TAI TRAN

Brisbane, Australia
0479 049 287 | Email: taitranz@ymail.com LinkedIn: linkedin.com/in/taitranz Portfolio: taitranz.com
GitHub: github.com/Taitranz Blog: blog.taitranz.com

PROFILE

Software engineer focused on outcomes: turning ambiguous requirements into shipped, maintainable software, owning delivery end-to-end, and balancing timelines, engineering capacity, and technical trade-offs. Primarily Python, but language-agnostic and happy to use the right tool for the job.

TECHNICAL SKILLS

- Languages: Python (primary), Java, C#, SQL, HTML, CSS, JavaScript.
- Frameworks & Libraries: FastAPI, PySide6 (Qt6), JavaFX, VisPy (OpenGL), NumPy, CuPy, Pandas, Flask, SQLAlchemy, Alembic.
- Databases: PostgreSQL, SQLite (WAL mode), SQLAlchemy ORM.
- Architecture & Practices: Clean Architecture, Domain-Driven Design, CQRS, Event-Driven Architecture, repository pattern, dependency injection, MVVM, type safety (MyPy), linting (Ruff), testing (pytest, pytest-qt, unit testing), CI with coverage gating, Logging & Observability, Information Security, REST APIs, Software Architecture Design, OOP.
- Tools & Platforms: Git/GitHub, Stripe, Cloudflare, OANDA API, Figma, MkDocs, Parquet.
- Domains: Real-time trading & charting, AI-native workflows, Cybersecurity, GPU Computing (CUDA), High-Performance Computing, Financial Data Visualisation, Machine Learning, UI/UX.

PROFESSIONAL EXPERIENCE

Lead Software Engineer (Contract) | Lewy Security (Agentic Security Hardening)

Nov 2025 - Present | Sydney, New South Wales, Australia (Remote)

Major Duties:

- **End-to-End Architecture Ownership:** Architected a modular security platform in Python using strict Plugin Architecture. Designed the "Agent Interface" abstraction to unify 12+ distinct tools (SAST, IaC) under strict type safety.
- **Security Analysis Orchestration:** Engineered a normalization engine that maps disparate tool outputs into a single "Finding" model. Built a parallelized pipeline to ingest and deduplicate findings locally without blocking developer workflows.
- **Hotspot Identification:** Implemented risk-density modelling to isolate the top ~10% of security-sensitive hotspots, enabling teams to prioritize manual review on high-risk surface areas rather than broad scanning.
- **AI-Assisted Triage:** Integrated LLM-based triage to cluster thousands of raw alerts into semantic groups. Mitigated false-positive fatigue by cross-referencing findings against known safe patterns and allowlists.
- **Controlled Remediation:** Implemented a "Human-in-the-Loop" fix workflow. Constrained LLM patch generation to scoped windows (<8k tokens) and verified syntax to ensure build-safety.
- **Auditability & Tooling:** Developed a developer-centric CLI optimized for near-instant startup via lazy module loading. Built a local SQLite tracking system to maintain a reproducible audit trail of detection events.

Successes:

- **Review Efficiency:** Transformed fragmented reviews into batch workflows, enabling triage of 50+ related issues in minutes rather than hours.
- **Cost & Performance:** Reduced LLM operational costs by ~40% via AST-based context slicing, ensuring only relevant code snippets were sent for inference.
- **Rapid Extensibility:** The modular design allowed new analysis domains (e.g., Secrets detection) to be added as plugins in days rather than weeks, with zero changes to core orchestration logic.
- **Stability:** Maintained >90% test coverage on the core normalization engine, ensuring consistent behavior across diverse project structures.

