AI-Powered Earnings Call Analysis System

Table of Contents

[1. Project Overview 1](#_Toc204636516)

[2. Choice of framework 1](#_Toc204636517)

[3. AutoGen System Architecture 2](#_Toc204636518)

[4. File Structure for KEY Components 3](#_Toc204636519)

[5. Report format & Agent Collaboration Details 4](#_Toc204636520)

[6. Notable Challenges & Solutions 8](#_Toc204636521)

[1. Agent Randomness 8](#_Toc204636522)

[2. Tool Integration 8](#_Toc204636523)

[3. Agent Communication & Feedback Loop 9](#_Toc204636524)

[4. Output Formatting 9](#_Toc204636525)

[5. Missing Metadata in Transcripts 9](#_Toc204636526)

[6. Agent Scheduling Errors 10](#_Toc204636527)

[7. Incomplete API Coverage 11](#_Toc204636528)

[8. Prompt for Investment Recommendations 12](#_Toc204636529)

# Project Overview

This project implements an advanced multi-agent AI system for analyzing earnings call transcripts and generating comprehensive investment reports. The system utilizes AutoGen's multi-agent framework to create a collaborative environment where different AI agents work together to produce accurate, detailed, and actionable investment insights. To enhance the depth and relevance of the analysis, agent tools are incorporated to call external financial data sources, allowing the system to enrich its insights with up-to-date market and company-specific information.

# Choice of framework

I considered using CrewAI for this project since it's a solid option for multi-agent orchestration, especially when tasks are linear and follow a clear role-based flow. However, for my use case, I needed something that supports more dynamic and collaborative interactions between agents, something where agents can not only perform tasks but also critique, build on, and refine each other’s work in real time. That’s where AutoGen’s GroupChat framework stood out. It’s much better suited for this kind of multi-agent collaboration, thanks to features like shared memory, contextual turn taking, and built-in support for tool use within a conversational setup. It just fits better for coordinating agents in a complex, back-and-forth financial analysis workflow.

# AutoGen System Architecture

A screenshot of a computer

AI-generated content may be incorrect.

**The AutoGen framework has four specialized agents:**

1. **Client Agent**: Acts as the project manager and quality controller. This agent decides when to terminate the [groupchat](https://microsoft.github.io/autogen/stable/user-guide/core-user-guide/design-patterns/group-chat.html) and to produce output or instruct writer to revise further.
2. **Writer Agent**: Responsible for initial report drafting based on transcript analysis and revision based on feedback from Analyst and Editor Agents.
3. **Analyst Agent**: Performs fact-checking, financial ratio calculations and news sentiment analysis using external data (Alphavantage API) via two agent tools:
   1. historicalfinancialdata(ticker, year, quarter) # Returns the current quarter, previous quarter's and previous year's financial data (EPS, cash flow, income statement, balance sheet) using Alpha Vantage.
   2. analyzemarketsentiment(ticker, year, quarter) # Fetches news articles about a company from Alpha Vantage within the 30 days prior to the earnings report date. (**Date Range Rationale**: we assume operational realism where we only have access to news before the reported earnings call, to prevent future-leaking from post-call news that might cause sentiment analysis to be bias).
4. **Editor Agent**: Ensures report quality, clarity, and completeness of report.

**Agent Workflow:**

Sequence: Writer → Analyst → Writer → Editor → Writer → Client

1. After Writer's initial draft: Select Analyst
2. After Analyst's feedback: Select Writer for revision
3. After Writer's revision: Select Editor
4. After Editor's feedback: Select Writer for final revision
5. After Writer's final revision: Select Client
6. After Client Checks: Output Report **/** Reloop to Revise

# 4. File Structure

/ (root project)

├── autogenAI.py # Main multi-agent orchestration script

├── app.py # Streamlit app for running the platform

├── requirements.txt

├── README.md

├── Documentation/ # Project documentation and diagram

├── Earnings2Insights/ # Main dataset and processing directory

│ ├── Dataset/

│ │ ├── ECTsum/ # ECTsum dataset

│ │ └── Professional/ # Professional dataset

│ ├── Convert\_MD\_Report\_To\_Json\_Tools/

│ │ ├── Convert\_all\_MD\_Reports.py # Batch conversion script for converting markdown reports to JSON format

│ │ └── print\_report\_based\_on\_ECC.py # Report viewer by ECC transcript ID

│ ├── Generated\_Reports/

│ └── Earnings2Insights\_Result\_final.json # Processed results for submission

│

├── research\_tools.py # Financial data and analysis tools

└── test\_research\_tools.py # Test script for testing tools

(for more details on implementation refer to README.md and the respective scripts)

# Report format & Agent Collaboration Details

**Report Format:**

A screenshot of a report

AI-generated content may be incorrect.

