

Jemuel Daniel

AI Engineer & Full-Stack Software Engineer

Manila, Philippines · Remote / International · UTC+8

[GitHub / jemueldaniel](#) [in LinkedIn / thisislondaniel](#) [✉ thisisdanielsph@gmail.com](mailto:thisisdanielsph@gmail.com)

Positioning

AI-native engineer who can move from model/research work to product UI, backend authority, durable data, system runtime and production verification.

ENGINEERING PROFILE

AI-first engineer and researcher who builds complete software systems from training/evaluation and benchmark infrastructure through retrieval, agent execution, polished interfaces, authenticated APIs/SDKs, MCP/WebMCP-oriented browser tooling, backend/data reliability, compiler/runtime systems, smart-contract flows and release/recovery engineering. I separate probabilistic intelligence from deterministic authority so complex systems stay useful, observable, recoverable and defensible.

- AI / ML engineering
- Research / benchmarks
- Agent systems
- API / SDK / MCP
- React / Next.js
- Playwright
- Custom language / compiler
- Rust / Solidity

DEMONSTRATED ENGINEERING DEPTH

Source-backed work, summarized for hiring

AI + MACHINE LEARNING

Training, evaluation and model infrastructure

- Python/PyTorch training/evaluation control planes with dataset admission, secret rejection, distributed execution configuration and checkpoint lineage.
- Held-out and independent evaluation gates that keep evidence separate from production acceptance.
- Adapter/QLoRA-style training patterns, GPU execution boundaries and deterministic evaluation controls.

RESEARCH + BENCHMARKS

Reproducible engineering research

- Hypothesis and benchmark design with frozen metrics, artifact identities, budgets and negative-evidence capture.
- Independent proof/decision boundaries so a run cannot accept its own result.
- Research work across model quality, agent/tool behavior, retrieval, runtime and failure analysis.

AGENTS + INTELLIGENCE

Tool-using, recoverable agent systems

- Model routing/fallback, tool use, persistent state and bounded retrieval/context reduction.
- Schedules/long-running work, scoped actions, evidence and replay-aware recovery.
- Unknown external outcomes reconcile the same operation identity before another effect is started.

FULL STACK + INTERFACES

Complex product surfaces made usable

- React/Next.js streaming workspaces, dashboards, search/filter flows, drawers/bottom sheets and mobile/mini-app surfaces.
- Keyboard/focus behavior, accessibility-minded states and explicit cancel/resume/reconcile UX.
- Node/Python services, auth, workers and SQL/PostgreSQL behind typed UI contracts.

API + DEVELOPER PLATFORM

API, SDK, MCP and browser-tool architecture

- Authenticated APIs/SDKs and JSON-RPC/MCP tool catalogs with actor/scope filtering and stable operation identity.
- Idempotency, payload binding, receipt-based outcomes, bounded paging and session recovery.
- WebMCP-oriented browser-tool contracts projected to the same backend authority without overstating production certification.

LANGUAGE + SYSTEMS

Compiler/runtime, Rust and transaction correctness

- Custom language work across lexer/parser, typed IR, lowering, deterministic VM execution and WebAssembly direction.
- Rust capability/lease/fencing components with deterministic identities and fail-closed tests.
- Solidity + wallet/transaction flows with replay protection, signatures, receipts/finality and settlement boundaries.

AI intelligence on top. Deterministic systems underneath.

My strongest work sits at the boundary between model intelligence and reliable software: models handle ambiguity, generation and strategy; typed contracts, backend authority, state, verification and recovery handle what must be correct.

AI

training/eval · retrieval · routing · agents

RESEARCH

hypotheses · benchmarks · evidence · replay

UI

streaming · responsive · mobile · accessibility

API

APIs · SDKs · MCP · browser tools

RUNTIME

services · jobs · idempotency · Postgres

TRUST

auth · contracts · fencing · rollback

CORE STACK		Strongest day-to-day footprint first
PRIMARY LANGUAGES	TypeScript · JavaScript / Node.js · Python · SQL / PostgreSQL · Solidity	
SYSTEMS / RUNTIME	Rust · compiler / HIR-MIR style IR / VM · WebAssembly / WIT direction · Shell / PowerShell	
AI / RESEARCH	PyTorch · training/evaluation · retrieval/context · agents/tools · benchmark and experiment design	
FRONTEND / TEST	React · Next.js · streaming/mobile interfaces · accessibility · Playwright-oriented browser automation	
PLATFORM / DELIVERY	API / SDK · MCP · WebMCP-oriented architecture · Docker · CI/CD · cloud/edge runtimes · observability	

ROLES I MATCH

AI Engineer / Applied AI Engineer

Training/evaluation, agent orchestration, retrieval/context, tool execution and reliability.

Research Engineer

Benchmark design, experimental protocols, evidence ledgers, negative-result analysis and reproducibility.

Full-Stack AI Engineer

React/Next.js, Node/Python backends, APIs, data stores, auth, streaming UX and deployment.

AI Platform / Developer Experience Engineer

SDKs, MCP, API contracts, browser-tool integrations, execution runtimes and capability discovery.

Compiler / Runtime or Smart Contract Engineer

Language/runtime systems, Rust, Solidity, transaction lifecycle, settlement/finality and fail-closed integration.

WHAT I CAN OWN END TO END

- AI-native product architecture from model/research boundary to user-visible result.
- Training pipelines, checkpoint lineage, held-out evaluation and promotion policy.
- Research/benchmark workflows with preregistered metrics, evidence identity and independent verification.
- Agent execution, model routing, retrieval/context, tool catalogs and long-running recovery.
- API/SDK/MCP integration and WebMCP-oriented browser-tool contracts.
- Frontend state machines: streaming, blocked, failed, cancelling, resumed, reconciled and verified.
- Backend command/action registries, validation, auth, idempotency and durable receipts.
- PostgreSQL schemas/migrations, transactional outbox and recovery-oriented durable state.
- Compiler/runtime engineering for a custom language, typed IR, deterministic VM and Wasm directions.
- Rust capability/lease/fencing systems with fail-closed security boundaries.
- Solidity clients/contracts and money-moving flows with finality and replay protection.
- Production hardening: tests, Docker, CI/CD, browser verification, edge/cloud runtime, observability, canary/readback and rollback.

PUBLIC CODE PROOF — DANIELCODES

AI / ML

Train/eval gates, dataset admission and content-addressed checkpoints.

Research

Hypothesis, benchmark identity, run receipts and independent decision.

API / MCP

Actor-scoped tools, operation IDs, idempotency and receipts.

Compiler / Rust

Tokenizer/parser/IR plus capability/fencing safety boundaries.

ENGINEERING QUALITY BAR

Truthful status

Source, tests, artifacts, deployment and live runtime are tracked separately.

Safe retry

Unknown external outcomes reconcile the same operation before another effect.

Precise boundaries

WebMCP is architecture/integration work; language self-hosting remains experimental engineering.