ALTITUDE AI by Allio Capital

Altitude AI — Technology Whitepaper

The intelligence layer that makes a frontier LLM trustworthy for money.

> "A frontier LLM is a powerful reasoner with no instruments of its own. Altitude is the purpose-built financial toolset it calls — the toolset is the product; the LLM is the operator."

Technology Whitepaper · Public release, June 2026 · alliocapital.com

EVIDENCE POLICY

> **Every quantitative claim reflects production behavior.**
>
> Where a capability is built but delivered behind a feature flag or not yet generally available, or a figure is illustrative rather than a measured result, it is labeled as such. The platform does not claim a realized return edge; its value is institutional-grade rigor, grounding, and honesty (§2.6, §9). This document does not constitute investment advice.

Audience: technical evaluators, builders, and decision-makers at firms embedding AI investment intelligence.

CONTENTS

01 — EXECUTIVE SUMMARY

An intelligence layer, not a chatbot.

Large language models are now extraordinary at language. They remain unable to *compute* — they cannot solve a constrained optimization, run a probabilistic regime model, fit a factor regression, or fetch

this morning's market data. Ask a bare model to "optimize this client's portfolio" or "tell me what macro regime we're in," and it returns a confident, plausible, and fundamentally un-auditable guess.

Altitude AI is the intelligence layer that closes that gap. A frontier LLM is a powerful reasoner with no instruments of its own; Altitude is the purpose-built financial toolset it calls — each tool a tuned, tested instrument that returns a *computed* result the model then explains. The toolset is the product; the LLM is the operator.

The platform rests on three reinforcing layers, all in production today:

1. **Grounded computation — "computed, not guessed."** Deterministic engines do the math (optimization, factor regression, regime classification, live data); the LLM only explains the result. Identical inputs produce bit-for-bit identical outputs.
2. **The validation & honesty layer.** A dedicated layer scores a backtested track record for reliability, quantifies its statistical uncertainty, and de-rates it when the evidence does not support the claim, applied even to the platform's own models. In a market where most "AI investing" tools present an overfit backtest as proof of skill, a vendor whose value proposition is a performance edge has little incentive to build a tool that honestly grades its own results. This layer is the platform's most defensible capability, and the rest of the paper treats it as such.
3. **Integration & distribution.** A Model Context Protocol (MCP) tool fabric, per-firm custom MCP servers, and a developer API put the engine inside a firm's own product and workflow. Once integrated, the engine is not displaced by a competitor shipping a better base model.

The proposition is the same for every audience: **institutional-grade portfolio construction, regime context, factor analysis, and honest validation — without a deep in-house quant stack to build and maintain.** This is not a data feed and not a chatbot. It is an intelligence layer, and this paper shows how it is built. The core engine is a production-refined, second-generation system; the sections that follow describe the mechanism and why it is defensible.

02 — THE INTELLIGENCE LAYER VS. A BARE LLM

The spine of the argument

This section is the spine of the paper. The claim is specific and falsifiable: for the questions that matter in investment management, the Altitude AI layer returns answers a bare LLM cannot — and we can show why, and how to measure it.

2.1 The core problem with asking an LLM directly

A transformer predicts the next token. It does not run a solver, an expectation-maximization loop, or a regression. So when a bare LLM is asked a quantitative finance question, it pattern-matches plausible numbers. Three structural failures follow:

- **No optimality.** "Optimize this portfolio" has a mathematically correct answer (a convex program). An LLM approximates, with no guarantee the result is optimal, feasible, or even that the weights sum to 1.

- **No live, broad data.** An LLM's knowledge is frozen at its training cutoff — no live prices, no current macro readings, only the most famous tickers. It cannot know today's regime or this ETF's current expense ratio.
- **A guessed number is not reproducible.** Self-generated figures are stochastic — the same question can yield different numbers run to run. For a fiduciary, a recommendation whose numbers cannot be reproduced is a compliance defect, not a feature.

The fix is to give the model instruments. Altitude AI is the purpose-built financial toolset the LLM calls — optimization, macro-regime classification, factor exposure, ticker analytics, live market data, and dozens more — each a tuned, tested instrument that returns a computed result the model then explains. The numbers are computed by deterministic engines, not generated by the model; the engines return identical output for identical inputs. The LLM is used only for the step it is genuinely best at: explaining an already-computed result in natural language.

