TODO: Write Python code to design an investment analysis chatbot using the Microsoft's AutoGen Framework.

(include a Google Colab link for submission)

Investment Analysis Chatbot

A specialised stock-analysis agent that evaluates equities through Warren Buffett's value-investing lens. It couples Microsoft Autogen's agent-orchestration with Google Gemini's OpenAI-compatible chat-completion API and a curated Yahoo Finance data tool.

Overview

```
User → Workflow → AssistantAgent ↔ Gemini LLM
                                ↳ get_stock_data_from_yfinance
```

1. **The user** provides a 'ticker' symbol (default ticker="AAPL").
2. **Workflow*** orchestrates logging, model client, tool registry and JSON parsing.*
3. **AssistantAgent** (Autogen) receives a structured system prompt detailing Buffett's ten principles and calls the `get_stock_data_from_yfinance` **FunctionTool**.
4. **Gemini 2 flash-lite** returns a **strict JSON** payload validated by Pydantic schemas.

> Other solutions were tried (~12 versions), but this is the simplest and most compatible with current resources and limitations, such as Multi-Agent, SelectChatGroup, RRChatGroup, RoutedAgent, model_client configuration/version, FunctionToolCall, data, schema design, etc.

Key Classes & Functions

Component	Responsibility
`Logger`	Timestamped console logging; attempts file append when the filesystem allows.
`ApiCredentialsManager`	Manage API credentials i.e. Google Gemini API key
`get_stock_data_from_yfinance`	Wrapper function to fetch data from yfinance API.
`Metrics` , `InvestmentAnalysis`	**Pydantic** models that define and validate the JSON response contract.

Component	Responsibility
`Workflow`	High-level orchestrator: registers the tool, instantiates `AssistantAgent`, sends the user prompt, extracts JSON from fenced blocks and returns a Python dict. Provides both `run()` (async) and `sync_run()` helper.
`main()`	CLI entry-point that times execution and prints both the raw JSON and a human-readable digest.

Design Decisions

Gemini via `OpenAIChatCompletionClient`

- Gemini is preferred over OpenAI because of its pricing.
- Gemini currently offers an OpenAI-compatible API (beta). So you can use the OpenAIChatCompletionClient with the Gemini API.
  - ref. https://microsoft.github.io/autogen/stable/user-guide/agentchat-user-guide/tutorial/models.html#gemini-experimental
- Current Gemini limitation: **structured_output + function_calling cannot be concurrent**. The agent therefore issues **one tool call** and then relies on `structured_output=True`; the JSON block is extracted with a regex fallback.

Value-Investing Focus

- The system prompt encodes Buffett's **ten-point checklist** — economic moat, ROE, debt ratios, etc.
- Responses are rated **0-10** and annotated with metric-by-metric scores for downstream dashboards.

For now, the scope is limited to a Stock Investment Analysis Chatbot focused on fundamental analysis, not a general-purpose chatbot, due to time constraints. General-purpose chatbot functionality can be implemented later.

Time constraints: 2 hours per weekday (weekends not provided), after corporate (consulting) working hours (typically after 7 PM).

Logging & Security

Custom Logger class for lightweight implementation, tailored to real requirements and formatting.

```
[2025-06-13T09:12:17+07:00][INFO][ApiCredentialsManager]
Attempting to access GEMINI_API_KEY
```

- **Every external-resource access** (API key fetch, Yahoo Finance call) is logged.
- On read-only environments the file write fails gracefully so the system still prints to STDOUT.
- Use print log with log file because need to print in GoogleColab.
- No API keys are persisted to disk or memory beyond the immediate call scope.

Code Style & Quality

- Use Python classes to facilitate future deployment as a .py module, but ensure compatibility for implementation in Colab.

Standard	Practice
PEP 8	Black-formatted, 88-char lines, snake_case, no wildcard imports.
PEP 257	Google-style docstrings for all public classes, methods and functions.
Type Hints	`typing` and `pydantic` ensure static analysis via mypy/pyright.
Exceptions	Broad `try/except` only at I/O boundaries; internal logic uses precise exception types.
Immutable config	All constants are `UPPER_SNAKE_CASE`; environment interaction isolated to one class.

