Proposal
TradeSymphony: The Multi Agentic Stock Trading System

01. Use Case Description

This use case explores a multi-agent simulation framework in which diverse trading agents autonomously execute trades on a designated stock, responding dynamically to various external factors. Agents - including Scraper, Neutralization, and Trading Agents - integrate real-time data from financial reports, market news, and macroeconomic indicators (such as interest rates, inflation, GDP, and employment data) with technical and fundamental market analysis. They also account for company-specific metrics, risk management strategies, and trading logistics like commissions, slippage, and tax implications. Underpinned by the LangGraph MAS framework and leveraging databases like MongoDB and Qdrant the system refines its decision-making process through continuous feedback and prompt adjustments, ultimately mimicking the complex dynamics of real-world trading environments.

The system is influenced by a broad spectrum of economic factors. This includes core macroeconomic indicators like central bank interest rate decisions, government fiscal policies, GDP growth, inflation (measured by CPI and PPI), and employment data. Financial market dynamics such as market sentiment (tracked via the VIX), credit spreads, currency exchange rates, and commodity prices are also crucial. Beyond these, the system incorporates geopolitical factors and critically, company-specific metrics like earnings per share (EPS), financial statement analysis, and even nuanced considerations such as the cross-elasticity of demand within a company's own product lines, reflecting a deep understanding of market interplay.

02. Technical Specifications

This system employs a multi-agent architecture built on the LangGraph MAS framework, with at least 5 agents. Specialized agents have been conceptualized to handle distinct tasks:

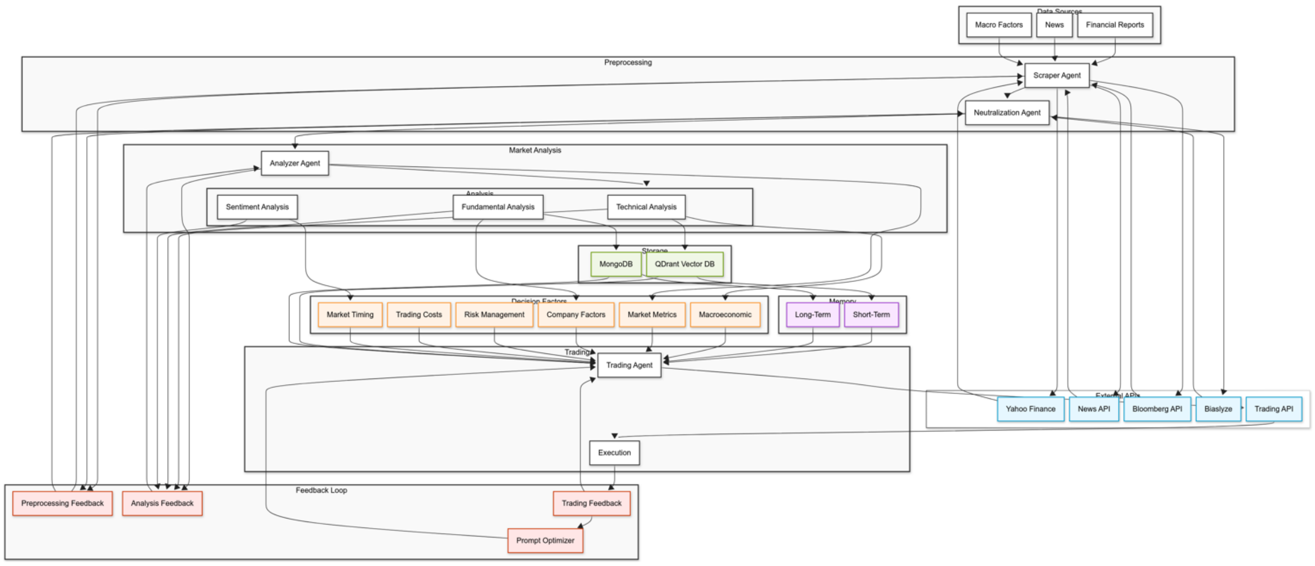
- **Scraper Agent**
  Responsible for gathering raw market data by integrating with external data providers such as the Yahoo Finance API, various financial news and data APIs, and even performing google searches with automatically generated queries to collect real-time information from financial reports, stock prices, market trends, and news on economics, finance, and politics. It operates on a time-triggered basis and then initializes the neutralization agent.
- **Neutralization Agent**
  Cleans and standardizes this data for consistent downstream processing. Will separate data into facts and opinions and use both information to improve final outcomes. The agent uses an Evaluator-Optimizer workflow that leverages a wide range of APIs (ex: Biaslyze API) to ensure the text meets the required level of unbiasedness.
- **Market Analysis Module**
  Extracts and refines the most important details from the normalized data using integrated Technical, Fundamental, and Sentiment Analysis components. It stores processed information in a high-speed Quadrant Vector Database for quick access and in MongoDB for robust, long-term storage, depending on access requirements.
- **Trading Agent**
  Leverages insights derived from the Market Analysis Agent to interact with a dedicated Trading API. This agent is responsible for executing trades based on real-time market conditions and runs continuously (24/7 or as defined by the trading timeframe).
- **Feedback Agent**
  Monitors executed trades and overall system performance by integrating with prompt optimization mechanisms. It employs an Evaluator-Optimizer architecture alongside an