2.2 Capability matrix — bare LLM vs. the Altitude AI layer

Capability	Bare LLM (alone)	Altitude AI intelligence layer
Portfolio optimization	Approximation; no feasibility/optimality guarantee	Convex optimization — global optimality for the convex formulation, constraints enforced exactly
Macro regime	No probability model; cannot classify	Probabilistic, economically-interpretable regime classification
Factor exposure	Recites factor names; cannot fit	Exact Fama-French five-factor betas via robust regression
Market data	Frozen at training cutoff; narrow	Broad ETF universe + macro series, refreshed intraday
Determinism / audit	Self-generated numbers are stochastic	Numbers are computed, not generated — identical inputs → identical output, bit-for-bit
Grounding	Numbers are generated tokens	Every number is a computed, tool-returned value
Reliability of a track record	None; will repeat a claimed Sharpe uncritically	Validation layer grades and de-rates it (§9)
Safety	Will emit "advice" freely	Advisory-only; no autonomous execution; disclaimer guardrail

The pattern is consistent: where the question requires computation, current data, or reproducibility, the bare LLM degrades to a guess and the layer returns a verifiable answer.

2.3 Worked example (illustrative): a computation an LLM cannot guarantee

The platform's optimizer performs constrained max-Sharpe Mean-Variance Optimization, solved via the standard convex reformulation, which yields a globally optimal, constraint-satisfying allocation. On a fixed five-asset example, the result demonstrates the properties a fiduciary needs. *(Inputs are representative and illustrative — not a recommendation or a realized result.)*

```
Optimal max-Sharpe allocation (illustrative):
  Asset A 0.0500   Asset B 0.0500   Asset C 0.1500
  Asset D 0.4000   Asset E 0.3500
  sum(weights) = 1.000000
  bounds [0.05, 0.40] respected = True

Determinism:        max weight deviation across 5 runs = 0.00e+00
Solution stability: 200 random feasible starts, max dev = 1.41e-06
```

Three properties no bare LLM can promise, demonstrated here:

- **Feasibility:** weights sum to exactly 1.0; every weight respects its bounds.
- **Solution stability:** 200 independent random starting points all converge to the same allocation.
- **Determinism:** repeated runs produce bit-for-bit identical weights. The recommendation is reproducible and auditable.

An LLM asked the same question returns numbers that look like an allocation. It cannot certify they are optimal, feasible, or repeatable — and in a fiduciary setting, "looks right" is not a standard.

2.4 Why a transformer literally cannot do this

The methods underpinning each answer are algorithms, not text: convex/quadratic programming for optimization; a Gaussian Hidden Markov Model for regime inference; ordinary least squares with robust standard errors for factor analysis; gradient-boosted trees and neural networks for forecasting, calibrated against the factor model. These run at machine precision in numerical libraries. A language model has no execution substrate for any of them. The intelligence layer's design insight is exactly this division of labor: the math is computed; the LLM explains.

2.5 How to measure the gap (reproducible benchmark protocol)

We publish a method, not invented head-to-head scores. A buyer can reproduce the comparison: (1) fix a question set spanning optimization, factor exposure, regime classification, and risk metrics; (2) establish ground truth with the platform's deterministic engines (exact by construction); (3) query a bare LLM with identical prompts; (4) score on constraint violations, distance from the optimal weight vector, factor-beta error, regime calibration, and run-to-run variance. The directly demonstrable half of this comparison is the platform's side: by construction, the layer returns feasible, optimal, reproducible results with zero constraint violations and zero run-to-run variance (§2.3). The protocol turns a structural argument into a number any reader can verify; it measures the tools' correctness and consistency — not a market-beating return.

2.6 Why "good tools" is the whole point

A frontier LLM is a brilliant reasoner with no instruments, and the failure modes that follow are the default, not the exception: weights that do not sum to 1, infeasible allocations, statistics it cannot actually compute, confidence it has no basis for. For a fiduciary, that is a compliance and liability problem. A well-built toolset removes those failure modes at the source. The platform does not claim — and does not depend on — a market-beating return edge; edges decay and are commodity. Its value is narrower and more durable: the right tool for each question, built to be correct and trustworthy, in the operator's hand.

