HANIF SHAIK

Computer Science Graduate — AI/LLM Systems, Technical Evaluation & Security

shaikhanif625@gmail.com | github.com/futureaihub | linkedin.com/in/hanif-shaik-025066392 | futureaihub.github.io/hanif-portfolio

PROFESSIONAL SUMMARY

Computer Science graduate focused on building, evaluating, and securing AI-powered software systems — drawn to problems where correctness, reliability, and security decide whether a system can be trusted. Experience includes cryptographic proof infrastructure for auditable AI decisions and structured evaluation of AI-generated code and agent designs against explicit correctness, robustness, and security criteria. Approaches every system the same way: understand the requirement, build it, test it, evaluate failure modes, secure it, and verify the result with evidence.

TECHNICAL SKILLS

Languages: Python, Rust, TypeScript, SQL

AI / LLM: LLM Output Evaluation, Rubric-Based Assessment, Failure-Mode Analysis, Agent Trust Boundaries, Policy Gating

Backend / Systems: FastAPI, REST API Design, Python & TypeScript SDKs, Alembic Migrations, CLI Tooling

Security / Verification: Ed25519 Signatures, Hash-Chained Audit Logs, RFC 3161 Timestamping, Offline Verification, Vulnerability Triage

Databases / Infra / Tools: PostgreSQL, SQLite, Redis, Docker, Docker Compose, pytest, Git

SELECTED ENGINEERING PROJECTS

Zorynex — Provable AI: Cryptographic Proof Infrastructure for AI Decisions

github.com/futureaihub/provable-ai • Python, FastAPI, PostgreSQL/SQLite, Docker

- Built a FastAPI backend (34 endpoints, 8 tag groups) that signs every recorded AI decision with Ed25519 and hash-chains it into an append-only, tamper-evident ledger, so any post-hoc modification is detectable.
- Built a governance layer restricting writes to approved model versions, agents, and policies, plus a standalone offline verifier (CLI and browser UI) so an auditor can confirm integrity without trusting the issuing system.
- Designed a deterministic proof_fingerprint identity check independent of signature verification, and authored a 593-test suite — including a chaos-testing module for database outages, signing-key failures, and disk-full conditions — across dual SQLite/PostgreSQL backends.
- Mapped the system's guarantees to regulatory requirements (SR 11-7, EU AI Act Art. 9/13, CFPB, GDPR Art. 17) and documented its honest limits: tamper-evident rather than tamper-proof, verifiable rather than trustless.

Agent Trust Benchmark — Adversarial Verification for Tool-Using AI Agents

github.com/futureaihub/agent-trust-benchmark • Python, OPA/Rego, SQLite, SHA-256, Pytest

- Built a deterministic verification framework that tests whether tool-using AI agents actually obey authorization policies and report outcomes truthfully, checking benchmark-owned ground-truth observation instead of trusting agent self-reports.
- Implemented an OPA/Rego policy layer that enforces authorization external to the agent's model loop, and a SHA-256 hash-chained evidence recorder verified independently of the LLM.
- Designed 11 deterministic evaluator checks (authorization enforcement, correct-target execution, role-escalation resistance, claim/tool-result consistency) and validated them against 30 adversarial red-team cases.
- Authored a 258-test pytest suite and ran 18 real-agent validation replays against a live model over OpenRouter — 18/18 evidence chains valid, 18/18 deterministic replay matches, 0 infrastructure errors.

AI SYSTEM & CODE EVALUATION

Structured technical review of AI-generated solutions against explicit, weighted criteria

- Evaluated two implementations of a distributed API rate limiter against weighted criteria (requirement fit, correctness, robustness, security), identifying a correctness failure that surfaces only under horizontal scaling — an in-memory per-IP counter silently multiplying its effective limit — and produced a remediation (token-bucket/Redis, atomic Lua-scripted state, authenticated-user key isolation, Retry-After headers), scoring the fix 8.2 vs. 4.0.
- Reviewed two AI-generated web vulnerability assessments of the same endpoint and distinguished a false-positive "reflected XSS" finding — parameter reflection without confirmed execution — from a verified assessment that tested payload execution in the rendered DOM, rather than escalating an unconfirmed finding as a real vulnerability.

EDUCATION

B.Tech in Computer Science & Engineering *2022 – 2026*

Sri Sarathi Institute of Engineering & Technology — Graduated