

DANIEL GUARNIERI

Full-Stack Software Engineer — AI-Assisted & Agentic Development

Clover, SC · (240) 750-9598 · Danny@3dcryptobots.com · github.com/DJG3DK · 3dcryptobots.com

Ask my work anything → demo.3dcryptobots.com — a chatbot trained on these projects. Pull on any claim below and it will tell you what the constraint was, what broke, and how it was fixed.

PROFESSIONAL SUMMARY

Full-stack developer with production systems live in the real world — e-commerce storefronts serving real customers, an autonomous coding agent, and an algorithmic trading system running live on my own capital — all built solo with React, TypeScript, Node.js, and Python. My longest-running project is an algorithmic trading system on Bybit perpetual futures that I run live with my own capital, where the cost of an unreviewed AI-generated bug is real money out of my own account. That constraint pushed me from using agentic tools to building them: 3D-Agent is my own LangGraph/deepagents coding agent, designed around the premise that an agent's own claim of "done" carries no authority — checks are re-run independently and an external review service has to sign off before anything merges. Background of 20+ years in automotive service management, which is where the underlying habits come from: isolate the variable, read the data, find the root cause, stay calm under pressure. Looking for a full-stack role where that combination — self-directed technical depth plus judgment about when to trust a tool and when to slow down — is the actual job.

TECHNICAL SKILLS

Languages	TypeScript, JavaScript, Python, SQL
Frontend	React 19, Vite, Tailwind CSS, component architecture, TypeScript typing/DTOs
Backend	Node.js/Express, FastAPI, REST APIs, WebSocket real-time systems, JWT & session auth, TOTP 2FA, CSRF/rate limiting
AI / Agent Eng.	LangGraph, deepagents, LangChain, LiteLLM model routing, LangSmith tracing, human-in-the-loop approval gates, agent memory & consolidation, Docker-sandboxed tool execution
AI-Assisted Dev	Claude Code and agentic workflows — shipping, reviewing, and validating AI-generated code in production
Quant / ML	Logistic regression, walk-forward validation, Platt calibration, backtesting, Sharpe/drawdown analysis
Data / Infra	PostgreSQL, Prisma, Redis/BullMQ, Docker, Stripe, Linux VPS deployment (Nginx, PM2), pytest/CI test suites

TECHNICAL PROJECTS

3D-Bot — Algorithmic Trading System | Solo project · 3DCryptobots LLC

Remote · Apr 2020 – Present · Node.js · TypeScript · Python · WebSocket

My first project and longest-running one — an algorithmic trading system on Bybit perpetual futures, rebuilt through several architectural iterations before reaching live operation. Currently trading my own capital only; multi-tenant SaaS work and a live track record are the remaining prerequisites before it takes users. Designed, built, and maintained solo, end-to-end, with AI-assisted and agentic development as the primary build method throughout.

- Architected the full system as two cooperating services — a live trading engine and a separate compute process — so backtests, parameter optimization, and ML retraining never delay a live order.
- Built a confluence-scored entry strategy combining support/resistance, EMA structure, Fibonacci levels, market structure, liquidity sweeps, and order-book depth, with cool-off and structure gates that veto weak setups before they're even scored.
- Added an ML entry gate (per-direction logistic regression models) trained with a temporal train/validation split and Platt calibration specifically to avoid look-ahead leakage, with a live calibration scorecard and an audit trail of every blocked entry.
- Diagnosed and rebuilt a survivorship-bias flaw in that gate: it had been training only on entries that already passed the upstream filters, learning $P(\text{win} \mid \text{approved})$ instead of $P(\text{win} \mid \text{signal fired})$. Widened the training population to all fired triggers and tightened the adoption bar.
- Built regime-adaptive walk-forward optimization that re-tunes strategy settings per market condition and only switches a pair to new settings live after they're validated against current live performance.
- Implemented security controls appropriate for real financial credentials: AES-encrypted API key storage, password + Telegram 2FA login, JWT auth on httpOnly/secure/sameSite cookies, CSRF protection, and rate limiting.
- Maintain a real automated test suite (auth flows, cookie handling, indicator/extractor parity, ML pipeline integration) as a deliberate practice — AI-assisted code gets tested and reviewed before it ships, not trusted by default.