Extensibility

- Projest Structure: setup as python package, manage dependencies via `uv` instead of `conda`
- Web app: Next.js, Vercel, Modal (cloud GPU)
- Add new agents
- Add new data sources
- Enable parallel processing
- And more

>

Setup

In [ ]:
```
# Check the environment
!conda env list
```

```
# conda environments:
#
base                  /Users/pupipatsingkhorn/miniconda3
datascience          * /Users/pupipatsingkhorn/miniconda3/envs/datascience
```

In [ ]:
```
# # Install libraries
# %pip install "autogen-agentchat" "autogen-ext[openai]"
# %pip install nest-asyncio
```

Import

In [ ]:
```
import asyncio
import json
import os
import re
import time
from datetime import datetime
from typing import Any, Dict, List, Optional

import yfinance as yf

# Microsoft's AutoGen imports
from autogen_agentchat.agents import AssistantAgent
from autogen_agentchat.base import Response
from autogen_agentchat.messages import TextMessage
from autogen_core import CancellationToken
from autogen_core.models import ModelInfo
from autogen_core.tools import FunctionTool
from autogen_ext.models.openai import OpenAIChatCompletionClient
from dotenv import load_dotenv
from pydantic import BaseModel, Field

import nest_asyncio

nest_asyncio.apply()
```

Settings

In [ ]:
```
class Logger:
    """Logger class for handling different types of log messages."""

    LOG_FILE: str = "log.log"

    def __init__(self, module_name: str) -> None:
        """
        Initialize logger with module name.

        Args:
            module_name: Name of the module using the logger
        """
        self.module_name: str = module_name

    def _write_log(self, level: str, message: str) -> None:
        """
        Write a log message to both console and file.

        Args:
            level: Log level (INFO, WARNING, ERROR)
            message: Log message
        """
        timestamp: str = datetime.now().isoformat()
```

```
        log_entry: str = f"[{timestamp}][{level}][{self.module_name}] {me
        print(log_entry)
        try:
            with open(self.LOG_FILE, "a", encoding="utf-8") as f:
                f.write(log_entry + "\n")
        except Exception as e:
            print(f"[{timestamp}][ERROR][Logger] Failed to write to log f

    def info(self, message: str) -> None:
        """Log informational messages with timestamp."""
        self._write_log("INFO", message)

    def warning(self, message: str) -> None:
        """Log warning messages with timestamp."""
        self._write_log("WARNING", message)

    def error(self, message: str) -> None:
        """Log error messages with timestamp."""
        self._write_log("ERROR", message)


class ApiCredentialsManager:
    """
    Configuration class for managing API credentials.
    Note: GEMINI_API_KEY is accessed via a local .env file and is not pro
    """

    API_KEY_NAME: str = "GEMINI_API_KEY"

    def __init__(self) -> None:
        """Initialize configuration and load environment variables."""
        self.logger: Logger = Logger("ApiCredentialsManager")
        self._load_environment()

    def _load_environment(self) -> None:
        """Load environment variables from .env file."""
        load_dotenv()
        self.logger.info("Environment variables loaded")

    def get_api_key(self) -> str:
        """
        Securely retrieves the API key from the environment variable.

        Returns:
            str: The API key.

        Raises:
            ValueError: If API key is not set in environment variables.
        """
        self.logger.info(f"Attempting to access {self.API_KEY_NAME}")
        api_key: Optional[str] = os.getenv(self.API_KEY_NAME)
        if not api_key:
            self.logger.error(f"{self.API_KEY_NAME} not set")
            raise ValueError(f"{self.API_KEY_NAME} environment variable n
        self.logger.info(f"{self.API_KEY_NAME} access successful")
        return api_key
```

