**AAI-520 – Final Team Project Report**

**Title: Multi-Agent Financial Analysis System**

**Group Number**: 13

**Team Leader/Representative:** Sethuraman Krishnasamy

**Full Names of Team Member(s):**

1. Sethuraman Krishnasamy

GitHub link: [AAI-520-Final-Team-Project/ at main · krishsethu/AAI-520-Final-Team-Project](https://github.com/krishsethu/AAI-520-Final-Team-Project/tree/main)

**Multi-Agent Financial Analysis System**

**Introduction**

This project presents a multi-Agent system for Stock Analysis designed to reason, act and deliver intelligent insights on stock investment opportunities. It orchestrates multiple specialized agents, mirroring real-world quant research workflows in investment firms.

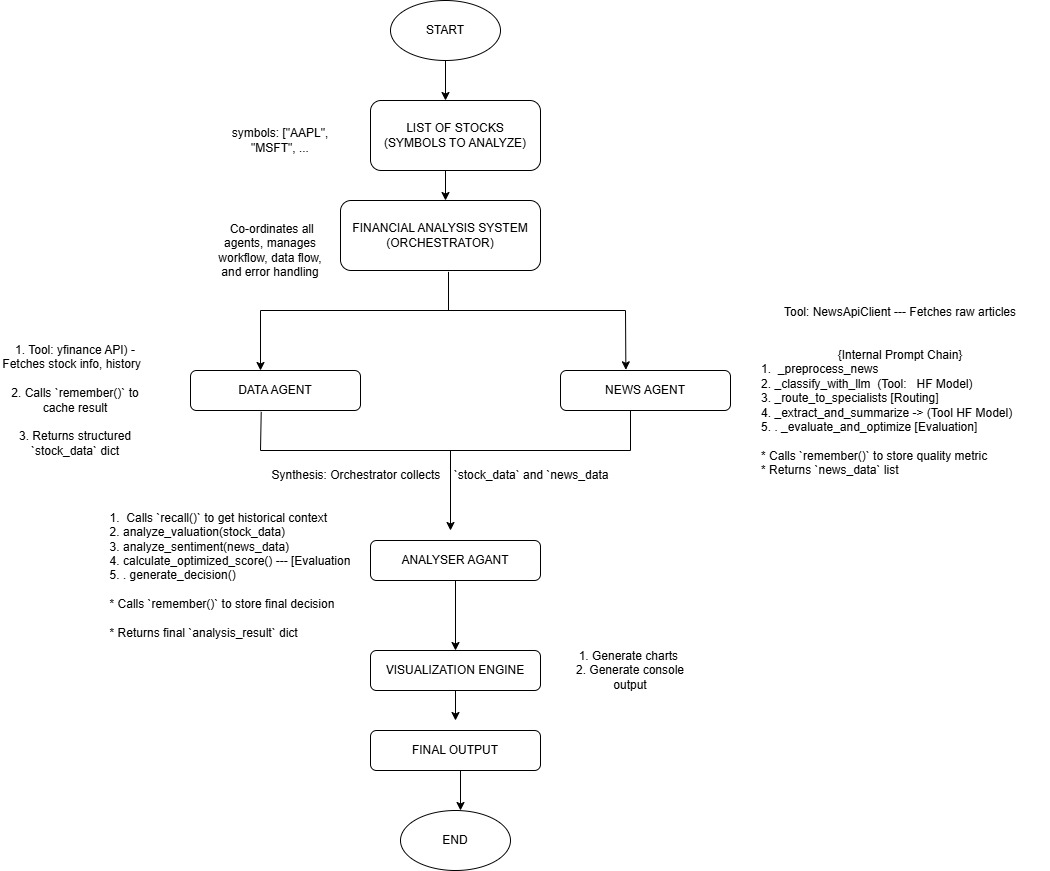
By delegating tasks, the system aims to overcome the limitations of a single model, achieving deeper, more focused analysis. It uniquely synthesizes hard quantitative metrics (like P/E ratios) with soft qualitative signals (like news sentiment). This fusion of data allows the system to generate a holistic, explainable investment thesis. The final output is not just a prediction, but a reasoned decision backed by evidence.

**Description of Selected Dataset:**

**Primary Data Sources:**

1. Yahoo Finance API: Sock prices, historical data, financial statements
2. Financial News APIs: [NewsAPI.org](https://newsapi.org/)

**I. Design and Work-flow diagram of Multi-Agent Financial Analysis System pipeline**

****

1. START & Orchestration

* START: This block represents the initiation of the entire process. In practice, this is when a user executes the main script, providing a list of stock symbols (like 'AAPL' or 'MSFT') that they want analysed.
* FinancialAnalysisSystem (Orchestrator): This is the central hub and "project manager" of the entire system. It's the first active component, receiving the user's list of symbols. Its primary role is not to perform the analysis itself, but to coordinate the workflow. It intelligently delegates specific, large-scale tasks to the specialized agents best suited for the job. It manages the flow of data between these agents, handles any errors that might occur, and ensures the entire process runs in a logical and efficient order from start to finish.

2. Parallel Data Gathering & Analysis

The Orchestrator immediately demonstrates its efficiency by launching two key agents in parallel. This is a crucial design choice because gathering financial data and fetching news articles are independent tasks that can take time (they are "I/O-bound"). Running them simultaneously dramatically speeds up the system.

* DataAgent (The Quantitative Analyst): This agent is a specialized process block responsible for acquiring all numerical and quantitative data. It connects to financial APIs, such as Yahoo Finance, to fetch a wide array of information. This includes real-time stock prices, key valuation metrics like the P/E (Price-to-Earnings) ratio and Market Cap, as well as historical price data. It also performs its own calculations on this raw data to derive important technical indicators, such as the Relative Strength Index (RSI), which helps measure market momentum.
* NewsAgent (The Qualitative Analyst): This agent is a sophisticated process block that runs concurrently with the DataAgent. Its mission is to gather and analyze all unstructured, qualitative data. It connects to news APIs (like NewsAPI.org) to find the latest articles related to a stock. It then processes these articles through its own internal multi-step pipeline, which uses Large Language Models (LLMs). This pipeline cleans the text, performs advanced sentiment analysis (classifying news as positive, neutral, or negative), summarizes long articles, and even includes a self-evaluation step to score the quality of its own analysis.

3. Synthesis & Decision-Making

* AnalysisAgent (The Core Synthesizer): This is the critical convergence point where the two parallel paths meet. The Orchestrator delivers the structured stock\_data (the numbers) from the DataAgent and the processed news\_data (the sentiment scores) from the NewsAgent to this agent. The AnalysisAgent's job is to act as the primary intelligence. It synthesizes these two very different types of information—quantitative facts and qualitative sentiment—using a weighted scoring model. It combines them to calculate a single, holistic Overall Score. Based on this score, it generates the final, actionable AI Decision (like "BUY", "HOLD", or "SELL") and constructs the detailed, human-readable Reasoning string that explains *why* it reached that conclusion.

4. Parallel Output Generation

After the AnalysisAgent completes its work, the Orchestrator now has the complete, final set of results. It once again proceeds in parallel to deliver this information to the user in two distinct formats simultaneously.

* VisualizationEngine (Charts) (The Visual Output): This output block receives the complete dataset (raw data, sentiment scores, and final analysis). Its purpose is to translate all the complex numbers and abstract scores into a suite of intuitive, easy-to-understand visual charts. This includes the radar chart (showing component scores), the sentiment pie chart (showing the news breakdown), the price history line chart, and the RSI gauge.
* Console Display (Table & Reasoning) (The Textual Output): Running at the same time, this output block is responsible for generating the textual report. It formats all the key data points into the comprehensive summary table that provides an at-a-glance overview of every stock. It also prints the detailed, multi-line Reasoning text generated by the AnalysisAgent, providing the crucial context and explainability behind the system's final decision.

