ChatGPT

Designing an Autonomous Multi-Agent Financial Trading System

Introduction and Objectives

Building a **fully autonomous, agentic financial trading system** (essentially a virtual hedge fund) requires careful planning. We will leverage OpenAI's new Agents SDK for orchestrating multiple AI agents, along with other OpenAI tools, to simulate hedge fund behavior on a **paper trading budget of $1,000**. The system's design must satisfy four key objectives:

1. **Integrate Diverse Free Data Sources:** Use open-source or free-tier APIs to ingest a broad range of financial data – including real-time stock prices, historical OHLCV data, news feeds, social sentiment (e.g. Reddit, Twitter), company fundamentals, earnings reports/analyst ratings, and macroeconomic indicators. We will identify free APIs for each data category and ensure the agents can pull this data on demand.
2. **Regulatory Compliance:** Even though this system only paper-trades (no real money), it should be designed **in accordance with financial regulations** across jurisdictions (U.S., EU, etc.). This means embedding compliance checks to prevent prohibited trading behaviors and ensuring transparency and audit trails as required by regulators [1] [2].
3. **Best Practices in Finance:** Incorporate strong financial domain knowledge ("financial literacy") into the AI agents, ensure all decisions are auditable and transparent, and implement robust risk management controls. The system should log its reasoning and trades for review, and enforce risk limits to protect the $1,000 virtual portfolio.
4. **Multi-Agent Role Architecture:** Define a clear hierarchy of **"agent executives"** (high-level governing agents) and **"agent roles"** (specialist functional agents) to mirror a hedge fund's organization. For example, separate agents for research analysis, strategy development, compliance oversight, risk management, trade execution, etc., all coordinated as a team.

By meeting these objectives, we will have a **modular, transparent, and realistic** AI-driven trading "firm" that can autonomously research markets and execute trades in simulation. Next, we outline the system's technical design, data integrations, compliance measures, best practices, and the detailed roster of agent roles.

System Architecture and OpenAI Agent Orchestration

We adopt a **multi-agent architecture** where each AI agent has a specialized role, and a top-level coordinator governs their interactions. OpenAI's Agents SDK will be used to orchestrate these agents in a "hub-and-spoke" model [3]. A central **Portfolio Manager agent** (the "hub") will break down tasks and call specialist agents (the "spokes") as needed, treating them as tools or subroutines [4] [5]. This **agents-as-tools pattern** keeps the workflow transparent and auditable by maintaining a single thread of control in the Portfolio Manager, while still allowing parallel analysis by experts [4].

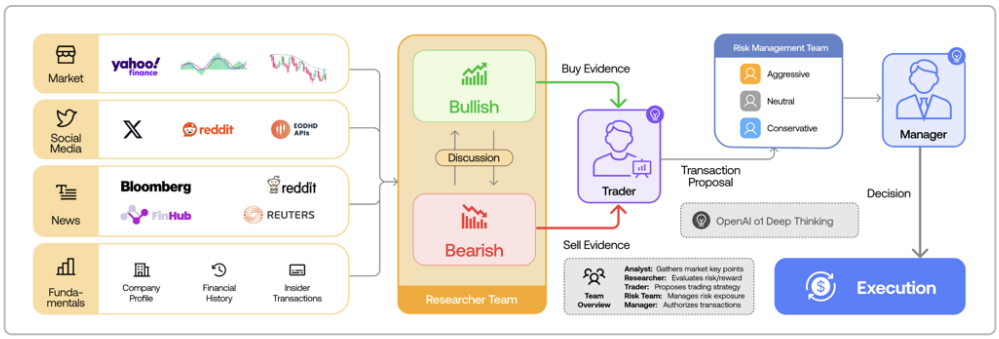

Figure: Multi-agent architecture for the autonomous trading system. Specialized analyst agents gather different types of market data (prices, social media, news, fundamentals) and feed a Researcher team that debates bullish vs. bearish perspectives. A Trader agent formulates a transaction proposal based on the evidence, which the Risk Management team evaluates against risk limits. Finally, a Portfolio Manager ("Manager") agent approves the decision and an Execution agent places the trade. This modular "team" approach mirrors a real trading firm and improves transparency and risk control [6] [7] .

OpenAI Tools Integration: The Agents SDK supports various tool types to empower our agents [8] . We will utilize **OpenAI's managed tools** like *WebSearch* (for real-time news retrieval) and *Code Interpreter* (for data analysis and running Python code) [9] . These tools let agents fetch up-to-date information and perform calculations without leaving the controlled environment. In addition, we will register **custom tools (Python functions)** to call external APIs for financial data (pricing, fundamentals, etc.) and to interact with a paper-trading API. The SDK's design simplifies tool integration – each function can be exposed to agents with defined input/output schema [10] .

All agent interactions and tool usages will be **logged for observability**, so that "every run follows best practices, is easy to debug, and produces outputs you can trust and review" [11] . By structuring the prompt strategies and using Agents SDK features, we aim for a system that is **robust, modular, and transparent** in its reasoning [12] [13] .

Data Sources and APIs for Market Intelligence

To enable informed trading decisions, the system will draw on a wide variety of data sources, all available through open-source or free-tier APIs. Below is a breakdown of the data categories and example sources:

- **Real-Time Stock Prices:** We will use free market data APIs to get live quotes and intraday prices for equities. For example, the **Alpha Vantage** API offers free real-time and historical market data for stocks (as well as forex and crypto) [14] . Another option is **IEX Cloud**, which has a free tier for real-time stock prices and market data [15] . These APIs typically have rate limits, but are sufficient for a small portfolio simulation. Agents can query current prices to make trading decisions (e.g. checking if a target entry price is reached).
- **Historical OHLCV Data:** Historical open-high-low-close-volume (OHLCV) price data is crucial for backtesting strategies and technical analysis. We can rely on sources like **Yahoo Finance** (via an open-source library like *yfinance*) or Alpha Vantage's time series endpoints to retrieve daily and intraday historical prices [14] . For instance, Alpha Vantage provides historical stock time series in various intervals (daily, weekly, etc.) via its free API, and Yahoo's data covers decades of price history.

