Zain Dana Harper

Seattle, Washington | zaindharper@gmail.com | harperz9.github.io
linkedin.com/in/zaindanaharper | github.com/HarperZ9

Curriculum Vitae | Systems Engineering, AI Infrastructure, and Field Operations

Profile

Independent systems engineer and technical writer building AI workstations, evaluation and accountability tools, language tooling, and graphics systems. My work connects a practical question to a verifiable implementation: what happened, what evidence supports it, and who authorized the next action. I combine a self-directed programming history with prior technical networking support and eleven years of full-time arboriculture and customer-facing operations.

Professional Experience

Independent Systems Engineer | Independent practice

Seattle / remote | 2023 to Present

- Own the architecture, implementation, integration, tests, documentation, and release work for Flywheel and related tools under the Zentropy Labs project identity. Work spans Python services and command-line tools, Flutter desktop software, Rust systems tooling, and browser interfaces.
- Develop model-agnostic workflows, evaluation harnesses, controlled-action interfaces, and source-linked records that can be checked without relying on a model to approve its own answer.
- Contribute focused fixes, regression tests, and documentation to externally maintained projects, responding to maintainer review and preserving the scope of each accepted change.

Freelance Technical Writer, Documentation, and Product Operations

Independent practice | Began 2017; project-based work

- Produced technical and compliance documentation, API and implementation guides, proposals, release notes, and operational playbooks. Turned implementation details into material usable by customers, developers, and reviewers.

Commercial Arboriculture and Field Operations | Legendary Tree

Seattle area | April 25, 2015 to June 2, 2026

- Combined ground-based tree-care work and safety coordination with client intake, site assessments, estimates, proposals, scheduling, and crew/vendor communication.
- Monitored rigging systems, clearances, and changing conditions while supporting climbers from the ground. Explained work options and constraints to clients and maintained continuity between the proposed scope and field execution.

Technical Networking Support | Xbox/Microsoft contract

Subcontracted through Stream/Convergys | Wilsonville, Oregon | 2014 to 2015

- Diagnosed connectivity and account issues by phone and chat using TCP/IP, DNS, NAT, firewall, and router troubleshooting; documented repeatable resolutions and customer-facing instructions.

Programming Background

Self-directed programming began with game modding, reverse engineering, bot and rotation logic, and runtime-memory tooling, then developed into systems engineering, graphics, compilers, and AI infrastructure. This is a project history, not a claim of equivalent years of paid software employment.

AI Platform and Agent Systems

Flywheel | Model-agnostic AI workstation and coding harness

Primary owned platform. The September 19, 2026 v1.0.1 release includes a Windows installer with a bundled engine. The source repository contains the Python runtime, Flutter desktop client, and browser fallback.

- Routes tasks to hosted or local models through an OpenAI-compatible gateway; keeps model selection separate from the task record and exposes command-line, desktop, and API entry points.

- Provides Rowan, a desktop assistant that translates requests into recorded tasks, together with permission-checked coding and tool-use workflows, session state, and inspectable outcomes.
- Composes research intake, workspace indexing, verification, orchestration, memory, authoring, and other tools through registered lanes and published interfaces. Bundled components and installable companions have distinct packaging requirements.
- Produces sealed, re-derivable receipts and offline MATCH, DRIFT, or UNVERIFIABLE outcomes. A reproducible receipt establishes the bounded check it records, not universal model accuracy or safety.

Accountable Surface | Governed actions and computer-use adapters

Built operator-granted action interfaces for filesystem, command, API, browser, and Windows UI Automation work. Actions carry explicit scope and postconditions; persisted journals preserve decisions and observed effects. Reversible adapters include restoration paths. Fake drivers make browser and desktop scenarios testable without uncontrolled live actions.

The v0.2.0 release provides source and wheel artifacts on GitHub. The project remains alpha; browser integrations are optional and a tool refusal or unknown result is not promoted into a successful action.

