

Autonomous Multi-Agent System for Generative Algorithmic Trading

System Architecture and Workflow Documentation

Group No. 24

Soham Hanmane

Rahul Kumar

Rohit Gautam

Shaswat

Adarsh Singh

November 14, 2025

Contents

1	Abstract	3
2	System Architecture Overview	3
3	Agent 1: Data Acquisition	3
3.1	Purpose	3
3.2	Operations	4
3.3	Output	4
4	Agent 2: Feature and Indicator Engineering	4
4.1	Purpose	4
4.2	Indicator Categories	4
4.3	Workflow	4
4.4	Quality Check	4
5	Agent 3: LLM-Based Strategy Generation	5
5.1	Purpose	5
5.2	Preliminary Quantitative Scan	5
5.3	LLM Prompt Construction	5
5.4	LLM Interaction	5
5.5	Parsing and Storage	5
6	Agent 4: Backtesting and Validation	6
6.1	Purpose	6
6.2	Workflow	6
6.3	Outputs	6
6.4	Ranking	6
7	Final Portfolio Construction	6
7.1	Purpose	6
7.2	Workflow	7
7.3	Output	7
8	Conclusion	7

1 Abstract

This document provides a detailed description of the workflow, internal mechanics, and interactions within our autonomous multi-agent pipeline for generative algorithmic trading. Instead of focusing on specific numerical results, the emphasis here is on how each agent operates, how data flows through the system, and how the architecture enables automated strategy discovery. The system integrates quantitative data engineering, rule-based analysis, large language model reasoning, and robust backtesting, forming a fully automated research loop capable of generating new trading strategies with minimal human intervention.

2 System Architecture Overview

The system is designed as a modular, multi-agent pipeline with clear separation of responsibilities. Each agent specializes in one stage of the quantitative research workflow. Data flows sequentially from one agent to the next, enabling a structured transformation from raw historical data to fully validated trading strategies.

The four-agent architecture is:

1. **Agent 1: Data Acquisition**
2. **Agent 2: Feature and Indicator Engineering**
3. **Agent 3: LLM-Based Strategy Generation**
4. **Agent 4: Backtesting and Validation**

This architecture resembles a traditional quant research pipeline but introduces generative intelligence at its core. All intermediate and final data products are stored in a unified dictionary:

```
out = {  
    'ranked': ...,  
    'test_returns': ...,  
    'llm_strategies': ...,  
    'backtests': ...,  
    'equal_weight': ...  
}
```

This design ensures complete reproducibility and traceability.

3 Agent 1: Data Acquisition

3.1 Purpose

Agent 1 collects all required historical market data for the selected universe of stocks. It acts as the system's data foundation.

3.2 Operations

- Connect to a market data source (e.g., Yahoo Finance).
- Download daily OHLCV data across the full time range.
- Align dates across all tickers.
- Clean and sanitize missing values.
- Ensure consistency of data types and column formats.

3.3 Output

For each ticker, Agent 1 produces a clean pandas `DataFrame` structured as:

```
Date | Open | High | Low | Close | Volume
```

These `DataFrames` are passed directly to Agent 2.

4 Agent 2: Feature and Indicator Engineering

4.1 Purpose

Agent 2 transforms raw price data into richer, feature-enhanced datasets by computing a diverse set of technical indicators.

4.2 Indicator Categories

- Trend Indicators (SMA, EMA, MACD)
- Momentum Indicators (RSI, ROC, Stochastic)
- Volatility Measures (ATR, Bollinger Bands)
- Volume Indicators (OBV, Volume Rate of Change)

4.3 Workflow

1. Receive OHLCV data from Agent 1.
2. Compute all selected indicators using rolling window methods.
3. Merge all indicator columns into a unified feature `DataFrame`.

4.4 Quality Check

The downstream presence of:

```
out['backtests']
```

confirms successful feature construction, because backtests cannot run without fully engineered datasets.

5 Agent 3: LLM-Based Strategy Generation

5.1 Purpose

This is the intelligence core of the pipeline. Agent 3 is responsible for discovering new trading strategies by combining quantitative analysis with LLM-based reasoning.

5.2 Preliminary Quantitative Scan

Before consulting the LLM, the system performs a structured rule scan:

1. Generate a set of simple indicator-based rules.
2. Backtest each rule on the training dataset.
3. Compute performance metrics: Sharpe, drawdown, CAGR, trade count.
4. Rank the rules and select the top signals.

This step ensures that the LLM works with meaningful, data-driven “evidence” instead of raw text prompts.

5.3 LLM Prompt Construction

The system constructs a detailed prompt containing:

- The top signals and their conditions.
- A description of market behavior observed in the training period.
- Instructions for generating entry and exit rules.
- Format requirements for structured output.

5.4 LLM Interaction

A large language model processes the prompt and generates:

- Multi-condition entry rules.
- Logical exit conditions.
- Optional filters (trend filters, volatility constraints, etc.).

5.5 Parsing and Storage

The natural-language output is parsed into structured rule definitions and stored in:

```
out['llm_strategies'][ticker]
```

6 Agent 4: Backtesting and Validation

6.1 Purpose

Agent 4 evaluates strategies generated by Agent 3 using a reliable and standardized backtesting framework.

6.2 Workflow

1. Translate LLM-generated rules into Python-executable conditions.
2. Attach them to a strategy class in the backtesting engine.
3. Perform full historical backtests on a validation dataset.
4. Measure risk-adjusted returns and consistency.

6.3 Outputs

Each ticker's backtest is stored in:

```
out['backtests'][ticker]
```

These objects include:

- Return metrics
- Drawdown data
- Trade logs
- Equity curves

6.4 Ranking

Agent 4 computes a composite score to select the most robust strategies. This ranking is stored in:

```
out['ranked']
```

7 Final Portfolio Construction

7.1 Purpose

After filtering and ranking strategies, the system constructs a diversified, multi-strategy portfolio.

7.2 Workflow

1. Select top-ranked strategies from Agent 4.
2. Extract their test-period return series.
3. Combine them using equal weights.
4. Produce an aggregated portfolio return series.

7.3 Output

Stored as:

```
out['equal_weight']
```

This represents the final, system-level output: a portfolio combining all validated strategies.

8 Conclusion

The autonomous multi-agent system provides an end-to-end workflow for generating and validating algorithmic trading strategies. The system's strength lies in its structured pipeline:

- Raw data is transformed into meaningful quantitative features.
- Preliminary analysis extracts real trading insights.
- A large language model synthesizes strategies using this evidence.
- A rigorous backtesting engine validates every hypothesis.
- The strongest strategies are combined into a final portfolio.

By separating responsibilities into modular agents, the pipeline achieves clarity, extensibility, and automation. The system can be re-run on different markets, assets, or time periods with minimal changes, enabling scalable quantitative research powered by generative intelligence.