These allow agents (or the Code Interpreter tool) to perform trend analysis or compute technical indicators over past data.

- **Financial Fundamentals & Earnings:** For company fundamental data (financial statements, ratios, earnings), we will call APIs such as **Financial Modeling Prep (FMP)** or Yahoo Finance's unofficial API. FMP's free tier provides fundamental data like income statements, balance sheets, cash flows, and even real-time financial metrics [16] [17]. Alpha Vantage also has endpoints for company overviews and earnings reports. Using these, a **Fundamental Analyst agent** can pull metrics like P/E ratios, revenue growth, and upcoming earnings dates. **Analyst ratings** and price targets can be accessed via free endpoints as well – for example, IEX Cloud provides a list of analyst price targets and recommendations on its free tier [18], and FMP offers analyst estimates and ratings data (e.g. consensus ratings).
- **News Feeds:** The system will stay informed of market news that could impact the portfolio. We will integrate a **news API or web scraper** for financial news. Options include the free tier of NewsAPI (which aggregates headlines from many outlets), or domain-specific feeds like **Finnhub's news API**, which delivers real-time news articles and even a sentiment score [19]. Another approach is using the Agents SDK's WebSearch tool: e.g. a **News Analyst agent** can search the web for the latest news on a given stock or economic event and summarize the findings. By pulling headlines and news summaries, the agents ensure that sudden news (earnings surprises, M&A announcements, policy changes, etc.) are factored into decisions.
- **Social Media Sentiment:** To gauge market sentiment from social platforms, we will use free APIs or scrapers for sources like Reddit and Twitter. For Reddit, one can use the official Reddit API (which is free for reasonable use) to fetch posts or comments from finance subreddits (e.g. r/stocks, r/wallstreetbets). A **Sentiment Analyst agent** could retrieve recent discussions about stocks in the portfolio and then apply sentiment analysis (potentially using an LLM or a simple NLP model) to determine if the crowd mood is positive or negative. Twitter's API is more limited now, but still allows searching recent tweets for a stock ticker on a free tier (with low rate limits). An alternative is leveraging an aggregate service: for example, **Finnhub's API** provides a *social sentiment endpoint* that directly gives sentiment from Reddit and Twitter for stocks [20] [21]. Using such sources, the agent can quantify public sentiment (e.g. bullish vs bearish mentions) as part of the decision process.
- **Macroeconomic Indicators:** Macro data (interest rates, inflation, GDP, etc.) will be pulled from free resources like the U.S. Federal Reserve's **FRED** API. FRED offers an extensive database of economic time series (e.g. unemployment rates, Treasury yields, CPI) accessible for free with an API key [22] [23]. The **World Bank API** and other central bank data feeds can provide global economic indicators. A **Macro Analyst agent** can periodically fetch relevant indicators (e.g. the latest Fed interest rate decision, GDP growth rates, PMI indexes) and feed that context to the strategy. This helps the system simulate a macro-aware hedge fund – for example, adjusting strategy if recession indicators are rising or if central bank policy shifts.
- **Alternative Data (Insider, ESG, etc.):** If needed, we can tap into other free data sets such as **insider trading records** or **ESG scores**. For instance, Finnhub's free tier includes insider transaction data and even insider sentiment analysis [19]. While these are optional, they could enhance certain strategies (e.g. flagging if many insiders are selling shares of a company).

Each data source above has a free or open component suitable for our budget. We will design agents to **gracefully handle API limits** (caching data when possible, limiting query frequency) so that the $1,000 paper portfolio isn't burned by data costs. In summary, by combining sources like Alpha Vantage, Yahoo Finance, Finnhub, IEX Cloud, NewsAPI, Reddit/Twitter, and FRED, the system gives the AI a **360° view of the**

market without requiring expensive data feeds. The diverse data will feed into different specialist agents as described next.

Regulatory Compliance by Design

Even in simulation, we will enforce policies to **mimic real-world financial regulations**. This ensures the system's behavior would be legally compliant if ever deployed in live trading, and it instills discipline in the autonomous agents. Key compliance considerations include:

- **Market Fairness & Anti-Manipulation:** The agents must not engage in strategies that regulators forbid (even inadvertently). In practice, this means no "market manipulation" tactics – e.g. the system will not attempt spoofing (placing fake orders), pump-and-dump schemes, or spreading false information. All trading decisions are based on public data and legitimate analysis. Regulatory bodies like the SEC and CFTC in the U.S. strictly prohibit manipulation and unfair practices [24] [1], and our compliance agent will monitor the strategies to ensure they remain within bounds. For example, if an agent proposed a trade that would rely on thinly traded volume to exploit other participants, the Compliance Officer agent would flag or veto it.
- **Segregation of Duties & Oversight:** Mirroring a real hedge fund, we incorporate **independent compliance and risk oversight agents** that operate separately from the trading decision agent. The **Compliance Officer agent** continuously monitors the other agents' activities (research and trading decisions) to check for any violation of rules or policies. This agent acts as an internal regulator, with the authority to halt or adjust the system's actions if something looks improper. For instance, if a strategy somehow tried to use non-public information (which shouldn't happen since our data sources are public), the compliance agent would catch that. In essence, no trade gets executed without a compliance "OK".
- **Multi-Jurisdictional Awareness:** The system's design accounts for both U.S. and EU regulatory frameworks. In the U.S., that means adhering to SEC guidelines for securities trading and CFTC rules for any derivatives – e.g. ensuring algorithmic strategies have proper risk controls (the SEC's Rule 15c3-5 mandates risk checks on trading algorithms [25]). In the EU, **MiFID II** regulations impose transparency, accountability, and circuit-breakers on algorithmic trading [2]. To simulate compliance, our system will, for example, maintain detailed logs of all decisions and data (for transparency/reporting), implement virtual "circuit breakers" (the risk agent can pause trading if markets become too volatile or if the portfolio's loss exceeds a threshold), and allow an audit of the algorithm's decision logic. The **audit trail** in particular is crucial – MiFID II requires firms to store trading algorithms' logs and communications, so our agents will preserve their conversation history and tool outputs for later review.
- **Know Your Customer (KYC) & Privacy:** Since this system trades only the house capital ($1,000 simulation) and doesn't interface with client funds, traditional KYC/AML doesn't directly apply. However, if extended to manage user funds or advice, it would need to incorporate those compliance steps too. The design is such that adding a KYC verification module or ensuring GDPR compliance for any personal data would be straightforward. Currently, no personal data is used (only market data), so privacy laws are not a major concern.
- **No Unauthorized Advice:** We will also ensure the system does not accidentally output personalized financial advice to any end-user, as that could trigger regulatory definitions of investment advisory. The system's outputs (trade decisions, analysis) are kept internal, and if any user-facing reporting is produced, it will include proper disclaimers that this is a simulation for research purposes only.