03 — THE PROBLEM & WHERE ALTITUDE AI SITS

Where the layer sits

3.1 The data boundary — bring your holdings, or connect your custodian

Altitude AI is deliberately an intelligence layer on data you already have, not a custodian building its own aggregation stack. The access model is: bring your holdings, or connect a data source you control. Today, bring-your-own-data is live via the custom-MCP architecture — a client wires in their own data source as a tool the engine consumes, without Altitude owning that plumbing (§8.1). An optional partner connector is the near-term roadmap for custodial account data. Either way, the layer returns computed analysis, optimization, and macro context. This is a stated boundary, not a gap: it keeps the engine focused on the hard, defensible part — the math, its grounding, and its honesty — rather than on a fragile in-house aggregation pipeline.

3.2 Why existing categories don't solve it

Category	What's missing
Market-data APIs	Raw data only — no reasoning or analytics layer
General LLM APIs	No financial domain math, tools, or live data
Fundamental data providers	Expensive, not developer-first, no AI layer
Institutional platforms	Enterprise-only; no open API model
Account aggregation	Transactional data only; no intelligence

Each adjacent category supplies one piece — data, or reasoning, or transactional plumbing — and leaves the buyer to build and govern the financial-intelligence layer on top. That layer is the gap Altitude AI fills.

3.3 Competitive landscape

Direct and adjacent players exist: developer-facing data/portfolio APIs, and advisor-AI features inside established platforms. **MCP exposure is no longer unique in this space, and tool-calling itself is not the innovation** — wiring an LLM to tools is now table stakes. Altitude AI's defensible position is the *composition*: **unified orchestration** of portfolio optimization + macro-regime context + factor and ticker analytics + bring-your-own MCP server, all behind a single grounded interface; a **validation layer held to an honest standard** (§9); and a **breadth of correctness-tuned tools** that is the actual work. The patent-pending element is the *adaptive ML optimization methodology* (§5) — not the textbook building blocks (mean-variance optimization, the Fama-French model) it composes.

3.4 Who builds on it

The strongest-fit segments are fintech / wealthtech SaaS embedding AI investment analysis behind their own UI; wealth-management platforms adding a grounded analytics layer; AI application builders who want financial domain tools with zero custom integration; and asset managers and model-portfolio teams who build model portfolios, apply regime overlays, and run optimization and factor analysis. For this last group the value is explicitly a cost one: the capability that traditionally requires staffing a team of CFAs and quants is delivered as a production service, so a firm can access institutional-grade portfolio construction without carrying the headcount and ongoing cost of building that function in-house. This applies to firms that could staff a quant team, not only those that cannot — the question is whether that expertise is worth paying for as salary and infrastructure or consuming as a service.

04 — PLATFORM ARCHITECTURE

Four layers, one tool fabric

Altitude AI is a set of independently deployed cloud microservices with strict per-environment isolation. Conceptually, the platform is organized into four layers: an **AI advisor / agent** layer; a **quantitative ML** layer (forecasting and large-universe optimization); an **optimization and macro** services layer; and a **commercial surface** (the developer API and MCP gateway).

Model Context Protocol (MCP) is the connective tissue. The optimizer, macro service, and ML pipeline each expose their capabilities as structured MCP tools; the AI advisor consumes them, and the same tool layer is exposed externally through the developer API's Remote MCP gateway. A typical query — *"what should I do with this client's portfolio given the macro environment?"* — flows: agent → MCP tool calls (regime, factor exposure, optimize) → computed results → LLM-synthesized, explainable recommendation.

Institutional-grade correctness, not a return claim

What this engine offers over a naive call to an off-the-shelf optimizer is not a claim to higher returns — it is **institutional-grade correctness**: realistic expected-return inputs, a downside-aware risk model, constraints that adapt to the universe, and a method that stays well-conditioned as the universe grows. These are the qualities that separate a result you can put in front of a client from one that merely runs.