5. End

* END: This final block signifies that both output paths are complete. The system has successfully finished its run, and the user has been provided with a complete, multi-faceted analysis, delivered in both visual (charts) and textual (table and reasoning) formats.

**II. Core Component: The Base Class**

1. **Research class (Base Class)**
2. **Description:** Foundational class inherited by all agents. Provides logging (with timestamps), simple in-memory storage for learning across runs, and recall functionality. It ensures all agents can "remember" data (e.g., previous analyses) in a serialized format for persistence.

Acts as the foundational base class (or "chassis") for all specialized agents (DataAgent, NewsAgent, AnalysisAgent). It is not an active agent in the pipeline flow but provides a core set of inheritable services, primarily persistent long-term memory and standardized logging. Its purpose is to allow agents to "learn across runs" by saving and retrieving information from a persistent file store.

1. **Key Methods/Features:**
   1. **load\_memory()**: Automatically called on agent initialization. Reads a dedicated .json memory file from disk (e.g., agent\_memory/NewsAgent\_memory.json), loading the agent's "memory" from all previous runs.
   2. **remember(key, value)**: The primary method for storing information. It serializes the value (using the DataSerializer) into a JSON-safe format, adds a timestamp, and calls save\_memory().
   3. **save\_memory()**: Writes the agent's entire current self.memory dictionary to its dedicated .json file, ensuring the data persists after the script finishes.
   4. **recall(key):** The primary method for retrieving information. It fetches the value associated with a key from the agent's memory (which includes all loaded past data).
   5. **log(message, level)**: A standardized logging utility inherited by all agents to ensure consistent console output.
2. **Pipeline Role:** Core enabler for all agents; supports **learning across runs** by serializing outputs (e.g. data) to avoid non-JSON issues like Timestamps.

Serves as the Core Architecture / Memory Subsystem. It is not a sequential step *in* the pipeline but rather a foundational layer *underneath* the specialized agents. Its role is to enable persistence and learning for all agents that inherit from it. This allows agents to cache expensive results (like API calls from DataAgent), store self-evaluation metrics (from NewsAgent's \_evaluate\_and\_optimize), or track historical decisions (from AnalysisAgent), enabling them to improve, adapt, and run more efficiently over time.

1. **Financial Analysis System (orchestrator Class)**
2. **Description:** Serves as the master orchestrator for the entire multi-agent financial analysis pipeline. It initializes and coordinates the specialized agents (DataAgent, NewsAgent, AnalysisAgent) and the VisualizationEngine. Its primary responsibility is to manage the end-to-end workflow for analyzing one or multiple stock symbols, ensuring data flows correctly between agents, handling errors gracefully, and consolidating the final results for presentation to the user in both tabular and graphical formats.
3. **Key Methods/Features:**

i. **\_\_init\_\_()**: Initializes instances of all required subordinate agents (DataAgent, NewsAgent, AnalysisAgent) and the VisualizationEngine, setting up the complete system architecture upon instantiation.

ii. **analyze\_stocks(symbols)**: Orchestrates the core analysis loop. For each symbol provided:

1. Calls DataAgent.fetch\_stock\_data() to get quantitative and technical data.

2. Calls NewsAgent.fetch\_news() to get processed news and sentiment analysis.

3. Calls AnalysisAgent.analyze\_stock() to synthesize data and generate a decision.

 4. Stores the results (stock\_data, news\_data, analysis\_result) for each symbol.

5. Triggers the generation of individual stock visualizations via create\_stock\_visualizations().

6. Includes error handling to manage failures during the analysis of any single stock.

 7. After processing all symbols, triggers summary visualizations via create\_summary\_visualizations().

iii. **create\_stock\_visualizations(...)**: Delegates the creation of individual stock plots to the VisualizationEngine, requesting charts for sentiment analysis, news categorization, decision breakdown, and technical indicators.

iv. **create\_summary\_visualizations(...)**: Delegates the creation of cross-stock summary plots (like recommendation distribution) to the VisualizationEngine.

v. **display\_comprehensive\_table(symbols)**: Acts as the primary user interface method. It first calls analyze\_stocks() to get the results, then uses helper functions (format\_market\_cap, get\_latest\_news\_title) to format the collected data, and finally prints a structured, comprehensive summary table comparing key metrics and the AI's decision/reasoning across all analyzed symbols.

1. **Pipeline Role:** Core enabler for all agents; supports **learning across runs** by serializing outputs (e.g., plans, data) to avoid non-JSON issues like Timestamps.
2. **Main Execution function (main())**
3. **Description:** Serves as the **master orchestrator** for the entire multi-agent financial analysis pipeline. It initializes and coordinates the specialized agents (DataAgent, NewsAgent, AnalysisAgent) and the VisualizationEngine. Its primary responsibility is to manage the end-to-end workflow for analyzing one or multiple stock symbols, ensuring data flows correctly between agents, handling errors gracefully, and consolidating the final results for presentation to the user in both tabular and graphical formats.
4. **Key Methods/Features:**
5. predefined\_symbols: Defines a static list of stock tickers (e.g., major tech stocks like "AAPL", "MSFT") that will be the target of the analysis run.
6. FinancialAnalysisSystem() Instantiation: Creates the main orchestrator object, which in turn initializes all the specialized agents (DataAgent, NewsAgent, AnalysisAgent, VisualizationEngine).
7. Calling system.analyze\_stocks(predefined\_symbols): Initiates the core multi-agent workflow coordinated by the FinancialAnalysisSystem for all specified symbols. This is where the bulk of the data fetching, processing, and analysis occurs.
8. Calling system.display\_comprehensive\_table(predefined\_symbols): Invokes the method responsible for formatting and printing the detailed, comparative results table to the console, providing a structured overview of key metrics and AI decisions for each stock.
9. Summary Report Generation: Calculates and prints high-level statistics after the analysis, including the number of stocks successfully analyzed and a distribution count of the final AI recommendations (e.g., number of "BUY", "HOLD", "SELL" decisions).
10. **Pipeline Role:** The main execution function is the core enabler for all agents. It supports learning across runs by serializing outputs (e.g., plans, data) to avoid non-JSON issues like Timestamps.

