Open in Colab

Group 5

AAI 520 - Final Project

Authors: Kevin Hooman, Spencer Cody, Tommy Poole

Fall 2025

Project Outline

This is an LLM powered research assistant that finds key financial details on publicly traded companies using their stock ticker. The assistant is able to plan and execute key tasks leveraging several tools in order to generate a final output.

Core Agent Functions

- Planning: The assistant has the ability to plan at each step.
- Tool Use: The assistant dynamically integrates APIs such as Yahoo Finance, SEC, as well as News APIs, we search APIs, and other tools.
- Self-Reflection: The assistnat has a specific agent focused on evaluation and output quality, including human in the loop feedback before concluding.
- Memory/Learning: The assistant also has memory to maintain context across runs.

Workflow Patterns

- Prompt Chaining: The agent dynamically follows a prompt chaining pattern between tool aclls including: News ingestion → preprocessing → classification → extraction → summarization
- Routing: The agent dynamically routes to specialist analyzers setup as tools to review earnings, draw graphs, and analyze market sentiment.
- Evaluator-Optimizer: The assistant ultimately performs an evaluation and optimizes the output incorporating human feedback.

Attribution:

This code leveraged several sources including.

1. LangGraph Intro to LangGraph course: https://academy.langchain.com/courses/take/intro-to-langgraph/
2. Hugging Face Course:https://huggingface.co/learn/agents-course/en/
3. Geek for Geeks Tutorial: https://www.geeksforgeeks.org/artificial-intelligence/introduction-to-langchain/
4. Perplexity AI for research and debugging. Accessed during October, 2025
5. Google Gemini within Colab for coding and debugging. Accessed during October, 2025

Setup

The project incorporates several tools and libraries which are imported here.

In [1]:
```
# %%capture --no-stderr
# %pip install --quiet -U \
#     langgraph \
#     langchain_openai \
#     langgraph_sdk \
#     langgraph-prebuilt \
#     langchain-perplexity \
#     huggingface_hub \
#     openai \
#     yfinance \
#     langchain-google-genai \
#     langchain-community \
#     transformers \
#     datasets \
#     bitsandbytes accelerate sentence-transformers \
#     faiss-cpu \
#     ddgs\
#     langchain-huggingface
```

In [11]:
```
#Import key libraries
# LLM Providers (OpenAI, Google Gemini, Perplexity AI, HuggingFace)
# LangChain and LangGraph Primitives (Agents, Tools, Chains, Prompts, Memory)
# Datasources and parsing helpers (yfinance, DuckDuckGo, Wikipedia, beautifulsoup,
# Typing and utility modules for memory and tool actions

import os
import requests
import re
import textwrap
import matplotlib.pyplot as plt
import pandas as pd
from datetime import datetime, timedelta
from pprint import pprint, pformat

from bs4 import BeautifulSoup

from langchain_core.tools import StructuredTool

from IPython.display import Markdown, display
from langchain_google_genai import ChatGoogleGenerativeAI
from langchain_community.chat_models import ChatPerplexity #updated
from langchain_openai import OpenAI, ChatOpenAI
from langchain_core.output_parsers import StrOutputParser
from langgraph.prebuilt import create_react_agent
import yfinance as yf
from langchain_community.tools.yahoo_finance_news import YahooFinanceNewsTool
from transformers import pipeline
from langchain_community.document_loaders import WebBaseLoader
from langchain.tools import tool as Tool

#Memory
from dataclasses import dataclass
from datetime import datetime
from typing import Dict, List, Optional, Any, Union
```

API Keys and LLMs

For experimentation, multiple LLMs were included. The following APIs are required in the code:

1. Gemini: https://ai.google.dev/
2. Hugging Face: https://huggingface.co/
3. OpenAI: https://openai.com/api/
4. Fin News:https://finnhub.io/docs/api/market-news
5. Tavily: https://www.tavily.com/

Add your keys for Gemini, HuggingFace, OpenAI, News, Tavily,and Perplexity in the "Secrets" section. (If you are running this notebook locallly, swith to (.env) + (dotenv))

Example: from dotenv import load_dotenv; load_dotenv(); then os.getenv("OPENAI_API_KEY") etc.

In the cells below we set up our APIs and initialize LLMs

In [12]:
```
#Run in Google Colab
# from google.colab import userdata

# #API Keys from ColAb Secrets
# gemini_key = userdata.get('GEMINI') #Used for agents in setup and validation
# hf_key = userdata.get('HF_TOKEN') #Used for agents in setup and validation, also
# openai_key = userdata.get('OPENAI')
# fin_news = userdata.get('FIN_API_KEY') #For web search on financial news
# tavily_key = userdata.get('TAVILY_API_KEY') #For web search
#######################################################################

gemini_key = os.getenv('GEMINI_API') #Used for agents in setup and validation
hf_key = os.getenv('HF_TOKEN') #Used for agents in setup and validation, also for s
openai_key = os.getenv('OPENAI_API')
fin_news = os.getenv('NEWS_API') #For web search on financial news
tavily_key = os.getenv('TAVILY_API') #For web search
```

Note: We experimented with more APIs during the project, such as Perplexity, and others, but landed on this core set.

In [13]:
```
#Download and set up Hugging Face model
from huggingface_hub import login #Connect to HF using API key accessed above
login(token=hf_key)

import warnings

#Suppress length warnings
warnings.filterwarnings("ignore", category=UserWarning, module="huggingface_hub.uti

llm_hf = pipeline(
    "text-generation",
    model="meta-llama/Llama-3.2-1B-Instruct",
    device= -1
    )
```

```
Note: Environment variable`HF_TOKEN` is set and is the current active token independ
ently from the token you've just configured.
config.json:   0%|          | 0.00/877 [00:00<?, ?B/s]
C:\Users\speco\anaconda3\envs\python310\lib\site-packages\huggingface_hub\file_downl
oad.py:143: UserWarning: `huggingface_hub` cache-system uses symlinks by default to
efficiently store duplicated files but your machine does not support them in C:\User
s\speco\.cache\huggingface\hub\models--meta-llama--Llama-3.2-1B-Instruct. Caching fi
les will still work but in a degraded version that might require more space on your
disk. This warning can be disabled by setting the `HF_HUB_DISABLE_SYMLINKS_WARNING`
environment variable. For more details, see https://huggingface.co/docs/huggingface_
hub/how-to-cache#limitations.
To support symlinks on Windows, you either need to activate Developer Mode or to run
Python as an administrator. In order to activate developer mode, see this article: h
ttps://docs.microsoft.com/en-us/windows/apps/get-started/enable-your-device-for-deve
lopment
  warnings.warn(message)
Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Fal
ling back to regular HTTP download. For better performance, install the package wit
h: `pip install huggingface_hub[hf_xet]` or `pip install hf_xet`
```

```
model.safetensors:   0%|          | 0.00/2.47G [00:00<?, ?B/s]
generation_config.json:   0%|          | 0.00/189 [00:00<?, ?B/s]
tokenizer_config.json:   0%|          | 0.00/54.5k [00:00<?, ?B/s]
tokenizer.json:   0%|          | 0.00/9.09M [00:00<?, ?B/s]
special_tokens_map.json:   0%|          | 0.00/296 [00:00<?, ?B/s]
Device set to use cpu
```

🤖 Local LLM (Hugging Face)

In this block we load an instruction tuned local model, `HuggingFaceTB/SmolLM2-1.7B-Instruct`, so the agent can generate text without external APIs. We initialize the tokenizer and model with `transformers`, then create a `text-generation` pipeline that caps output length (`max_new_tokens=256`), keeps responses focused (`temperature=0.7`), and runs on CPU by default (`device=-1`, switch to a GPU id if available).

We wrap the pipeline with LangChain's `HuggingFacePipeline`, yielding `llm_hf` as a drop in LLM for our agents.

In [15]:
```python
from transformers import AutoModelForCausalLM, AutoTokenizer, pipeline
from langchain_huggingface import HuggingFacePipeline

model_id = "HuggingFaceTB/SmolLM2-1.7B-Instruct"
tokenizer = AutoTokenizer.from_pretrained(model_id)
model = AutoModelForCausalLM.from_pretrained(model_id)

hf_pipe = pipeline(
    "text-generation",
    model=model,
    tokenizer=tokenizer,
    max_new_tokens=256,
    temperature=0.7,
    device=-1
)

llm_hf = HuggingFacePipeline(pipeline=hf_pipe)
```

```
Device set to use cpu
```

In [16]:
```python
#Setup Gemini to use in Agents
llm_gemini = ChatGoogleGenerativeAI(
    model="gemini-2.0-flash",
    google_api_key=gemini_key
)
```

Setting the temperature to zero for OpenAI LLM helps us maximize determinizm and reduce hallucination in steps that must be exact.