- **Expected returns** are estimated with the Fama-French five-factor model via robust regression, with bounds applied to prevent unstable inputs from distorting the optimization.
- **Risk** is modeled with a downside-focused (semicovariance) approach rather than naive sample covariance — a more conservative, more defensible posture for client capital.
- **Optimization** is convex Mean-Variance Optimization, so the solve returns a globally optimal, constraint-satisfying allocation, deterministically (§2.3). Constraints are *adaptive* — diversification floors and concentration caps that tighten with universe size and data quality, preventing both under-diversification and over-concentration.
- **Configurable per use case.** The instrument set, target volatility, weight bounds and locks, cash target, lookback window, and expected-return model are all caller-supplied, so the same engine serves different mandates, risk budgets, and house views.
- **Large universes.** For large universes, classical MVO degrades because the covariance matrix is poorly estimated. The engine applies **Nested Clustered Optimization** — hierarchically clustering correlated assets, optimizing within well-conditioned clusters, then across them — which substantially reduces estimation error while preserving the optimization's mathematical properties.
- **Forecasting.** Return forecasts feeding the large-universe optimizer are produced by neural-network and gradient-boosting models, calibrated against the factor model as a floor on predictions.
- **Asset-class extensibility.** The optimizer is asset-class-agnostic by construction — it operates on any instrument with reliable return/price history. Additional universes are a data-onboarding step, not a re-engineering of the engine.
- **Automation.** The pipeline runs end-to-end on an automated schedule, from ingestion → forecasting → optimization → output validation → storage.

What is patent-pending — and what is not

The novel, patent-pending element is the adaptive ML optimization methodology — the clustered-optimization-plus-forecasting pipeline and its adaptive constraint logic — *not* mean-variance optimization or the Fama-French model, which are established techniques the engine composes correctly. The defensibility is in the composition, the tuning, and the breadth, reinforced by the patent over the methodology.

Context and explainability, not a market-timing signal

A dedicated ETL service ingests a broad set of FRED economic indicators across Growth, Inflation, Risk-Premium, and Discount-Rate categories, plus an economic-events calendar, on an intraday schedule timed to US market windows.

The platform's production regime classifier is a **four-state Gaussian Hidden Markov Model** — Expansion / Inflationary Growth / Disinflationary Slowdown / Stagflation — trained on roughly two decades of monthly macroeconomic data. The number of states is not chosen by hand; it is **selected by a formal model-selection criterion**, and the training procedure is engineered specifically to produce *stable, economically interpretable* regimes rather than a degenerate fit. The classifier is **live in production**, refreshing monthly; macro-regime state and regime-conditioned ETF views are served from it today. Two deeper surfaces — **signal attribution** (a growth / inflation / liquidity / market decomposition of the current regime) and a **transition-risk / regime-confidence** view — are built and served, with exposure on the public API rolling out per environment.

What the regime model is for — and what it is not

The regime model is a
macro-context and explainability instrument
: it tells an advisor *which* environment we are in, *why* (the signal decomposition), and *how settled* that reading is (transition risk). It is
not presented as a source of return edge.

We tested whether conditioning a strategy on the regime model improved out-of-sample, risk-adjusted outcomes; it did not produce a durable improvement, so the platform does not claim regime conditioning adds alpha or reduces drawdown. The value is context and transparency for the human making the decision — not a market-timing signal. As with every tool on the platform, the regime model is a rigorously built, validated decision-support instrument, not a realized return claim (§9).

07 — THE AI ADVISOR LAYER · CORE CAPABILITY #3

Where computed intelligence becomes language

- **Agent.** A tool-calling agent on a frontier large language model (provider-abstracted), returning a *structured recommendation object* rather than free text. The reasoning model is a substitutable substrate — the proprietary value is the tools it calls and the data behind them, not the base model.
- **Tool orchestration via MCP.** A resilient client with connection pooling, per-server circuit breakers, and a tool-schema cache. The agent's tool surface is gated by context, so write-capable tools are unavailable in read-only settings.
- **Safety / compliance.** Advisory-only — the agent never executes autonomously; a disclaimer guardrail is enforced; sensitive identifiers are redacted in logs.
- **Cost engineering.** Prompt caching splits the system prompt into a cached static segment and a per-request dynamic segment, substantially reducing input cost on cached context at scale, with per-firm token tracking and budgets.
- **Streaming.** Responses stream via Server-Sent Events, so users see the first tokens of an analysis while the model is still generating rather than waiting for the full response.

This closes the loop on §2: the layer computes with deterministic algorithms and uses the LLM only for the step it is uniquely good at — grounded explanation.

08 — THE COMMERCIAL SURFACE

REST, Remote MCP, and custom-MCP extensibility

43+
Production MCP tools across chat, portfolio, macro/regime, tickers, usage

80
REST endpoints, incl. an OpenAI-compatible chat-completions path

9
Client integrations validated end-to-end

The intelligence layer is reachable through a **Remote MCP server** exposing **43+ production MCP tools** (chat, portfolio, macro/regime, tickers, usage) and a **REST API of 80 endpoints** (including an OpenAI-compatible chat-completions path) with a committed, auto-generated OpenAPI spec. Access is API-key authenticated (keys hashed, never stored in plaintext) with strict firm-scoped tenant isolation, usage metering, and an append-only audit log. **9 client integrations have been validated end-to-end**,

including Claude.ai, n8n, Dify, AnythingLLM, Flowise, LibreChat, Google Sheets, Excel, and GitHub Copilot.