By embedding a dedicated compliance agent and rigorous logging, the autonomous fund behaves **"compliantly by construction."** It is worth noting that algorithmic trading is legal in most jurisdictions *if* you obey the rules [1]. Our system's culture (reflected in agent prompts and reward mechanisms) will emphasize **fairness, transparency, and investor protection**, just as a real compliance department would. This not only avoids legal issues but also cultivates trust in the system's outputs.

Financial Literacy and Domain Expertise

To make sound trading decisions, the AI agents must demonstrate strong **financial literacy** – in other words, an understanding of financial concepts, instruments, and analysis techniques on par with a human analyst. We will achieve this in several ways:

- **Domain-Specific Agent Prompts:** Each specialist agent (Fundamental, Technical, etc.) will be prompted with instructions that enforce proper financial analysis frameworks. For example, the fundamental analysis agent's prompt will remind it to look at revenue, earnings, margins, debt levels, etc., and maybe even include formulas or criteria (like "if debt-to-equity > X, flag high leverage risk"). The prompts will also use correct financial terminology and context so that the LLM responds in the appropriate domain register. Essentially, we'll encode a mini "financial analyst training" into the system prompts.
- **Knowledge Base and Tools:** The system can be equipped with financial reference data and the ability to query definitions or formulas. For instance, if an agent needs to recall what *MACD* or *RSI* indicator means, it could either have that in its prompt context or use a reference tool. Incorporating **financial libraries** via the Code Interpreter tool can also help – e.g. using Python's `pandas` or `ta` libraries to compute indicators ensures mathematically correct outputs rather than relying purely on the LLM's memory. By using such tools for quantitative finance tasks, we sidestep potential hallucinations and leverage well-tested financial logic.
- **Specialized Models or Fine-tuning:** If needed, we could integrate domain-tuned language models. The financial AI space has models like BloombergGPT or FinGPT that are trained on finance text and thus have deeper understanding of financial nuances [26] [27]. Our design will likely use OpenAI GPT-4 (via the Agents SDK), which is already quite knowledgeable, but fine-tuning on financial data or adding few-shot examples of good financial analysis can further improve literacy. For example, showing the fundamental agent an example of an earnings analysis paragraph can guide it to produce similarly structured insights.
- **Multi-Agent Cross-Verification:** Because we have multiple agents with different perspectives, they can catch each other's mistakes or unreasonable assumptions. The **Researcher team** stage of our workflow explicitly has a **Bullish analyst and a Bearish analyst** debating [28]. This means for any given investment idea, one agent will argue the positive case and another will argue the negative case. This reflective discussion forces the agents to justify their reasoning with evidence ("buy evidence" vs "sell evidence" as per the design), highlighting different facets of the financial analysis. By having to convince a skeptical counterpart, each agent tends to be more thorough and fact-driven. This approach draws on the idea of reflection and debate among LLM agents to enhance factual accuracy [29] [30]. In practice, the bullish vs bearish debate will surface important financial details (e.g. "Company ABC's cash flows are strong and it has pricing power" vs "But ABC is trading at a high P/E and could be overvalued if growth falters"), leading to a well-rounded view.
- **Grounding in Data:** The agents will wherever possible ground their outputs in real data pulled from the APIs. Instead of saying "I think the stock is cheap," an agent would retrieve the actual P/E or peer comparison to evidence that claim. Since our architecture allows agents to use tools mid-thought, an

agent can literally query a company's latest earnings or an economic indicator while formulating its analysis. This ensures the content is anchored to **updated, factual information** [5] [8], boosting financial correctness.

- **Financial Reasoning Checks:** We will include some rule-based sanity checks for the agents' conclusions. For example, if an agent recommends investing 50% of capital into a single volatile penny stock, that shows lack of prudent diversification – the risk agent or portfolio manager can push back citing basic portfolio theory. Similarly, the system knows not to confuse, say, revenue with profit or macro with micro effects (these could be encoded as common error checks). Essentially the higher-level agents (PM, compliance) will act as mentors ensuring the specialist agents' outputs make sense financially.

Overall, the design strives to make the AI **"think" like a knowledgeable investor**. The combination of structured prompts, data grounding, internal debates, and oversight should yield analysis that is **original, evidence-based, and financially sound**, rather than naive or purely statistical. In the OpenAI multi-agent example, they emphasized outputs that are *"original, risk-aware, and rigorously reviewed"* [31] – our system will do the same, ensuring each trade decision is supported by solid financial rationale.

