

Tristan Mitsuji O'Donnell | Software Engineer

Los Angeles, CA | tristanodonnell100@gmail.com | 818-439-9662

[linkedin.com/in/tristan-odonnell-dev](https://www.linkedin.com/in/tristan-odonnell-dev) | github.com/TristanODonnell | tristanodonnellportfolio.com

SUMMARY

Software engineer focused on backend systems and data infrastructure: DAG-driven execution engines, structured pipelines, and reproducible analytics workflows. Seeking software engineering internships, with a focus on backend and data systems.

TECHNICAL SKILLS

Languages: Python, SQL, TypeScript, JavaScript, C++, C#, HTML/CSS

Frameworks & Libraries: FastAPI, SQLAlchemy, Alembic, Rest APIs, React, Pydantic

Systems & Design: Background Workers, State Machines, Idempotent Systems, DAG Orchestration, Retry Semantics, Transaction Management

Databases & Data: PostgreSQL, Data Modeling, Migrations, Pandas, NumPy, Parquet

Machine Learning: Scikit-learn, Random Forest, Feature Engineering

Testing & DevOps: Pytest, Git, Docker, CI/CD (GitHub Actions), Environment Configuration

Observability: OpenTelemetry, Grafana, Loki (LGTM stack), Dashboarding

PROJECTS

US Equities Autonomous Trading Platform | Backend Architecture & Data Infrastructure | 2026 – Present

Python, FastAPI, PostgreSQL, SQLAlchemy, Alembic, React, Docker

- Architected a full-stack quantitative trading platform (FastAPI, PostgreSQL, React) spanning contracts, storage, application, and API layers across 23 subpackages and 90+ REST endpoints, with 89 Alembic migrations tracking iterative schema evolution.
- Designed a defense-in-depth trading safety system built for eventual live deployment: four chained pre-trade gates (environment, account allowlist, armed/disarmed state, DB-persisted kill switch), shadow-mode compute-but-suppress execution, and configurable pre-trade risk limits (gross, per-symbol, and sector caps); implemented concurrency-safe, idempotent capital reallocation using PostgreSQL advisory locks.
- Built runtime-enforced research-integrity guards — look-ahead and survivorship-bias checks — validated with exact-assertion tests, plus a from-scratch Black-Litterman portfolio optimizer numerically verified against the CAPM formula.
- Engineered a deterministic historical backtesting and simulation engine (T+N settlement, seeded stochastic fills, byte-identical replay) sharing execution-policy code with the trading path to guarantee sim/live parity ahead of a planned paper-trading rollout.
- Instrumented the platform for production observability — OpenTelemetry tracing and metrics across 261 instruments, queried and visualized via a self-hosted Grafana/Loki (LGTM stack) — and maintained 4,080 automated tests (unit tier plus a Dockerized-Postgres CI integration tier) with near-zero skip/xfail debt, including a schema-drift suite that diffs every data contract against its ORM model.
- Built a staged strategy-research funnel — registry-driven candidate generation (grid/random/mutation, ~854-skeleton composite-rule assembler) filtered through cheap-simulation → full-window → walk-forward → Monte Carlo validation stages — with deterministic SHA-256 seeding, lineage-hashed simulation caching, and checkpoint/resume for restartable batch runs.
- Completed a 522-trading-day (2023–2024), \$1M-notional backtest across a 100-symbol universe: 522/522 ticks succeeded, 29,654 real fills executed, ending at +261.7% gross return with a -61.5% max drawdown under a 12% governance limit.

Config-Driven DAG Execution Engine | Workflow Orchestration & Runtime Architecture | 2026

Python, YAML, DAG Compilation, Retry Semantics, Run-State Persistence

- Architected a compile-then-execute DAG workflow engine that transforms declarative YAML pipeline specifications into validated execution plans, with dependency graph validation, cycle detection, and topological scheduling.
- Designed a retry-aware runtime with a per-step state machine and structured run-state persistence (run_manifest.json).
- Built artifact tracking and a step registry abstraction to decouple orchestration logic from execution logic and enable reproducibility.

Algorithm Selection Meta-Learning System | ML Experimentation & Model Evaluation | 2026

Python, Scikit-learn, NumPy, Random Forest, Feature Engineering

- Designed a supervised meta-learning system to predict the optimal graph-search algorithm (BFS/DFS/A*) from structural maze features, using a labeled dataset of 144 mazes generated via oracle benchmarking.
- Engineered a topology-aware feature set (dead-end ratio, reachable ratio, shortest path length, open-space density) to encode search-space complexity.
- Trained and evaluated a Random Forest (200 trees) achieving 100% test accuracy and zero average regret, introducing regret-based evaluation to measure real decision cost beyond accuracy.

Pokémon Data Explorer | Full-Stack / Backend Developer | 2025

Python, FastAPI, React

- Built a full-stack Pokémon data explorer with normalized Pydantic models, TTL caching, and CORS-configured deployment on Render.

PROFESSIONAL EXPERIENCE

Project Associate — Global Early Careers / Talent Brand | Activision Blizzard, Irvine, CA | 2022 – 2024

- Supported national internship campaigns that drove 3.3M site visits and 3K+ applications across 835+ schools through coordinated branding and outreach initiatives.

EDUCATION

Bachelor of Science — Applied Computer Science, University of Colorado Boulder | 2026 – 2028

Bachelor of Science — Business Administration, California State Polytechnic University, Pomona | 2021

Certification: Unity Developer Bootcamp, UC Irvine Continuing Education | 2024

Relevant Coursework: Data Structures, Algorithms, Discrete Structures, Computer Architecture, Linear Algebra