Tools

In [ ]:
```python
def get_stock_data_from_yfinance(ticker: str) -> Dict[str, Any]:
    """
    Retrieve stock data from Yahoo Finance for a specific ticker,
    including key performance indicators.

    Args:
        ticker: The stock ticker symbol (e.g., 'AAPL' for Apple).

    Returns:
        Dictionary containing stock data and key performance indicators
    """
    logger: Logger = Logger("get_stock_data_from_yfinance")
    try:
        logger.info(f"ExternalAPI, service=yfinance, type=request, ticker
        try:
            stock: yf.Ticker = yf.Ticker(ticker)
        except Exception as e:
            logger.error(f"Failed to access yfinance API for {ticker}: {e
            return {}
        logger.info(f"ExternalAPI, service=yfinance, type=response, ticke

        info: Dict[str, Any] = stock.info

        return {
            "company": info.get("shortName", "N/A"),
            "sector": info.get("sector", "N/A"),
            "industry": info.get("industry", "N/A"),
            "market_cap_b": round(info.get("marketCap", 0) / 1_000_000_00
            "pe_ratio": info.get("trailingPE", "N/A"),
            "forward_pe": info.get("forwardPE", "N/A"),
            "eps_ttm": info.get("trailingEps", "N/A"),
            "dividend_yield": round(info.get("dividendYield", 0) * 100, 2
            if info.get("dividendYield")
            else "N/A",
            "profit_margin": round(info.get("profitMargins", 0) * 100, 2)
            if info.get("profitMargins")
            else "N/A",
            "roe": round(info.get("returnOnEquity", 0) * 100, 2)
            if info.get("returnOnEquity")
            else "N/A",
            "debt_to_equity": info.get("debtToEquity", "N/A"),
            "beta_5y": info.get("beta", "N/A"),
            "week_high_52": info.get("fiftyTwoWeekHigh", "N/A"),
            "week_low_52": info.get("fiftyTwoWeekLow", "N/A"),
            "analyst_rating": info.get("recommendationKey", "N/A").capita
            if info.get("recommendationKey")
            else "N/A",
            "target_price": info.get("targetMeanPrice", "N/A"),
        }

    except Exception as e:
        logger.error(f"Error retrieving data for {ticker}: {e}")
        return {}
```

