

Three products, one pipeline

Maxwell Connolly · August 2026

SUMMARY

Kemily has built three products with substantially different requirements: an enterprise platform where the binding constraint is data isolation, a coaching tool where it is direct manipulation, and a consumer application where it is visual identity. All three were produced by one operator using a single development pipeline. This paper describes what each product is, what each demanded, and what was actually delivered. It does not describe the pipeline itself; that is Paper 02.

1 Scope

The three products below were not selected to demonstrate a thesis. They are ordinary products with ordinary requirements, developed for different markets over the same period. They are presented together because their requirements diverge sharply, and that divergence is what makes the common production method worth examining.

PRODUCT	DESCRIPTION	BINDING CONSTRAINT	STAGE
Vestria	Enterprise quality-assurance platform for hotel groups: inspection, scoring, deficiency tracking, and the finance workflow that reimburses auditors.	Multi-tenant data isolation; consolidation of several existing systems.	Foundation complete; product surfaces specified and unbuilt.
Victrix	Team operating system for competitive athletics: routine design, roster, scheduling, parent communication.	Direct-manipulation canvas; a coherent component system.	In use by a working coach.
Fretburn	Guitar-drilling application built on short repeatable exercises, sold once rather than by subscription.	A distinctive visual identity held consistent across twelve screens.	Design locked; prebuild.

Each section below states the requirement, then what was built against it, then what can be shown. Where a claim cannot be evidenced, that is stated.

2 Vestria — consolidation and data isolation

2.1 The requirement

Hotel groups run recurring quality audits across every property in their portfolio. In current practice that work is distributed across four systems: a commercial inspection product, spreadsheets for scoring and deficiency tracking, email for the review and approval cycle, and a separate expense process for reimbursing the auditors who travel to conduct the inspections.

Vestria covers that lifecycle in one platform. The finance workflow is included, which is the component most competing products omit and the one that turns four systems into one.

A second requirement shaped most design decisions: reduce the input required from the operator. Scoring thresholds, property hierarchy, brand standards and eligibility rules are customer-configurable surfaces rather than support requests. Standards documents are ingested and extracted rather than re-keyed. The audit workflow carries state forward so that an inspector confirms rather than re-enters.

2.2 The isolation requirement

A multi-tenant platform holding one hotel group's audit data must make that data unreachable from another tenant's session. Multi-tenancy is a well-understood problem with established patterns; nothing in this implementation is novel. The relevant observation is that an agent-built product, produced by one operator, reached the standard an enterprise security review expects rather than the working-but-indefensible state that fast solo builds commonly reach.

The implementation enforces isolation at the database rather than in application code, so that a missing filter in a single query cannot expose data:

- **82 row-level-security policies** across six migrations, default-deny, with FORCE enabled.
- The runtime connects through a connection pooler as a privileged role, so policies would not otherwise apply. Every user-facing query therefore runs inside a transaction that drops to a non-privileged role and sets the tenant identifier as a database setting the policies read.
- Both the role change and the identifier are transaction-scoped. They reset on commit and cannot persist to a later request that shares the same physical connection, which is the failure mode that appears under load.
- The tenant identifier has one trusted source: a membership-verifying resolver that fails closed. It is never read from a request body, query string, or client-supplied header.
- A cross-tenant isolation suite runs in continuous integration as its own named check. The distinction between enabled and proven matters here; the suite executes against

a policy-capable database on every change.

2.3 What cannot be shown

Vestria has no screenshots in this paper. Its foundation is built — identity, tenancy, domain calculations, system console, operational documentation — and its customer-facing screens are not. The audit workspace, evaluation report, dashboard and finance surfaces are specified and carded behind a build gate. Presenting a mockup as the product would misrepresent the state of the work.

3 Victrix — direct manipulation

3.1 The requirement

Competitive cheer coaches choreograph on paper: eight-count sheets and formation maps that change continuously. Roster, scheduling and parent communication live in separate applications. Victrix consolidates these into one team system.