8.1 Custom MCP — bring your own data and tools

The architecturally distinctive surface is **custom MCP**: a firm registers its own external MCP server so that its tools and data — a CRM, a tax engine, a custodial connection, an internal data store — are composed alongside Altitude AI's own tools behind the same grounded agent, without Altitude building or owning that integration plumbing. The firm keeps its own credentials.

The controls are first-class, because a customer-supplied tool surface is exactly the kind of thing a security reviewer scrutinizes:

- **Credential confidentiality** — credentials are envelope-encrypted at rest; the plaintext is never persisted and never returned through the API (a client sees only a "has-credential" flag).
- **Network safety** — server URLs are validated as public and re-validated at connection time; redirects are refused; connections are strictly outbound.
- **Tenant isolation** — custom tools are scoped to the registering firm, with namespaced tool names that cannot collide with built-in tools, enforced per request.
- **Auditability** — every registration, update, revocation, and credential access writes an append-only audit entry.

Crucially, the full path is built — not just registration. When a request carries a firm context, the agent fetches that firm's registered servers, connects over guarded transport, namespaces and isolates their tools per tenant, and **invokes them during live reasoning** — validated end-to-end against a real external MCP server, with cross-tenant isolation and usage/audit logging confirmed. The capability is delivered behind a feature flag and entitlement-gated for staged rollout, and the platform does not ship turnkey, pre-built connectors it does not maintain. This turns the §3.1 data boundary into an extensibility advantage: bring-your-own-data is a first-class part of the product.

09 — STRATEGY VALIDATION & THE HONESTY LAYER

The most defensible capability

A computed answer is only as trustworthy as the discipline that validates it. The platform's most defensible capability is the layer that does exactly that — and it is the one most directly aligned with a fiduciary's duty.

9.1 The problem the layer solves

A backtest is an in-sample fit to a fixed history, and three well-documented mechanisms turn a plausible-looking backtest into a misleading one: **look-ahead / leakage** (information unavailable at decision time inflates in-sample results), **best-of-N selection bias** (across enough trials, one strategy looks excellent by chance), and **overfitting** (a model flexible enough to memorize the sample reports a Sharpe that does not generalize). The quantitative-finance literature formalizes these failures: the **Deflated Sharpe Ratio**

and the **Probability of Backtest Overfitting** establish that, absent correction for the number of trials and the statistical uncertainty of the estimate, a reported Sharpe ratio is not interpretable as skill.

This is also a compliance matter. Marketing rules govern how performance and hypothetical/backtested results may be presented. A platform that surfaces a track record without quantifying its reliability transfers a liability to its customers; a platform that attaches a methodology-grounded reliability grade to every backtested result gives its customers a compliance artifact.

9.2 What the layer computes

Given a backtested strategy result, the validation layer returns a reliability **trust score** and an **A–F honesty grade**; the **statistical uncertainty** of the performance estimate (standard errors, confidence intervals, and the Probabilistic Sharpe Ratio), so a figure is never reported without its uncertainty; **flags** for the failure modes above — overfitting (which caps the grade), absence of a holdout, selection bias, and backfilled or synthetic history; and a hard rule that withholds a track-record claim entirely when the evidence does not support it. The method is grounded in the published backtest-overfitting literature, so the grade is academically defensible rather than a house heuristic.

> **Coverage, stated honestly**
>
> The layer grades **backtested strategy results today** — the platform's model strategies across risk levels. Single-stock / per-ticker guidance does **not** yet carry an honesty grade (there is no per-instrument track record to grade under the current contract); extending the discipline to that surface, and exposing the validation tools on the public API, are the next steps on the roadmap. "Live internally" is not the same as "exposed on the public API," and this paper keeps that line clear.