**2. Specialized Agents**

1. **Earnings Analyzer (Specialized Agent)**
   1. **Description:** Acts as a specialized sub‑agent under the *Routing Pattern* hierarchy of the NewsAgent. It focuses on detecting and interpreting *earnings‑related news* such as quarterly results, EPS beats/misses, and guidance updates. Its duty is to extract clear, financially‑relevant insights that represent corporate performance impact.
   2. **Key Methods/Features:**
2. Embedded Keyword Detection (earnings\_keywords): Maintains a dedicated vocabulary set (*earnings, EPS, revenue, quarter, guidance, beat, miss etc.*) to classify items as earnings‑related.
3. Insight Extraction: Derives structured insights (e.g., “Earnings exceeded expectations”, “Future guidance provided”) from the headline context.
4. Impact Scoring (calculate\_earnings\_impact): Computes a 0–1 impact value where positive terms (beat, surprise, raise guidance) increase score and negative terms (miss) reduce it.
5. Structured Output: Returns a dictionary with flags – is\_earnings, insights, impact\_score, and category = "earnings" for integration by NewsAgent.
   1. **Pipeline Role:** Serves as the Earnings‑focused Classifier within the NewsAgent Routing system, supporting sentiment and summary derivation for earnings‑centric articles, feeding clean signals to the AnalysisAgent for valuation and profitability context alignme
6. **Market Analyzer (Specialized Agent)**
7. **Description:** Dedicated to evaluating macro‑level and marketwide news about indices, interest rates, or economic events. It filters content with market‑wide scope (the Dow, NASDAQ, inflation, Fed rate moves) and infers overall market climate sentiment.
8. **Key Methods/Features**
   * 1. Market Keyword Registry (market\_keywords): Looks for core macroeconom ic triggers (“Fed”, “interest rate”, “rally”, “selloff”, “inflation”).
     2. Sentiment Assignment: Labels detected news as Positive (“market rally”, “bull run”) or Negative (“selloff”, “bear”, “drop”).
     3. Insight Collection: Generates short phrases highlighting market momentum (“Positive market momentum”, “Negative market pressure”).
     4. Categorical Output: Returns structured dict {'is\_market', 'sentiment', 'insights', 'category':'market’.
9. **Pipeline Role:** **Planning function**: First step; enables dynamic workflow (e.g., 6-step plan: fetch → news → analyze → sentiment → report → evaluate).
10. **Company News Analyzer (Specialized Agent)**
11. **Description:** Analyzes company‑specific stories such as product launches, M&A announcements, and leadership changes. Acts as a focused agent for corporate activity detection within the NewsAgent Routing layer.
12. **Key Methods/Features:**
    * 1. **Domain Categorization:**  Scans news titles for keywords (spanning “product”, “launch”, “acquisition”, “merger”, “CEO”, “executive”) to label items as product, corporate action, or management.
      2. **Relevance Scoring (calculate\_relevance**): Computes a 0–1 relevance factor weighted by explicit company mention and priority keywords (breakthrough, major, strategic).
      3. **Output Schema:** Provides {'is\_company\_news', 'categories', 'relevance\_score', 'category':'company\_news'} for cross‑agent integration.
13. **Pipeline Role:** **Planning function**: Complements the Earnings and Market Analyzers by handling micro‑level, entity‑specific information. Feeds critical qualitative signals to the AnalysisAgent to capture innovation, management quality, and partnership themes affecting company valuation.
14. **News Agent (Specialized Agent)**
15. **Description:** Retrieves recent news articles for a symbol and preprocesses them (e.g., concatenates title/summary, cleans text). Serializes processed news.
16. **Key Methods/Features:**
    1. **fetch\_news(symbol):** The main orchestration method that manages the entire end-to-end pipeline, from ingestion to evaluation.
    2. **ingest\_news(symbol):** Fetches raw news articles from the NewsApiClient, which is more robust than the basic yf.Ticker method.
    3. **\_process\_news\_pipeline(...):** The core **PROMPT CHAINING** method. It sequentially executes the full workflow for a single article
       * 1. \_preprocess\_news (cleans text, extracts metadata)
         2. \_classify\_with\_llm (generates sentiment)
         3. \_route\_to\_specialists (categorizes news)
         4. \_extract\_and\_summarize (creates a summary)
    4. **classify\_with\_llm(...):** Sentiment analysis is handled by the model cardiffnlp/twitter-roberta-base-sentiment-latest. This is enhanced by:
       * 1. \_add\_financial\_context: A prompt engineering step that adds guiding text.
         2. \_apply\_financial\_context\_adjustments: A correction layer that uses financial rules to override or boost the LLM's output.
    5. \_**fallback\_sentiment\_analysis(...):** A robust, keyword-based backup system (with contextual boosts) that provides sentiment if the LLM fails.
    6. **\_route\_to\_specialists(...):** Implements the **ROUTING** pattern, directing news to specialized modules like EarningsAnalyzer or MarketAnalyzer.
    7. **\_extract\_and\_summarize(...):** Uses the facebook/bart-large-cnn model to generate a dynamic, AI-powered summary of the article.
    8. **\_evaluate\_and\_optimize(...):** An **EVALUATOR** function that scores the quality of the sentiment and summaries, enabling a feedback loop for continuous improvement.
17. **Pipeline Role:** Prompt chaining start: News ingestion/preprocessing; feeds cleaned text to sentiment analysis.
18. **Data Agent (Specialized Agent)**
    1. **Description:**

Serves as the specialized agent responsible for the ingestion and preparation of structured financial and technical data. It reliably fetches quantitative information (company profile, historical stock prices, analyst recommendations) from external APIs, calculates key technical indicators (RSI and MACD), and meticulously serializes this data into a standardized format for seamless integration with downstream analytical agents.

* 1. **Key Methods/Features:**

1. fetch\_stock\_data(symbol): The primary orchestration method for this agent. It coordinates the retrieval of stock information (stock.info), 3-month price history (stock.history), and triggers the calculation of technical indicators.
2. API Interaction (yf.Ticker): Uses the yfinance library (yf.Ticker) to retrieve the raw stock data (profile, price history). This demonstrates Tool Use.
3. calculate\_rsi(price, \_period\_ = 14): Determines the Relative Strength Index (RSI) based on the provided price history DataFrame. It returns the latest RSI value and a dictionary indicating trends (Bullish/Bearish, Oversold/Overbought).
4. calculate\_macd(prices): Calculates the Moving Average Convergence Divergence (MACD) indicator and its signal line based on the provided price history DataFrame. It returns the latest MACD and signal line values.
5. Data Structuring & Serialization: Organizes the fetched data and calculated indicators into a comprehensive dictionary. Crucially, it converts the price history DataFrame into a dictionary (hist.to\_dict()) for easy serialization, ensuring the data can be stored and passed between agents without compatibility issues.
   1. **Pipeline Role:**

Functions as the primary Data Ingestion Tool User within the pipeline, operating in parallel with the NewsAgent. Its role is expanded to provide not only foundational quantitative fundamental data (like P/E ratio, market cap) but also key technical indicators (RSI, MACD) required by the AnalysisAgent and potentially the ReportingAgent for generating technical analysis insights and recommendations.

