Technical Proposal: Autonomous Financial Research Agent System

To: Shaheer

From: Yan

Date: November 24, 2025

Subject: Proposed Architecture and Execution Plan for Financial Research Crew

1. Executive Summary

Following an evaluation of the **OpenAI Financial Agent** examples and the **ScholarAI** agentic architecture, this proposal outlines the plan to develop a similar autonomous research system using the **CrewAI framework**.
While OpenAI's examples demonstrate single-agent capabilities, **CrewAI** was selected for this implementation due to its native support for **Role-Based Agent Orchestration**. This allows us to decouple the "Researcher" (data gathering) from the "Analyst" (quantitative reasoning) and the "Writer" (reporting), resulting in higher accuracy, reduced hallucinations, and modular maintainability.

2. Proposed Architecture

The system will utilize a **Hierarchical Crew Architecture**. Unlike a sequential flow where Agent A simply passes data to Agent B, a "Manager Agent" will oversee the process, delegating sub-tasks, reviewing quality, and requesting iterations if the data is insufficient.

2.1 System Context Diagram

This diagram illustrates how the Crew interacts with external tools, APIs, and the persistent memory layer.

Code snippet

```
graph TD
    subgraph "External Interfaces"
        UI[User Interface / CLI]
        APIs[Financial & Search APIs]
    end

    subgraph "CrewAI Runtime Environment"
        Manager[<b>Manager Agent</b><br/><i>Orchestrator & QC</i>]

        subgraph "Worker Agents"
```

```
        Researcher[<b>Researcher Agent</b><br/><i>Web & News Scraper</i>]
        Analyst[<b>Financial Analyst Agent</b><br/><i>Quant & Ratios</i>]
        Writer[<b>Reporting Agent</b><br/><i>Synthesis & Formatting</i>]
    end

    Memory[(<b>Shared Memory</b><br/><i>Vector DB / Embeddings</i>)]
end

%% Flows
UI -->|Trigger Request| Manager
Manager --Delegates Task--> Researcher
Manager --Delegates Task--> Analyst
Manager --Delegates Task--> Writer

Researcher <-->|Search & Fetch| APIs
Analyst <-->|Fetch Market Data| APIs

Researcher -.->|Store Context| Memory
Analyst -.->|Retrieve Context| Memory
Writer -.->|Retrieve Context| Memory

Writer -->|Draft Report| Manager
Manager -->|Final Output| UI
```

2.2 Agent Interaction Flow (Sequence)

This sequence diagram details the operational logic: how the Manager coordinates the "Analyze" request to ensure data is gathered *before* analysis occurs.

Code snippet

```
sequenceDiagram
    participant User
    participant Manager as Manager Agent
    participant Res as Researcher
    participant Anl as Analyst
    participant Wri as Writer
    participant Mem as Memory/Tools

    User->>Manager: "Analyze TSLA Q3 Strategy"

    loop Information Gathering
        Manager->>Res: Task: Gather Q3 News & Filings
        Res->>Mem: Search Web / SEC
        Mem-->>Res: Raw Data
        Res-->>Manager: Data Collected
    end

    loop Financial Analysis
```

```
        Manager->>Anl: Task: Calculate Growth & Ratios based on Data
        Anl->>Mem: Retrieve Raw Data
        Anl->>Anl: Compute Metrics
        Anl-->>Manager: Quantitative Insights
    end

    Manager->>Wri: Task: Compile Investment Memo
    Wri->>Mem: Retrieve Context (Data + Insights)
    Wri-->>Manager: Draft Report

    alt Quality Check Failed
        Manager->>Res: Request: Missing Competitor Info
        Res-->>Manager: New Data
        Manager->>Wri: Request: Update Report
    else Quality Check Passed
        Manager->>User: Final Report
    end
```

---

3. Project Scaffolding (Code Structure)

To ensure scalability and alignment with enterprise standards (similar to the ScholarAI structure), the project will follow this directory structure:

Plaintext

```
financial-research-crew/
├── .env                    # API Keys (OpenAI, Serper, Financial Data)
├── pyproject.toml          # Dependencies
├── README.md               # Documentation
├── src/
│   └── finance_crew/
│       ├── __init__.py
│       ├── main.py         # Entry point to kick off the Crew
│       ├── config/
│       │   ├── agents.yaml    # Definition of Agent Roles & Backstories
│       │   └── tasks.yaml     # Definition of Tasks & Deliverables
│       ├── agents/
│       │   ├── __init__.py
│       │   └── research_agents.py # Custom agent logic (if extending BaseAgent)
│       ├── tools/
│       │   ├── __init__.py
│       │   ├── search_tools.py    # Wrappers for Serper/Google
│       │   ├── finance_tools.py   # Wrappers for YahooFinance/Polygon
│       │   └── calculator_tools.py
│       └── crew.py         # The Crew orchestration logic
```

```
├── notebooks/              # Jupyter notebooks for prototyping
│   └── experiment_1_tools.ipynb
└── tests/
    ├── test_agents.py
    └── test_tools.py
```

---

4. Execution Plan (5-Week Timeline)

The project will be executed in **4 sprints** over 5 weeks, focusing on an iterative "Crawler-Walker-Runner" approach.

4.1 Gantt Chart Schedule

Code snippet

```
gantt
    title Financial Agent Development Timeline
    dateFormat  YYYY-MM-DD
    axisFormat  %W

    section Phase 1: Setup
    Repo Init & Tool Selection      :a1, 2025-11-25, 3d
    API Integration (Search/Fin)    :a2, after a1, 4d

    section Phase 2: Agents
    Researcher Agent Impl           :b1, 2025-12-02, 4d
    Analyst Agent Impl              :b2, after b1, 3d
    Unit Testing Agents             :b3, after b2, 2d

    section Phase 3: Orchestration
    Manager & Crew Logic            :c1, 2025-12-11, 4d
    Memory/Vector DB Integ          :c2, after c1, 3d
    Output Formatting               :c3, after c2, 2d

    section Phase 4: Polish
    UI/CLI Wrapper                  :d1, 2025-12-22, 3d
    E2E Testing & Validation        :d2, after d1, 3d
    Documentation & Handoff         :d3, after d2, 2d
```

4.2 Detailed Task Breakdown

Week 1: Infrastructure & Tooling (Discovery)

- Initialize Git repository and Python environment.
- **Deliverable:** Validated connection to LLM (OpenAI/Groq) and successful API calls to Search (SerperDev) and Finance data sources (yfinance/AlphaVantage).

Week 2: Agent Definition (Development)

- Define `agents.yaml` with specific "backstories" to control hallucinations.

- Implement the **Researcher** (focus: breadth of data) and **Analyst** (focus: depth of calculation).
- **Deliverable:** Unit tests showing the Researcher can find a 10-K filing and the Analyst can calculate a P/E ratio.

Week 3: Orchestration & Memory (Core Logic)

- Implement crew.py utilizing the Hierarchical Process.
- Integrate crewai[tools] memory features (ChromaDB/Qdrant) to allow agents to recall previous search steps.
- **Deliverable:** A functional script where a single prompt triggers the full multi-agent chain.

Week 4: Refinement & Interface (UI/UX)

- Build a lightweight Streamlit or CLI interface.
- Implement "Human-in-the-loop" features (allowing the user to approve the research plan before execution).
- **Deliverable:** Beta version of the tool ready for internal demo.

Week 5: Documentation & Handover

- Write system architecture documentation.
- Create a "Prompt Engineering Guide" for future maintenance.
- **Deliverable:** Final Codebase and User Manual.
