Christopher McNamara

Ormond Beach, FL · 100% Remote · chris@ormondlabs.com · github.com/cmm219 · linkedin.com/in/chris-mcnamara-823b41408 · Ormond Labs LLC

FULL-STACK PRODUCT ENGINEER — BACKEND, APPLIED AI, DATA SYSTEMS

Engineer building production-grade full-stack, backend, and AI systems in Python, React, and PostgreSQL: reconciliation and auditability, LLM orchestration and RAG, safe data import, and the operational monitoring to run them unattended.

SUMMARY

- Ship and operate production systems end-to-end: REST API design, PostgreSQL schema and migrations, auth and RBAC, billing, WebSockets, cloud deploy. Primary stack Python/FastAPI/SQLAlchemy and React/TypeScript.
- Build for correctness under adversarial conditions: immutable audit journals, runtime invariant checks, atomic rollback, risk-scored approvals, and fail-closed handling on any ambiguity.
- Applied-AI depth: multi-provider LLM routing (OpenAI, Claude, Grok), RAG and vector retrieval, and evaluation harnesses with holdout windows and statistical significance gating.

EXPERIENCE

Ormond Labs LLC — Founder & Software Engineer · Remote · 2023–Present

Independent studio building production full-stack and AI systems for my own products and for a private operator.

Production financial-operations platform (built and operated for a private operator) — Python, FastAPI, SQLAlchemy, PostgreSQL, vanilla JS, WebSockets, JWT, APScheduler

- Sole engineer of a production platform that moves real money daily: **200+ REST endpoints, 40+ tables, 24,000+ transactions and ~$56M in gross ledgered volume**, with role-based access control and real-time WebSocket updates.
- Built a **near-real-time reconciliation engine** that continuously compares an operator's daily spreadsheet against the system's computed ledger and alerts the team to mismatches (rows in one system but not the other, plus amount deltas), with failure-isolated comparison streams and residual-based gap classification, so a spreadsheet in daily use could be reconciled without abandoning it.
- Enforced correctness with a **runtime balance invariant** (|full − ledger − activity| < $0.01) that trips before a bad number can propagate; zero unresolved deltas above one cent across all accounts.
- Built **safe multi-source bulk import**: layered validation, SHA-256 + fingerprint dedupe, a dry-run preview sharing one detector with the commit path, staging tables, atomic commit/rollback, and fail-closed handling of ambiguous records.
- Made every state change auditable: a **three-layer append-only audit trail** (ORM listener + session guard + database trigger, CI-probed) with request-id correlation, immutable adjustment journals (voids as counter-entries), and risk-scored dual approvals.
- Ran it unattended: an open → pending-close → closed → locked period state machine, and a **job-monitoring framework** (heartbeats + external dead-man's-switch + an ~8-check watchdog daemon) so failures surface before they matter.

SELECTED PROJECTS

Aura — Full-stack AI content SaaS (built on a personal v1; built, not launched) — FastAPI, PostgreSQL, SQLAlchemy/Alembic, React 19, TypeScript, Vite, Tailwind, Stripe, Render/Cloudflare

- Built a multi-tenant AI SaaS end-to-end: per-tenant JWT auth, tiered Stripe billing, PostgreSQL with SQLAlchemy/Alembic migrations, a React 19/TypeScript frontend, and cloud deploy.
- Built a provider-agnostic LLM router with fallback chains across OpenAI, Claude, and Grok: task-type routing and transparent handling of differing provider API formats.

- In the personal v1, built a statistical evaluation lab: up to 200 experiments per session across 4 non-overlapping holdout windows, promotion gated on 3-of-4 windows improving at $p<0.05$, with config versioning and rollback.

OLAT — Systematic-trading research apparatus — Python 3.11, SQLite, pytest (263 tests), Alpaca paper API

- Built a 7-phase research apparatus to validate trading edge before deploying capital: an append-only signal ledger with SQL-enforced immutability and point-in-time timestamps, a deterministic backtester with injectable lookahead guards and a pre-registered null, an adversarial AI research agent forced to produce disconfirming evidence, an event-sourced paper portfolio, and an approval-gated broker integration with a hash-chained audit.
- Then wrote the NO-GO memo myself: 263 tests prove the infrastructure, not the edge — so no capital deployed. Every phase passed dual-model review before merge.

Ormond Labs — Automated lead-generation and website-builder pipeline — Python, Node.js, Playwright, Claude API, Apify

- Built an end-to-end pipeline that scrapes and scores local businesses with weak web presence, generates positioned marketing copy and multi-touch outreach with the Claude API, and produces deployable static-site mockups, with human approval gates before anything sends.
- Staged, resumable architecture: per-lead artifact folders, a CSV-backed CRM as the pipeline's state source of truth, JSONL run logs, and two build modes (AI-generated vs real-photo assets).

AGENTIC ENGINEERING SYSTEM

- Built multi-model development tooling in early 2026 (run logs on disk): an autonomous build loop with Claude as builder and GPT/Grok as adversarial auditors, and a policy-driven consensus orchestrator with multi-round plan debate, deadlock escalation, and an enforced launch gate — a pattern I still run today as dual-model review on every money-path change (documented catches include a five-figure accounting bug).
- Operate a Postgres-native RAG/memory pipeline as production infrastructure for my agents — two-source ingestion (notes + symbol-aware code indexing), hybrid retrieval over 3,000+ documents, and a write-back protocol so engineering context compounds across sessions — adopted after building a measurement harness to prove it earned its place.

SKILLS

- **Languages:** Python, JavaScript, TypeScript, SQL, Bash, PowerShell
- **Backend:** FastAPI, SQLAlchemy, Alembic, Node.js, REST API design, WebSockets, JWT auth, async patterns
- **Frontend:** React 19, Next.js 16, Vite, Tailwind CSS, Chart.js; Textual, Flet, Electron
- **Data:** PostgreSQL, SQLite (WAL), Supabase (RLS, edge functions), Prisma, ELT pipelines, schema design and migrations, RAG, embeddings, vector retrieval
- **AI / LLM:** OpenAI GPT, Claude API, xAI Grok, OpenRouter, local models (Ollama, LM Studio); multi-agent orchestration, task-type model routing, provider fallback, evaluation harnesses
- **Infra / DevOps:** Docker, Vercel, Cloudflare (Workers, Pages, R2), Render, Turborepo, git worktrees, VPS deploy
- **Testing / QA:** pytest, Playwright, automated code review, dry-run and rollback verification

BACKGROUND

Self-taught engineer since 2023; 23 repositories built in under three years with production ownership across the stack. Fifteen-plus years as a professional poker player before software: probabilistic thinking, EV analysis, and bankroll/risk discipline that show up directly in the immutable journals, invariant checks, and fail-closed defaults above.