1. **Analysis Agent (Specialized Agent**
   1. **Description:** Synthesizes structured financial data (from DataAgent) and processed news analysis (from NewsAgent) into a comprehensive, multi-factor investment assessment. It calculates individual scores for valuation, profitability, technicals, sentiment, analyst ratings, and market position, then combines these into a weighted overall score, potentially adjusted based on the quality of the input news analysis, to generate a final investment decision and supporting rationale.
   2. **Key Methods/Features:**
      1. **analyze\_stock(stock\_data, news\_data):** The main orchestration method. It receives inputs from upstream agents and coordinates calls to various internal analysis functions to compute component scores and the final decision.
      2. **analyze\_valuation(stock\_data):** Evaluates the stock's valuation attractiveness based on metrics like P/E ratio and forward P/E, returning a score (0-100).
      3. **analyze\_profitability(stock\_data):** Assesses the company's financial health using profit margins and revenue growth, returning a score (0-100).
      4. **analyze\_technical(stock\_data):** Analyzes technical momentum using indicators like RSI (including oversold/overbought conditions and trend), returning a score (0-100).
      5. **analyze\_sentiment(news\_data):** Calculates an enhanced sentiment score based on the processed news. It weights the sentiment score of each article by the confidence level and whether an LLM was used, providing a more reliable aggregate sentiment score (0-100).
      6. **analyze\_analyst(stock\_data):** Scores the consensus analyst recommendation (e.g., strong buy, hold) and the potential upside based on the target price versus the current price, returning a score (0-100).
      7. **analyze\_market\_position(stock\_data):** Evaluates the company's size (market cap) and relative volatility (beta), returning a score (0-100).
      8. **\_calculate\_optimized\_score(...):** Implements the **EVALUATOR-OPTIMIZER** pattern. It first calculates a base score by applying predefined weights (self.decision\_weights) to each component score. It then adjusts this base score using a \_calculate\_quality\_boost derived from the average quality metrics (sentiment confidence, summary quality, LLM usage) of the input news\_data, rewarding analyses based on high-quality inputs.
      9. **generate\_decision(overall\_score):** Translates the final optimized overall score into a categorical investment decision (e.g., "STRONG BUY", "HOLD", "SELL") and assigns a confidence level.
      10. **generate\_detailed\_reasoning(...):** Creates a human-readable explanation for the final decision by highlighting the key contributing factors identified during the component analysis (e.g., "Attractive valuation", "Negative news sentiment").
   3. **Pipeline Role:** **Routing/Analysis**: Acts as the **central analysis and decision-making hub**. It sits downstream from the Data Agent and News Agent, receiving their processed outputs. Its primary role is to **synthesize** diverse quantitative and qualitative inputs, apply a weighted scoring model (refined by input quality evaluation), and produce the final, actionable output: an investment recommendation with supporting evidence and reasoning, ready for the Reporting Agent
2. **Visualization Agent (Specialized Agent)**
3. **Description:** Operates as a non‑analytical but highly interactive visual agent transforming outputs from Data, News, and Analysis Agents into intuitive dashboards. It helps users trace how sentiments, technical metrics, and recommendations interact across multiple equities.
4. **Key Methods/Features:**
   * 1. **Sentiment Visualization (create\_sentiment\_analysis\_chart):** Generates pie/bar comparisons of news sentiment distribution for each symbol.
     2. **Decision Breakdown (create\_decision\_breakdown\_chart):** Depicts weighted component scores on bar and radar charts for each agent score category.
     3. **Recommendation Summary (create\_recommendation\_summary):** Aggregates buy/hold/sell signals across multiple stocks into pie charts, confidence bars, and heatmaps.
     4. **Technical and News Category Charts:** Render time‑series price and RSI visuals plus news category distribution.
     5. **Matplotlib & Seaborn Integration:** Implements aesthetic color maps and annotations for clarity.
5. **Pipeline Role:** Acts as the Presentation Layer Agent, integrating quantitative analytics and qualitative insights into visual formats. It bridges AI outputs to human interpretability within the FinancialAnalysis
6. **Reporting Agent (Specialized Agent**
7. **Description:** Compiles analysis into a full report: summaries, recommendations (e.g., BUY/SELL based on P/E + sentiment), risks/opportunities.
8. **Key Methods/Features:**
   1. generate\_report(...): Builds structured dict (financial\_summary, recs).
   2. generate\_recommendation(...): Rule-based logic (e.g., low P/E + positive sentiment → STRONG\_BUY).
   3. identify\_risks/opportunities(...): Lists issues (e.g., high beta) or upsides (e.g., strong margins).
9. **Pipeline Role:** **Output generation**: Converges data/sentiment; produces actionable insights (e.g., "HOLD" with risks).
10. **Evaluation Agent (Specialized Agent)**
11. **Description:** Self-assesses report quality (completeness, consistency, rationale) and generates feedback/suggestions for improvement.
12. **Key Methods/Features:**
    * 1. evaluate\_analysis(symbol, report): Scores 0-1 (e.g., field checks for completeness).
      2. generate\_feedback(...): Lists issues (e.g., "inconsistent sentiment-rec").
      3. generate\_improvement\_suggestions(...): Actionable tips (e.g., "add metrics").
13. **Pipeline Role:** **Evaluator-optimizer pattern**: Final step; enables **self-reflection** with feedback loop (e.g., low score → refine report).

**II. DESIGN AND IMPLEENTATION DETAILS OF COMPLEX AGENTS:** The design and implementation details of some of the complex agents such as News Agent, Data Agent and Analysis agent are described below.

**1. News Agent - Sentiment Analysis Component: LLM Model and Pipeline details:**

**i) Design Details:**

The News Agent represents a sophisticated AI-powered financial news processing platform that intelligently analyses, categorizes, and summarizes market-moving information. This system combines advanced machine learning, domain expertise, and software engineering patterns to transform raw news articles into actionable investment insights.

The system implements a multi-stage processing pipeline and is built on three core architectural patterns.

**a) PROMPT CHAINING:**

The Prompt Chaining orchestrates sequential AI transformations where each news item flows through preprocessing, sentiment classification, specialized analysis, and summarization stages.

Instead of a single, monolithic analysis, each news item undergoes a multi-stage transformation. The output of each stage becomes the refined input for the next, ensuring that the final summary is built upon layers of contextual understanding. This ensures comprehensive and layered understanding of financial content.

The sentiment classification is LLM powered. The model's classification is provided by a set of financial heuristics. For example, a product launch classified as "Neutral" is automatically elevated to "Positive"). The output is a structured sentiment object, like {'sentiment': 'Positive', 'score': 0.95, 'context\_adjusted': True}. It generates a financially relevant sentiment score that goes beyond simple keyword analysis.

**b) ROUTING PATTERN**

Automatically directs content to domain-specific analyzers - Earnings Analyzer for financial results, market analysts for generic macroeconomic news such as such as "inflation," "interest rates," "GDP," or "Federal Reserve" and Company news Analyzer for corporate developments. This enables precision analysis tailored to each news type's unique characteristics.

The Routing Pattern acts as the system's intelligent triage centre, ensuring that the right news gets to the right expert. By automatically directing content to domain-specific analysers, the system delivers a precise analysis tailored to each news type's unique characteristics.

**c) EVALUATOR-OPTIMIZER:**

This continuously monitors processing quality through multiple metrics, enabling real-time performance assessment and systematic improvement. The system self-diagnoses issues and suggests optimizations for maintaining high-quality outputs.

The Evaluator-Optimizer pattern is what elevates the system from a simple automated tool to an intelligent agent capable of self-reflection. It functions as an internal quality control department, continuously assessing the performance of its own AI-driven analyses. By scoring its outputs against key metrics and generating actionable feedback, this pattern creates a continuous learning loop, allowing the system to identify its own weaknesses and systematically improve its accuracy and reliability over time.

**d) EVALUATOR-OPTIMIZER PATTERN:**

The Self-Reflective Quality Control System

The Evaluator-Optimizer pattern is what elevates the system from a simple automated tool to an intelligent agent capable of self-reflection. It functions as an internal quality control department, continuously assessing the performance of its own AI-driven analyses. By scoring its outputs against key metrics and generating actionable feedback, this pattern creates a continuous learning loop, allowing the system to identify its own weaknesses and systematically improve its accuracy and reliability over time.

This is the mechanism that builds trust in the agent's insights.

The Three Stages of Self-Improvement

The process of evaluation and optimization follows a clear, cyclical path.

**1) Stage 1: The Quality Audit (Quantitative Scoring):**

After the ReportingAgent synthesizes its analysis, the EvaluationAgent steps in to perform a quantitative audit. It doesn't assess the content itself, but rather the quality of the AI processes used to generate that content. It measures specific, pre-defined metrics

**2) Stage 2: The Actionable Critique (Feedback Generation):**

Raw scores are just numbers; their value comes from interpretation. In this stage, the agent translates the quantitative scores from the audit into qualitative, actionable feedback. It compares the average scores against internal benchmarks.

**3) Stage 3: The Optimization Loop (Closing the Feedback Loop):**

This is the most critical stage, where reflection leads to action. The feedback generated in Stage 2 is passed back to the OrchestratorAgent, closing the loop and enabling improvement.

The platform demonstrates exceptional contextual intelligence by understanding financial terminology, recognizing product launch significance, and interpreting business events through an investment lens. It enhances generic sentiment analysis with financial domain knowledge, boosting confidence scores for market-moving events like product announcements and earnings reports.

Robust error handling and graceful degradation ensure system reliability, with comprehensive fallback mechanisms that maintain service availability even when individual components fail. The architecture supports continuous learning and adaptation, making it increasingly effective at identifying financially relevant patterns over time.