In [17]:
```python
#Setup Open AI to use
llm_openai = ChatOpenAI(
    model="gpt-3.5-turbo",  # update to your prefered/available model
    openai_api_key=openai_key,  # Your OpenAI API key
    temperature=0.0              # update needed
)
```

In [18]:
```
#Test APIs
print("OpenAI Response:")
response_openai = llm_openai.invoke("Explain how neural networks work in 10 words o
print(response_openai.content)

print("\nGemini Response:")
response_gemini = llm_gemini.invoke("Explain how neural networks work in 10 words o
print(response_gemini.content)

print("\nHugging Face Response:")
response_hf = llm_hf.invoke("Explain how neural networks work in 10 words or less."
print(response_hf) # Keeping this as 'response_hf' for now to match the original pr
```

```
OpenAI Response:
Neural networks learn patterns from data to make predictions.

Gemini Response:
Learn patterns from data through interconnected nodes.

Hugging Face Response:
Explain how neural networks work in 10 words or less.

1. Input **X**, data, or pattern.
2. Transfer **Y** to **Z**, hidden layer.
3. **Z** to **W**idden layer.
4. **W** hidden layer to **A**, output layer.
5. **A** to **O**utput, prediction.
6. Backprop, **Y**-**X** to correct **A**, **Z**-**W**.
7. Repeat until **A** matches **Y**, optimizes **Z** to **X**.

How does a neural network work?
```

Learning/Memory

Memory across analysis runs

🧠 Session Memory (why and how)

This block gives the agent a lightweight way to learn across runs. We keep a small, bounded store of notes per ticker (`MemoryItem` records with symbol, Q&A, timestamp, and metadata) inside `SessionMemory`. New takeaways are written with `remember`, and oldest entries are evicted automatically both per symbol and globally to keep memory lean. We can fetch the most recent insight with `latest` or a specific answer with `recall`. A simple `extract_symbol` helper pulls a likely ticker from free text, and `as_text` normalizes tool or agent outputs before saving. In our pipeline, the planner reads the latest note to adjust parameters and after generating the brief we write a concise takeaway so the next run starts smarter.

In [19]:
```python
#MEMORY AGENT
#Session-scoped memory

@dataclass
class MemoryItem:
    symbol: str
    question: str
    answer: str
    created_at: str
    meta: Dict[str, Any]

class SessionMemory:
    def __init__(self, max_items: int = 200, max_per_symbol: int = 10):
        self._store: Dict[str, List[MemoryItem]] = {}
        self.max_items = max_items
        self.max_per_symbol = max_per_symbol

    def remember(self, symbol: str, question: str, answer: str, **meta) -> None:
        symbol = (symbol or "GENERIC").upper().strip()
        item = MemoryItem(
            symbol=symbol,
            question=(question or "").strip(),
            answer=(answer or "").strip(),
            created_at=datetime.utcnow().isoformat(timespec="seconds"),
            meta=meta or {}
        )
        bucket = self._store.setdefault(symbol, [])
        bucket.append(item)
        if len(bucket) > self.max_per_symbol:
            del bucket[0 : len(bucket) - self.max_per_symbol]
        self._cap_global()

    def recall(self, symbol: str, question: Optional[str] = None) -> Optional[str]:
        symbol = (symbol or "GENERIC").upper().strip()
        bucket = self._store.get(symbol, [])
        if not bucket:
            return None
        if not question:
            return bucket[-1].answer
        q = (question or "").strip()
        for item in reversed(bucket):
            if item.question == q:
                return item.answer
        return None

    def latest(self, symbol: str) -> Optional[MemoryItem]:
        symbol = (symbol or "GENERIC").upper().strip()
        bucket = self._store.get(symbol, [])
        return bucket[-1] if bucket else None

    def _cap_global(self):
        all_items = []
        for sym, bucket in self._store.items():
            for it in bucket:
                all_items.append((it.created_at, sym, it))
        if len(all_items) <= self.max_items:
            return
        all_items.sort(key=lambda x: x[0])  #Oldest first
        to_drop = len(all_items) - self.max_items
        cutoff = set(id(it) for _, _, it in all_items[:to_drop])
        for sym in list(self._store.keys()):
            self._store[sym] = [it for it in self._store[sym] if id(it) not in cuto



def extract_symbol(text: str) -> str:
    """
    Grab a likely ticker from the user_input like 'Analyze the SPY stock ticker'.
    Simple heuristic: first ALL-CAPS token 1-5 chars (e.g., AAPL, MSFT, SPY).
    Falls back to 'GENERIC' if none found.
    """
    if not text:
        return "GENERIC"
    candidates = re.findall(r"\b[A-Z]{1,5}\b", text)
    #Light filter for common English words
    stop = {"THE","AND","FOR","WITH","FROM","THIS","THAT","YOUR","HAVE","HOLD"}
    for c in candidates:
        if c not in stop:
            return c
    return "GENERIC"

def as_text(x: Any) -> str:
    """
    Normalize whatever comes back from planner/tools/evaluator/optimizer into a str
    Works with LangChain AgentExecutor outputs (dict), AIMessage, or raw str.
    """
    try:
        # AIMessage / ChatMessage
        if hasattr(x, "content"):
            return str(x.content)
        # Agent-like dicts
        if isinstance(x, dict):
            if "output" in x and isinstance(x["output"], str):
                return x["output"]
            if "messages" in x and isinstance(x["messages"], list):
                return "\n\n".join(
                    (m.content if hasattr(m, "content") else str(m))
                    for m in x["messages"]
                )
        # plain string
        if isinstance(x, str):
            return x
        return str(x)
    except Exception:
        return str(x)


def query_memory(stock_symbol, session_memory):
    print('query memory stock symbol = ', stock_symbol)
    if stock_symbol in session_memory._store:
        recent_memory = session_memory._store.get(stock_symbol)[-1].answer
        return recent_memory
```

In [20]:
```
SESSION_MEMORY = SessionMemory() #Create session memory for overall function
```

Tool Agent

The tools agent performs the majority of the agent functions outlined for this project. By giving one agent access to these tools and specialists, it is able to plan, and dynamically select which tool is the best one to accomplish the goal.

This approach leverages the automony that agents can have to perform dynamic research tasks.

First the tools are defined, including testing/debugging code step by step.

Tool Functions

In [53]:
```
#Yahoo Finance Tool
def get_stock_summary(ticker: str) -> str:

    """Gets detailed financial info for a given stock symbol."""

    try:
        stock = yf.Ticker(ticker) #Use teh defined stock ticker
        data = stock.info

        #Return as much data from the API as you can to fill in the report
        return (
            f"Company: {data.get('longName', 'N/A')} ({ticker})\n"
            f"Sector: {data.get('sector', 'N/A')}\n"
            f"Price: ${data.get('regularMarketPrice', 'N/A')}\n"
            f"Market Cap: {data.get('marketCap', 'N/A')}\n"
            f"P/E Ratio: {data.get('trailingPE', 'N/A')}\n"
            f"52-Week High: {data.get('fiftyTwoWeekHigh', 'N/A')}\n"
            f"52-Week Low: {data.get('fiftyTwoWeekLow', 'N/A')}\n"
            f"Dividend Yield: {data.get('dividendYield', 'N/A')}\n"
            f"Beta: {data.get('beta', 'N/A')}\n"
            f"Revenue: {data.get('totalRevenue', 'N/A')}\n"
            f"Profit Margin: {data.get('profitMargins', 'N/A')}\n"
        )
    except Exception as e:
        return f"Error retrieving data for {ticker}: {e}"

#Define the LangGraph tool
yahoo_api_tool = StructuredTool.from_function(
    name="YahooFinanceAPI",
    func=get_stock_summary,
    description="Retrieves detailed Yahoo Finance company data such as price, valua
)
```

In [55]:
```
#For debugging
print(yahoo_api_tool.invoke("AAPL"))
```

```
Company: Apple Inc. (AAPL)
Sector: Technology
Price: $252.29
Market Cap: 3744081903616
P/E Ratio: 38.341946
52-Week High: 260.1
52-Week Low: 169.21
Dividend Yield: 0.41
Beta: 1.094
Revenue: 408624988160
Profit Margin: 0.24295999
```