A screenshot of a screen shot of a chart

AI-generated content may be incorrect.A screenshot of a document

AI-generated content may be incorrect.A close-up of a document

AI-generated content may be incorrect.

A screenshot of a document

AI-generated content may be incorrect.A screenshot of a document

AI-generated content may be incorrect.

**Agent Pseudo Dialogue:**

1. Writer agent drafts the above report but leaves the following subsections empty with placeholders for feedback agents to populate:
   1. Section 1 (Financial Analysis): Financial Ratio Table //Populated by Analyst
   2. Section 2 (Market Analysis): News Sentiment //Populated by Analyst
   3. Section 4 (Investment Recommendation): [Whole section] //Populated by Editor

Rationale for drafting report based solely on transcript:

1. Base requirement of hackathon which is to analyse based on transcript only
2. In the event tools call fails to retrieve external financial data to enrich the analysis, at least writer agent has extracted applicable financial metrics from the transcript to work with.
3. Analyst Agent then calls Historicalfinancialdata and analyzemarketsentiment tools (more details see [section 3](#_AutoGen_System_Architecture)) to provide feedback to writer to **update** the financial metrics table and **populate** the rest of section 1 (financial analysis).

Rationale to **not** populate financial ratio in the initial draft by writer: The financial ratios are typically not available in the EC transcript and is calculated during Historicalfinancialdata function call using the financial data (metrics) from alphavantage api, with the goal to enrich the overall financial analysis.

1. Editor Agent then proceed to generate section 4 (investment recommendations) based on the **updated** section 1 to 3 (financial analysis, market analysis and risk assessment), while also providing feedbacks regarding the completeness and formatting of the report (table, indents, space , etc)
2. Client Agent review the latest report generated by the writer and runs against its own checklist to decide whether the report is of satisfactory standards.

If yes, terminate, and the app extracts writer’s latest report to display in the streamlit UI.

Else, it will call the writer agent to revise based on its own feedback. At this point, the writer may or may not call the feedback agents again to remediate based on the feedback from client. Once remediated, the writer will handoff the report to the client for another round of check. This cycle repeats for two round to balance between accuracy and cost (LLM token).

# 6. Notable Challenges & Solutions

### 1. Agent Randomness

**Challenge:** Output varied, with writer agent omitting important details. For example, formula for financial ratios.

A screenshot of a black and white screen

AI-generated content may be incorrect.

**Solution:**

1. Enhanced the system prompt for writer agent with **explicit reminders s**uch as instructing the Writer Agent to **always include formulas** when generating financial ratios.
2. **Increased the temperature** value to **0.95** to promote more **consistent and deterministic responses**.

### 2. Tool Integration

#### a. Historical Financial Data API

**Challenge:** Difficulty in retrieving accurate data for current quarter, previous quarter, and previous year to compute financial ratios.

**Solution:** Refined the tool functions (historicalfinancialdata) to **correctly calculate the financial quarter end dates** based on the target company's fiscal calendar.

#### b. News Sentiment API

**Challenge: We should extract news around the reported date range for a more accurate news sentiment analysis and**reported date for quarter report is typically later than the quarter end date

**Solution:** Updated (analyzemarketsentiment) to **derive reportedDate from the quarter report**

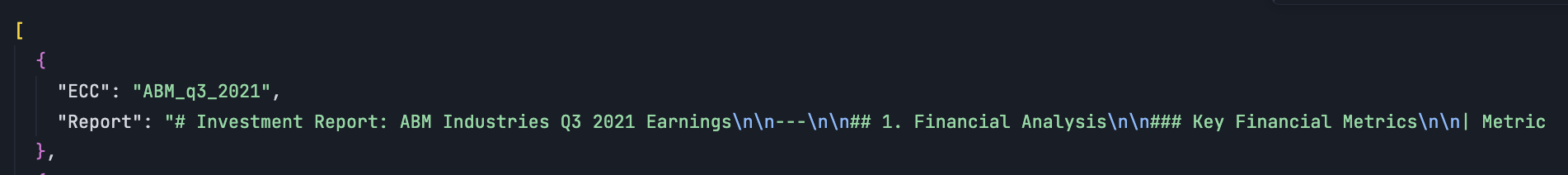
### 3. Agent Communication & Feedback Loop