This system acts as an integrated investment analytics tool for informed financial decisions to investors, analysts, and financial institutions seeking to process vast amounts of news data efficiently while extracting meaningful, actionable intelligence for informed decision-making.

**ii) News Agent - Details of LLM Model and Pipeline:** The news agent uses a powerful pretrained LLM model

1. Pre-Trained Model: A powerful, pre-trained RoBERTa-based AI model, fine-tuned by Cardiff University on Twitter data is used to perform highly accurate sentiment analysis (classifying text as positive, negative, or neutral).
2. The pipeline function from the Hugging Face transformers library is used for classifying text sentiment.Below are the details
3. pipeline("sentiment-analysis", ...): This is the core command. The Hugging Face library builds a complete, end-to-end tool for the specific task of "sentiment-analysis". The pipeline automatically handles all the complex steps as mentioned below:

1) **Tokenization:** Takes the raw text (e.g., a news headline) and converts it into a numerical format the model can understand.

2) **Inference:** Feeds the numbers through the AI model to get a prediction.

3) **Post-processing:** Converts the model's numerical output back into a human-readable result (like "Positive" or "Negative").

1. model="cardiffnlp/twitter-roberta-base-sentiment-latest": This specifies exactly the pre-trained model.
2. RoBERTa-base: It's built on RoBERTa, a powerful and robust language model developed by Facebook AI.
3. cardiffnlp: It has been fine-tuned by the NLP research group at Cardiff University.
4. sentiment-latest: It's specifically trained on a huge dataset of tweets to be exceptionally good at identifying sentiment (Positive, Negative, Neutral) in modern, informal text.
   1. **Data Agent - Data Acquisition and Feature Extraction**
      * 1. This Agent is responsible for collecting raw financial market data and transforming it into meaningful, actionable metrics.
        2. It involves:

i) Fetching historical stock data over a relevant time window from reliable data sources (e.g., Yahoo Finance APIs).

ii) Accessing fundamental company information such as the price of the stock, P/E ratios, analyst price targets, earnings dates, and volume.

* + - 1. The system derives key technical indicators, such as the Relative Strength Index (RSI), which provides insight into market momentum and helps identify overbought or oversold stock conditions. The design leverages time series computations on closing prices to yield trend signals that indicate bullish or bearish market behaviours. By integrating both static company fundamentals and dynamic technical signals, this stage prepares a rich dataset for further evaluation.
  1. **Analysis Agent - Analytical Evaluation and Decision Support**
     + 1. Building upon the features generated in the data acquisition phase, this component systematically evaluates the stock's attractiveness and market sentiment through multiple lenses:
       2. It applies a weighted scoring framework where dimensions such as valuation (P/E ratios), profitability (profit margins and revenue growth), technical indicators (RSI analysis, momentum), sentiment derived from news, analyst recommendations, and overall market position are quantified.
       3. Sentiment analysis is enhanced by aggregating news-derived sentiment scores, weighted by confidence and whether advanced NLP (like LLMs) was used, ensuring more reliable sentiment inputs.
       4. A composite overall score is calculated by combining component scores adjusted through quality feedback mechanisms to reflect data integrity and processing reliability.
       5. The system generates intuitive decision signals (e.g., "BUY", "HOLD", "SELL") with associated confidence levels based on overall scores.
       6. Detailed reasoning accompanies each decision, explaining contributions from valuation, technical conditions, news sentiment, and analyst outlook, thereby improving transparency.
       7. These two sections together form a powerful pipeline: from raw financial and market data ingestion, through technical and fundamental feature extraction, to multi-faceted analytical evaluation yielding actionable investment recommendations.
       8. Such modular, interpretable, and extensible design enables continuous refinement and integration of new data sources or analytical techniques, advancing AI-driven financial decision support.

**III. OUTPUT ANALYSIS:**

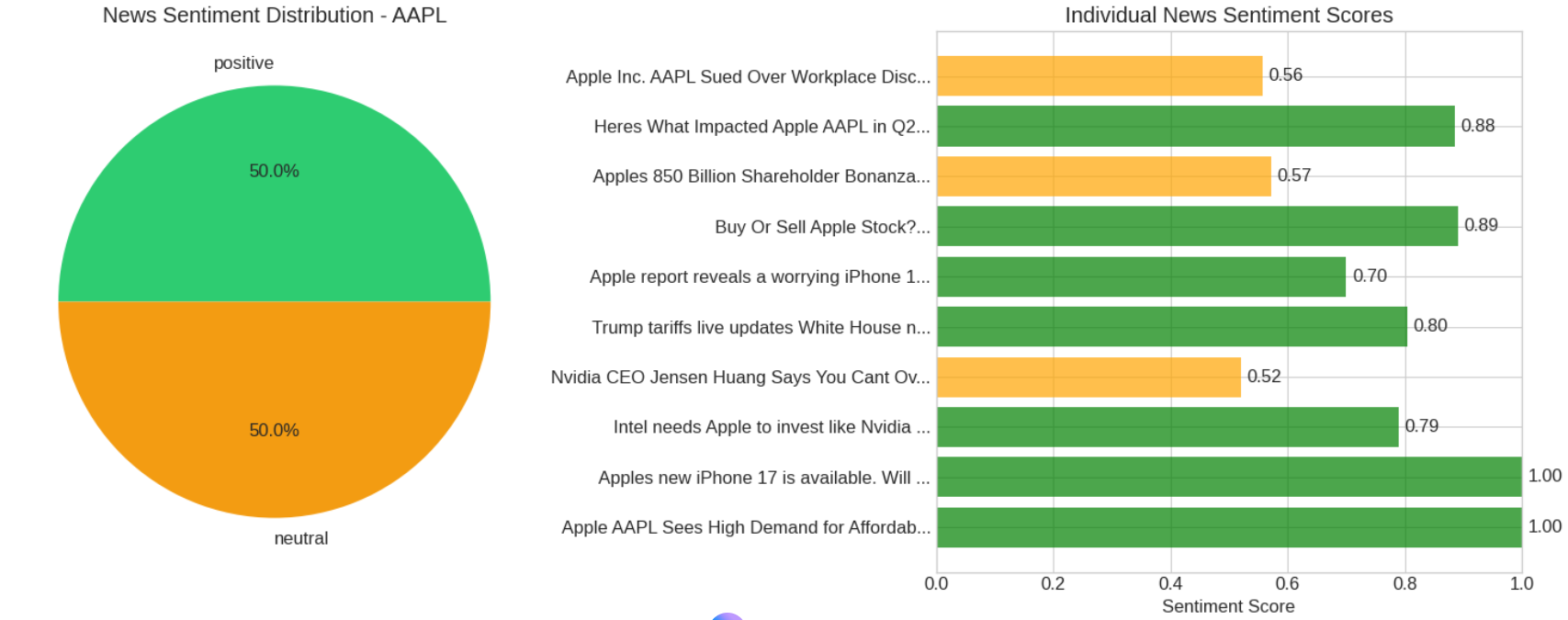
The below graphical analysis summarizes the findings of the multi-agent system for finance analysis. The system has been tasked with a comprehensive evaluation of a predefined list of 10 major stocks in technology industry: AAPL (Apple), , GOOG (Alphabet C), NVDA (NVIDIA), GOOGL (Alphabet A), ORCL (Oracle), AMD (Advanced Micro Devices), INTC (Intel), MSFT (Microsoft), AMZN (Amazon), and META (Meta Platforms).