Agent

In [ ]:
```python
# -- Pydantic models --
```

```python
class Metrics(BaseModel):
    """Model for representing a relevant metric with its rating."""

    metric: str = Field(description="Name of the metric")
    rating: int = Field(description="Rating from 0-10, where 10 is best")


class InvestmentAnalysis(BaseModel):
    """Model for representing a complete investment analysis."""

    reasoning: str = Field(
        description="Detailed reasoning for the investment decision based
    )
    rating: int = Field(
        description="Overall investment rating from 0-10, where 10 is bes
    )
    relevant_metrics: List[Metrics] = Field(
        description="List of relevant metrics with individual ratings"
    )
    improvement_requirements: str = Field(
        description="Areas where the company needs to improve"
    )


# -- Workflow --
class Workflow:
    def __init__(self, ticker: str) -> None:
        """
        Initialize the Workflow for a stock analysis.

        Args:
            ticker: The stock ticker symbol (e.g., 'AAPL' for Apple)
        """
        self.logger: Logger = Logger("Workflow")
        self.ticker: str = ticker
        self.get_stock_data_tool: FunctionTool = FunctionTool(
            get_stock_data_from_yfinance,
            description="Retrieve financial data for a stock using yfinan
            strict=True,
        )

    async def run(self) -> Dict[str, Any]:
        """
        Run the stock analysis based on Warren Buffett's principles.

        Returns:
            Dictionary with the analysis results
        """
        self.logger.info(f"Analyzing stock: {self.ticker}")

        # Initialize model_client
        self.logger.info("Initializing model client")
        model_client: OpenAIChatCompletionClient = OpenAIChatCompletionCl
            model="gemini-2.0-flash-lite",
            model_info=ModelInfo(
                vision=True,
                function_calling=True,
                json_output=True,
                family="unknown",
```

```
            structured_output=True,
        ),
        api_key=ApiCredentialsManager().get_api_key(),
    )

    system_message: str = f"""You are an expert investment analyst sp

    INVESTMENT PRINCIPLES:
    1. Value Investing
    - Focus on intrinsic value vs. market price
    - Seek great businesses at fair prices
    - Avoid fair businesses at great prices

    2. Long-term Perspective
    - Minimum 10-year investment horizon
    - Ignore short-term market fluctuations
    - Focus on sustainable competitive advantages

    3. Business Understanding
    - Only analyze businesses within your expertise
    - Focus on simple, understandable business models
    - Avoid complex or rapidly changing industries

    4. Economic Moat Analysis
    - Brand power and customer loyalty
    - Network effects and switching costs
    - Cost advantages and economies of scale
    - Regulatory advantages and patents

    5. Management Quality
    - Track record of capital allocation
    - Alignment with shareholder interests
    - Conservative financial policies
    - Transparent communication

    6. Financial Health
    - Strong balance sheet metrics
    - Consistent earnings growth
    - High return on equity (ROE)
    - Low debt-to-equity ratio
    - Strong free cash flow

    7. Predictability
    - Stable revenue streams
    - Consistent profit margins
    - Low capital expenditure needs
    - Predictable business cycles

    8. Margin of Safety
    - Significant discount to intrinsic value
    - Conservative growth assumptions
    - Buffer against market volatility
    - Risk management considerations

    9. Dividend Analysis
    - Sustainable payout ratio
    - History of dividend growth
    - Cash flow coverage
    - Management's capital allocation priorities
```

```
        10. Circle of Competence
            - Industry expertise
            - Competitive dynamics understanding
            - Regulatory environment knowledge
            - Market positioning assessment

        ANALYSIS TASK:
        Analyze {self.ticker} stock using the get_stock_data tool to retr

        1. Reasoning (Detailed Analysis)
        - Evaluate against each investment principle
        - Highlight key strengths and weaknesses
        - Consider industry-specific factors
        - Address potential risks and mitigants

        2. Overall Rating (0-10)
        - 9-10: Exceptional investment opportunity
        - 7-8: Strong investment with minor concerns
        - 5-6: Average investment with mixed signals
        - 3-4: Below average with significant concerns
        - 0-2: Poor investment opportunity

        3. Key Metrics Analysis
        - Select 5-7 most relevant metrics
        - Rate each metric from 0-10
        - Explain rating rationale
        - Compare against industry benchmarks

        4. Improvement Requirements
        - Specific areas needing improvement
        - Concrete action items
        - Timeline expectations
        - Success metrics

        RESPONSE FORMAT:
        Your response must be a valid JSON object following the Investmen

        InvestmentAnalysis schema: {InvestmentAnalysis.model_json_schema(
        """

        self.logger.info("Initializing InvestmentAnalystAgent")
        analyst: AssistantAgent = AssistantAgent(
            name="InvestmentAnalyst",
            description="An investment analyst who follows Warren Buffett
            model_client=model_client,
            system_message=system_message,
            tools=[self.get_stock_data_tool],
            reflect_on_tool_use=True,
        )

        self.logger.info("Creating request_message")
        user_prompt: str = f"Please conduct a comprehensive investment an
        request_message: TextMessage = TextMessage(
            content=user_prompt,
            source="user",
        )

        self.logger.info("Sending request_message to Agent")
        response_message: Response = await analyst.on_messages(
            [request_message],
```

```
            cancellation_token=CancellationToken(),
        )
        self.logger.info("Received response_message from Agent")

        final_content: str = response_message.chat_message.content  # ```
        try:
            json_match: Optional[re.Match[str]] = re.search(
                r"```json\s*(.*?)\s*```", final_content, re.DOTALL
            )
            if json_match:
                json_str: str = json_match.group(1)
                parsed_data: Dict[str, Any] = json.loads(json_str)
                self.logger.info(
                    f"Successfully extracted and analyzed JSON for {self.
                )
                return parsed_data
        except Exception as json_error:
            self.logger.error(f"Failed to extract JSON: {json_error}")
            return {
                "reasoning": "FAILED",
                "rating": -1,
                "relevant_metrics": [],
                "improvement_requirements": "FAILED",
            }  # Fallback

    def sync_run(self) -> Dict[str, Any]:
        """
        Synchronous wrapper for the run method.

        Returns:
            Dictionary with the analysis results
        """
        self.logger.info("sync_run() called")
        return asyncio.run(self.run())
```

