AGENTICAMIT

AI SYSTEMS, EXPLAINED FOR WORK

Five AI projects worth building

Five original projects for proving you can define the task, expose the mechanism, handle failure, and attach evidence.

MECHANISM-FIRST PROOF-ATTACHED USEFUL-TOMORROW

SCAN FIRST. BUILD ONE.

Each project gets one page: what to build, why it matters, the system flow, four build steps, a practical stack, proof criteria, metrics, and primary sources.

07 PAGES / 05 PROJECTS / ONE PAGE EACH

CHOOSE YOUR SIGNAL

Choose the skill you want to prove.

The strongest project is the one that makes your judgment visible.

01 **Output Contract Firewall**
Reliable model outputs
INTERMEDIATE

02 **Context Budget Studio**
Context engineering
INTERMEDIATE

03 **Grounding Ledger**
Evidence and provenance
INTERMEDIATE

04 **Agent Replay Lab**
Agent observability
ADVANCED

05 **AI Change Impact Mapper**
AI developer tooling
ADVANCED

PORTFOLIO PROOF RUBRIC

01 Task defined

02 Mechanism visible

03 Failure handled

04 Evidence attached

PROJECT 01 / RELIABLE MODEL OUTPUTS

INTERMEDIATE

Output Contract Firewall

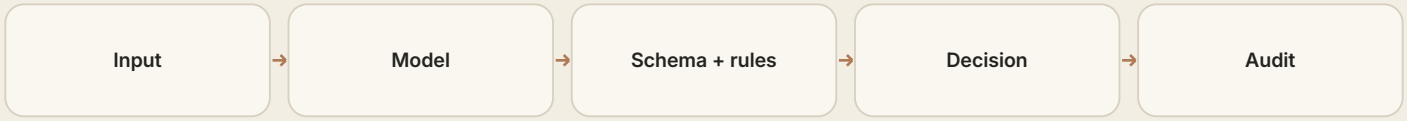
WHAT YOU'RE BUILDING

A validation gateway between an LLM and downstream software. Every response is accepted, repaired once, or quarantined.

WHY THIS PROJECT MATTERS

This proves you can turn probabilistic output into a typed, audited trust boundary instead of relying on a prompt and hoping for the best.

SYSTEM FLOW



BUILD IT

1. Write one versioned JSON contract plus valid and invalid fixtures.
2. Separate schema, semantic, and policy checks.
3. Allow one repair attempt, then run every check again.
4. Log the decision and test normal, edge, and adversarial cases.

SUGGESTED STACK

API	Python + FastAPI
CONTRACT	Pydantic / JSON Schema
MODEL	Structured-output LLM
STORAGE	SQLite audit log
TESTS	pytest fixtures

PROOF

No known high-severity fixture is accepted; every request leaves an audit record.

FAILURE RECEIPT Validator error must quarantine, never pass through.

MEASURE

- Unsafe acceptance
- Quarantine rate
- P95 latency

PRIMARY SOURCES

Structured Outputs / Eval Best Practices / OWASP Injection

PROJECT 02 / CONTEXT ENGINEERING

INTERMEDIATE

Context Budget Studio

WHAT YOU'RE BUILDING

A laboratory that compares which context should reach a model under a hard token budget. Quality, cost, latency, and access decisions stay visible.

WHY THIS PROJECT MATTERS

This proves you can treat context as an engineered resource, compare strategies fairly, and explain when more tokens make an answer worse.

SYSTEM FLOW

Query → Candidates → Access filter → Budget plan → Answer + eval

BUILD IT

1. Create queries, candidate context items, and expected facts.
2. Apply access controls before ranking any candidate.
3. Compare recency, similarity, value-per-token, and diversity.
4. Plot answer quality against tokens, cost, and latency.

SUGGESTED STACK

CORE	Python
TOKENS	Model tokenizer
SEARCH	Embeddings + vector index
STORAGE	SQLite / Parquet
UI	Streamlit comparison

PROOF

Every plan stays in budget, respects access scope, and explains every selection.