Auditability and Transparency

Auditability is a first-class requirement – every decision the system makes should be explainable and traceable. To achieve this, we implement multiple layers of logging, reporting, and human-readable output:

- **Comprehensive Logging:** The system will maintain a log of every agent's actions, intermediate outputs, and tool calls. Each time the Portfolio Manager agent delegates a task to a specialist (e.g. asking the News agent for an update on a company), that interaction is recorded. Likewise, data fetched from APIs, analysis performed (like code outputs from the Code Interpreter), and the final trade decision rationale will all be logged with timestamps. This creates a **detailed audit trail** that can be reviewed later to understand how a conclusion was reached. These logs could be stored in a structured format (JSON or a database) for easy querying (e.g. "show me all trades decided in the last week and the reasons"). If a regulator or developer asks "why did the AI buy this stock?", we can pull the conversation and see that "the Fundamental agent said earnings were up 20%, the Sentiment agent noted positive Reddit buzz, the Risk agent approved it given low portfolio exposure, etc.". This level of traceability is crucial to trust the system.
- **Structured Outputs & Memos:** When the agents collaborate to form an investment thesis, the result can be compiled into a structured report. In the OpenAI example, the Portfolio Manager agent ultimately produced an **"Investment Memo"** for a stock [32]. We can adopt a similar approach: after analysis, the system generates a brief memo explaining the trade – including an **Executive Summary, Key Factors (fundamental, technical, sentiment), Risk Assessment, and Recommendation**. This memo not only guides the internal decision (and is stored for audit) but can also be an artifact for human overseers to read. By enforcing a template, we make sure the AI's reasoning is **explicitly documented** in natural language. The memo acts as the AI's written justification for the trade, which is exactly what an analyst at a hedge fund would produce when pitching an idea [5] [33].
- **Real-Time Monitoring Dashboard:** We plan to build a simple dashboard that displays the system's portfolio status and recent actions, pulling from the logs. This could include current holdings, profit/loss, and a feed of recent agent decisions (with explanations). Such a dashboard provides

transparency for developers or stakeholders to observe the AI's behavior in real time. If something odd is seen (e.g. a sudden flurry of trades), one can drill down into the logs/memos to investigate.

- **Regular Audits and Reviews:** We will institute a process (perhaps an automated agent that acts as an internal auditor) to periodically review the trade logs and performance. For example, a **"Compliance audit agent"** could run weekly and scan the logs for anything that looks like a rule violation or anomaly. It can then produce a compliance report highlighting, say, "No instances of prohibited strategies were found. All trades had accompanying rationale. One trade had an unusually low risk-reward ratio – flagged for review." This adds an extra layer of assurance and can be shown to demonstrate compliance measures.
- **Explainability to Users:** If the system were ever giving output to users (e.g. showcasing a trade or advice), it should explain the reasoning. We can leverage the fact that the multi-agent system inherently produces explanations in the collaboration process – by surfacing those as needed. For instance, if the system suggests adding a position in Tesla, it could also present the key bullet points from the analysis (fundamentals strong, technical uptrend, low downside risk, etc.). This is aligned with the principle of **"glass box" AI** in finance, where model decisions must be interpretable.
- **Observability Tools:** The OpenAI Agents SDK itself is built to facilitate observability – it handles the agent loop and tool use in a way that can be inspected [34]. We will utilize any debugging or trace utilities the SDK provides during development to ensure we capture all relevant info. Moreover, by centralizing control in the Portfolio Manager agent (using the agent-as-tool approach), we reduce the complexity of following the chain of thought; there is one orchestrator that "knows" the overall plan and can narrate it. This contrasts with a fully decentralized agent network that might be harder to audit. Our design choice thus favors **transparent orchestration**.

In summary, the system is designed to be **transparent by default** – every step is recorded and rationalized. This not only satisfies regulatory expectations (e.g. MiFID II's requirements for algorithmic trade records) but also helps us iterate and improve the system by analyzing its decision logs. As OpenAI's guide noted, a major benefit of multi-agent tool-based systems is that they are *"transparent, auditable, and scalable"* [33]. We uphold that by treating each agent's output as a piece of a puzzle that is retained and can be examined individually or in whole.

Risk Management Framework

Effective **risk management** is critical for any trading entity, and we imbue the AI system with robust risk controls to protect the $1,000 virtual portfolio from excessive losses or volatile exposures. The Risk Management function is both a set of automated rules and a dedicated **Risk Manager agent (or team)** that oversees positions continuously [7]. The key elements of our risk framework include:

- **Position Sizing and Diversification:** The system will follow conservative position sizing rules. For example, no single trade will allocate more than a certain percentage (say 10%) of the total $1,000 capital to one asset. This ensures diversification and that one bad trade can't wipe out the portfolio. The **Risk Manager agent** will enforce these limits – if a Trader agent's proposal exceeds them, it will scale it down or reject it. Over time, as the portfolio hopefully grows, the system can recalculate position limits accordingly. We may implement rules like the Kelly criterion or volatility-scaling for sizing, but capped by a max percentage for safety.
- **Stop-Loss and Take-Profit Levels:** For each position initiated, the system will set a stop-loss threshold (e.g. if the stock drops 5% from entry, sell to limit further loss) and optionally a take-profit level (e.g. take profit if gain exceeds 10%). These thresholds can be determined by the strategy or set

globally. The Execution agent and Risk agent collaborate to monitor these – if a live price feed (even delayed) shows a stop-loss hit, the Execution agent will "paper sell" that position immediately to simulate disciplined risk-cutting. This mimics how real trading systems have automatic stop orders.