Lead Software Engineer | Hooper Music Studio (Contract)

Nov 2025 - Jan 2026 | Brisbane, Queensland, Australia (On-site)

Major Duties:

- **System Architecture:** Architected a modular SaaS platform using FastAPI and PostgreSQL. Designed a strict schema with 20+ SQLAlchemy entities to enforce referential integrity across tightly coupled domains (Users, Schedules, Financial Ledgers).
- **Advanced Scheduling Engine:** Engineered a deterministic booking kernel that evaluates recurring patterns against complex constraints (holidays, 24-hr lockouts). Implemented temporal logic to handle "series regeneration" on conflict, ensuring schedule consistency without data corruption.
- **Proprietary Financial Core:** Built a double-entry accounting system to replace manual tools. Implemented idempotent transaction processing and atomic writes to ensure 100% financial accuracy between the bank feed and internal P&L.;
- **Payments & Commerce:** Integrated Stripe Webhooks with a digital asset store. Built a reconciliation layer that links point-of-sale events directly to the General Ledger, automating revenue recognition.
- **Security & Ops:** Enforced strict RBAC and OAuth2 authentication boundaries. Established a "Shift Left" quality pipeline using strict typing (mypy) and ruff linting to catch logic errors in complex accounting flows before deployment.

Successes:

- **Digital Transformation:** Reduced administrative overhead by ~90% by replacing fragmented spreadsheets with a unified event-driven workflow for payroll and invoicing.
- **Efficient Frontend:** Delivered a responsive UI using a lightweight, dependency-free ES6 architecture. Reduced bundle size and maintenance overhead by avoiding heavy frameworks for the core administrative dashboard.
- **Rapid Feature Delivery:** Leveraged a modular service layer to ship complex business logic (e.g., Lesson Escalations, Credit systems) in 2-week sprints while maintaining zero regression in the financial core.

Lead Software Engineer (Contract) | Valgo Trading

Jun 2025 - Dec 2025 | Brisbane, Queensland, Australia (Hybrid)

Major Duties:

- **System Architecture:** Architected a modular desktop platform (200k+ LOC) using Python and Clean Architecture. Applied CQRS to strictly separate deterministic trading logic from statistical analysis, ensuring high testability and modular deployment.
- **AI Agent Integration:** Engineered an autonomous decision layer using fine-tuned LLMs. Mitigated hallucination risks by grounding agent reasoning in structured "Market State" objects rather than raw price noise.
- **GPU Rendering Engine:** Built a custom VisPy/CUDA pipeline to render 10,000+ interactive candlesticks. Wrote custom shaders to achieve 160fps performance, bypassing the render-loop bottlenecks of standard Python plotting libraries.
- **Concurrency & Threading:** Developed a thread-safe Event Bus to manage high-throughput data ingestion. Solved GIL-related UI freezing during heavy AI inference by implementing a non-blocking "snapshot" pattern for state synchronization.
- **Execution & Data:** Integrated real-time OANDA feeds into a unified state engine. Designed the system to use identical logic for both backtesting and live execution, ensuring simulation validity.

Successes:

- **Performance Optimization:** Achieved >160fps rendering for high-density datasets by migrating from CPU-bound plotting to a custom GPU pipeline, eliminating UI latency.
- **Reliable Autonomy:** Delivered a stable "Human-in-the-loop" system where LLM decisions are transparently explained and visually correlated with market structure in real-time.
- **Stability:** Eliminated race conditions between the AI and UI threads by enforcing immutable state snapshots, ensuring data consistency during high-volatility events.

KEY PROJECTS

PySide6 Date Range Popover - Reusable Date/Range Picker Component - Python, PySide6, Qt, pytest, pytest-qt, MkDocs

- Built a reusable date/date-range picker widget with clean Qt signals to decouple UI from business logic.
- Packaged as an installable library with semantic versioning, documentation (MkDocs), and CI-tested coverage (pytest + pytest-qt).

EDUCATION

Bachelor of Information Technology - Major: Computer Science; Second Major: Computational and Simulation Science

Queensland University of Technology (QUT), Brisbane

WRITING & TECHNICAL COMMUNICATION

- “Valgo: An Agentic Trading System for EUR/USD” - deep-dive on an agentic EUR/USD trading platform combining Clean Architecture, event-driven design, and an LLM-backed agent to go from raw market data to autonomous execution.
- “Your AI Assistant Doesn’t Need Better Inference. It Needs Better Documentation” - described a two-layer documentation architecture (AGENTS.md entry point + .agent tree) to improve AI-assisted development by giving tools structured, high-signal context.
- “NOV11 Valgo Progress Update” - documented architectural decisions, performance characteristics, and lessons learned from scaling a Clean Architecture trading terminal.

REFERENCES

References available upon request; reference page will be provided.