The central object cannot be represented as a form. A routine is a timeline of counts; each count positions a set of athletes on a mat; each athlete carries a role that determines what they should be doing at that moment. No arrangement of text inputs expresses this, so the product is built around direct manipulation of a canvas.

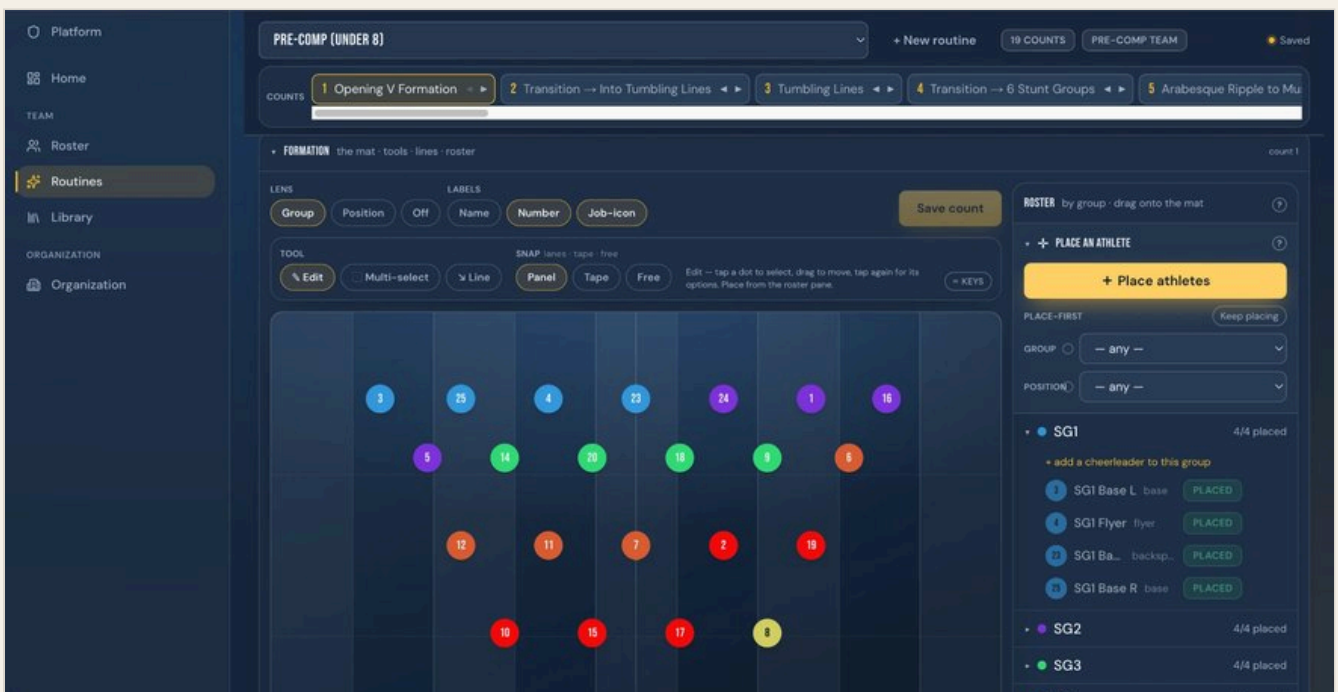


Figure 1 — the routine builder, captured from the running application. Count timeline across the top; positioned athletes on the mat; lens and snap controls at left; stunt-group roster at right.

3.2 What the interface does

— **Count timeline.** A routine is navigated as named counts — an opening formation, a transition into tumbling lines, a transition into stunt groups — rather than as a list of

records. The example shown holds nineteen counts.

- **Lenses.** The same athlete markers recolor by group, by position, or not at all, with labels switchable between name, number and job icon. One formation is viewed through several organizing principles without duplicating state.
- **Snap modes.** Positions snap to panel, to tape, or place freely. Mat geometry is modeled, so markers land where a coach would place them rather than where a cursor was released.
- **Structured roster.** Athletes are grouped into stunt groups carrying explicit roles — base, flyer, backspot — and placement completeness is tracked per group. The roster models the sport rather than listing names.