- **Continuous Portfolio Monitoring:** The Risk Manager agent continuously evaluates the portfolio's risk metrics. This includes tracking metrics like **daily P&L**, **cumulative return**, **volatility of portfolio**, **Sharpe ratio**, and **drawdown**. If volatility or drawdown exceeds predefined levels (for instance, more than a 10% drawdown from peak equity), the risk agent might temporarily reduce position sizes or advise the Portfolio Manager agent to pause new trades. The TradingAgents research showed improved performance by actively managing risk exposure and even having different risk profiles (aggressive, neutral, conservative) to choose from [7]. We can similarly adjust the risk stance of the system (e.g. if market conditions are very uncertain, switch to a conservative mode that uses smaller positions or hedges).
- **Pre-Trade Risk Checks:** Before any trade is executed, the Risk agent does a checklist: Will this trade violate any exposure limit? How will it affect sector or asset class concentration? Does it introduce excessive leverage (if margin were allowed, though in our $1k simulation we may avoid leverage altogether)? Only if the trade passes these checks will the Portfolio Manager give final approval. This is analogous to a real risk officer approving a trader's request. For example, if the system already has 50% of capital in tech stocks and the trader agent wants to add another tech stock, the risk agent might flag that as concentrated risk – prompting either a rejection or a reduction of that trade size.
- **Scenario Analysis and Stress Testing:** Periodically, the risk management module can run scenario tests on the portfolio. Using historical data or simulated shocks, it can estimate what happens to the portfolio if, say, the market drops 5% in a day or if a particular stock gaps down on bad news. These stress-test results feed back into strategy adjustments. If a scenario reveals too much downside, the system might rebalance to safer assets or hold more cash. This forward-looking risk analysis is something we can implement using the Code Interpreter tool (e.g. run a Python script to stress test the current positions against historical worst-case scenarios).
- **Risk-Adjusted Decision Making:** The Portfolio Manager agent will factor risk metrics into its decision-making, not just raw returns. For instance, it might favor a trade with slightly lower expected return if it comes with significantly lower risk (higher Sharpe ratio). We will encode into the prompt that the goal is **maximizing risk-adjusted returns** rather than just returns. This ensures the agents inherently consider volatility and uncertainty in their reasoning. The TradingAgents framework showed improvements in Sharpe ratio by doing multi-agent risk management [35] [36], indicating the value of such approach.
- **Regulatory Risk Compliance:** As mentioned earlier, regulations (like MiFID II) require certain risk controls for algorithmic trading – such as automatic shutdown if things go haywire (our equivalent: if losses exceed X, stop trading) and safe order execution mechanisms. We implement a **"kill switch"**: if portfolio loss exceeds a critical threshold (for example 20% of the capital, i.e. $200), the system will freeze new trading and alert (in simulation this could just be a logged event) [37]. This prevents a runaway algorithm scenario. Additionally, the Execution agent will simulate **best execution** practices: it won't buy or sell all at once in illiquid conditions (though with $1k and paper trading, liquidity isn't a big issue, but we simulate the idea by maybe using average price fill). It will also avoid marking the open or close of market with huge orders (again mostly theoretical for us).

By combining these measures, we expect the system to behave prudently. Even if the AI agents are very confident in a trade, the **Risk layer imposes discipline** akin to a human risk officer tapping the brakes. This not only preserves capital in the simulation but also aligns with how real funds operate under risk mandates. The result should be a **more stable equity curve** and avoidance of catastrophic losses, enabling the AI to learn and adapt over longer periods.

Agent Roles and Responsibilities

A core aspect of our design is defining distinct **agent roles**, much like roles in a hedge fund's organization. We have two tiers of agents: **"Executive" agents that govern and make high-level decisions,** and **Specialist "staff" agents that perform analysis or execution tasks.** Below is a detailed list of the agents in our system and their roles:

- **Portfolio Manager (PM) – *Executive Agent*:** The Portfolio Manager agent is the central coordinator and decision-maker [38]. It receives the overall trading goal or query (e.g. "maximize returns with minimal risk") and orchestrates the other agents to achieve it. The PM agent breaks down tasks and assigns them: it asks analysts to gather insights, asks the strategy agent for trade proposals, consults risk and compliance for approvals, and finally decides which trades to execute. It synthesizes information from all sources into a coherent plan. In effect, this agent acts as the **CIO (Chief Investment Officer)** of the AI fund – balancing inputs and making final strategy calls. It also ensures the workflow follows proper procedures and best practices (for example, waiting for all analysis to come in before making a decision) [4]. The PM agent is also responsible for producing summary reports or "investment memos" from the team's analysis for auditing [32]. No trade goes through without the PM's sign-off, and it will only proceed if both the Compliance and Risk agents have cleared the trade.

- **Compliance Officer – *Executive Agent*:** This agent serves as the AI firm's compliance department. It continuously monitors all communications and proposed actions by other agents to ensure they meet regulatory and ethical standards. The Compliance agent has rules encoded (or learned) about things like insider trading, market manipulation, and general conduct. For instance, if the Research team is discussing a rumor that is not from a credible public source, the Compliance agent might issue a warning or disregard that input (simulating how compliance would disallow acting on unverified info). Before the Portfolio Manager can approve a trade, the Compliance Officer must "green-light" it, confirming that the trade doesn't violate any regulations (US or EU), internal policies, or risk limits. This agent also ensures that proper records are kept – it might prompt the PM agent to document the rationale if it's missing. Essentially, the Compliance agent is the **guardian of legality and ethics**, making sure the autonomous system's behavior would pass external scrutiny. It also prepares compliance reports and can halt trading if a serious concern arises.

- **Risk Manager – *Executive Agent*:** Acting as the chief risk officer, the Risk Manager agent's job is to **protect the portfolio from undue risk**. It evaluates each strategy and trade proposal in the context of current portfolio exposure. For example, it will check how a new trade affects sector allocation, leverage, and potential drawdown. The Risk agent has the authority to veto or ask for modification of trades that don't fit the risk profile (e.g. too large position size, or adding risk when the portfolio is already stressed). It also continuously monitors market conditions – if volatility spikes or if a portfolio asset moves dramatically, it might advise the Execution agent to adjust (e.g. tighten stop-losses or reduce positions). The Risk Manager agent essentially ensures the system's activity stays within predefined risk appetite (which can be configured to aggressive, neutral, or conservative modes). It interfaces closely with the PM and Trader agents: after the Trader agent proposes a transaction, the Risk agent either confirms it's within limits or provides feedback like "reduce quantity from 100 shares to 50 shares due to risk." By having a dedicated risk agent, we mirror real firms' two-tier approach (traders vs. risk oversight) and significantly improve safety [7].