For each stock, the system's VisualizationEngine generates a standardized suite of charts based on the processed data from the DataAgent, NewsAgent, and AnalysisAgent. This suite includes:

1. News Sentiment Distribution (Pie Chart)
2. Individual News Sentiment Scores (Bar Chart)
3. News Category Distribution (Bar Chart)
4. Decision Component Breakdown (Bar Chart)
5. Component Radar Chart
6. Price History (Line Chart)
7. Relative Strength Index (RSI) (Gauge Chart)
8. **CHART OUTPUTS AND DESCRIPTIONS:**

Different Charts of one of the stocks AAPL (from the list of stocks mentioned above) is being described below.

* + - 1. **Chart 1: News Sentiment Distribution of Stock AAPL** This output provides a detailed two-part summary of the analysis by NewsAgent for the stock "AAPL".



1. **What it is:** This pie chart shows the aggregated sentiment of all news articles processed for Apple stock.
2. **Analysis:** The chart indicates a perfectly balanced sentiment distribution among the analysed articles:
   * 50.0% Positive (Green): Half of the news items were classified as having a positive sentiment.
   * 50.0% Neutral (Orange): The other half were classified as neutral.
   * Noteworthy Absence: There were no articles classified as "Negative" in this batch.
3. **Agent Contribution:** This chart visualizes the summary that the AnalysisAgent would compute. The AnalysisAgent takes the full list of processed articles from the NewsAgent, counts the occurrences of each sentiment label ("positive", "neutral", "negative"), and calculates these percentage breakdowns.
   * + 1. **Chart 2: Individual News Sentiment Scores (The Micro View)**
4. **What it is:** This horizontal bar chart provides a detailed, article-by-article breakdown of the sentiment scores that were aggregated to create the pie chart.
5. **Analysis:** It plots each news headline against its calculated "Sentiment Score" (from 0.0 to 1.0).
   1. **NewsAgent's Core Output:** This is a direct visualization of the output from the NewsAgent's \_classify\_with\_llm and \_apply\_financial\_context\_adjustments methods. Each bar represents one processed news item.
   2. **High-Scoring Positive (Green) Articles:** Headlines like "Apple’s new iPhone 17 is available..." and "Apple AAPL Sees High Demand..." received perfect 1.00 scores, indicating the NewsAgent's high confidence in their positive nature.
   3. **Mid-Scoring Neutral (Orange) Articles:** Headlines like "Apple Inc. Sued Over Workplace Disc..." (0.56) and "Nvidia CEO Jensen Huang Says..." (0.52) received scores close to 0.5, which the AnalysisAgent (or the underlying NewsAgent's classification) correctly interprets as "Neutral".
   4. **Connecting the Charts:** You can see how this bar chart feeds the pie chart: there are 5 green bars (Positive) and 5 orange bars (Neutral), leading to the 50/50 split.
6. **Overall Insight from the Agents' Output**

This visualization powerfully combines the high-level summary from the AnalysisAgent with the granular evidence from the NewsAgent.

A user can instantly see the "What" (50/50 positive/neutral split) from the pie chart and then immediately investigate the "Why" by looking at the bar chart to see *which specific articles* were positive (e.g., "High Demand") and which were neutral (e.g., "Sued Over Workplace Disc."). This provides both a quick summary and the necessary detail for a human to validate the AI's findings.

* + - 1. **Chart 3: Decision component Breakdown and Radar chart:**

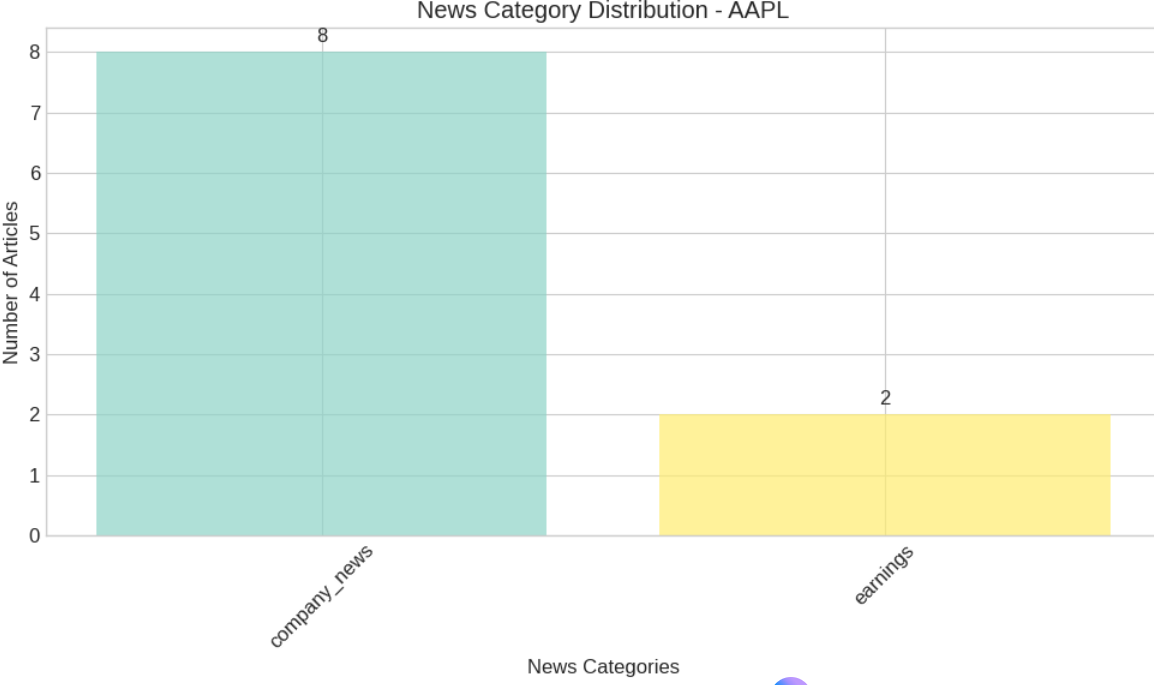
These two charts visualize the final scores (0-100) calculated by the Analysis Agent for each key factor.

A screenshot of a graph

AI-generated content may be incorrect.

1. **Analysis:** The charts show opposing forces at play, creating a "tug-of-war" effect. Pulling strongly in Apple's favor are its high scores for Profitability (75.0) and Sentiment (76.1).This indicates that the DataAgent found strong profit margins and revenue growth, while the NewsAgent's LLM-driven analysis of recent news was overwhelmingly positive.
2. **Balancing Factor:** However, these strengths are being moderated by a mediocre Valuation score (55.0), which is the lowest component. This suggests the DataAgent determined that AAPL's stock price is high relative to its earnings (P/E ratio), making it less attractive from a value perspective.
3. **Other Components:** The Technical (60.0), Analyst (65.0), and Market Position (60.0) scores are all neutral-to-positive, providing a stable floor but not driving a strong "buy" signal. This combination leads to the balanced "HOLD" recommendation seen in the summary table.

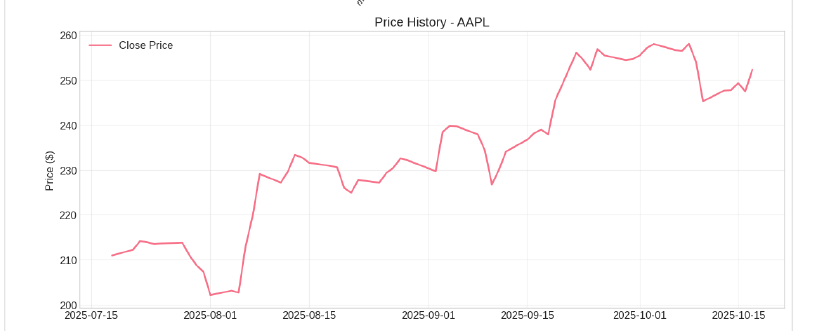
### Chart 4: News Category Distribution



This chart shows the output of the NewsAgent's Routing pattern (\_route\_to\_specialists function).