In [ ]:

```
def main(ticker: str = "AAPL") -> None:
    """
    Main function to run the stock analysis.

    Args:
        ticker: The stock ticker symbol to analyze (default: "AAPL")
    """
    start_time: float = time.time()
    logger: Logger = Logger("main")

    # Create and run the workflow
    logger.info("Initializing workflow")
    workflow: Workflow = Workflow(ticker=ticker)
    logger.info("Running workflow")
    result: Dict[str, Any] = workflow.sync_run()

    # Print JSON version
    result_jsonstr: str = json.dumps(result, indent=4)
    print(f"\n{'-' * 50}\nJSON Result:\n{result_jsonstr}\n{'-' * 50}\n")

    # Print human-readable version
    def print_human_readable_result(result: Dict[str, Any]) -> None:
        print("\n" + "-" * 50)
        print(f"INVESTMENT ANALYSIS FOR {ticker}")
        print("-" * 50)
```

```
        print("\nREASONING:")
        print(result["reasoning"])

        print(f"\nOVERALL RATING: {result['rating']}/10")

        print("\nRELEVANT metrics:")
        for metric in result["relevant_metrics"]:
            print(f"- {metric['metric']}: {metric['rating']}/10")

        print("\nIMPROVEMENT REQUIREMENTS:")
        print(result["improvement_requirements"])

        print("\n" + "-" * 50)

    print_human_readable_result(result)

    end_time: float = time.time()
    logger.info(f"Total time: {end_time - start_time:.2f} seconds")


main()
```

```
[2025-06-13T20:54:31.201278][INFO][main] Initializing workflow
[2025-06-13T20:54:31.202795][INFO][main] Running workflow
[2025-06-13T20:54:31.202891][INFO][Workflow] sync_run() called
[2025-06-13T20:54:31.203008][INFO][Workflow] Analyzing stock: AAPL
[2025-06-13T20:54:31.203054][INFO][Workflow] Initializing model client
[2025-06-13T20:54:31.203331][INFO][ApiCredentialsManager] Environment vari
ables loaded
[2025-06-13T20:54:31.203460][INFO][ApiCredentialsManager] Attempting to ac
cess GEMINI_API_KEY
[2025-06-13T20:54:31.203922][INFO][ApiCredentialsManager] GEMINI_API_KEY a
ccess successful
[2025-06-13T20:54:31.300260][INFO][Workflow] Initializing InvestmentAnalys
tAgent
[2025-06-13T20:54:31.300425][INFO][Workflow] Creating request_message
[2025-06-13T20:54:31.300487][INFO][Workflow] Sending request_message to Ag
ent
[2025-06-13T20:54:37.827900][INFO][Workflow] Received response_message fro
m Agent
[2025-06-13T20:54:37.829389][INFO][Workflow] Successfully extracted and an
alyzed JSON for AAPL

------------------------------------------------
JSON Result:
{
    "reasoning": "AAPL, as a technology company, has a strong brand and a
massive installed base, demonstrating a significant economic moat through
brand power and network effects. Its ecosystem lock-in through hardware, s
oftware, and services creates high switching costs for its customers. Howe
ver, the technology sector is subject to rapid change and innovation, maki
ng long-term predictability a challenge. The company's financial health is
robust, with consistent earnings growth and a strong balance sheet. Manage
ment has a proven track record, but the company operates in a competitive
industry where maintaining a competitive advantage over the long term is c
rucial. The analysis will focus on its financial performance, competitive
position, and growth prospects, as well as its ability to adapt to changes
in the tech landscape. We will also assess management's capital allocation
decisions, including dividends and share repurchases. The current market p
rice will be compared to intrinsic value, and a margin of safety will be c
onsidered. The long-term sustainability of its competitive advantages in a
rapidly evolving market will be a key consideration.",
    "rating": 7,
    "relevant_metrics": [
        {
            "metric": "Revenue Growth",
            "rating": 7,
            "rationale": "AAPL has demonstrated consistent revenue growth
over the past decade, driven by new product releases and expansion into se
rvices. However, growth has slowed in recent years as the market matures.
The rating reflects solid, but not exceptional, growth. (Industry average:
5-8%)"
        },
        {
            "metric": "Gross Margin",
            "rating": 8,