In [56]:
```
import xml.etree.ElementTree as ET

#Define a summarizer for SEC content
def summarize_sec_document(doc_url: str) -> str:
    headers = {"User-Agent": "tpoole@sandiego.edu"}
    resp = requests.get(doc_url, headers=headers)
    if resp.status_code != 200:
        return f"Could not access doc: {doc_url}"
    doc_text = resp.text

    # Try XML parsing first (most filings)
    try:
        root = ET.fromstring(doc_text)
        values = []
        for item in root.iter():
            if item.tag.lower() in ['transactionamount', 'transactioncode', 'issuer
                values.append(f"{item.tag}: {item.text}")
        return " | ".join(values) if values else "No key XML findings found."
    except ET.ParseError:
        # Fallback: try HTML/text parsing
        soup = BeautifulSoup(doc_text, 'html.parser')
        summary = soup.get_text(separator=' ', strip=True)
        return summary[:400]  # Return first 400 chars as summary

def get_sec_filings(ticker: str) -> str:
    cik_url = "https://www.sec.gov/files/company_tickers.json"
    headers = {"User-Agent": "your_email@example.com"}
    cik_resp = requests.get(cik_url, headers=headers)
    if cik_resp.status_code != 200:
        return f"SEC.gov rejected request: {cik_resp.status_code}"

    try:
        cik_data = cik_resp.json()
        cik_lookup = {item['ticker']: item['cik_str'] for item in cik_data.values()
        cik = cik_lookup.get(ticker.upper())
    except Exception as e:
        return f"Error parsing CIK data: {e}"

    if not cik:
        return f"CIK for {ticker} not found."
    filings_url = f"https://data.sec.gov/submissions/CIK{cik:0>10}.json"
    filings_resp = requests.get(filings_url, headers=headers)
    try:
        data = filings_resp.json() if filings_resp.status_code == 200 else {}
        filings = data.get('filings', {}).get('recent', {})
        if not filings: return f"No filings found for {ticker}."
        forms = filings.get('form', [])[:3]
        filing_dates = filings.get('filingDate', [])[:3]
        primary_docs = filings.get('primaryDocument', [])[:3]
        accessions = filings.get('accessionNumber', [])[:3]

        result = []
        for form, date, doc, acc in zip(forms, filing_dates, primary_docs, accessio
            # Build the SEC document URL:
            doc_url = f"https://www.sec.gov/Archives/edgar/data/{int(cik)}/{acc.rep
            summary = summarize_sec_document(doc_url)
            result.append(f"{form} on {date}:\n{doc_url}\nSummary: {summary}\n")
        return "Latest filings with summaries:\n" + "\n".join(result)
    except Exception as e:
        return f"Error loading filings for {ticker}: {e}"

sec_api_tool = StructuredTool.from_function(
    name="SECEDGARAPI",
    func=get_sec_filings,
    description="Retrieves and summarizes key findings from recent SEC filings by t
)
```

In [25]:
```
#For debugging & Validation
print(sec_api_tool.invoke("AAPL"))
```

```
Latest filings with summaries:
4 on 2025-10-17:
https://www.sec.gov/Archives/edgar/data/320193/000205091225000008/xslF345X05/wk-form
4_1760740266.xml
Summary: SEC FORM
            4 SEC Form 4 FORM 4 UNITED STATES SECURITIES AND EXCHANGE COMMISSION Was
hington, D.C. 20549 STATEMENT OF CHANGES IN BENEFICIAL OWNERSHIP Filed pursuant to S
ection 16(a) of the Securities Exchange Act of 1934 or Section 30(h) of the Investme
nt Company Act of 1940 OMB APPROVAL OMB Number: 3235-0287 Estimated average burden h
ours per response: 0.5 Â Â Check this box if no longer

4 on 2025-10-17:
https://www.sec.gov/Archives/edgar/data/320193/000163198225000009/xslF345X05/wk-form
4_1760740212.xml
Summary: SEC FORM
            4 SEC Form 4 FORM 4 UNITED STATES SECURITIES AND EXCHANGE COMMISSION Was
hington, D.C. 20549 STATEMENT OF CHANGES IN BENEFICIAL OWNERSHIP Filed pursuant to S
ection 16(a) of the Securities Exchange Act of 1934 or Section 30(h) of the Investme
nt Company Act of 1940 OMB APPROVAL OMB Number: 3235-0287 Estimated average burden h
ours per response: 0.5 Â Â Check this box if no longer

144 on 2025-10-16:
https://www.sec.gov/Archives/edgar/data/320193/000195004725008030/xsl144X01/primary_
doc.xml
Summary: Form 144 Filer Information UNITED STATES SECURITIES AND EXCHANGE COMMISSION
Washington, D.C. 20549 Form 144 NOTICE OF PROPOSED SALE OF SECURITIES PURSUANT TO RU
LE 144 UNDER THE SECURITIES ACT OF 1933 FORM 144 144: Filer Information Filer CIK 00
02050912 Filer CCC XXXXXXXX Is this a LIVE or TEST Filing? LIVE TEST Submission Cont
act Information Name Phone E-Mail Address 144: Issuer Information Name o
```

In [26]:

```python
#A tool to visit webpage to collect information
#This was built for the SEC, but they didn't like this web crawler, and did not per

STOCK_SITES = {
    "tipranks": "https://www.tipranks.com/stocks/{ticker}/stock-analysis",
    "marketbeat": "https://www.marketbeat.com/stocks/{ticker}/",
    "zacks": "https://www.zacks.com/stock/quote/{ticker}",
    "simplywallst": "https://simplywall.st/stocks/us/{ticker}"
}

def visit_webpages_for_symbol(ticker: str) -> dict:
    """
    Fetches and reads webpage information for a given stock ticker from all defined
    Returns a dictionary with site names as keys and page content as values.
    """
    results = {}
    for site, url_pattern in STOCK_SITES.items():
        try:
            url = url_pattern.format(ticker=ticker)
            loader = WebBaseLoader(web_path=url)
            documents = loader.load()
            results[site] = documents[0].page_content[:1000]
        except Exception as e:
            results[site] = f"Error fetching {url}: {e}"
    return results
```

```
# Define the tool for agent use
visit_webpages_tool = StructuredTool.from_function(
    func=visit_webpages_for_symbol,
    name="VisitWebpagesForSymbol",
    description="Fetches and reads webpage information for a given stock ticker fro
)
```

In [27]:
```
#Print for debug
result = visit_webpages_tool.invoke({"ticker": "AAPL"})
pretty_str = pformat(result, width=100)
lines = pretty_str.split('\n')
for line in lines[:25]:
    print(line)
```

```
{'marketbeat': '\n'
               '\n'
               '\n'
               '\n'
               '\n'
               '\n'
               '\r\n'
               '\tStock Lists | Top Stocks by Interest, Sector, Exchange and More\r
\n'
               '\n'
               '\n'
               '\n'
               '\n'
               '\n'
               '\n'
               '\n'
               '\n'
               '\n'
               '\n'
               '\n'
               '\n'
               '\n'
               '\n'
               '\n'
               '\n'
               '\n'
```

In [29]:
```
#Finanical news lookup
def get_fin_news(symbol: str) -> str:
    api_key = fin_news
    url = "https://newsapi.org/v2/everything"
    params = {
        "q": symbol,
        "apiKey": api_key,
        "sortBy": "publishedAt",
        "language": "en"
    }
    response = requests.get(url, params=params)
    if response.status_code != 200:
        return f"API error {response.status_code}: {response.text[:200]}"
    data = response.json() #Return data in JSON
    articles = data.get("articles", [])
    if not articles:
        return f"No news found for {symbol}. Full message: {data.get('message', '')
    return "\n".join([a["description"] or a["title"] for a in articles[:7]])


#Define tool
news_api_tool = StructuredTool.from_function(
    name="NewsAPI",
    func=get_fin_news,  #Assumes you've defined this class
    description="Finds recent financial news on stock symbol"
)
```

In [31]:

```
#For debugging
print(get_fin_news("AAPL"))
```

```
Key Points






Apple’s iPhone 17, Air, and AI tailwinds might power a breakout in the stock.




The strength in iPhone 17 sales is a major boon for the stock, but let’s not forget
about iPhone Air’s potential, the coming foldable iPhone, and the …
TradingView historical and live data downloader with advanced features
The absence of profitable AI services doesn't undermine the massive buildout; it jus
tifies it.
People counting how many devices Apple is likely to sell this year are likely focuse
d on the latest iPhones. But the company has plenty of other new devices ...
Investors are increasingly pouring money into non-productive gold, raising alarm for
the global economy while BTC lags behind.
Professional quantitative trading research platform with ML-powered backtesting, mul
ti-source options analysis, portfolio management, and interactive Plotly visualizati
ons
Investors weighed corporate earnings against the escalating US-China trade war.
```

In [82]:

```
#Web search tool with Tavily
def get_fin_news_tavily(symbol: str) -> str:
    """
    Searches for and extracts news impacting the given stock symbol.
    Performs a deep extraction with Tavily's advanced read (HTML parsing and markdo
    """
    query = f"News on {symbol}"
    api_url_search = "https://api.tavily.com/search"
    api_url_extract = "https://api.tavily.com/extract" #Added to try and get better

    #Search for related news articles
    payload_search = {
        "query": query,
        "api_key": tavily_key,
        "max_results": 15, #Increased up from 5 for final runs
        "search_depth": "advanced",#Search for more contextually relevant pages
        "include_raw_content": False
    }

    search_response = requests.post(api_url_search, json=payload_search)
    if search_response.status_code != 200:
        return f"Tavily search error: {search_response.status_code} {search_respons

    results = search_response.json().get("results", [])
    #print(results)
    links = [r.get("url") for r in results if r.get("url")]

    #Extract full article contents using Tavily Extract API
    if not links:
        return "No articles found for this stock symbol."

    payload_extract = {
        "urls": links,
        "api_key": tavily_key,
        "extract_depth": "advanced", #Deeper DOM parsing, better for long articles
        "include_images": False,
        "format": "markdown" #Returns Markdown-formatted summaries
    }

    extract_response = requests.post(api_url_extract, json=payload_extract)
    if extract_response.status_code != 200:
        return f"Tavily extract error: {extract_response.status_code} {extract_resp

    extracted_articles = extract_response.json().get("results", [])
    if not extracted_articles:
        return "No extractable content retrieved."

    #Structure extracted results for downstream summarization or agent reflection
    summaries = [] #Instantiate blank list
    #print(extracted_articles)
    for article in extracted_articles: #Add to list
        link = article.get("url", "")
        content = article.get("raw_content", "No text content")
        summaries#.append({'link':link,
                          'content': content
                         })
    #return summaries

#Define Tool
tavily_news_tool = StructuredTool.from_function(
    name="TavilyNews",
    func=get_fin_news_tavily,
    description="Searches for and extracts detailed financial news impacting the gi
)
```