- **Fundamental Analysis Agent – *Research Specialist*:** This agent focuses on **company fundamentals**. For any given stock or investment under consideration, it pulls relevant fundamental data: financial statements (revenue, profit, margins), growth trends, debt levels, valuation ratios (P/E, P/B, etc.), and recent earnings results. It then provides an analysis of the company's financial health and intrinsic value. For instance, it might report "Company X's revenue grew 15% last quarter and it beat earnings expectations by 10% [39]. It has a healthy balance sheet with low debt, and its P/E of 20 is below industry average, indicating potential undervaluation." Conversely, it will also flag fundamental red flags like declining margins or excessive debt. The Fundamental agent essentially answers "Is this business sound and is its stock reasonably valued?" and gives a **long-term investor's perspective** on quality.

- **Technical Analysis Agent – *Research Specialist*:** This agent is responsible for **chart and technical indicator analysis**. It looks at price history and patterns for assets of interest. Using historical OHLCV data, it calculates technical signals such as moving averages, RSI, MACD, support/resistance levels, trendlines, etc. The agent will produce insights like "Stock Y is in an uptrend above its 50-day moving average; however, RSI is entering overbought territory." It might detect chart patterns (e.g. "a bullish breakout from a consolidation") or note momentum and volume trends. The Technical agent's perspective is shorter-term and trading-oriented, complementing the Fundamental agent. It essentially tells the team **"what are the price dynamics and timing signals?"**. If a stock has great fundamentals but the technicals are currently bearish, the system may time its entry more cautiously. This agent uses data from sources like Yahoo/Alpha Vantage for charts and possibly indicator formulas built-in or via libraries.

- **Sentiment Analysis Agent – *Research Specialist*:** The Sentiment agent gauges **market mood and crowd sentiment** around the assets. It scrapes or queries social media (Twitter/X, Reddit, Stocktwits) and news sentiment to see what people are saying. For example, it may report "Market sentiment on Stock Z is strongly positive on Reddit forums, with many users citing a new product launch. Twitter sentiment is neutral [40] [41]." It could also incorporate sentiment from news articles (some APIs return a sentiment score for news). By quantifying bullish vs bearish sentiment, this agent helps the AI understand the psychological factor – e.g. is there euphoria (which might mean overbought conditions) or fear (potentially a contrarian buy signal)? The Sentiment agent might use tools like natural language processing to analyze text, or leverage an API like Finnhub that directly provides sentiment metrics [19]. Its output is an estimate of how optimistic or pessimistic the public and media are about the asset or market in question.

- **News Analyst Agent – *Research Specialist*:** While the Sentiment agent focuses on tone, the **News Analyst** focuses on the factual news content. This agent keeps track of any recent headlines or developments related to the portfolio or watchlist. It might use the WebSearch tool to find the latest articles or a curated news feed. The agent then summarizes key news: "Company X just announced a partnership with ABC Corp" or "The Federal Reserve signaled a possible rate hike next quarter" or "Industry Q is facing new regulations [39]." The News analyst ensures **no important current event is missed** by the trading strategy. It provides context like earnings announcements, economic data releases, geopolitical events, etc. In effect, this agent answers "What just happened or is happening that we should know about?" and provides a concise briefing to the rest of the system.

- **Macro Analyst Agent – *Research Specialist*:** The Macro analyst focuses specifically on **macroeconomic and sector-level data**. It looks at indicators such as interest rates, inflation,

employment, as well as sector performance and rotation. For example, it might note "Tech sector has been outperforming for 6 months, but with interest rates rising, a rotation to value stocks is expected" or "Oil prices have surged 10% this month, which could benefit energy stocks." This agent uses data from sources like FRED for macro stats and perhaps sector ETFs for sector trends. Its role is to give the big-picture backdrop so that trades align with macro tailwinds and avoid fighting broad economic trends. The Macro agent might work closely with the News agent when major economic news hits (like Fed decisions). In some designs, the Macro perspective was included in news/fundamental, but we list it separately to ensure dedicated attention to **top-down analysis** (important for a hedge fund's strategic allocation decisions).

- **Researcher Team (Bullish & Bearish) – *Strategy Sub-Agents*:** After the individual analysts (fundamental, technical, sentiment, news, macro) gather evidence, their findings are fed to a **Researcher team** that debates the investment case [28]. We implement this as two agents with opposing mandates: one "Bullish Researcher" agent whose bias is to find why an investment might be a good idea, and one "Bearish Researcher" agent whose bias is to find why it might be a bad idea. They each take the analysts' inputs and form arguments. Then they engage in a structured discussion (via the Portfolio Manager or a special mediator agent) to weigh pros and cons. This process surfaces the **risk/reward tradeoffs** clearly. For instance, the Bullish researcher might say "Given strong earnings and positive sentiment, I believe Stock X will rise" while the Bearish counters "But the technicals show a possible reversal and the stock is not cheap." The outcome of their debate is an internally balanced view that is passed to the Trader agent. This mimics an internal investment committee discussion, ensuring that **both upside potential and downside risks** are considered before finalizing a strategy [42]. (If needed for efficiency, we could integrate this debate into the Portfolio Manager's reasoning instead of separate agents, but conceptually it's useful).

