observing a sensitive desktop environment.

System design

- **Desktop client:** React 19 inside Tauri v2, with Rust owning the global shortcut, audio, screen capture, panels, and overlay behavior.
- **Product backend:** Fastify and PostgreSQL handle authentication, user preferences, usage metering, billing, and provider access.
- **Secure AI boundary:** provider keys stay server-side in production rather than shipping inside the desktop bundle.
- **Feedback surface:** responses combine language, visual guidance, and cursor context so the learner can act without losing their place.

KAIRO / CASE STUDY

Designing for useful, bounded assistance

The goal was helpful context, not an always-on recorder or an autonomous computer operator.

Privacy and control decisions

Screen context is treated as sensitive input rather than a permanent stream. The design makes collection explicit, local when possible, and constrained by the application's state.

- Block capture for a conservative set of password, email, wallet, banking, messaging, and photo applications.
- Discard a frame if the frontmost application changes while capture is in progress.
- Exclude Kairo's own guidance surfaces from normal production capture.
- Resize transmitted frames before they leave the device.
- Keep direct provider credentials restricted to local development paths.

What I owned

I took Kairo from the product interaction through the full system boundary: desktop behavior, visual guidance, backend APIs, data, authentication, metering, billing, and model-provider integration. That meant continuously balancing an ambitious user experience with reliable operational behavior and a clear privacy posture.

Result and relevance

Kairo is now used by **100+ users** and was selected for the **Sarvam Startup Program**. It is evidence of end-to-end product building: framing a real user problem, designing the AI workflow, building the native and backend systems, shipping the product, and iterating with users.

For an AI-native product team, Kairo demonstrates the way I work: start from a constrained workflow, use AI where it creates real leverage, keep humans in control, and own the technical details required to make the product dependable.