In [86]:
```
#get_fin_news_tavily("AAPL")
```

In [87]:
```
#For debugging
#print(get_fin_news_tavily(symbol="AAPL"))
```

In [88]:
```
#from langchain_core.tools import tool
#Create the pipeline once, outside the function for efficiency
sentiment_classifier = pipeline(
    "sentiment-analysis",
    model="distilbert-base-uncased-finetuned-sst-2-english"
)

@tool
def analyze_sentiment(text: str) -> int:
    """Analyzes sentiment using DistilBERT."""
    result = sentiment_classifier(text)[0]
    return 1 if result["label"].upper() == "POSITIVE" else -1
```

```
Device set to use cpu
```

In [89]:
```
# Example usage:
print(analyze_sentiment.invoke({"text": "I love this product!"}))
```

```
1
```

In [90]:
```
def earnings_analyzer(symbol: str, context: str = "", yahoo_tool=None, sec_tool=Non
    # Use dummy data if no tool is provided
    yahoo_data = yahoo_tool if yahoo_tool else yahoo_dummy
    sec_data = sec_tool if sec_tool else sec_dummy

    # Example analysis logic
    analysis = (
        f"Earnings Analysis for {symbol}:\n"
        f"Yahoo Finance: {yahoo_data['summary']}\n"
        f"SEC Filing: {sec_data['filing_text']}\n"
    )
    return analysis

#Define tool
earnings_specialist = StructuredTool.from_function(
    name="EarningsAnalyzer",
    func=earnings_analyzer,
    description="Analyzes earnings context and info for a stock symbol, using memor
)
```

In [91]:
```
#For debugging

yahoo_dummy = {
    "symbol": "AAPL",
    "earnings": {
        "quarter": "Q3 2025",
        "eps_actual": 2.15,
        "eps_estimate": 2.10,
        "revenue_actual": 82000000000,
        "revenue_estimate": 81500000000,
        "surprise": 0.05,
        "date": "2025-07-28"
    },
    "summary": "Apple reported better-than-expected earnings for Q3 2025, beating a
}
```

```
sec_dummy = {
    "symbol": "AAPL",
    "filing_type": "10-Q",
    "filing_date": "2025-07-28",
    "filing_text": (
        "Apple Inc. reported net sales of $82 billion for the quarter ended June 20
        "with net income of $19 billion. The company highlighted strong performance
    )
}

print(earnings_analyzer("META"))
```

```
Earnings Analysis for META:
Yahoo Finance: Apple reported better-than-expected earnings for Q3 2025, beating ana
lyst estimates on both EPS and revenue.
SEC Filing: Apple Inc. reported net sales of $82 billion for the quarter ended June
2025, with net income of $19 billion. The company highlighted strong performance in
its services segment.
```

In [92]:
```
#Add Duck Duck Go
from langchain_community.tools import DuckDuckGoSearchRun
search = DuckDuckGoSearchRun()

duck_duck_go_search = StructuredTool.from_function(
    name="DuckDuckGoSearch",
    func=search.invoke,
    description="Useful for when you need to answer questions about current events.
)
```

In [93]:
```
duck_duck_go_search.invoke('GOOG')
```

Out[93]:
```
"2 days ago - Google LLC (/ˈɡuː.ɡəl/ ⓘ, GOO-gəl) is an American multinational tec
hnology corporation focused on information technology, online advertising, search
engine technology, email, cloud computing, software, quantum computing, e-commerc
e, consumer electronics, and artificial intelligence (AI). January 8, 2025 - Track
GOOG Stock with real-time price updates, overview, analysis, insider insights, and
Smart Score ratings. Get Alphabet Class C news, earnings, and stock analysis — all
in one place at TipRanks. September 2, 2025 - U.S. District Judge Amit Mehta ruled
against the most severe consequences that were proposed by the Department of Justi
ce, including the forced sale of Google's Chrome browser, which provides data that
helps its advertising business deliver targeted ads. 📉 Why I'm NOT Buying Google
Stock ( GOOG ) in 2025 | AI Threat to Alphabet's Profits? 🤖 Everyone is bullish o
n Alphabet ( GOOG ) right now. It's trading at a h... February 20, 2025 - GOOG sto
ck's bullish probabilities suggest a compelling long opportunity, despite fears of
generative AI disrupting Alphabet's dominance."
```

In [46]:
```
#Define the plotting function
def plot_stock_52_week(symbol: str) -> str:
    """
    Plots the last 52 weeks (1 year) of stock closing prices for the given symbol.
    Returns the filepath to the saved chart image.
    """
    try:
        end_date = datetime.today()
        start_date = end_date - timedelta(weeks=52)
        # Download 1-year historical stock data
        data = yf.download(symbol, start=start_date, end=end_date)

        if data.empty:
            return f"No data found for {symbol}. Please check the symbol or try aga

        # Plot closing price over time
        plt.figure(figsize=(10, 5))
        plt.plot(data.index, data['Close'], label='Closing Price', color='dodgerblu
        plt.title(f"{symbol} Stock Price (Last 52 Weeks)", fontsize=14, pad=15)
        plt.xlabel("Date", fontsize=12)
        plt.ylabel("Price (USD)", fontsize=12)
        plt.grid(True, linestyle='--', alpha=0.6)
        plt.legend()

        # Save plot to file
        filename = f"{symbol}_52week_chart.png"
        plt.savefig(filename, bbox_inches='tight')
        plt.close()

        return f"Stock chart saved as {filename}"
    except Exception as e:
        return f"Error generating chart for {symbol}: {e}"

#Wrap as a LangGraph Tool
stock_chart_tool = StructuredTool.from_function(
    name="StockChart52Weeks",
    func=plot_stock_52_week,
    description="Plots and saves a 52-week stock price chart for the given symbol u
)
```

In [50]:
```
from datetime import datetime, timezone
#from langchain.tools import Tool

def get_timestamp(_: str = "") -> str:
    """
    Returns the current date and time in ISO 8601 format.

    This tool can be used by the agent to log events, timestamp analyses,
    or align data with real-world time context. It supports chaining with
    other workflow nodes that require temporal awareness.
    """
    return datetime.now(timezone.utc).isoformat()

timestamp_tool = StructuredTool.from_function(
    name="TimestampTool",
    func=get_timestamp,
    description="Returns the current date and time in ISO 8601 format, useful for l
)
```

In [51]:
```
print(get_timestamp())
```

```
2025-10-18T17:48:28.722735+00:00
```

In [98]:
```
#Create the tools list
tools = [
    yahoo_api_tool,
    sec_api_tool,
    news_api_tool,
    #tavily_news_tool,
    earnings_specialist,
    duck_duck_go_search,
    stock_chart_tool,
    analyze_sentiment,  # Use the analyze_sentiment function here
    timestamp_tool,
    visit_webpages_tool
]
```

Define Tools Agent

Note: This agent leverages the LangGraph ReAct agent framework. The basis of this agent is an LLM that is prompted to reason about what it should do, take an action, and then observe the results to evaluate if it should act again, or be done. In addition to that basic framework, the prompt contains critical functions and workflows for the agent to perform.

This prompt engineering is a balance to get a thorough enough agent to select multiple tools, while not creating an agent that is either too lazy, quitting with too few API calls, or too aggresive.