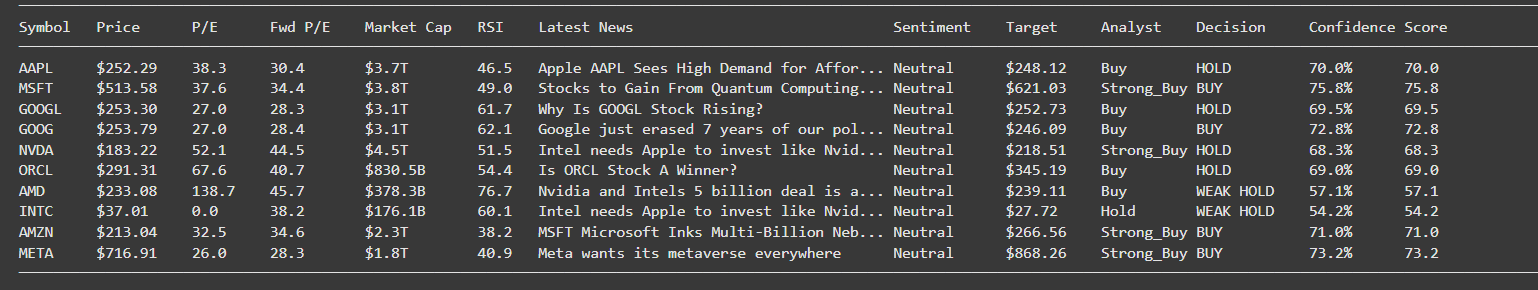
1. Analysis: Out of the 10 articles processed for AAPL, the NewsAgent successfully categorized 8 as "company\_news" (e.g., product updates, partnerships, legal news) and 2 as "earnings" (related to financial results).
2. Significance: This demonstrates the agent's ability to differentiate news types, ensuring that earnings reports (which are highly significant) are tagged appropriately. This classification directly feeds the AnalysisAgent's context.

### Chart 5: Price History & Relative Strength Index (RSI)





### The Price History and RSI are generated from data fetched and calculated by the DataAgent.

1. **Price History:** The line chart shows AAPL's stock price over the last three months (approx. July-Oct 2025). It displays a clear uptrend, peaking around the beginning of October before a recent, minor pullback.
2. **RSI:** The gauge chart shows the current RSI value is approximately **46.5**. This value is firmly in the neutral territory (between the "Oversold" threshold of 30 and the "Overbought" threshold of 70).
3. **Combined Insight:** The technical picture supports the AnalysisAgent's neutral **Technical score (60.0)**. The stock is neither over-extended nor oversold; it is in a "holding" pattern after its recent run-up, justifying a neutral technical stance.
4. **FINAL FINANCIAL STOCK RECOMMENDATION OF THE 10 STOCKS (WORK FLOW AND TABLE OUTPUT EXPLANATION):**

The output displayed in the comprehensive table along with different parameters is the final products of a collaborative workflow of multi-agents, where each agent plays a crucial role: The role of each agent in the determination of output is described below

* + 1. **FINANCIAL ANALYSIS SYSTEM (ORCHESTRATOR)**
       1. Initiation & Coordination: Starts the analysis for each symbol (analyze\_stocks).
       2. Data Flow Management: Passes the symbol to DataAgent and NewsAgent, then passes their outputs (stock\_data, news\_data) to the AnalysisAgent.
       3. Result Consolidation: Colle cts the final analysis\_result along with the inputs.
       4. Output Formatting & Display: Calls helper functions (format\_market\_cap, get\_latest\_news\_title) and uses the collected data to construct and print the final output
    2. **DATA AGENT**
       1. Data Ingestion (Tool Use): Fetches raw quantitative data using APIs (yf.Ticker).
       2. Technical Calculation: Computes RSI and MACD.
       3. Serialization: Prepares the data in a usable dictionary format.
       4. Output Contribution: Directly provides the values for columns like Current Price, P/E Ratio, Forward P/E, Market Cap, RSI, Analyst Target, and Analyst Recommendation in the final table.
    3. **NEWS AGENT**
    4. **Data Ingestion & Preprocessing** (Tool Use & Prompt Chaining Start): Fetches news (NewsApiClient), cleans text (\_clean\_text).

**Example:** For AAPL, it shows "Apple unveils the new iPhone with adv...". This title was originally ingested by \_ingest\_news and preprocessed by \_clean\_text within the NewsAgent.

* + 1. **LLM Analysis (Summarization):**

1. The NewsAgent generates summaries (\_extract\_and\_summarize). While the summaries themselves aren't directly shown in the *final table*, the *quality score* (summary\_quality\_score) calculated during this step is passed along.
2. **Example:** This quality score is used *internally* by the AnalysisAgent (\_calculate\_quality\_boost) to potentially adjust the Overall Score. So, although not directly visible, the summarization *quality* influences the final Overall Score (e.g., 85.3 for AAPL).
   * 1. **LLM Analysis (Sentiment Classification):**
3. The NewsAgent classifies the sentiment of each article using its LLM pipeline (\_classify\_with\_llm) and applies financial context/corrections (\_add\_financial\_context, \_apply\_financial\_context\_adjustments).
4. The *aggregated result* of these classifications (weighted by confidence, as done in AnalysisAgent.analyze\_sentiment) is displayed in the News Sentiment column.
   1. Example: For AAPL, the aggregate sentiment is Positive. For MSFT, it's Neutral. These labels are derived from the individual sentiment analyses performed by the NewsAgent.
      1. **LLM Analysis (Prompt Chaining):**
         1. Classifies sentiment with context injection and correction (\_classify\_with\_llm, \_add\_financial\_context, \_apply\_financial\_context\_adjustments).
         2. Summarizes articles (\_extract\_and\_summarize).
      2. **Routing:** 
         1. The NewsAgent categorizes news (\_route\_to\_specialists). This categorization isn't directly shown in the final table. However, insights extracted by these specialists (\_extract\_key\_insights) are used by the AnalysisAgent to construct the Reasoning string.
         2. **Example:** The reasoning for AAPL ("Strong profit margins; Positive news sentiment; Analyst target suggests 13.2% upside") includes "Positive news sentiment", confirming the sentiment analysis contribution. Other parts might implicitly come from insights gathered during routing (though not explicitly shown in this simple reasoning example).
      3. **Evaluation:** 
         1. The NewsAgent assesses its own analysis quality (\_evaluate\_and\_optimize) and passes these metrics (like sentiment\_confidence and summary\_quality\_score) along with the processed news data.
         2. Example: The AnalysisAgent uses these quality metrics in \_calculate\_quality\_boost to modify the final Overall Score. A high average confidence/quality from the NewsAgent's output for AAPL's news likely contributed positively to its final score of 85.3. Conversely, lower quality news analysis for another stock might slightly penalize its score
      4. **Confidence Score: Analysis of Output, Description and Derivation**

The Confidence score in your output table represents the system's degree of certainty in its final AI Decision (e.g., "BUY", "HOLD", "SELL"). It's essentially a numerical measure of how strongly the combined analytical factors support the given recommendation. A higher confidence score indicates that the various components analyzed (valuation, sentiment, technical, etc.) predominantly align with the final decision.

The Confidence score is derived directly within the AnalysisAgent, specifically in the generate\_decision method. This method takes the final, potentially optimized overall\_score (which ranges from 0 to 100) as its input.

1. **Thresholds Determine Decision**: The overall\_score is first compared against predefined thresholds (80, 70, 60, 50, 40) to determine the categorical AI Decision.
2. **Score Normalization and Capping:**

The overall\_score is normalized (divided by 100 to get a value between 0 and 1). This normalized score is then compared with a maximum confidence cap specific to that decision category (e.g., 0.95 for "STRONG BUY", 0.85 for "BUY"). The *lower* of these two values is chosen as the final Confidence score.