Articulate | Writing analysis and accountable editing

Built deterministic prose analysis with register, mode, and genre profiles, alongside optional model-assisted editing. Includes command-line, library, MCP, language-server, and SARIF surfaces; math-span protection preserves formulas during supported editing flows. Content-free audit receipts record checks without retaining the underlying prose. Version 0.3.0 was released September 20, 2026.

Agent tools, memory, and coordination

Relay supplies model-agnostic coding-agent tooling; Plexus connects tool interfaces; Mneme retains memories with source checks; Canon consolidates assistant instructions and context. Bulletin provides a signed-identity public message board. These systems contribute distinct capabilities to the larger platform rather than representing separate claims of deployment at customer scale.

Evaluation, Systems, and Security Engineering

Terminal State Fixtures and Crucible

Designed exhaustive terminal-state evaluation cases and deterministic scoring that inspect the environment after an agent runs. The MLflow fixture family includes 324 cases; the component-forge suite includes 337 cases and 5,058 frozen expectations. These are fixture counts, not measured model-performance gains. Crucible provides structured worker/verifier evaluation with explicit match, drift, and missing-evidence outcomes. Third-party private staging is not represented as public hosting or endorsement.

EMET | Byte-integrity witnessing

Developed Python, Rust, Node, and Go implementations around shared conformance cases and portable verification records. The witness checks byte-level integrity, anchors, and source correspondence; it does not grant permission, certify semantic truth, or replace an operator decision. Cross-language checking makes implementation differences visible.

BuildLang | Rust compiler and developer tooling

Developed a systems-language toolchain with typed capability effects, type inference, parser and intermediate-representation work, native C output, shader output, and language-server support. The compiler makes categories of ambient access visible to its type and build machinery. Experimental backends and linear-type work remain distinguished from the supported path.

Phantom | Hardware-identity inspection and reversible system changes

Developed a Rust-based Windows and Linux utility with deterministic profiles, read-only audit, backup-before-write behavior, validation, restoration, and platform packages. The released v1.1.1 scope is userland-reversible identifiers. Kernel and firmware layers are modeled, not shipped.

Controlled assessment and operator tooling

Private systems work includes Array for approval-gated assessment orchestration and containment records, Lab/Seed for controlled-lab fixtures and detection-engineering workflows, and Runtime/ORCA for local workspaces, reports, bundles, and release provenance. Public descriptions cover architecture and safe interfaces; private implementations, target material, and operational details are not published here.

Research intake and workspace tooling

Built Gather for source-carrying research intake, Index for workspace and symbol maps, and Forum for model orchestration with recorded decisions. These tools make source selection, handoffs, and task context inspectable across a multi-repository product.

Graphics, Color, and Runtime Work

Developed HLSL and C++ graphics and game-modding work spanning post-processing, Direct3D 11 integration, shader debugging, GPU traces, and shared-memory interfaces. Related public projects include Elder ENB, SkyrimBridge, ENB Runtime Core, Brender Archival, and Engine Revival. Existing engines and collaborators retain their own attribution.

Color and creative-tool work includes Build Color, Calibrate Pro, Studio Engine, and browser-based studio/print surfaces: perceptual color spaces, color difference, HDR/tone-mapping tools, display characterization, and generative visual systems. These are engineering projects, not claims of professional calibration certification.

Selected Open-Source Contributions

The following changes were accepted into externally maintained repositories. They are contributions, not employment at those organizations. The broader public contribution record separately tracks merged, open, and closed without merge outcomes.