3D-Agent — Autonomous Coding Agent | Source-available · github.com/DJG3DK/3D-Agent

Python · LangGraph · deepagents · FastAPI · PostgreSQL · React/Vite · Docker

An agent that plans, builds, verifies, and ships real production changes against real repositories, with a human-in-the-loop review stage. Built to solve my own problem: agentic coding tools that confidently report success on work that doesn't actually pass.

- Designed the verification gate as the core architectural idea — the agent's own "done" status carries no authority. The gate independently re-runs the project's real typecheck/lint/test suite, then hands the commit to a separate review service; a NEEDS_FIXES verdict loops back into the same thread with findings, and only READY merges and deploys.
- Solved the mid-plan commit problem — committing while the agent's own todo list still has open items means the reviewer critiques deliberately-unfinished work and bounces the task on findings that describe scheduled work. Implemented as a nudge budget rather than a hard gate, so a model that stops maintaining its list can't strand working code permanently.
- Sandboxed all tool execution in a Docker container that mounts only the target repo, so a shell command can't reach other projects or host secrets; delegated work to constrained subagents (a read-only investigator with no write or shell tools at all, and a test-writer required to run checks itself before reporting done).
- Made every task recoverable — escalated, stopped, budget-exhausted, or orphaned by a backend restart mid-run. Full graph state persists in a Postgres checkpointer, so resuming continues the same inner thread instead of starting over; the dashboard detects orphaned tasks and offers a resume.
- Built persistent per-project agent memory with a scheduled consolidation pass that distills episodic task records into durable patterns, plus on-demand skill files so situational domain knowledge doesn't permanently bloat the context window.
- Made model selection a runtime concern, not a code change — every role (planner, coder, investigator, test-writer, reviewer, classifier) is a named alias pinned in a shared LiteLLM router and swappable from the dashboard. A per-turn classifier escalates hard problems to a stronger model.
- Enforced two limits the model cannot override: a per-task USD budget ceiling checked after every model call on both the coordinator and every subagent, and an iteration ceiling on the work/verify loop.
- Shipped a real-time React/Vite dashboard — live task output over WebSockets with a REST snapshot on reconnect (so a page opened mid-task shows real history instead of starting blank), inline approval controls, multi-viewer support, and admin analytics for spend by project and category, per-role model usage, and tool-call reliability.
- Hardened observability before enabling it — wrote a redacting LangSmith client that scrubs credential-shaped values, DSN passwords, bearer tokens, JWTs, and PEM blocks from trace payloads, since the agent's bash tool can read arbitrary sandbox files and the approval gate can't know what a command's output will contain.
- Full auth stack: argon2id password hashing, RFC 6238 TOTP 2FA with one-time recovery codes, revocable server-side session cookies, and per-user repo allow-lists. Covered by a pytest suite running against in-memory stores and mocked model calls — no live infrastructure required.

Portfolio Agent | demo.3dcryptobots.com

Public conversational résumé — built on 3D-Agent

- Built a public-facing agent that answers questions about my work — architecture decisions, what broke, and how it was fixed — with web search available and explicit boundaries preventing disclosure of source code, schemas, or trading strategy specifics.

Artistic Gamut | Full-Stack Developer

Remote · Live in production

- Built a full storefront and admin panel (React, MongoDB, Stripe) with automated WebP image processing and USPS shipping integration.
- Deployed and maintain the platform on a Linux VPS via Nginx and PM2, including point-of-sale sync with EPOS Now for in-person retail.

3DSteals — E-Commerce Platform | Solo project · 3dsteals.com

Remote · Live in production · pnpm monorepo — NestJS · Prisma/PostgreSQL · React + Vite · BullMQ · Stripe

A dropshipping platform built as three deployables from one monorepo — customer storefront, admin console, and a NestJS API that owns the data model, supplier integrations, order routing and payments. Products come from multiple dropship suppliers, each with its own API, quirks and identifier scheme.