- **Trader Agent – *Strategy/Decision Agent*:** The Trader agent (or group of agents) takes the consolidated research and decides on specific **trading actions**. This agent answers the question: *"Given all the analysis, what trades should we make, at what price, and when?"* It will formulate a trading plan – e.g. "Buy 10 shares of Stock X at market price, set a stop loss at 5% below entry, and target a 15% upside" or "Short 20 shares of Stock Y if it breaks below $50 support, because fundamentals are weak." The Trader agent essentially converts research insight into executable strategy. It also decides on portfolio allocation – how to weight different positions, how much cash to keep, etc., within the limits allowed. The trader must time entries and exits, so it uses technical signals for timing. In the design, the Trader agent works under the oversight of the Portfolio Manager (and Risk/Compliance). It proposes transactions (which can be thought of as *limit orders, market orders, etc. in the simulation*) that then go through approvals. In some literature, Trader agents also continuously adjust or rebalance the portfolio in reaction to market changes [43]. Our Trader agent will do the same: it doesn't just make one decision and stop; it will frequently evaluate if the current positions are still optimal or if some should be trimmed or added. It's the **action-oriented brain** of the system, implementing the strategy in practice.

- **Execution Agent – *Operational Agent*:** The Execution agent is responsible for carrying out the trades in the paper trading environment. Once a trade proposal is approved by PM/Compliance/Risk, it is handed to the Execution agent to simulate its execution. This agent connects via API to a **paper trading brokerage** or an internal simulator. For instance, we could use the **Alpaca API's paper trading mode** (which is free) to execute orders – the Execution agent would send a buy/sell order to Alpaca's paper trading account and get a fill confirmation. Alternatively, we maintain our own ledger:

the Execution agent updates our portfolio state (reducing cash, adding the stock position) at the price at that moment (perhaps using the real market price fetched via API). The Execution agent also handles **order logic**: if the plan is to place a limit order or to dollar-cost average, it will simulate those details. It can introduce realistic execution conditions, like partial fills or slippage, if we want our simulation to be more lifelike. Additionally, the Execution agent publishes the updated portfolio holdings to the logs and any dashboard. In summary, this agent is the **bridge between decisions and portfolio state**, ensuring each approved trade is duly reflected in the simulation. It also can be tasked with routine operations, like checking each morning if any stop-loss orders should trigger, or if dividends were "paid" on a stock (those can be simulated too, adding to cash).

- **Logging & Reporting Agent – *Operational/Support Agent*:** This supporting agent (or functionality) handles the creation of periodic reports and the maintenance of the trade journal. While all agents contribute to logging, this agent ensures logs are structured and summarized. It can, for example, compile a **daily summary report**: what the portfolio value is, what trades were made, current allocations, and notable events or rule violations (if any). It might also prepare a **performance report** at intervals, calculating the portfolio's return, Sharpe ratio, etc., comparing it to benchmarks. The Reporting agent assists the Compliance and Risk agents by highlighting out-of-bound metrics (e.g. "Warning: weekly loss exceeded 5%"). Essentially, it automates the role of an operations analyst who keeps track of the fund's status and generates statements. These reports are useful for both internal tuning and for demonstrating the system's accountability. If a human were overseeing this AI fund, the Reporting agent's outputs would be what they read to understand how things are going.

Each of the above agents works together as a **cohesive team**, with clear responsibilities much like employees in a firm. By designing these roles explicitly, we ensure modularity (each agent focuses on its task) and **accountability** (we can pinpoint which agent recommended what). As a result, the overall system is more robust than a monolithic agent – for example, if the Technical analyst fails to spot something, the Fundamental or Risk agent might still catch the issue. This division of labor and structured communication is why multi-agent collaboration is powerful for complex tasks [44]. In our case, *"each agent has a specific role: one watches the news, one analyzes the charts, one decides orders, one monitors risk, one keeps a trade journal"* [45], to quote a recent description. By simulating a miniature hedge fund staff with AI agents, we get the benefits of specialization while maintaining a unified purpose through the Portfolio Manager's orchestration.

Simulation Environment and Testing Plan

Finally, a few notes on how we will implement the **paper trading simulation** and test the system end-to-end with the $1,000 virtual budget:

- **Paper Trading API:** We plan to use a broker's paper trading API like Alpaca or Interactive Brokers' demo, which allows us to execute simulated trades with real market data. Alpaca's free paper trading account, for example, starts with a configurable balance (we'll set $1,000) and lets us send REST orders. The Execution agent can be easily integrated with such an API (placing orders and querying positions via HTTP requests). This gives us realistic fills and P&L tracking without risking money. Alternatively, we can simulate trades offline by logging the price at trade time from our data feed – however, using a broker API adds realism (e.g. simulating slight delays and the need for order IDs, etc.).

