Scott Weiss

Founder, CrewWork Inc. Lead Software Engineer, PowerSchool. Portland, Oregon.

- scott.a.weiss@gmail.com
- github.com/scottweiss
- linkedin.com/in/scottweissdev

Scott builds CrewWork, a self-hosted platform for building features, repairing errors, and improving code quality. His work connects autonomous implementation to independently produced evidence and human review. At PowerSchool, he leads frontend engineering work on the component framework and build infrastructure behind products used by more than 60 million students.

CrewWork

Founder and sole engineer

CrewWork plans and implements software changes, verifies them against tests and quality gates, and accepts only platform-produced checks as completion evidence. The platform also proposes repairs to its own codebase through draft pull requests that a person reviews.

- Designed the verification model. Every check is stamped with its source, platform or model, at one splice point and written last, so a model report of "done" over failing evidence fails and a model report of "failed" over passing evidence triggers repair.
- Made one 27B open-weight model (Qwen3.8) sufficient for planning, implementation, review, and extraction by building a two-stage extraction pipeline: natural-language analysis first, then JSON-schema-validated typed extraction with typed rejection instead of silent coercion.
- Built the typed context graph that compiles about twenty node types (repository summary, service map, test overview, API contracts, run history) into budget-bounded prompts with layered hash caching, so a small model keeps the evidence that matters inside its window.
- Compiled prompts across three trust tiers, immutable system and developer instructions, trusted reference, and untrusted data, with fenced zones and instruction-override neutralization, to keep repository content separate from higher-trust instructions and reduce instruction-override risks.
- Ran all untrusted execution (previews, test runs, agent sessions, self-repair candidates) in gVisor kernel-isolated containers with hard CPU, memory, and process caps, and built a test engine covering ten language stacks on digest-pinned images, with network isolation verified before test execution.
- Implemented Platform Self-Repair, where the platform finds, fixes, and validates bugs in its own codebase and opens draft pull requests for human review. Promotion is gated by attested evidence from the incumbent build, protected authority paths, and a quality gate that blocks any new lint, type, or security suppression.

At a glance

- Independent validation evidence for reviewable changes
- Local open-weight generation with separate semantic-search embeddings
- Bounded execution and isolated candidate workspaces

Experience

Lead Software Engineer, PowerSchool

2020 to present

- Built Neon, an internal TypeScript web component framework with React adapters, after benchmarking Lit, Solid, and others against the failures of the Polymer stack it replaced. Bundles are more than four times smaller than the alternatives, and components run with zero framework runtime inside legacy Java/JSP and PHP pages as well as modern React apps. Products built on it serve more than 60 million students.
- Created the build and release infrastructure from scratch: webpack, then a migration to Vite, CI/CD, npm publishing to Artifactory, automated cross-browser testing, and deployment automation on AWS.
- Built the React documentation site used by more than 600 developers, with real-time search, an interactive component playground, and live code examples.
- Architected micro-frontends with Module Federation so teams deploy independently while the user experience and internationalization stay consistent.
- Implemented right-to-left and internationalization support across the whole component library, enabling deployment in more than 90 countries including Middle Eastern and Asian markets. Cut CSS bundle size by 70 percent and brought the library to WCAG 2.1.
- Served as frontend lead and internal consultant embedded with multiple teams, and as a founding member of the frontend resource group. Translated Figma designs into accessible components across several products.

Front-End Developer, Hoonuit

2019 to 2020

- Standardized legacy frontends spanning more than fifteen years of code.
- Converted static design assets into responsive web components and layouts.

Developer, Tembo

2014 to 2019

- Built web applications from scratch for state education departments, with responsive components handling more than 80,000 dynamic student reports.
- Developed reusable frontend systems and patterns standardized across deployments in DC, New Jersey, New Mexico, Texas, and Tennessee.
- Implemented interactive data visualization and mapping for large datasets.

Skills

Frontend: TypeScript, React, native JavaScript and Web Components, CSS, Vite, webpack, Module Federation, performance, accessibility (WCAG)

Backend: Python, FastAPI, async SQLAlchemy, PostgreSQL, Redis, Qdrant, event-driven domain architecture

AI systems: Context compilation, structured extraction, prompt trust boundaries, provenance-stamped verification, provider-neutral model routing, autonomous agents

Infrastructure: Docker Compose, gVisor, Nginx, Prometheus, Grafana, Loki, Tempo, OpenTelemetry, Sentry, CI/CD

Design systems: Component libraries, design tokens, Figma handoff, cross-framework compatibility

Education

B.S., Information Technology, Drexel University, 2014