In [99]:
```
#Tools Agent Prompt & Function

ANALYSIS_PROMPT = """
System Role: You are a detailed Financial Research Agent designed to analyze compan

**Your goal** For {symbol}, provide a ~50 word investment summary, based on extensi

Include {context_memory} to improve your answers.

NEVER INCLUDE YOUR THOUGHT PROCESS IN THE RESULTS.


**Core Functions**

1. Planning:
Develop and print out a plan for each given stock symbol. Include a plan for a mult

Example steps: “Collect company overview → Analyze financials → Cross-check recent

2. Tool Use:
Use all of your tools by default to hit APIs and data sources dynamically, such as

TOOL USAGE REQUIREMENTS:
- Refer to the memory for useful data.
- ALWAYS include a timestamp of the run near the top of the report.
- Use at least 4 different tools for each analysis
- Cross-reference information from multiple sources
- If a tool fails, try alternative tools for the same data
- Always explain your reasoning between tool calls

3. Self-Reflection:
Evaluate your output’s completeness, correctness, and coherence at each stage. If k

4. Memory/Learning:
Retain useful context from prior analyses (observations, missing data, errors, patt


**Workflow Logic**

Follow a defined sequence for every research request:
* Ingest → Preprocess → Classify → Extract → Summarize

1. Ingest: Gather relevant news, filings, and metrics. Use the webpage visit tool o

2. Preprocess: Clean, standardize, and interpret results from multiple tools.

3. Classify: Route data to the correct analysis path (earnings, news, or market).

4. Extract: Pull core signals, events, or values.

5. Summarize: Synthesize insights in your final output.

6. Routing:
  * Choose appropriate specialist analyzers according to data type:

  * NewsAnalyzer for press, sentiment, or event narratives.

  * MarketAnalyzer for indices, performance data, and macro context.

7. Evaluation: Review reasoning and completeness, recall tools again if needed.


**Final Report**
At the end of each session, create a final summary report that contains:

Updated output from EVERY tool as well as the TIMESTAMP near the top.

NEVER include your internal planning information in the report.

Include citations such as [1] followed by a hyperlink at the bottom of the page for

Respond in clear, organized Markdown, including headings, bullet points, and labele

**STOPPING RULE:** Once you have basic financials, recent news, and market context,

Be thorough - this analysis will inform major investment decisions.

"""

#Define the agent
tools_agent = create_react_agent(
    model=llm_openai, #Can use gemini here
    tools=tools,
    prompt=ANALYSIS_PROMPT, #Feed in the prompt outlined above
    debug=False
)
```

```
C:\Users\speco\AppData\Local\Temp\ipykernel_5656\2913891809.py:81: LangGraphDeprecat
edSinceV10: create_react_agent has been moved to `langchain.agents`. Please update y
our import to `from langchain.agents import create_agent`. Deprecated in LangGraph V
1.0 to be removed in V2.0.
  tools_agent = create_react_agent(
```

In [100…

```
#User Input
user_input = "COST" #Search for information on this company ticker

# Create your formatted prompt
formatted_prompt = ANALYSIS_PROMPT.format(
    symbol=user_input, # ex. "TGT"
    context_memory=query_memory(user_input, SESSION_MEMORY)
)
```

```
query memory stock symbol =  COST
```

In [101…

```
#Build & run agent call to ensure this step works
tools_output = tools_agent.invoke({
    "messages": [
        {"role": "user", "content": formatted_prompt}
    ]
})
```

In [102…

```
#Function to make print out look nicer
def clean_answer (text):
    msg = text["messages"][-1]
    return msg.content
```

In [103…

```
display(Markdown(clean_answer(tools_output)))
```

Investment Summary for Costco Wholesale Corporation (COST)

- **Company Overview:**
  - **Sector:** Consumer Defensive
  - **Price:** $936.33
  - **Market Cap:** $414.96 billion
  - **P/E Ratio:** 51.56
  - **Dividend Yield:** 0.56%
  - **Revenue:** $275.24 billion
  - **Profit Margin:** 2.94%
- **Recent Filings:**
  - **8-K on 2025-10-15:** Link to Filing
  - **3 on 2025-10-09:** Link to Filing
  - **10-K on 2025-10-08:** Link to Filing
- **Latest News:**
  - Consumer trends shifting towards western and fusion wear impact ethnic fashion sales.
  - India investing in AI and VR for economic growth and skill development.
  - Argentina's austerity program lowers inflation but faces challenges before elections.
  - Phillies considering trading or releasing Nick Castellanos.
- **Market Analysis:**
  - **TipRanks:** Link
  - **MarketBeat:** Link
  - **Zacks:** Request unsuccessful.
  - **Simply Wall St:** Page not found.

For further details, refer to the provided links for filings and market analysis tools.

[1] Timestamp: 2025-10-18 18:20:18 UTC

Self Reflection & Evaluation Agent

This agent evaluates the tools agent output, and allows for human feedback to then further refine and optimize the report.

In [104…

```
#Define Self Evaluation agent

EVAL_PROMPT = """
You are an expert evaluator. Your primary job is to give feedback on the analysis b
```

```
Instructions:

- Always display the full analysis/summary input *exactly as received* at the start
- Provide your commentary (improvement, completeness, feedback) **separately after
- If human feedback is supplied, include your response to it at the end **without c

FORMAT STRICTLY LIKE THIS:
---
Original Analysis:
{input}

Evaluator Commentary:
[Your bullet points: Completeness, Succinctness, Accuracy, Clarity, Human Feedback

---

Never rewrite or summarize the original analysis. Only provide clear, constructive

--- Human Feedback ---
{human_feedback}


"""


#Define agent
evaluator_agent = create_react_agent(
    model=llm_openai,
    tools = [], #No tools for this agent
    prompt=EVAL_PROMPT,
    debug=False
)
```

```
C:\Users\speco\AppData\Local\Temp\ipykernel_5656\2297441607.py:31: LangGraphDeprecat
edSinceV10: create_react_agent has been moved to `langchain.agents`. Please update y
our import to `from langchain.agents import create_agent`. Deprecated in LangGraph V
1.0 to be removed in V2.0.
  evaluator_agent = create_react_agent(
```

In [105…

```
#This function is to provide a clean input to the evaluator agent

def get_eval_input(final_output, human_feedback, memory_context):
    return f"""
            {final_output}

            --- Human Feedback ---
            {human_feedback if human_feedback else "no human feedback provided"}

            --- Memory ---
            {memory_context}
                """
```

Optimization Loop

Below is the primary function used to run the agent. The workflow goes as follows:

Passes the prompt to the to the tools agent. Displays the tool agent's response to the user. The user inputs any feedback they have to the response. The session memory is queried if there are any relevant past questions about the specific stock The tool's response, the human feedback, and the memory (if there is a relavent one) are all passed to the evaluator agent. The evaluator agent critiques and improves upon the tool's response. The user is prompted if they would like to refine the response further. "y" leads to another iteration of the evaluator agent on its latest response; "n" ends the loop. Lastly, the user question, stock symbol, and final response are appended to the session memory object for future reference.

In [106…

```python
#This function runs the agents in an optimization kind of loop, to iterate based on
def optimization_loop(tools_agent,
                      evaluator_agent,
                      formatted_prompt,
                      session_memory,
                      PRINT_TOOL_MSGS=True
                     ):
    #Run tools agent
    print("Conducting research...")
    tools_output = tools_agent.invoke({
    "messages": [
        {"role": "user", "content": formatted_prompt}
        ]
      })

    #Clean print out the tool messages
    if PRINT_TOOL_MSGS:
        for msg in tools_output['messages']:
            if msg.content:
                print(f'{msg.content} |  tool {msg.name}\n')

    final_output = list(tools_output.values())[-1][-1].content
    print("\n--- Analysis Summary ---")
    print(final_output)

    #Ask user for feedback
    human_feedback = input("\nPlease enter your feedback (areas to improve, missing

    #Grab memory if there is one.
    stock_symbol = user_input
    memory_context = query_memory(stock_symbol, session_memory)



    evaluator_input = get_eval_input(final_output, human_feedback, memory_context)
    #Evaluate and revise summary using feedback
    eval_payload = {
        "messages": str(evaluator_input),
    }
    print("\nRunning evaluator with feedback...")
    revised_output = evaluator_agent.invoke(eval_payload)
    final_output = revised_output['messages'][-1].content
    print("\n--- Revised Summary ---")
    print(final_output)
    #Optional: Loop for more feedback
    while True:
        more = input("\nWould you like to refine further? (y/n): ")
        if more.lower().startswith("y"):
            human_feedback = input("Enter any further feedback:\n")
            evaluator_input = get_eval_input(final_output, human_feedback, None)
            eval_payload = {
                            "messages": str(evaluator_input),
                        }

            revised_output = evaluator_agent.invoke(eval_payload)
            final_output = revised_output['messages'][-1].content
            print("\n--- Refined Revised Summary ---")
            print(final_output)
        else:
            break

    user_question = as_text(user_input)
    symbol = extract_symbol(user_question)
    final_answer = as_text(final_output)   # Use your utility function

    SESSION_MEMORY.remember(symbol, user_question, final_answer)

    print("\nWorkflow complete. Final output above.")
    return clean_answer(revised_output)  #Return for pretty print
```

Execution Function, Set The Agent Free

The actual running of the agents in sequence with human feedback

In [107…

```
#Here again for easy redefining
#User Input
user_input = "NVDA" #Search for information on this company ticker

# Create your formatted prompt
formatted_prompt = ANALYSIS_PROMPT.format(
    symbol=user_input,
    context_memory=query_memory(user_input, SESSION_MEMORY)
)
```

```
query memory stock symbol =  NVDA
```