The canvas renders through a dedicated graphics layer rather than DOM elements, because dragging dozens of positioned markers at interactive frame rates exceeds what a document layout engine is designed for.

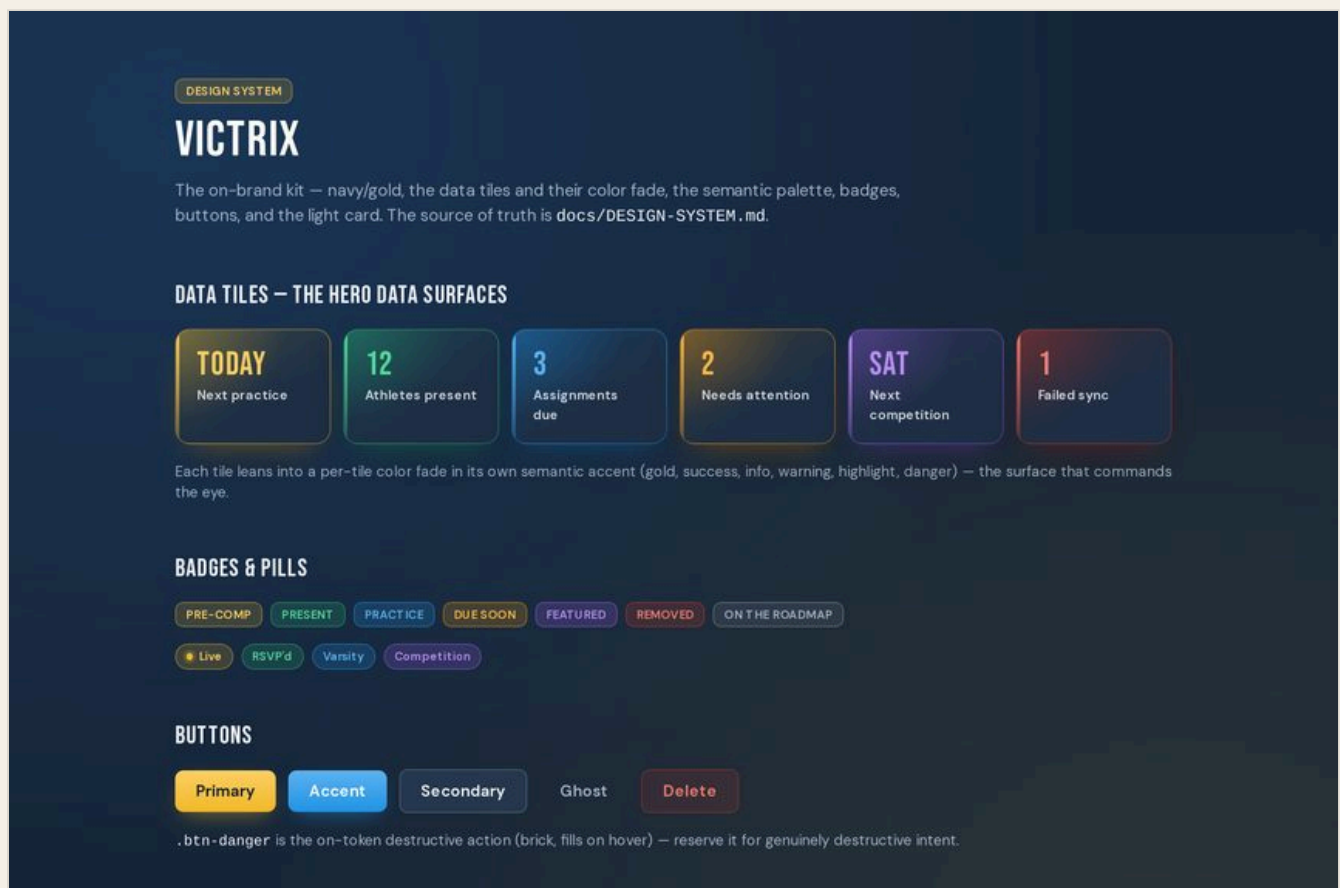


Figure 2 — the design system, rendered live from the application at its own route.

