Ali Zeeshan

SOLANA PROTOCOL ENGINEER · RUST / SVM · DEFI & SETTLEMENT

+91 76708 84956 | mohdalizeeshan4@gmail.com
github.com/alizeeshan1234 | x.com/m_a_zeeshan | zeeshan-site.vercel.app | LinkedIn

SUMMARY

Solana protocol engineer building **security-critical financial infrastructure** — on-chain settlement, collateral and margin accounting, lending markets, AMMs, multisig custody and oracle-driven pricing. Shipped production Rust / Anchor / native Pinocchio for a non-custodial perpetuals DEX, and designed systems where **correctness protects real capital**: loss-bounding circuit breakers, formal verification as a merge gate, and adversarial threat modeling. Comfortable taking a mechanism from research notes to audited production code. Available full-time.

HIGHLIGHTS

- **Production perps DEX at Flash.Trade** — shipped the trade-execution and settlement stack for a non-custodial leveraged/synthetic-asset exchange, with **sub-50 ms** on-chain execution, an outflow circuit breaker that bounds exploit and oracle-manipulation losses, and formal verification enforced as a CI merge gate.
- **Contributed to Solana's core tooling** — merged a compute-unit optimization guide into the Solana Foundation developer docs, and landed three PRs in `pinocchio-stake-program` (StakeStateV2 and required account layouts).
- **svmscope** — authored and published a Rust crate (`crates.io/crates/svmscope`) with a hosted web UI and TypeScript SDK: decode, replay and mutate any mainnet transaction in an embedded SVM, replay it against the exact account state of any slot in the last 30 days, step through it with per-instruction and per-CPI account state, profile its compute down to individual functions, and watch the programs a protocol depends on — an on-chain registry (Anchor) plus an engine that replays a protocol's real traffic against a dependency's old and new binary after every upgrade, with signed webhook alerts.
- **Alpenglow consensus benchmark** — independent, reproducible measurement of Solana's new consensus (SIMD-0326): 218 ms vs 13.2 s p50 finality, with Agave LocalCluster fault-testing to the liveness boundary.

EXPERIENCE

Flash.Trade — Solana Smart Contract Developer

`Aug 2025 – Sept 2026`

Shipped production Rust and Anchor for a non-custodial perpetuals DEX supporting leveraged crypto and synthetic-asset trading, with a focus on settlement correctness, collateral safety and security.

- Built the **trade-execution and settlement stack** for opening, increasing, decreasing and closing positions, plus limit and stop-loss / take-profit orders, with session-key authorization for low-friction, self-custodial trading.
- Designed and shipped an **outflow circuit breaker** with USD-aggregate and per-token leaky buckets, atomically enforced at transfer settlement to bound exploit and oracle-manipulation losses.
- Drove the **MagicBlock Ephemeral Rollup** integration, moving latency-sensitive execution onto the rollup and achieving **sub-50 ms** on-chain execution.
- Built a **formal-verification and fuzzing pipeline** (Kani, Certora Prover, Crucible, LiteSVM, invariant tests) enforced in CI before security-sensitive instructions merge.
- Architected a permanent dual-layer transaction archive with redundant ingestion and automatic gap repair; captured **4.6M+ transactions** and served them through a Solana-RPC-compatible endpoint.
- Shipped builder-code fee attribution, account delegation, self-custodial deposit/withdrawal via Magic Actions, Pyth Lazer pricing and compute-unit optimizations across critical paths.

SELECTED PROJECTS

svmscope — Transaction Replay Engine, Step Debugger, Compute Profiler & Dependency Monitoring · Published Rust Crate · Creator

2026

`crates.io/crates/svmscope` · `github.com/alizeeshan1234/svmScope` · `svmscope.vercel.app`

A debugging and testing toolkit that starts from a real transaction: any mainnet signature is decoded, its entire world reconstructed (every account and program binary), replayed locally in an embedded SVM at the exact state of its slot, stepped through, profiled, and turned into a permanent regression test. Published on crates.io (`cargo add --dev svmscope`) with a hosted web UI (vanilla JS) and a TypeScript SDK over the same HTTP engine (axum), deployed as a container on Render with GitHub Actions.