9.3 Why this is hard to replicate

Capability	How readily a competitor reproduces it	Basis
Grounded computation (§5–§7)	Partially	The LLM-plus-tools pattern is known; the breadth and correctness-tuning of the toolset is the work
Validation / honesty layer	**Difficult for incumbents**	A vendor whose value proposition is a performance edge has little incentive to ship a tool that honestly de-rates its own product

Capability	How readily a competitor reproduces it	Basis
Integration / distribution (§8)	Slowly; never retroactively	MCP exposure is reproducible; integrations a firm already depends on are not un-wired

The barrier is not that the validation layer is *impossible* to copy; the techniques are in the public literature. It is twofold. First, an incentive conflict: a firm that monetizes alpha has little reason to publish the instrument that would honestly grade that alpha, because the two products work against each other. Second, the ordinary combination of switching cost and correctness-tuned breadth: a firm that has wired the engine into its product does not re-wire it because a competitor ships a better base model. A platform whose product *is* the honest instrument carries no such conflict, and its integration advantage compounds over time.

The strongest demonstration is that the platform applies the standard to its own models — and reports the result honestly even when it is unflattering. Altitude AI claims **no return edge.** Held to a disciplined out-of-sample test, the platform's own ETF models show **strong in-sample skill but no demonstrated, durable out-of-sample edge** — over the test window they did not beat simply holding the market. Far from hiding that, the platform treats it as the flagship case of the validation layer doing its job: an in-sample backtest that looks like skill, correctly de-rated once it is tested the honest, forward way. The models function as structured priors the system reasons over, and the **validation discipline, not the models' performance, is the product.** A layer willing to flag its owner's own models is the clearest possible evidence the grade means something. *(A separate internal experiment to add macro-regime conditioning likewise produced no durable edge and was shelved — §6.)*

9.4 How this reconciles with "no track record claimed"

These two statements are not in tension, and it is worth stating why directly. **In production, the layer grades two things: the *customer's* strategies that pass through it, and the platform's own models — which function as structured priors the system reasons over, not as a performance product.** "No track record is claimed" refers to the platform not asserting a realized return edge for itself. "Trust is the product" refers to the integrity of the grading instrument. The proof that the instrument is honest is precisely that it grades its owner's models without exception — including shelving an experiment that did not earn its keep. The grader is the product; the graded models are inputs to it.

10 — SECURITY, COMPLIANCE & RELIABILITY

Built to be reviewed

- **Defense in depth:** a web application firewall at the edge, TLS 1.3 in transit and AES-256 at rest, API-key hashing, complete firm-scoped data isolation, MCP injection guards, append-only audit logging, and multi-account environment isolation.

- **Advisory-only by design:** the agent never executes trades autonomously; a disclaimer guardrail is enforced.
- **Reliability:** a 99.9% platform availability target on top of high-availability managed infrastructure.
- **SOC 2 Type II:** Allio is in the readiness and controls-remediation phase ahead of a SOC 2 Type II examination. No SOC 2 report has been issued, and SOC 2 Type II is **not** claimed as complete.

11 — SCOPE & EVIDENCE STATUS

A strict no-suppositions standard

- **No realized return track record — and none is claimed.** The platform's value is the quality, correctness, and honest validation of its tools (§2.6, §9), not a market-beating return edge. We do not present realized investment performance or an alpha claim.
- **Worked-example inputs are illustrative.** The §2.3 example uses representative inputs to demonstrate determinism, feasibility, and solution stability; it is not a recommendation or a realized result.
- **The bare-vs-tools comparison (§2.5) is a published protocol, not measured results.** The directly demonstrated half is the platform's own determinism and feasibility (§2.3).
- **Live vs. forthcoming is labeled.** The four-state regime model is live for regime state and regime-conditioned views; its deeper surfaces (signal attribution, transition-risk) are built and served, with public-API exposure rolling out per environment. The validation layer is live internally for backtested strategies; single-stock grading and public-API exposure are roadmap. Custom MCP — including the agent-side runtime path — is built and end-to-end verified, delivered behind a feature flag for staged rollout.
- **The regime model is macro-context, not a return signal.** A regime-conditioned-vs-agnostic test showed no durable out-of-sample edge; the model is positioned for context and explainability only.
- **MCP exposure is not unique to Altitude AI;** tool-calling is not the innovation. The differentiation is unified orchestration + honest validation + the patent-pending adaptive ML methodology (§3.3, §5).
- **SOC 2 Type II is in pre-audit readiness, not complete.**

This precision is the point: a reader can rely on every claim that is stated, because anything not yet generally available is scoped here rather than glossed.