**Challenge:** Writer Agent sometimes ignored feedback from Analyst Agent due to unclear prompts. For example, after the analyst agent hands off back to writer agent, the writer agent did not update the financials ratio table and left it blank which caused it to fail when the client check and creates unnecessary loops and wastes LLM tokens.

**Solution:** Added **explicit placeholders and structured instructions** in the Writer Agent’s system prompt to **trigger strict revision behavior** upon receiving feedback.

### 4. Output Formatting

**Challenge:** Final output needed to be a **single-line string** under the report key in JSON, but agents generated multi-line text.



**Solution:** Developed a convert\_report.py script to **replace newlines with \n**, preserving formatting in a single-line JSON-compliant string.

### 5. Missing Metadata in Transcripts

**Challenge:** Some earnings transcripts lacked **ticker, quarter, or year**, leading to incorrect tool function arguments.

A screenshot of a computer

AI-generated content may be incorrect.

**Solution:** Extracted these metadata values directly from the **transcript file name** to ensure correct API calls. For example, CPF\_q4\_2019 => ticker: CPF, quarter: q4, year: 2019

6. **Challenges**: Earnings Call transcript timestamp and Folder Name mismatch. In the example below, the folder enclosing the transcript for Hilltop Holding’s Q4 2019 Earnings Call is enclosed in the the folder named ‘HTH\_q4\_2020’. There are numerous of such instances which caused the analyst agent to fetch financial data for the wrong quarterly reports, since my API call arguments: Ticker, Year and Quarter metadata values depend on the file name of each transcript.

A screenshot of a computer

AI-generated content may be incorrect.

**Solution**: I renamed the transcript file name to match the content of the transcript as seen in the green box above, so that I can extract the correct metadata for my API calls.

### 6. Agent Scheduling Errors

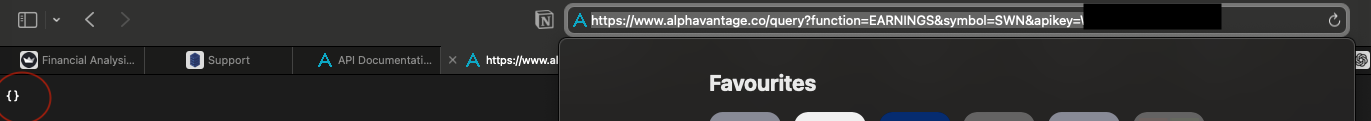
**Challenge:** Occasionally, agents executed **out of the defined sequence**, breaking workflow logic and triggering consecutive reply cap which terminates the current loop and restarts the whole analysis causing unnecessary token consumption. A screenshot of a computer

AI-generated content may be incorrect.

**Solution:** [Pending – exploring manual sequencing]

### 7. Incomplete API Coverage

**Challenge:** Some companies (e.g., Southwestern Energy Company, SWN) had **no available data** in AlphaVantage.



A screenshot of a computer program

AI-generated content may be incorrect.

**Solution:** In such cases, we fallback to **earnings transcript content** to extract necessary financials manually.

8. Prompt for Investment Recommendations

**Challenge**:

Providing instructions for investment recommendations for next day, week and month based on the 3 analyses.

**Solution**: I structured the investment recommendation instructions to ensure each timeframe (next day, next week, and next month), reflects a different analytical emphasis, while still incorporating all three core perspectives: financials, market dynamics, and risk. The idea is to weight these differently based on what realistically drives stock movements in each window. For example, next-day recommendations lean more heavily on financial metrics like EPS or revenue beats, since markets tend to react immediately to hard numbers. Next-week calls shift toward market positioning and sentiment as investor focus broadens to sector trends and competitive standing. For the next month, I want the agent to consider the company’s strategic trajectory, macroeconomic conditions, and fundamental value. That said, each recommendation should still be holistic, factoring in financial performance, market signals, and risk exposure because real-world investing never hinges on just one lens.