- **Decodes** the full cross-program call tree — instructions, accounts and arguments named from on-chain IDLs or native layouts, SOL/token balance deltas, per-program compute units, Address Lookup Table resolution, and failures resolved to their IDL error names.
- **Replays** in an embedded SVM (LiteSVM), reconstructing every touched account and program — following the upgradeable-loader programdata pointer for the ELF, rewinding pre-transaction SPL balances so swaps replay faithfully.
- **What-if lab** — mutate account state by field name, warp the Clock by slots/epochs/seconds, toggle runtime feature gates, and assert on outcomes and named account fields with a declarative Check DSL; can also build, sign, submit and capture transactions directly from an Anchor IDL.
- **Step debugger** — one real execution, with account state captured on entry and exit of every instruction at every depth through an observer I added to the runtime (a LiteSVM hook plus a patched program runtime), so every step and every CPI shows exactly what it changed; edit a field, argument or raw instruction data, skip or reorder instructions, and re-run against the same captured world. Parity tests prove tracing changes neither verdict, compute nor final state.
- **Compute profiler** — traces every BPF instruction the replay executes and renders a per-program, per-function flamegraph with syscall cost attribution; a build pipeline rebuilds verified mainnet programs byte-for-byte from their on-chain build records so their functions carry real Rust names (SPL Token, Phoenix, Kamino and others), with a code-shape corpus naming library functions inside stripped binaries.
- **Replay at a slot** — exact historical replay: every account and program binary rebuilt as it stood at any slot in a 30-day window, with earlier transactions of the same block replayed first; state comes from a self-run account recorder, an account-changes stream and each transaction's own record, and a fidelity certificate labels every input with its provenance rather than presenting an unproven number as exact.
- **Dependency watch** — on-chain monitoring for programs a protocol calls but does not control: an Anchor registry program (devnet) where a protocol registers with proof of its upgrade authority and lists its dependencies; the engine verifies entries against the program's mainnet upgrade authority, holds every dependency's current binary, and on a redeploy replays the protocol's recent mainnet transactions twice on identical state — old binary vs new — reporting newly failing transactions, new errors and compute deltas, delivered as ed25519-signed webhook alerts. First measurement: a Pump AMM deploy cost every Jupiter swap routed through it ~265 CU, reproduced across eight runs against the chain's own recorded compute.
- **Instruction naming for every program type** — Anchor (current and legacy IDLs), Shank/Codama, native layouts, bundled IDLs, and recovery of handler names straight from a program's binary when no IDL exists anywhere; plus an instruction-introspection view showing which instructions read the Instructions sysvar and exactly what they see.
- **Hermetic fixtures** — freeze transaction, accounts, ELFs, IDLs and the recorded on-chain outcome into one JSON file that replays deterministically offline in CI, with on-chain-fidelity assertions.
- **Engineering rigor** — security-audited with no reachable panics on untrusted input and a hard no-silently-passing-tests guarantee; ~200 tests including adversarial/DoS, an Anchor test suite against a local validator, exact-byte Borsh and trace-parity suites, wire-format snapshot tests, typed internal IDL model, MSRV, CI, full API docs.

Beethoven — Integrated On-Chain Exchange & Capital-Routing Protocol · Anchor

2026

`github.com/alizeeshan1234/beethoven-trade`

A single protocol unifying a yield-bearing fund, cross-margin lending, perps and an AMM, with adapters that route capital across external Solana venues — the flow-of-funds orchestration a collateral protocol requires.

- Yield-bearing fund with live NAV accounting and governance-gated capital allocation; a shared lending and perps market over one margin and oracle layer; protocol adapters routing funds and swaps across venues through a uniform CPI interface.
- Wrote the shared **math kernel** — fixed-point arithmetic, interest accrual, funding, liquidation and NAV — as the single source of truth for solvency-critical accounting.