1. **Output:**

The method returns both the decision string and the calculated confidence score (as a float between 0 and 1), which is then formatted as a percentage in the final table.

Example:

1. If overall\_score is 75.8 , the decision is "STRONG BUY". The confidence is min(0.85.3, 75.6/100) = min(0.95, 0.758) = 0.758
2. If overall\_score is 57.1, the decision is "WEAK HOLD". The confidence is min(0.65, 57.1/100) = min(0.65, 0.571) = 0.571. The table shows 57.1%.
3. **Multi-Agent (Analysis Agent, Data Agent and News Agent) Role in Confidence score:**

The Confidence score is primarily generated and impacted by the AnalysisAgent.

1. It's a direct output of the AnalysisAgent's decision-making logic (generate\_decision).
2. It's fundamentally tied to the overall\_score, which the AnalysisAgent calculates by synthesizing inputs from the DataAgent and NewsAgent and potentially applying optimizations based on the NewsAgent's evaluation metrics (part of the Evaluator-Optimizer pattern implemented within the AnalysisAgent).

While the inputs from DataAgent and NewsAgent influence the overall\_score and thus *indirectly* affect the confidence, the *calculation and capping* of the confidence value itself happen entirely within the AnalysisAgent.

* + 1. **Output Contribution:**
       1. **Directly Visible:** The Latest News title comes from the ingested news. The News Sentiment label is the aggregated result of the NewsAgent's classification.
       2. **Indirectly Influential:** The sentiment scores, confidence levels, and summary quality scores generated by the NewsAgent are used by the AnalysisAgent to calculate the final Overall Score, Confidence, AI Decision, and parts of the Reasoning.
       3. **Two example Outputs and reason:**

1. AAPL: Strong profit margins; Positive news sentiment; Analyst target suggests 13.2% upside

Analysis Agent reasoning is based on Data Agent and News Agent data

1. MSFT: Reasonable valuation; Strong profit margins; Analyst target suggests 9.7% upside

Analysis Agent reasoning is based on Data Agent and News Agent data

* + 1. **Conclusion:**

This clearly shows how the final, user-facing output is a direct result

of the orchestrated collaboration between the specialized agents, each contributing its specific analysis to the overall assessment.

**IV FUTURE ENHANCEMENTS**

The current multi-agent system provides a robust foundation for automated financial analysis. Future enhancements can focus on deepening the analytical capabilities, improving adaptability, expanding data sources, and enhancing user interaction.

Enhancing Agent Capabilities

* DataAgent - Broader Data Integration:
  + More Sources: Integrate additional APIs for macroeconomic data (e.g., interest rates, inflation from FRED), alternative data (e.g., social media trends, satellite imagery), competitor data, and more granular financial statement details (e.g., cash flow statements, balance sheets).
  + Data Quality Handling: Implement more sophisticated methods for handling missing or inconsistent data from APIs.
* NewsAgent - Deeper Contextual Understanding:
  + Advanced LLMs/Fine-tuning: Utilize larger, more powerful LLMs or fine-tune existing models specifically on financial news datasets to better understand nuance, causal relationships, and sector-specific jargon.
  + Retrieval-Augmented Generation (RAG): Implement RAG to allow the NewsAgent to pull in relevant historical context or specific company information when analyzing a new article, leading to richer summaries and sentiment analysis.
  + Multilingual Support: Extend capabilities to process news from multiple languages.
* AnalysisAgent - Sophisticated Modeling & Risk Assessment:
  + Dynamic Weighting: Move beyond fixed weights. Implement models where the importance of factors (valuation vs. sentiment vs. technicals) can change based on market regime (e.g., high volatility vs. stable growth) or sector.
  + Machine Learning Models: Incorporate ML models (e.g., regression, classification) trained on historical data to predict potential price movements or risk levels, complementing the rule-based scoring.
  + Dedicated Risk Analysis: Introduce specific risk scoring based on factors like volatility (beta), debt levels, geopolitical news, and regulatory changes.
* EvaluationAgent - Autonomous Optimization:
  + Richer Metrics: Develop more nuanced quality metrics, potentially using LLMs to score the coherence and insightfulness of the reasoning.
  + Automated Parameter Tuning: Enable the EvaluationAgent's feedback to directly trigger adjustments in other agents' parameters (e.g., automatically adjusting summary length in NewsAgent if quality scores are consistently low).
* New Agents:
  + RiskAgent: Dedicated agent to assess various risk factors (market, credit, operational, geopolitical).
  + MacroeconomicAgent: Focuses on analyzing broader economic trends and their potential impact on specific stocks or sectors.
  + PortfolioAgent: Manages a virtual portfolio based on the system's recommendations, tracking performance and suggesting rebalancing.

Improving System Architecture & User Experience

* Asynchronous Processing: Re-architect parts of the system for asynchronous operation, allowing agents like DataAgent and NewsAgent to work concurrently more effectively, speeding up analysis, especially for multiple stocks.
* Enhanced Error Handling & Recovery: Implement more granular error tracking and potentially add mechanisms for agents to retry failed tasks or adapt the plan if a specific data source is unavailable.
* Scalability: Optimize API usage (caching), memory management, and potentially distribute agent workloads for analyzing larger numbers of stocks.
* User Interface (UI): Develop a web-based dashboard for easier interaction, allowing users to input stocks, view results and visualizations dynamically, and potentially customize analysis parameters (like decision weights).

Next Steps (Logical Progression)

1. Refine Core Agents: Start by enhancing the existing DataAgent (more data points) and NewsAgent (fine-tuning LLMs/RAG) as they provide the foundational input.
2. Develop UI: Create a simple web interface for better usability.
3. Implement EvaluationAgent Optimization: Make the feedback loop actively tune parameters.
4. Introduce New Specialized Agents: Add agents like RiskAgent or MacroeconomicAgent incrementally.
5. Build Advanced Features: Add backtesting and portfolio management capabilities once the core analysis is deemed reliable.

**V. CONCLUSION:**

This project successfully developed a multi-agent financial analysis system capable of autonomous reasoning and delivering intelligent investment insights.

Architecture effectively mirrors real-world research workflows by orchestrating specialized agents for planning, tool use, and self-reflection. Its core strength lies in the sophisticated synthesis of quantitative data from the DataAgent with nuanced, LLM-driven sentiment analysis from the NewsAgent.

The AnalysisAgent then capably provides these inputs into a weighted, explainable decision.

Furthermore, the implementation of an EvaluationAgent demonstrates a crucial self-assessment capability, enabling a feedback loop for continuous improvement. The final output, a comprehensive table supported by detailed reasoning and visualizations, provides transparent and actionable analysis.

This project successfully demonstrates the potential of collaborative AI agents to automate and elevate complex financial research. Ultimately, the system serves as a potent decision-support tool, efficiently transforming raw data into reasoned investment perspectives.

**VI. REFERENCES :**

1. Xiao Liu, et al. (2023). *AgentBench: Evaluating LLMs as agents*. Preprint manuscript submitted for publication. arXiv. <https://arxiv.org/abs/2308.03688>

2. Zhiheng Xi, Wenxiang Chen., et al. (2023). *The rise and potential of large language model based agents: A survey*. Preprint manuscript submitted for publication. arXiv. <https://arxiv.org/abs/2309.07864>

3. Li, Y., Wang, S., Ding, H., & Chen, H. (2023). Large language models in finance: A survey. *arXiv*. <https://doi.org/10.48550/arXiv.2311.10723>

4. Wang, L., et al. (2023). *A survey on large language model based autonomous agents*. Preprint manuscript submitted for publication. arXiv. <https://arxiv.org/abs/2308.11432>