- Normalised several supplier APIs into one catalog model, with stock and price sync and order routing to whichever supplier fulfils a given item.
- Diagnosed a class of failure worth more than the fix: one supplier exposes two product identifiers and its detail endpoint accepts either — but only under the matching parameter name. Querying with the wrong one returns Success and an empty result rather than an error, which is indistinguishable from "this product doesn't exist". Staging failed for products plainly in the catalog until the lookup was made to try one identifier, then the other, before concluding anything is missing.

- Built an in-admin AI sourcing agent that researches products against market evidence — ad-intelligence and scraping tools for what is actually selling — then cross-references the connected suppliers' catalogs so a candidate must clear both demand and fulfillable margin before an operator sees it.
- Hardened that agent against its own worst failure: the candidate list is JSON the model writes by hand, so a hallucinated product ID doesn't fail loudly — it imports the wrong real product. Every ID is now verified against what the conversation actually saw before it can be staged, and anything unverified is stripped with the reason surfaced.
- Server-side Stripe verification that the payment intent's amount and currency match the order before it is marked paid, plus an explicit oversell path that flags for manual review rather than silently accepting an unfulfillable order.
- Image-quality scoring (resolution, subject framing, edge clipping, blur) that auto-promotes the best gallery image to hero, so poorly cropped supplier photography doesn't land on category pages.
- ~680 tests across the monorepo plus a CI-mirroring suite: typecheck, lint, unit, coverage, dependency audit, build, database drift and seed checks, e2e, and a secret scan.

C&S Home & Occasion Services | Client work · cshomeandoccasionservices.com

Remote · Live in production · React + TypeScript · NestJS · PostgreSQL

- Marketing site and lead-capture system for a home services business. Pages are prerendered to static HTML so a crawler never has to execute JavaScript; only the enquiry form talks to an API, which keeps the site up even when the API is not.
- Built an automated SEO audit that runs against the live site the way a crawler does, with JavaScript disabled — because an empty prerendered shell still returns HTTP 200 and still looks correct in a browser. Testing the built artifact would have passed.

PROFESSIONAL EXPERIENCE

Service Advisor | Gastonia Nissan — Hudson Automotive

Gastonia, NC · May 2026 – Present

- Ranked in the top 3 of the department for labor and parts sales within the first month.
- Built a new customer base quickly while retaining loyal clients who now drive in from the previous store.

Service Advisor | Honda of Indian Trail — Hudson Automotive

Indian Trail, NC · Jul 2024 – May 2026

- Maintained a strong Customer Satisfaction Index in a high-volume Honda franchise; recognized by store leadership for retention and performance.

Service Advisor | Foreign Cars Italia

Charlotte, NC · Jul 2023 – Jul 2024

- Primary service contact for Maserati clients; scope grew to include Aston Martin, Alfa Romeo, and Bentley as trust in client handling increased.

Earlier — Service Management | Ourisman, AutoNation, Fitzgerald, Leith, Stadium Toyota, Jerry's Chevrolet

VA, MD & NC · 2001 – 2019

Advisor through to **Service Manager**, running full department operations — scheduling, technician teams, escalations and performance reporting on Reynolds & Reynolds DMS. Departments consistently ranked top of region for CSI and sales. Built an aftermarket accessory department from scratch at Stadium Toyota.

EDUCATION & SELF-DEVELOPMENT

- Self-taught developer since 2020 — JavaScript, TypeScript, React, Node.js, Python — learned by building and shipping real production systems rather than through tutorials or coursework.
- Independent study of agent architecture: LangGraph state machines, tool-use sandboxing, agent memory and consolidation, and human-in-the-loop verification design.
- Independent study of quantitative finance: technical analysis, market microstructure, backtesting methodology, and risk management frameworks.
- 20+ years leading high-volume technical operations — the same root-cause diagnosis and pressure-tested decision-making now shapes how I build and validate software.
- Active GitHub: github.com/DJG3DK