            "rationale": "AAPL consistently maintains high gross margins d
ue to its brand strength, pricing power, and efficient supply chain. This
is a key indicator of its economic moat. (Industry average: 30-50%)"
        },
        {
            "metric": "Return on Equity (ROE)",
```

```
            "rating": 9,
            "rationale": "AAPL has a very high ROE, indicating efficient u
se of shareholder equity. This is a sign of strong profitability and effec
tive capital allocation. (Industry average: 15-25%)"
        },
        {
            "metric": "Debt-to-Equity Ratio",
            "rating": 9,
            "rationale": "AAPL maintains a conservative balance sheet with
a low debt-to-equity ratio. This provides financial flexibility and reduce
s risk. (Industry average: varies)"
        },
        {
            "metric": "Free Cash Flow",
            "rating": 8,
            "rationale": "AAPL generates significant free cash flow, which
it uses for share buybacks, dividends, and strategic investments. (Industr
y average: varies)"
        },
        {
            "metric": "Dividend Yield and Growth",
            "rating": 7,
            "rationale": "AAPL pays a dividend and has a history of increa
sing it, though the yield is not exceptionally high. This is a positive si
gnal of capital allocation. (Industry average: varies)"
        }
    ],
    "improvement_requirements": "AAPL should focus on: 1) Sustaining innov
ation to maintain its competitive edge in existing product categories and
entering new ones. 2) Diversifying revenue streams to reduce reliance on h
ardware sales. 3) Managing regulatory risks and potential antitrust scruti
ny. 4) Improving supply chain resilience. Success metrics include maintain
ing or growing market share in key product categories, growing services re
venue, and expanding into new markets or product categories. Timeline expe
ctations should focus on consistent progress and adaptation over a 5-10 ye
ar timeframe."
}
--------------------------------------------------


--------------------------------------------------
INVESTMENT ANALYSIS FOR AAPL
--------------------------------------------------

REASONING:
AAPL, as a technology company, has a strong brand and a massive installed
base, demonstrating a significant economic moat through brand power and ne
twork effects. Its ecosystem lock-in through hardware, software, and servi
ces creates high switching costs for its customers. However, the technolog
y sector is subject to rapid change and innovation, making long-term predi
ctability a challenge. The company's financial health is robust, with cons
istent earnings growth and a strong balance sheet. Management has a proven
track record, but the company operates in a competitive industry where mai
ntaining a competitive advantage over the long term is crucial. The analys
is will focus on its financial performance, competitive position, and grow
th prospects, as well as its ability to adapt to changes in the tech lands
cape. We will also assess management's capital allocation decisions, inclu
ding dividends and share repurchases. The current market price will be com
pared to intrinsic value, and a margin of safety will be considered. The l
ong-term sustainability of its competitive advantages in a rapidly evolvin
```

g market will be a key consideration.

OVERALL RATING: 7/10

RELEVANT metrics:
- Revenue Growth: 7/10
- Gross Margin: 8/10
- Return on Equity (ROE): 9/10
- Debt-to-Equity Ratio: 9/10
- Free Cash Flow: 8/10
- Dividend Yield and Growth: 7/10

IMPROVEMENT REQUIREMENTS:
AAPL should focus on: 1) Sustaining innovation to maintain its competitive edge in existing product categories and entering new ones. 2) Diversifying revenue streams to reduce reliance on hardware sales. 3) Managing regulatory risks and potential antitrust scrutiny. 4) Improving supply chain resilience. Success metrics include maintaining or growing market share in key product categories, growing services revenue, and expanding into new markets or product categories. Timeline expectations should focus on consistent progress and adaptation over a 5-10 year timeframe.

[2025-06-13T20:54:37.829790][INFO][main] Total time: 6.63 seconds