Data tiles, semantic badges, buttons and filter controls are defined once and composed across the product.

3.3 Development observation

Victrix has had a working coach using it on live routines since early in its development. Her feedback drove a complete redesign of the component system. Separately, and without prompting, she requested the data model that was already queued for implementation. That convergence was treated as validation of the model rather than as a feature request.

The practice that resulted: design discovery now runs before implementation on every product. Establishing the visual direction against a real user's reaction to inexpensive artifacts costs a day. Establishing it after the engine is written costs a rewrite.

4 Fretburn — visual fidelity at production

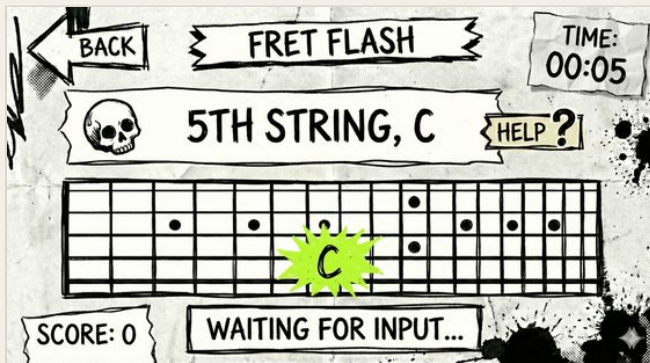
4.1 The requirement

Fretburn drills guitar fundamentals — fretboard memory, tonal recognition, speed — against competitors that are polished, well-funded and subscription-based. It competes on identity instead: high-contrast black and white, photocopied zine texture, hand-drawn throughout, sold once.

For a discretionary consumer purchase in a crowded category, that identity is the product's differentiator. This creates a production problem. Concept art is generated freely and can look however it looks. A real screen is assembled from reusable components, must display live data, and normally loses most of the character in translation.

The question this product tested: can an agent-driven pipeline close that gap? The comparisons below place the generated concept beside the screen built from the component library and rendered by script.

CONCEPT ART



BUILT FROM THE COMPONENT LIBRARY

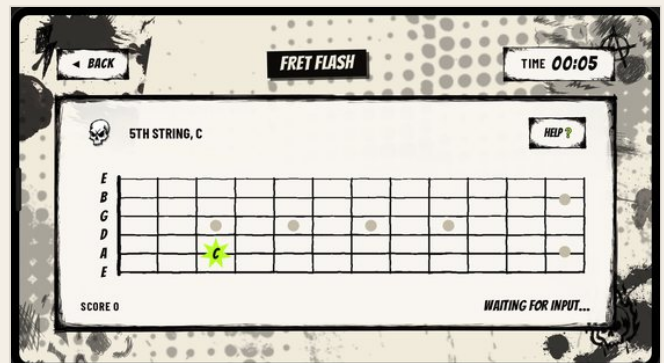
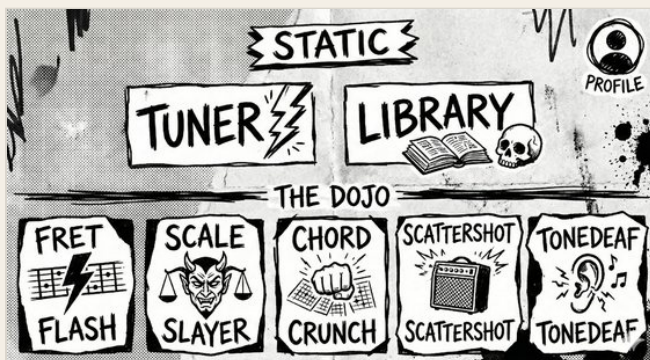


Figure 3 — Fret Flash, the core drill.

CONCEPT ART



BUILT FROM THE COMPONENT LIBRARY



Figure 4 — Dashboard. The concept predates a product rename, which is visible in the title treatment.

