BA – Agentic AI Based Financial Advisor

Mayank Mokhere (BAC0068)

AI Hands-On Task Report

27th January 2026

Table of Contents

1 Objective of the Task

This task focuses on building an **Agentic AI-based Financial Advisor** that analyzes current market conditions and provides **short-term and long-term investment recommendations**. The solution follows a **multi-agent architecture**, where each agent performs a specialized function such as market analysis or investment advisory, enabling collaborative and informed decision-making.

The system is implemented using the **PydanticAI framework** to manage agent interactions and ensure structured, validated outputs. An **open-source Large Language Model (LLM)** is used via a local runtime to avoid dependency on proprietary services. .

Key Objectives:

1. Build an Agentic AI-based Financial Advisor using a multi-agent approach.
2. Integrate an open-source LLM using Ollama for local model execution.
3. Use PydanticAI for agent orchestration and structured outputs.
4. To generate short-term and long-term investment strategies
5. Ensure clean, modular, and well-documented code.

2 Introduction to PydanticAI Framework

PydanticAI is an agent-oriented AI framework designed to build **structured, reliable, and multi-agent AI systems**. It enables developers to define intelligent agents with clear roles while enforcing schema-validated outputs using Pydantic models.
Instead of relying on unstructured language model responses, PydanticAI focuses on **deterministic and explainable agent behavior**, making it particularly suitable for domains such as finance, healthcare, and enterprise applications.

Advantages of AutoGen

1. Enforces **structured and validated outputs** using Pydantic schemas
2. Supports **multi-agent and orchestrated workflows**
3. Improves reliability and reduces hallucinations
4. Easy integration with open-source and local LLMs (e.g., Ollama)

Limitations of AutoGen

1. API changes across versions require careful handling
2. Less suitable for highly creative, free-form tasks
3. Requires well-defined schemas and planning during design
4. Additional validation overhead may impact speed for simple tasks

3 Tools and Technologies Used

- Programming Language: Python
- Development Environment: Visual Studio Code
- Framework: PydanticAI
- Language Model: LLaMA 3.2 (open source) via Ollama
- Libraries Used: pydantic-ai,pydantic,ollama

All tools used in this project are open-source, ensuring cost-free development and execution.

4 System Design and Workflow

The system is designed using **multiple agents** coordinated by a **central orchestrator**.

4.1 Agents Description

- **Market Analysis Agent:** This agent analyzes current global and Indian financial market conditions.It analyzes market trends, and general financial sentiment to provide a clear understanding of the current market scenario.
- **Short-Term Investment Agent:** This agent focuses on identifying investment opportunities with a **short-term time horizon**, typically ranging from a few weeks to a few months. Using market insights, it emphasizes momentum, volatility, and near-term trends to suggest tactical investment options suitable for short-term gains.
- **Long-Term Investment Agent:** This agent concentrates on **long-term investment strategies** aimed at sustained growth and wealth creation. It considers stability, fundamental strength, and long-term market outlook to recommend investments suitable for extended holding periods.

4.2 Role of the Orchestrator

The **Orchestrator** acts as the central controller of the Agentic AI system. It is responsible for coordinating the execution of all agents and managing the overall workflow.
The orchestrator ensures:

- Agents communicate in a fixed sequence.
- Each agent operates independently
- The workflow terminates automatically after all agents complete their tasks.
- The output remains structured and in readable format.

5 Execution Flow

- The program starts from the main execution file.
- The orchestrator runs the **Market Analysis Agent** to assess market conditions.
- The market insights are passed to the **Short-Term** and **Long-Term Investment Agents**.
- The orchestrator aggregates all agent outputs and generates the final financial report.

User
↓
Orchestrator
↓
Market Analysis Agent
↓
Short-Term Investment Agent
↓
Long-Term Investment Agent
↓
Final Financial Advisory Report

6 File System Structure

The project follows a clean and modular folder structure:

```
agentic_financial_advisor/
|
├── agents/
|   ├── market_analysis_agent.py
|   ├── short_term_investment_agent.py
|   └── long_term_investment_agent.py
|
├── orchestrator/
|   └── financial_advisor_orchestrator.py
|
├── schemas/
|   └── investment_schema.py
|
├── utils/
|   └── llm_configuration.py
|
├── main.py
├── requirements.txt
```


7 Conclusion

This project successfully demonstrates the design and implementation of an **Agentic AI-based Financial Advisor** using a multi-agent architecture. By dividing responsibilities among specialized agents for market analysis and investment strategy, the system produces structured and explainable short-term and long-term investment recommendations. The use of the PydanticAI framework ensures reliable agent coordination and validated outputs, while the integration of an open-source LLM through Ollama enables local execution without dependency on proprietary services.

Overall, the solution highlights how agent-based AI systems can enhance financial decision-making through modular design, clarity, and scalability. The project provides a strong foundation for future enhancements such as real-time market data integration, advanced risk analysis, and portfolio optimization, making it suitable for both academic evaluation and practical extension.