- DeepEval #2822: fixed unhashable tool-output and nested-parameter failures in tool-correctness evaluation; merged July 2, 2026. github.com/confident-ai/deepeval/pull/2822
- TOMLKit #549: preserved dotted-key siblings when replacing a value with an array of tables; merged July 14, 2026. github.com/python-poetry/tomlkit/pull/549
- Datasette #2815: corrected error handling for malformed composite primary-key row URLs; merged July 7, 2026. github.com/simonw/datasette/pull/2815
- Free Law Project #820: documented runnable database-free fast tests; merged August 25, 2026. github.com/freelawproject/litigant-portal/pull/820
- Hebbian Robotics #157: tested reserved-column drift against the actual queryable episode schema; merged August 25, 2026. github.com/Hebbian-Robotics/hflow/pull/157
- Voxwire #44: added offline WebSocket recording and replay regression coverage; merged August 15, 2026. github.com/vectorvoyager358/voxwire/pull/44

Technical Competencies

Languages and runtimes: Python, Rust, C++, Dart/Flutter, TypeScript/JavaScript, Lua, HLSL, C#, PowerShell, and Bash. Additional cross-language conformance work includes Go and Node implementations.

AI infrastructure and quality: multi-provider/local-model routing, tool-use loops, task decomposition, MCP, evaluation design, deterministic fixtures, regression testing, worker/verifier separation, failure analysis, and terminal-state scoring.

Verification and controlled actions: capability effects, operator grants, action envelopes, source provenance, hash-chained journals, offline replay, scoped postconditions, conformance suites, and release evidence.

Software delivery: Git/GitHub, GitHub Actions, Python packaging, command-line design, Windows/Linux installers, CMake, test automation, static websites, release documentation, and checksums.

Graphics and domain knowledge: D3D11/HLSL, shader and runtime debugging, color-space and tone-mapping work, GPU instrumentation, shared-memory interfaces, and applied arboriculture/horticulture from field practice.

Working Practice

AI-assisted development is part of the workflow. Architecture, integration, source review, test design, release decisions, and responsibility for the submitted work remain mine. Public examples separate observed results from interpretation and retain negative or inconclusive findings rather than treating every completed run as success.

Research Agenda

My independent research examines how an AI system can leave evidence that another system or person can check without inheriting its authority. Workstreams include evaluation provenance and verifier independence; live-state drift; proof-carrying action envelopes; the relationship between memory, uncertainty, and behavior; and explicit human authorization for consequential actions. Robotics is a research interest, not a claim of professional robotics deployment experience.

Selected Publications and Research Records

Independent systems papers, preprints, research notes, and archived manuscripts with DOI records. These records are not peer reviewed. An archived philosophical manuscript is not an awarded academic degree.

EMET: An Authority-Incapable Byte-Level Integrity Witness

Systems paper, 2026. Byte-integrity witnessing and the separation of observation from authority. doi.org/10.5281/zenodo.21230267

BuildLang: Accountable Compute via Typed Capability Effects

Systems paper, 2026. Capability effects and accountable compilation. doi.org/10.5281/zenodo.21231253

Witnessed Independence: Recording Whether a Verifier Graded Its Own Work

Preprint, 2026. doi.org/10.5281/zenodo.21232206

Proof Packets: A Derive-Don't-Trust Envelope for Accountable Agent Actions

Preprint, 2026. doi.org/10.5281/zenodo.21231406

The Personhood-Gate Handoff

Research note, 2026. doi.org/10.5281/zenodo.21234475

Re-Perceived Effects

Research note, 2026. doi.org/10.5281/zenodo.21231311

The Witnessing Spine

Archived research corpus, 2026. doi.org/10.5281/zenodo.20778927

Conferred Existence

Long-form philosophical manuscript and archived corpus, 2026. doi.org/10.5281/zenodo.20773724

Education

High School Diploma | Wilsonville High School, Wilsonville, Oregon | 2013

Self-directed study in systems programming, compilers, graphics, color science, AI evaluation, epistemics, and philosophy of mind. Practical arboriculture and plant-science knowledge is grounded in field work, not a separately claimed professional license.

Research and Portfolio Links

ORCID: 0009-0001-7175-5393 | orcid.org/0009-0001-7175-5393

Publications: harperz9.github.io/publications.html | Portfolio: harperz9.github.io/portfolio.html