- **Initial Portfolio Setup:** The simulation will begin with $1,000 in cash and no holdings. The system might first run a **calibration phase** – the agents gather initial data on a watchlist of assets (maybe a selection of 10–20 stocks or ETFs to focus on) and assess current market conditions. Then the Portfolio Manager may decide an initial allocation (even if that's staying mostly in cash if conditions aren't favorable). This warm-up ensures the agents have a baseline understanding before trading.
- **Run Frequency:** The system could run in two modes – **continuous mode**, where it monitors markets in real-time (or at least during market hours) and reacts to events, or **batch mode**, where it makes decisions once a day (like at market close or open). Given our multi-agent complexity and using free data (which might not be true tick-by-tick real-time), a reasonable approach is to run the decision loop daily. Each day after market close, the agents can reassess and decide adjustments for the next day. We could also run a morning check for any overnight news (for gap risk) and intraday checks if something triggers (like a stop-loss event). This way we simulate a **swing trading or daily decision frequency** strategy. Continuous intraday trading could be added if we have a reliable live feed, but that would also consume more API calls and require faster loops – with $1k, our strategy likely leans more medium-term.
- **Testing and Iteration:** We will initially test the system on historical data (backtesting) by replaying past market scenarios. Using the Code Interpreter, we can run a simulation loop where historical prices/news are fed in sequentially and the agents "trade" on that. This will validate that agents behave logically (e.g. they don't churn the portfolio too much, compliance catches what it should, etc.) and that the performance is at least reasonable. After tweaking based on backtests, we will deploy the system in a live paper trading mode going forward. We'll monitor performance metrics like cumulative return and Sharpe ratio to see if the AI fund outperforms baseline strategies (the TradingAgents paper, for instance, benchmarked against simple strategies like buy-and-hold and saw better Sharpe and drawdowns with their multi-agent system [35] [36]).
- **Budget for API and OpenAI usage:** We also consider the computational budget. Our plan sticks to free-tier data sources, so direct costs there are zero. The main cost could be API usage of OpenAI (each agent's queries to GPT-4 incur tokens). With a thoughtful prompt design and perhaps using GPT-3.5 for some lower-level agents, we can contain those costs. If we assume a certain number of prompts per day, we will ensure it fits within a reasonable budget (which could be part of that $1,000 if it was meant for development cost). Given that $1,000 is plenty for OpenAI API calls in a prototype phase, we are comfortable here.
- **Safety Mechanisms:** During the simulation, if anything goes wrong (e.g. an agent outputs nonsense or an error occurs calling an API), the system is designed to fail safely – ideally by skipping that trade or holding cash. All exceptions and anomalies will be caught and logged by the framework (with the help of the Agents SDK error handling). This is akin to having circuit-breakers: better to do nothing than to do something catastrophic if an agent malfunctions. We will test such scenarios (e.g. what if the sentiment API is down? Does the PM just proceed without that info? It should.).

With this plan, we will gradually build confidence in the autonomous fund. We will refine the agents' prompts and logic as we observe it trading in paper mode. The end goal is a system that can autonomously research and trade a small portfolio, **demonstrating the feasibility of an AI-driven hedge fund in miniature**. If successful, we can consider scaling it up (both in capital and complexity), knowing that we have a solid foundation of compliance, risk management, and modular agents.

Conclusion

In this technical design, we laid out a comprehensive blueprint for an **autonomous multi-agent trading system** that behaves like a hedge fund – sourcing rich financial data, respecting regulatory guardrails, applying rigorous analysis, and managing risk – all on a modest $1k paper trading portfolio. By leveraging OpenAI's Agents SDK, we orchestrate a team of specialized AI agents (from analysts to trader to risk manager) that collaborate transparently to make investment decisions [44]. The system uses only free and open data APIs, proving that even with zero data cost one can simulate sophisticated market strategies with AI. Crucially, we have baked in best practices from finance: every trade is justified with **financial literacy** (solid fundamental/technical reasoning), logged for **auditability**, and filtered through strict **risk management** and **compliance** checks – ensuring the AI's actions remain both smart and safe [2] [33].

This design document serves as a roadmap for implementation. The next steps would involve building out each agent (writing prompts, connecting APIs, implementing the decision loop) and then iteratively testing the system's performance. We expect challenges in fine-tuning the agents' cooperation and avoiding information overload, but the structured approach (hub-and-spoke agent orchestration and clearly defined roles) should make debugging easier [12]. In essence, we will be **creating a miniature, self-regulating AI hedge fund**, where humans set the objectives and constraints, and the AI agents do the heavy lifting of thinking and trading.

If successful, such a system could not only simulate hedge fund behavior for research, but also pave the way for more autonomous financial decision-making tools (always under proper oversight). We'll proceed carefully, treating our AI agents as we would human portfolio managers – with trust but also verification. With robust monitoring and continuous improvement, this multi-agent system has the potential to deliver consistent, intelligent trading performance, all while remaining transparent and compliant in its operations.

---

[1] [2] [24] [25] [37] Is Algorithmic Trading Legal? Rules Explained
https://nurp.com/wisdom/is-algorithmic-trading-legal-understanding-the-rules-and-regulations/

[3] [4] [5] [8] [9] [10] [11] [12] [13] [31] [32] [33] [34] [38] Multi-Agent Portfolio Collaboration with OpenAI Agents SDK
https://cookbook.openai.com/examples/agents_sdk/multi-agent-portfolio-collaboration/multi_agent_portfolio_collaboration

[6] [7] [28] [35] [36] [39] [42] [43] Your Guide to the TradingAgents Multi-Agent LLM Framework | DigitalOcean
https://www.digitalocean.com/resources/articles/tradingagents-llm-framework

[14] Free Stock APIs in JSON & Excel | Alpha Vantage
https://www.alphavantage.co/

[15] [18] Top 5 Free Financial Data APIs for Building a Powerful Stock Portfolio Tracker ⚡ - DEV Community
https://dev.to/williamsmithh/top-5-free-financial-data-apis-for-building-a-powerful-stock-portfolio-tracker-4dhj

[16] [17] Free Stock Market API and Financial Statements API... | FMP
https://site.financialmodelingprep.com/developer/docs

[19] [20] [21] Exploring the finnhub.io API | IBKR Quant
https://www.interactivebrokers.com/campus/ibkr-quant-news/exploring-the-finnhub-io-api/

22 23 Macro Data APIs and where to find them : r/algotrading
https://www.reddit.com/r/algotrading/comments/zc6m0p/macro_data_apis_and_where_to_find_them/

26 27 29 30 TradingAgents: Multi-Agents LLM Financial Trading Framework
https://arxiv.org/pdf/2412.20138

40 6 Powerful Real-Time Stock Market Sentiment Analysis Tools
https://www.strikingly.com/blog/posts/6-powerful-real-time-stock-market-sentiment-analysis-tools

41 Finnhub Stock APIs - Real-time stock prices, Company ...
https://finnhub.io/

44 45 TradingAgents: How Multi-Agent LLMs Can Run a Tiny “AI Trading Firm” for You | by CodeBun | Coding Nexus | Nov, 2025 | Medium
https://medium.com/coding-nexus/tradingagents-how-multi-agent-llms-can-run-a-tiny-ai-trading-firm-for-you-f48338fdef48