In [109…

```
#Execution function
final_analysis = optimization_loop(tools_agent,
                                   evaluator_agent,
                                   formatted_prompt,
                                   SESSION_MEMORY
                                  )
```

```
Conducting research...

System Role: You are a detailed Financial Research Agent designed to analyze compani
es, securities, and markets with a structured workflow.

**Your goal** For NVDA, provide a ~50 word investment summary, based on extensive re
serach.

Include Original Analysis:
# Investment Summary for NVDA

NVIDIA Corporation (NVDA) is a technology company with a market cap of $4.46 trillio
n, a P/E ratio of 52.05, and a profit margin of 52.41%. Recent filings indicate acti
ve securities activities [1]. The stock has shown resilience in the AI sector, with
positive analyst ratings and significant chip shipments approved by the US [1].

- **Financials**:
  - Price: $183.22
  - 52-Week High: $195.62
  - 52-Week Low: $86.62
  - Dividend Yield: 0.02%
  - Revenue: $165.22 billion

- **Recent News**:
  - NVIDIA remains a top AI stock with strong fundamentals [1].
  - Micron's decision impacts server chip supply in China [1].
  - Analysts bullish on chipmaker stocks like NVDA, TSM, and AMD [1].

This information suggests NVDA's strong position in the market, backed by positive s
entiment and financial performance.

[1] [SEC Filings and News Sources](#)

--- Human Feedback ---
no feedback

--- Memory ---
None

Evaluator Commentary:
- Completeness: The analysis provides a comprehensive overview of NVIDIA Corporation
(NVDA), covering key financial metrics, recent news, and market performance.
- Accuracy: The financial details and recent news are accurately presented, giving a
clear picture of NVDA's standing in the market.
- Clarity: The information is well-structured and easy to follow, making it simple f
or investors to understand NVDA's current position and potential investment opportun
ities.
- Citations: Proper citation of sources is included, enhancing the credibility of th
e analysis.

Overall, this investment summary effectively conveys the strengths and positive outl
ook for NVDA as an investment option. to improve your answers.

NEVER INCLUDE YOUR THOUGHT PROCESS IN THE RESULTS.
```

```
**Core Functions**

1. Planning:
Develop and print out a plan for each given stock symbol. Include a plan for a multi
-step research process that defines what data to collect (e.g., news, market data, f
ilings), what tools to use, and how to verify results.

Example steps: "Collect company overview → Analyze financials → Cross-check recent f
ilings → Summarize."

2. Tool Use:
Use all of your tools by default to hit APIs and data sources dynamically, such as Y
ahoo Finance, SEC EDGAR, news APIs, and any specialist analyzers. Choose the right t
ools and calls autonomously based on the goal of each stage. Return structured findi
ngs.

TOOL USAGE REQUIREMENTS:
- Refer to the memory for useful data.
- ALWAYS include a timestamp of the run near the top of the report.
- Use at least 4 different tools for each analysis
- Cross-reference information from multiple sources
- If a tool fails, try alternative tools for the same data
- Always explain your reasoning between tool calls

3. Self-Reflection:
Evaluate your output's completeness, correctness, and coherence at each stage. If ke
y data (e.g., P/E, recent filings, or market indicators) appears missing or uncertai
n, perform iterative refinement using another reasoning pass or additional tool call
s.

4. Memory/Learning:
Retain useful context from prior analyses (observations, missing data, errors, patte
rns in company performance). Use this "brief memory" to improve future research qual
ity and efficiency.


**Workflow Logic**

Follow a defined sequence for every research request:
* Ingest → Preprocess → Classify → Extract → Summarize

1. Ingest: Gather relevant news, filings, and metrics. Use the webpage visit tool on
any URLS.

2. Preprocess: Clean, standardize, and interpret results from multiple tools.

3. Classify: Route data to the correct analysis path (earnings, news, or market).

4. Extract: Pull core signals, events, or values.

5. Summarize: Synthesize insights in your final output.

6. Routing:
  * Choose appropriate specialist analyzers according to data type:

  * NewsAnalyzer for press, sentiment, or event narratives.
  * MarketAnalyzer for indices, performance data, and macro context.

7. Evaluation: Review reasoning and completeness, recall tools again if needed.


**Final Report**
At the end of each session, create a final summary report that contains:

Updated output from EVERY tool as well as the TIMESTAMP near the top.

NEVER include your internal planning information in the report.

Include citations such as [1] followed by a hyperlink at the bottom of the page for
further review.

Respond in clear, organized Markdown, including headings, bullet points, and labeled
data sources for readability.

**STOPPING RULE:** Once you have basic financials, recent news, and market context,
conclude your analysis. Do not seek additional tools or data.

Be thorough - this analysis will inform major investment decisions.

 |  tool None

2025-10-18T18:34:57.166752+00:00 |  tool TimestampTool

Company: NVIDIA Corporation (NVDA)
Sector: Technology
Price: $183.22
Market Cap: 4460857262080
P/E Ratio: 52.051136
52-Week High: 195.62
52-Week Low: 86.62
Dividend Yield: 0.02
Beta: 2.123
Revenue: 165217992704
Profit Margin: 0.52414
 |  tool YahooFinanceAPI

Latest filings with summaries:
144 on 2025-10-17:
https://www.sec.gov/Archives/edgar/data/1045810/000192109425001272/xsl144X01/primary
_doc.xml
Summary: Form 144 Filer Information UNITED STATES SECURITIES AND EXCHANGE COMMISSION
Washington, D.C. 20549 Form 144 NOTICE OF PROPOSED SALE OF SECURITIES PURSUANT TO RU
LE 144 UNDER THE SECURITIES ACT OF 1933 FORM 144 144: Filer Information Filer CIK 00
01197649 Filer CCC XXXXXXXX Is this a LIVE or TEST Filing? LIVE TEST Submission Cont
act Information Name Phone E-Mail Address 144: Issuer Information Name o

144 on 2025-10-16:
https://www.sec.gov/Archives/edgar/data/1045810/000192109425001268/xsl144X01/primary
_doc.xml
Summary: Form 144 Filer Information UNITED STATES SECURITIES AND EXCHANGE COMMISSION
Washington, D.C. 20549 Form 144 NOTICE OF PROPOSED SALE OF SECURITIES PURSUANT TO RU
```

```
LE 144 UNDER THE SECURITIES ACT OF 1933 FORM 144 144: Filer Information Filer CIK 00
01197649 Filer CCC XXXXXXXX Is this a LIVE or TEST Filing? LIVE TEST Submission Cont
act Information Name Phone E-Mail Address 144: Issuer Information Name o

4 on 2025-10-15:
https://www.sec.gov/Archives/edgar/data/1045810/000119764925000048/xslF345X05/wk-for
m4_1760564872.xml
Summary: SEC FORM
            4 SEC Form 4 FORM 4 UNITED STATES SECURITIES AND EXCHANGE COMMISSION Was
hington, D.C. 20549 STATEMENT OF CHANGES IN BENEFICIAL OWNERSHIP Filed pursuant to S
ection 16(a) of the Securities Exchange Act of 1934 or Section 30(h) of the Investme
nt Company Act of 1940 OMB APPROVAL OMB Number: 3235-0287 Estimated average burden h
ours per response: 0.5 Â Â Check this box if no longer
 |  tool SECEDGARAPI

AVGO is red hot after Broadcom announced an OpenAI deal. However, shares trade at a
forward P/E near 65x.
Fears spread about regional banks' exposure to bad loans and credit quality, spurrin
g an exodus by investors to safe havens.
Investor fears over regional banks' exposure to bad loans and credit quality eased w
hile talks between the US and China appeared to be back on track.
Many investors find it hard to think past the likes of Nvidia (NVDA) or Palantir Tec
hnologies (PLTR) when it comes to the artificial intelligence space. But ASML (ASML)
is an alternative AI stock that is worth considering as it eyes a fresh entry amid a
stron…
Micron shares lost ground Friday following a report that the company will stop provi
ding server chips to data centers in China. Monitor these key chart...