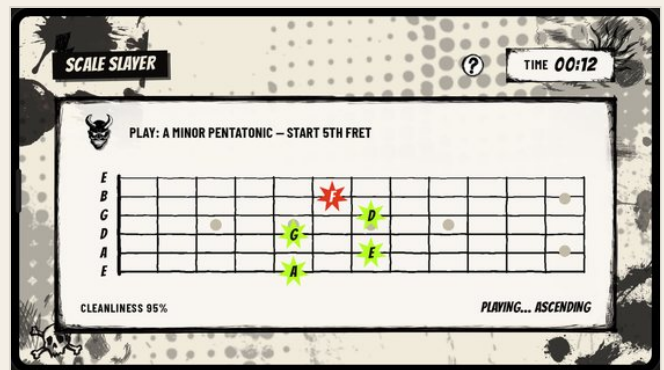
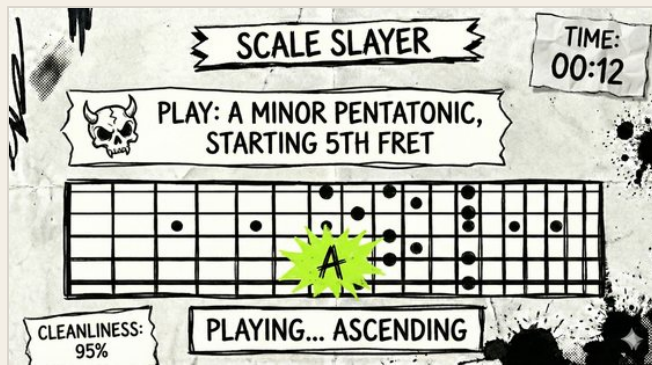


Figure 5 — Scale Slayer.

The differences that appear are deliberate. Built screens carry state the concepts did not: string labels on the fretboard, correct note positions, a running score, a live timer. The visual language survived translation; the placeholder content did not.

4.2 How the identity is held

Consistency across twelve screens is a production problem rather than an artistic one. The pipeline that solves it has four parts.

- **A component audit per screen** produces a ranked list of missing assets, so generation is driven by a gap list rather than by intuition.
- **One component library.** Every mark on every screen comes from a single set of 64 cutouts, each generated in isolation on a locked palette, background removed, transparency verified against a non-white surface, and committed in both raw and cut form.
- **One master contact sheet** renders the whole library together and serves as the completeness audit. Gaps become visible rather than inferred. It has driven several rounds of gap-filling and once prompted the regeneration of a frame that read too heavily beside its neighbors, which is a judgment only available when the set is seen at once.
- **Screens are composed as markup** from the library and rendered to images by script, one file per device frame.



Figure 6 — the master component library: 64 cutouts on transparency, dated, rendered as a single sheet for completeness audit.

The fourth item is the one that changed the economics. Before that script existed, altering a shared component meant reworking each screen individually. After it, a component is changed once and all twelve screens regenerate in a single command. The reduction is not in how fast the design work happens; it is in what the unit of work is.

Two decisions were recorded rather than arrived at by default. The product collects no data — login, authentication and the backend were cut deliberately, which reduces the privacy surface to almost nothing and is consistent with a one-time purchase. And an end-to-end review of every rendered screen is a gate that must pass before implementation begins, matching the readiness gate applied to every other product.

5 Observation

The three products required different things. Vestria required a security posture that survives enterprise review. Victrix required an interaction model that a document layout engine cannot provide. Fretburn required a visual identity to survive translation from concept to component.

They were produced by one operator, over the same period, using one delivery pipeline, one security standard and one review discipline. No product had a dedicated team, and none was built by hand at the volume the output suggests.

That raises the question this paper does not answer: what does the production method have to look like for one person to hold three products to this standard, and how is the output verified when most of it is machine-generated?

RELATED DOCUMENTS

Paper 02 — Helm: an experiment in AI continuity. The system that produces this work: its architecture, how work is managed and dispatched, and the instrument used to certify that its memory is reliable rather than merely plausible.

Paper 03 — The engineering pipeline. The delivery machinery underneath all three products: discovery, gating, testing, and the security controls referenced in §2.2.

Paper 04 — The measured standard. The security standard the pipeline enforces, measured from the inside: the adjusted scorecard, the watchers, and the safety nets.