FAILURE RECEIPT Preserve one case where extra context lowers quality.

MEASURE

- Task score
- Quality / 1K tokens
- P95 latency

PRIMARY SOURCES

OpenAI Evals / Eval Best Practices / Redis Semantic Cache

PROJECT 03 / EVIDENCE AND PROVENANCE

INTERMEDIATE

Grounding Ledger

WHAT YOU'RE BUILDING

A claim-level evidence system that connects important answer claims to exact, versioned source spans and abstains when critical support is missing.

WHY THIS PROJECT MATTERS

This proves you understand the difference between showing a citation and demonstrating that the cited evidence actually supports the claim.

SYSTEM FLOW

Question → Retrieve → Split claims → Verify spans → Publish / abstain

BUILD IT

1. Version a small corpus with source IDs and content hashes.
2. Generate answers that reference stable source IDs.
3. Split answers into claims and link exact evidence spans.
4. Classify support and block unsupported critical claims.

SUGGESTED STACK

API	Python + FastAPI
RETRIEVAL	FAISS / Chroma
CONTRACT	Pydantic records
STORAGE	SQLite / Postgres
REVIEW	React / Streamlit

PROOF

Every critical claim opens to an exact excerpt; unsupported critical claims are withheld.

FAILURE RECEIPT A related citation that does not entail the claim must fail.

MEASURE

- Citation precision
- Unsupported publish
- Abstention quality

PRIMARY SOURCES

Eval Best Practices / OpenAI Evals / OWASP Injection

PROJECT 04 / AGENT OBSERVABILITY

ADVANCED

Agent Replay Lab

WHAT YOU'RE BUILDING

An append-only event system for reconstructing, replaying, forking, and comparing agent runs without repeating external side effects by default.

WHY THIS PROJECT MATTERS

This proves you can debug a nondeterministic system from evidence: model calls, tools, state changes, approvals, versions, costs, and artifacts.

SYSTEM FLOW

Agent run → Event log → Replay → Fork → Diff timeline

BUILD IT

1. Define versioned events for model, tool, state, approval, and error.
2. Instrument one agent and store large artifacts by hash.
3. Rebuild state using recorded model and tool outputs.
4. Fork before a failure and compare the first divergence.

SUGGESTED STACK

RUNTIME	Python / TypeScript
TRACING	OpenTelemetry
STORAGE	SQLite / Postgres
TIMELINE	React
SAFETY	Docker sandbox

PROOF

Recorded events rebuild final state; default replay performs no external writes.

FAILURE RECEIPT Secrets must be redacted before the trace is persisted.

MEASURE

- Trace completeness
- Replay success
- Added overhead

PRIMARY SOURCES

OpenTelemetry GenAI / OWASP Injection / Eval Best Practices

PROJECT 05 / AI DEVELOPER TOOLING

ADVANCED

AI Change Impact Mapper

WHAT YOU'RE BUILDING

A dependency graph that maps prompt, model, tool, retrieval, policy, and dataset changes to the smallest defensible set of checks in CI.

WHY THIS PROJECT MATTERS

This proves you can connect an AI-system change to behavioral risk, required evaluation evidence, and an explainable merge decision.

SYSTEM FLOW

Repo diff → Artifact graph → Risk rules → Required checks → CI report

BUILD IT

1. Inventory AI artifacts, evaluation suites, dashboards, and owners.
2. Discover dependencies from source code and configuration.
3. Classify a pull-request diff and traverse its impact graph.
4. Generate checks with a visible rule ID and reason path.

SUGGESTED STACK

CORE	TypeScript / Python
PARSING	AST / tree-sitter
GRAPH	NetworkX / graphlib
CI	GitHub Actions
STORAGE	SQLite

PROOF

Every required check has a reason path; high-risk changes need evidence or a waiver.

FAILURE RECEIPT Stale and low-confidence dependencies must remain visible.

MEASURE

- Impact recall
- Unneeded checks
- CI time added

PRIMARY SOURCES

GitHub Actions / OpenAI Evals / Eval Best Practices