Key Points







More arguments are being made that stocks are in an AI-driven bubble.





Fundamentals counter that argument, with analysts continuing to be bullish on chipma
ker stocks.






NVDA, TSM and AMD continue to hold “Buy” ratings with …
NVIDIA Corporation (NASDAQ:NVDA) is one of the Hottest Mega-Cap Stocks of 2025. On O
ctober 9, Reuters reported that NVIDIA Corporation (NASDAQ:NVDA) had received approv
al from the United States to ship several billion worth of its chips to the United A
rab Emi… |  tool NewsAPI

1 |  tool analyze_sentiment

# Investment Summary for NVDA

**NVIDIA Corporation (NVDA)** is a leading technology company with a market cap of
```

```
$4.46 trillion, a P/E ratio of 52.05, and a profit margin of 52.41%. Recent filings
show active securities activities. NVDA has demonstrated resilience in the AI secto
r, with positive analyst ratings and significant chip shipments approved by the US.

- **Financials**:
  - Price: $183.22
  - 52-Week High: $195.62
  - 52-Week Low: $86.62
  - Dividend Yield: 0.02%
  - Revenue: $165.22 billion

- **Recent News**:
  - NVDA remains a top AI stock with strong fundamentals.
  - Micron's decision impacts server chip supply in China.
  - Analysts bullish on chipmaker stocks like NVDA, TSM, and AMD.

**Sentiment Analysis**: The sentiment around NVDA is positive, reflecting its strong
position in the market and investor confidence.

This information suggests NVDA's robust standing in the market, supported by positiv
e sentiment, financial performance, and analyst outlook.

---

**Data Sources**:
- Yahoo Finance for company financials
- SEC EDGAR for recent filings
- NewsAPI for recent news
- Sentiment analysis for sentiment evaluation

[1] [SEC Filings and News Sources](#) |  tool None


--- Analysis Summary ---
# Investment Summary for NVDA
```

```
**NVIDIA Corporation (NVDA)** is a leading technology company with a market cap of
$4.46 trillion, a P/E ratio of 52.05, and a profit margin of 52.41%. Recent filings
show active securities activities. NVDA has demonstrated resilience in the AI secto
r, with positive analyst ratings and significant chip shipments approved by the US.

- **Financials**:
  - Price: $183.22
  - 52-Week High: $195.62
  - 52-Week Low: $86.62
  - Dividend Yield: 0.02%
  - Revenue: $165.22 billion

- **Recent News**:
  - NVDA remains a top AI stock with strong fundamentals.
  - Micron's decision impacts server chip supply in China.
  - Analysts bullish on chipmaker stocks like NVDA, TSM, and AMD.

**Sentiment Analysis**: The sentiment around NVDA is positive, reflecting its strong
position in the market and investor confidence.
```

```
This information suggests NVDA's robust standing in the market, supported by positiv
e sentiment, financial performance, and analyst outlook.

---

**Data Sources**:
- Yahoo Finance for company financials
- SEC EDGAR for recent filings
- NewsAPI for recent news
- Sentiment analysis for sentiment evaluation

[1] [SEC Filings and News Sources](#)
```

```
query memory stock symbol =  NVDA

Running evaluator with feedback...

--- Revised Summary ---

Original Analysis:
# Investment Summary for NVDA

NVIDIA Corporation (NVDA) is a leading technology company with a market cap of $4.46
trillion, a P/E ratio of 52.05, and a profit margin of 52.41%. Recent filings show a
ctive securities activities. NVDA has demonstrated resilience in the AI sector, with
positive analyst ratings and significant chip shipments approved by the US.

- **Financials**:
  - Price: $183.22
  - 52-Week High: $195.62
  - 52-Week Low: $86.62
  - Dividend Yield: 0.02%
  - Revenue: $165.22 billion

- **Recent News**:
  - NVDA remains a top AI stock with strong fundamentals.
  - Micron's decision impacts server chip supply in China.
  - Analysts bullish on chipmaker stocks like NVDA, TSM, and AMD.

**Sentiment Analysis**: The sentiment around NVDA is positive, reflecting its strong
position in the market and investor confidence.

This information suggests NVDA's robust standing in the market, supported by positiv
e sentiment, financial performance, and analyst outlook.

---

**Data Sources**:
- Yahoo Finance for company financials
- SEC EDGAR for recent filings
- NewsAPI for recent news
- Sentiment analysis for sentiment evaluation

[1] [SEC Filings and News Sources](#)

            --- Human Feedback ---
            good job

            --- Memory ---
            Original Analysis:
# Investment Summary for NVDA

NVIDIA Corporation (NVDA) is a leading technology company in the AI sector with a ma
rket cap of $4.46 trillion. Key financial metrics include:
- Price: $183.22
- P/E Ratio: 52.05
- Profit Margin: 52.41%
- Revenue: $165.22 billion
```

```
## Recent News
- NVIDIA remains a top AI stock with strong fundamentals.
- Micron's decision impacts server chip supply in China.
- Analysts are bullish on chipmaker stocks like NVDA, TSM, and AMD.

## Market Performance
- NVDA's stock price has shown resilience, with a 52-week high of $195.62 and a low
of $86.62.
- The company received approval for significant chip shipments to the US.

## Sentiment Analysis
- Sentiment analysis indicates positive sentiment towards NVDA and the chipmaker sec
tor.

This data suggests NVDA's strong market position, supported by positive sentiment, f
inancial performance, and analyst ratings.

For further details, refer to [SEC Filings and News Sources](#).

---

**Data Sources:**
- Yahoo Finance
- SEC EDGAR
- News API
- Sentiment Analysis

*Timestamp: 2025-10-18T18:21:00.558338+00:00*

            --- Human Feedback ---
            I think you did great

            --- Memory ---
Original Analysis:
# Investment Summary for NVDA

NVIDIA Corporation (NVDA) is a technology company with a market cap of $4.46 trillio
n, a P/E ratio of 52.05, and a profit margin of 52.41%. Recent filings indicate acti
ve securities activities [1]. The stock has shown resilience in the AI sector, with
positive analyst ratings and significant chip shipments approved by the US [1].

- **Financials**:
  - Price: $183.22
  - 52-Week High: $195.62
  - 52-Week Low: $86.62
  - Dividend Yield: 0.02%
  - Revenue: $165.22 billion

- **Recent News**:
  - NVIDIA remains a top AI stock with strong fundamentals [1].
  - Micron's decision impacts server chip supply in China [1].
  - Analysts bullish on chipmaker stocks like NVDA, TSM, and AMD [1].

This information suggests NVDA's strong position in the market, backed by positive s
entiment and financial performance.
```

```
[1] [SEC Filings and News Sources](#)

--- Human Feedback ---
no feedback

--- Memory ---
None

Evaluator Commentary:
- Completeness: The analysis provides a comprehensive overview of NVIDIA Corporation
(NVDA), covering key financial metrics, recent news, market performance, and sentime
nt analysis, offering a well-rounded view for potential investors.
- Accuracy: The financial details, recent news, and sentiment analysis are accuratel
y presented, contributing to a reliable assessment of NVDA's current standing in the
market.
- Clarity: The structure of the analysis is clear and organized, making it easy for
readers to grasp the key points and make informed decisions regarding NVDA as an inv
estment option.
- Citations: Proper citation of sources is maintained, enhancing the credibility and
transparency of the information provided.

Overall, this investment summary effectively conveys the strengths and positive outl
ook for NVDA, serving as a valuable resource for investors considering this company.
Great job on compiling a thorough and well-presented analysis.

Human Feedback Summary:
The human feedback provided positive reinforcement for the analysis, acknowledging t
he quality of the work done.

Response to Human Feedback:
Thank you for the positive feedback! If you have any specific suggestions for improv
ement or further details you'd like to see in the analysis, feel free to share.
Workflow complete. Final output above.
```