APPENDIX A — METHODOLOGY REFERENCES

Grounded in the literature

The platform's methods are grounded in established quantitative-finance and machine-learning literature, including:

- Fama, E. & French, K. — the five-factor asset-pricing model (expected-return estimation).
- Bailey, D. & López de Prado, M. — *The Deflated Sharpe Ratio* and *The Probability of Backtest Overfitting* (the statistical basis for the validation layer, §9).
- López de Prado, M. — *Nested Clustered Optimization* and advances in financial machine learning (large-universe optimization, §5).
- Standard treatments of Gaussian Hidden Markov Models and robust regression (regime classification, §6; factor analysis, §5).

APPENDIX B — ILLUSTRATIVE CAPTURED SESSION

Bare LLM vs. Altitude AI, one live session

This appendix accompanies the benchmark protocol in §2.5 with a single captured session, provided as an illustration rather than as measured benchmark scores. Each question was run twice against the same underlying model: once with the Altitude AI tools on, which is the computed answer, and once with them off, which is a general-purpose model answering from training data. The bare-model lines in quotation marks are taken from the tools-off run. The Altitude AI figures are actual outputs from one live session, shown as illustrative because they move with the market; they are not performance claims, and per §11 the platform claims no realized return edge. The regime read is macro-context, not a return signal (§6); the validation grades follow the discipline in §9, whose public-API exposure is staged per §11.

> **Reference portfolio, fixed for every question**
>
> SPY 40%, QQQ 20%, AGG 25%, GLD 10%, VWO 5%.

Question	Bare LLM (tools off)	Altitude AI (tools on, computed)
Beta of this portfolio?	"Without access to live market data or a portfolio analytics tool, I can't compute a precise, current beta." Offers a textbook estimate.	Runs a per-holding stock model and weights the results: portfolio beta of approximately 0.71, with the per-holding breakdown. Identical on every run.
Sharpe ratio?	"Calculating a precise, current Sharpe ratio isn't something I'm able to do in this mode." Returns the formula.	Computes from realized return and volatility: Sharpe of approximately 1.45, with Sortino, expected return, volatility, and max drawdown.

Question	Bare LLM (tools off)	Altitude AI (tools on, computed)
Why is SPY down?	"I don't have access to live market data, so I can't tell you exactly why SPY is moving." Lists generic drivers.	Reads current data: SPY's intraday price versus the prior close, and the specific sectors dragging the move.
What macro regime are we in?	"I don't have access to live macro regime data, so I can't tell you the current classified regime."	Returns the four-state HMM read (§6): Inflationary Growth at approximately 98.5% posterior, with the coincident-nowcast caveat; returns uncertain when the signal is weak.
Build a portfolio (constraints given).	Writes a plausible allocation, but the weights are not an optimizer's output: no guarantee they sum to 1, respect a risk budget, or are optimal.	Maps the profile to a target volatility, runs the convex optimizer (§5), then validates the result: constraint-satisfying weights with a 1-to-10 risk level.
Write a compliant paragraph on last month's performance.	"Fabricating specific return figures, even approximate ones, would be misleading and could create compliance issues for your client communications."	Computes the real period figures first, then writes the paragraph around those values; every figure in the prose is a computed, tool-returned value.
Improve the alpha on this portfolio.	"Without live portfolio analytics, optimizer output, or current regime data, I can't compute precise alpha improvements."	Re-optimizes, then runs an out-of-sample validation check: holds the new weights over a disjoint forward window and reports predicted versus realized Sharpe and the gap, flagging overfitting rather than asserting an edge (§9). The regime model supplies context only (§6).

Question	Bare LLM (tools off)	Altitude AI (tools on, computed)
Is this strategy any good? (Sharpe 0.84 backtest)	Gives sound qualitative caveats (overfitting, look-ahead, survivorship, costs) but cannot compute any of them: no standard error on the 0.84, no probability the edge is real.	Runs the validation layer (§9): Sharpe standard error, 95% confidence interval, and Probabilistic Sharpe Ratio, with flags for missing out-of-sample tests, backfill, and selection bias, then two grades: is the measurement honest, and is there a real edge.

The pattern is the one §2 predicts structurally. Where a question requires computation, current data, or reproducibility, the bare model returns a plausible-looking answer that breaks the constraints making it meaningful, and the layer returns a computed, reproducible one. This session is a single illustration of the §2.5 protocol; it is not a measured benchmark result.