Autonomous Agent framework (as exemplified by Agent Recipes) to continuously refine trading decisions and optimize system operations.

In addition to these, the system incorporates several external APIs:

- **Yahoo Finance API** (and similar financial data APIs) for the Scraper Agent to pull stock data and financial reports.
- **News API** (ex: NewsAPI.org) for gathering financial news.
- **Bloomberg Open API** for additional market data.
- **Biaslyze API** for evaluating and optimizing text naturally.
- **Trading API** for executing trades.

03. Agentic Flow Diagram



The full diagram can be viewed here.

04. Database Integration

The application uses multiple databases to manage and store data effectively:

- **MongoDB**
  Acts as the primary storage solution for historical market data, agent decisions, and performance metrics. It provides robust, long-term data storage, ensuring data integrity and scalability for extensive archival and analytical purposes.
- **Qdrant Vector DB**
  Facilitates advanced data analytics and high-dimensional similarity searches, which are critical for nuanced sentiment analysis and technical evaluations. QDrant is designed with a two-layer storage approach:
  - Short-Term Storage: An in-memory section optimized for fast, real-time data access and rapid query performance.
  - Long-Term Storage: A persistent, on-disk (memmap) section that securely archives historical vector data, ensuring scalability and cost-effective storage.

This dual approach enables QDrant to deliver immediate, low-latency insights while maintaining a comprehensive repository for long-term analysis.

05. Memory Framework

The plans to implement a dual-layer memory framework that supports both short-term and long-term memory. This ensures that our agents can react promptly to current market conditions while also learning from historical data to improve future decision-making.

- **Short-Term Memory:**
  Utilizes in-memory caches to quickly adapt to current market conditions and immediate feedback from trade executions. This will help provide immediate feedback from trade executions, enabling rapid responses to volatile market movements. In addition, QDrant will be integrated into our short-term memory layer. QDrant's high-speed, in-memory vector search capabilities allow the quick retrieval and comparison of relevant data points, enhancing real-time decision-making.
- **Long-Term Memory:** Relies on persistent databases (like MongoDB) to store historical data, trade logs, and performance analytics, enabling agents to learn and improve over time. Addiitonally, QDrant's on-disk (memmap) storage option will be used to archive the vector data efficiently, ensuring cost-effective scaling.

This memory architecture ensures that our agents not only react to the current environment but also learn from historical patterns, enhancing future decision-making.

06. Business Value

The core business value of TradeSymphony lies in its capacity to:

- **Mitigate Risk:** More nuanced understanding of market drivers, thereby reducing exposure to unforeseen volatility and potential losses.
- **Foster Competitive Advantage:** Delivery of timely insights and adaptive strategies, allows users to capitalize on emerging opportunities and navigate market fluctuations with greater agility.
- **Drive Operational Efficiency:** Automation of data processing and analysis streamlines workflows, freeing up valuable resources and enabling trading professionals to focus on higher-level strategic initiatives.
- **Improve Quantifiable Performance:** Through detailed logging and performance tracking, TradeSymphony facilitates rigorous back testing and forward testing of trading strategies, enabling continuous optimization and measurable improvements in trading outcomes.

Ultimately, TradeSymphony empowers financial institutions to navigate market complexities with increased confidence and precision, driving improved performance and delivering tangible ROI.

07. Optional Feature - Prompt Auto-Optimizer

Dynamically improves AI trading agent performance by continuously analyzing real-time trade outcomes and market data. It uses machine learning, including reinforcement learning and adaptive algorithms, to refine input prompts and adjust model parameters. This optimization leverages both immediate market conditions (in-memory caches, QDrant short-term storage) and historical trends (MongoDB, QDrant on-disk storage) for context-aware and scalable adjustments, ultimately enhancing decision accuracy, reducing latency, and driving better trading performance.