Final Report & Discussion

In [ ]:
```
#Beutify printout of
display(Markdown((final_analysis)))
```

Memory Validation

In [110…
```
#Ensure the memory is working as intended
#After running the user query through your whole pipeline:
user_question = user_input
symbol = extract_symbol(user_input)
final_answer = as_text(final_analysis)   # Use your utility function

SESSION_MEMORY.remember(symbol, user_question, final_answer)
```

In [112…
```
# Later, you can recall the latest answer for "AAPL":
prev = SESSION_MEMORY.recall("NVDA")
```

```
if prev:
    print("Previous answer for stock:", prev)
```

```
Previous answer for stock: Original Analysis:
# Investment Summary for NVDA

NVIDIA Corporation (NVDA) is a leading technology company with a market cap of $4.46
trillion, a P/E ratio of 52.05, and a profit margin of 52.41%. Recent filings show a
ctive securities activities. NVDA has demonstrated resilience in the AI sector, with
positive analyst ratings and significant chip shipments approved by the US.

- **Financials**:
  - Price: $183.22
  - 52-Week High: $195.62
  - 52-Week Low: $86.62
  - Dividend Yield: 0.02%
  - Revenue: $165.22 billion

- **Recent News**:
  - NVDA remains a top AI stock with strong fundamentals.
  - Micron's decision impacts server chip supply in China.
  - Analysts bullish on chipmaker stocks like NVDA, TSM, and AMD.

**Sentiment Analysis**: The sentiment around NVDA is positive, reflecting its strong
position in the market and investor confidence.

This information suggests NVDA's robust standing in the market, supported by positiv
e sentiment, financial performance, and analyst outlook.

---

**Data Sources**:
- Yahoo Finance for company financials
- SEC EDGAR for recent filings
- NewsAPI for recent news
- Sentiment analysis for sentiment evaluation

[1] [SEC Filings and News Sources](#)

            --- Human Feedback ---
            good job

            --- Memory ---
            Original Analysis:
# Investment Summary for NVDA

NVIDIA Corporation (NVDA) is a leading technology company in the AI sector with a ma
rket cap of $4.46 trillion. Key financial metrics include:
- Price: $183.22
- P/E Ratio: 52.05
- Profit Margin: 52.41%
- Revenue: $165.22 billion

## Recent News
- NVIDIA remains a top AI stock with strong fundamentals.
- Micron's decision impacts server chip supply in China.
- Analysts are bullish on chipmaker stocks like NVDA, TSM, and AMD.

## Market Performance
```

```
- NVDA's stock price has shown resilience, with a 52-week high of $195.62 and a low
of $86.62.
- The company received approval for significant chip shipments to the US.

## Sentiment Analysis
- Sentiment analysis indicates positive sentiment towards NVDA and the chipmaker sec
tor.

This data suggests NVDA's strong market position, supported by positive sentiment, f
inancial performance, and analyst ratings.

For further details, refer to [SEC Filings and News Sources](#).

---

**Data Sources:**
- Yahoo Finance
- SEC EDGAR
- News API
- Sentiment Analysis

*Timestamp: 2025-10-18T18:21:00.558338+00:00*

            --- Human Feedback ---
            I think you did great

            --- Memory ---
Original Analysis:
# Investment Summary for NVDA

NVIDIA Corporation (NVDA) is a technology company with a market cap of $4.46 trillio
n, a P/E ratio of 52.05, and a profit margin of 52.41%. Recent filings indicate acti
ve securities activities [1]. The stock has shown resilience in the AI sector, with
positive analyst ratings and significant chip shipments approved by the US [1].

- **Financials**:
  - Price: $183.22
  - 52-Week High: $195.62
  - 52-Week Low: $86.62
  - Dividend Yield: 0.02%
  - Revenue: $165.22 billion

- **Recent News**:
  - NVIDIA remains a top AI stock with strong fundamentals [1].
  - Micron's decision impacts server chip supply in China [1].
  - Analysts bullish on chipmaker stocks like NVDA, TSM, and AMD [1].

This information suggests NVDA's strong position in the market, backed by positive s
entiment and financial performance.

[1] [SEC Filings and News Sources](#)

--- Human Feedback ---
no feedback

--- Memory ---
```

```
None

Evaluator Commentary:
- Completeness: The analysis provides a comprehensive overview of NVIDIA Corporation
(NVDA), covering key financial metrics, recent news, market performance, and sentime
nt analysis, offering a well-rounded view for potential investors.
- Accuracy: The financial details, recent news, and sentiment analysis are accuratel
y presented, contributing to a reliable assessment of NVDA's current standing in the
market.
- Clarity: The structure of the analysis is clear and organized, making it easy for
readers to grasp the key points and make informed decisions regarding NVDA as an inv
estment option.
- Citations: Proper citation of sources is maintained, enhancing the credibility and
transparency of the information provided.

Overall, this investment summary effectively conveys the strengths and positive outl
ook for NVDA, serving as a valuable resource for investors considering this company.
Great job on compiling a thorough and well-presented analysis.

Human Feedback Summary:
The human feedback provided positive reinforcement for the analysis, acknowledging t
he quality of the work done.

Response to Human Feedback:
Thank you for the positive feedback! If you have any specific suggestions for improv
ement or further details you'd like to see in the analysis, feel free to share.
```

Discussion

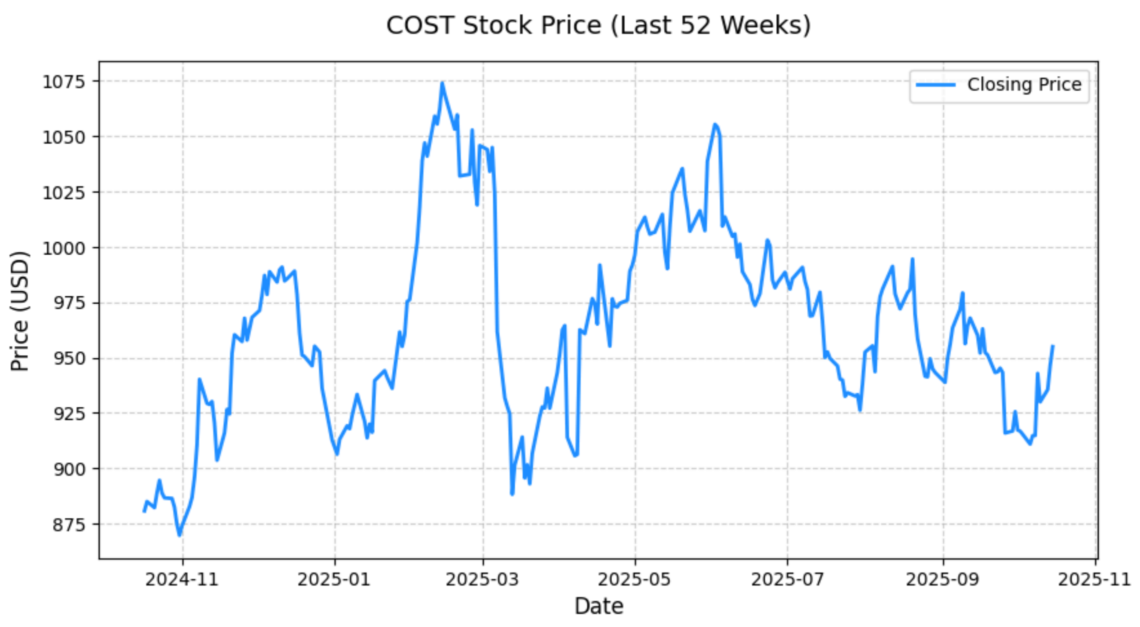
This project outlines an agentic financial research assistant with four key agent functions, and multiple workflows to produce a finanical report.

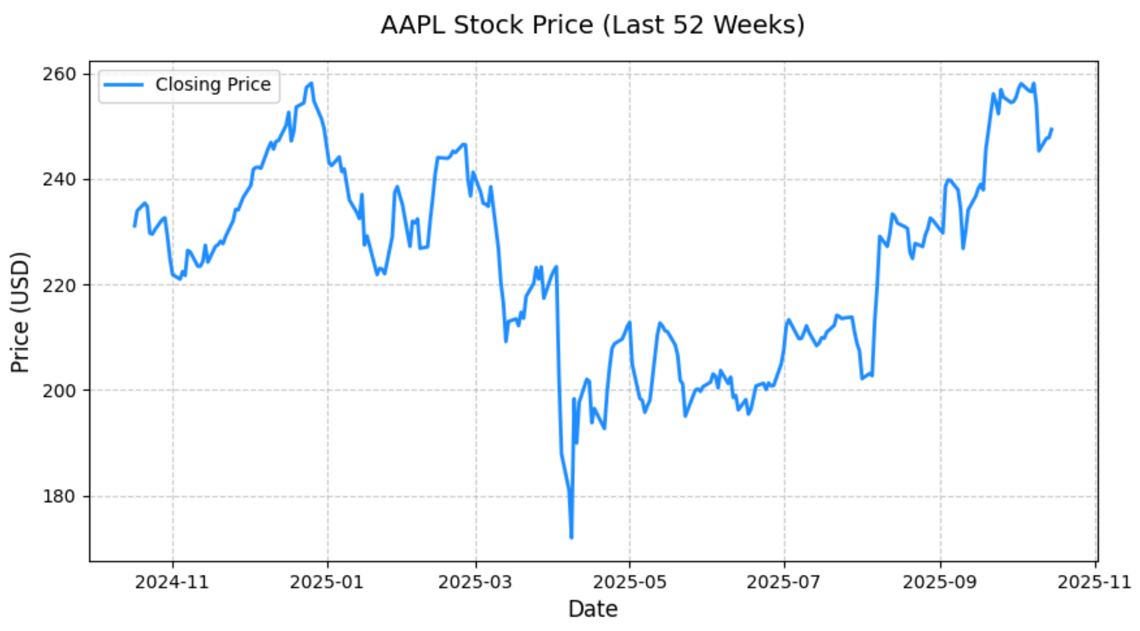
Ideas Future Improvements

- Adding an actual LangGraph graph and state to run the execution would improve consistency across runs.
- Focusing on specific tools for enhancement. Some tools did not produce meaningful output for the analysis, there is room for improvement with the tools that were used, and also additional tools that should be used.
- Adding state for memory to improve context across each run and between runs.
- Adding more specialists, with sub graphs to run within the tools agent. This would have enabled more complex analysis, but was more than we could accomplish in the scope of this course.
- Deploy to production somewhere that others could use it.

Example graphs drawn by tools

Drawn 10/16/25


COST Stock Price (Last 52 Weeks)
Closing Price
Price (USD)
1075
1050
1025
1000
975
950
925
900
875
Date
2024-11
2025-01
2025-03
2025-05
2025-07
2025-09
2025-11


AAPL Stock Price (Last 52 Weeks)
Closing Price
Price (USD)
260
240
220
200
180
Date
2024-11
2025-01
2025-03
2025-05
2025-07
2025-09
2025-11