Solana Circuit Breaker · Open Source · Security Primitive

2026

- A reusable on-chain primitive that auto-halts anomalous vault outflows using rolling time windows, dual-authority control, timelocked policy changes and an auto-trip lockout — integrable through one CPI check to bound the blast radius of an exploit.
  `github.com/alizeeshan1234/solana-circuit-breaker`

Anqa — Private Perpetuals DEX · Creator · Open Source

2026 — Present

`github.com/anqa-labs/anqa-core` · `anqa-red.vercel.app`

- Architected a working **private perpetuals DEX across nine devnet markets**: a zero-copy CLOB, Percolator risk kernel, Pyth marks, margin accounting and one-click session-key execution, with orders and liquidation levels kept private inside a **TEE-attested rollup** while trades settle with public, verifiable evidence on the base layer.
- Published a reproducible, credential-free **privacy proof** verifying the venue's TEE read-boundary against live devnet, plus supervised keepers, settlement/oracle/funding cranks and a resident market maker.

Alpenglow Crash Test · Open Source · Consensus Measurement & Fault Testing 2026

- Independent, reproducible benchmark of Solana's Alpenglow consensus (SIMD-0326): measured live-cluster finality (**218 ms vs 13.2 s p50**), decomposed the finality path, and fault-tested real Agave `LocalCluster` deployments to the 60%-online liveness boundary with a TowerBFT control. Rust → WASM votor engine and a five-way validation dossier. `github.com/alizeeshan1234/alpenglow-viz`

More Solana Programs · Open Source 2025 – 2026

- Native-Pinocchio **lending/borrowing market** and an **on-chain multisig** (member management, weighted voting, proposal execution) · Liquid Staking (Shank IDLs, Mollusk) · Multi-Asset AMM · Pyth price-feed integration · flash loans, vaults, escrows, token vesting and Ephemeral-Rollup transfer programs. `github.com/alizeeshan1234` — 25+ Solana repos.

OPEN SOURCE

- Merged the **compute-unit optimization guide** into Solana Foundation developer documentation.
- Contributed three merged PRs to **pinocchio-stake-program**, including `StakeStateV2` and required account layouts.
- Benchmarked **p-token** (SIMD-0266) against SPL Token across all 25 instructions with live testnet CU measurements — ~96% average compute reduction (transfer: 78 vs 4,645 CU).
- Integrated MagicBlock ER into Phoenix and Manifest order books; published private-book and on-rollup oracle variants. Fixed Token-2022 CPI compatibility in a Pinocchio token module.

SKILLS

Languages — Rust · TypeScript · JavaScript · C · Python · SQL

Solana & SVM — Anchor · Pinocchio · MagicBlock ER / PER · Percolator · Magic Actions · Pyth (Hermes, Lazer, push oracles) · SPL Token · Token-2022 · CPI · PDAs · Session Keys · Zero-copy Accounts · CU Optimization

DeFi & Settlement — Lending / borrowing markets · Perpetuals (margin, funding, liquidation) · AMMs (constant-product, multi-asset) · yield-bearing vaults & NAV accounting · collateral routing & capital allocation · cross-protocol adapters · fixed-point solvency math · outflow circuit breakers

Custody & Security — On-chain multisig · dual-authority & timelocked controls · threat modeling · adversarial privacy verification · Kani · Certora Prover · Crucible · formal verification · invariant design · Mollusk · LiteSVM

Monitoring & Observability — on-chain dependency monitoring · signed webhook alerting · account-changes streams (Substreams, gRPC) · historical state reconstruction · replay-based upgrade impact analysis

Data & Infra — Yellowstone gRPC / Geyser indexing · custom RPC services · Parquet · ClickHouse · PostgreSQL · WebSocket streaming · Docker · AWS · GitHub Actions

EDUCATION & CREDENTIALS

ICFAI University
B.Sc. Computer Science

- **Turbin3** — Builders, Accelerated Builders and Advanced Pinocchio graduate.
- **Blueshift** — verified Solana program challenges completed across Pinocchio and Anchor.