**Product Requirements Document (PRD) — Phase 1**

**Product:** Multi-Agent Investment Advisor  
**Phase 1 UI:** CLI (Phase 2: Streamlit)  
**Date:** Aug 26, 2025  
**Owner:** [Team/You]

**1) Summary & Objectives**

We aim to build a **CLI-first**, multi-agent investment advisor system orchestrated with **LangGraph**, powered by **OpenAI models** (GPT-4.1 as default).

The system will use **Model Context Protocol (MCP)** tools for company news, financial data, and structured analysis. The workflow mirrors the current Python prototype: **gather news → gather financial data → analyze → recommend**.

**Success Criteria (Phase 1):**

* CLI produces **two Markdown reports**:
  + Analysis.md – synthesis of company news + fundamentals.
  + Recommendation.md – Buy/Hold/Sell decision with rationale.
* All external data access occurs via **MCP tools**.
* Agents run in **LangGraph orchestration**, not sequential scripts.
* All reasoning and generation handled by **OpenAI API**.

**2) Scope (Phase 1)**

**In Scope**

* **Orchestration:** LangGraph-based graph with four agent roles.
* **Agents & Roles:**
  1. **News Agent** – gather latest company news.
  2. **Financial Data Agent** – collect structured financials.
  3. **Analyst Agent** – produce analysis (combines news + fundamentals).
  4. **Recommendation Agent** – generate Buy/Hold/Sell output.
* **Tools (via MCP):**
  1. search.duckduckgo (company news).
  2. finance.yahoo.price (real-time price).
  3. finance.yahoo.company\_info (snapshot data).
  4. finance.yahoo.income\_statements (financial reports).
* **LLM:** OpenAI GPT-4.1 (primary), fallback GPT-4o-mini for faster runs.
* **CLI UX:** Command-line utility (invest-advisor --ticker AAPL).
* **Output Reports:** Markdown-formatted, timestamped, citing data sources.
* **Config:** .env for API keys, YAML/JSON for tool + model settings.
* **Observability:** Logging of node transitions, MCP tool calls, API usage.

**Out of Scope (Phase 1)**

* GUI/Streamlit interface (planned Phase 2).
* Multi-ticker batch runs.
* Advanced analytics (portfolio optimization, trade execution).

**3) Users & Use Cases**

**Target Users:** Retail investors, analysts, product managers, and students needing quick, structured investment insights.

**Use Cases:**

1. Produce **fast, explainable company analysis** with latest news + fundamentals.
2. Generate a clear **Buy/Hold/Sell recommendation** with assumptions.
3. Highlight **key risks, opportunities, and data citations**.

**4) Functional Requirements**

**4.1 CLI**

* Command:
* invest-advisor --ticker AAPL [--market-suffix .NS] [--out ./reports] [--date 2025-08-26] [--verbose]
* Generates Analysis.md and Recommendation.md reports.

**4.2 LangGraph Orchestration**

**Nodes (Agents):**

* **News Agent → Financial Data Agent → Analyst Agent → Recommendation Agent**.
* Orchestrator passes context/results across agents.
* Retry on MCP failures (configurable).

**4.3 Agents**

* **News Agent**: Summarizes latest news (headline, date, relevance, URL).
* **Financial Agent**: Collects company snapshot + income statements (PE/EPS, market cap, 52-week range, margins, etc.).
* **Analyst Agent**: Writes structured Markdown analysis (Overview, Financials, News, Risks, Insights).
* **Recommendation Agent**: Generates Buy/Hold/Sell stance with rationale, price context, and watchouts.

**4.4 MCP Tools**

* **DuckDuckGo** for recent news.
* **Yahoo Finance** (yfinance) for structured data:
  + company\_info(symbol)
  + income\_statements(symbol)
  + current\_price(symbol)

**4.5 LLM (OpenAI)**

* **Primary Model:** gpt-4.1 (for detailed, structured reasoning).
* **Fallback:** gpt-4o-mini (fast summaries/testing).
* Prompts require structured, concise outputs with **citations**.

**4.6 Output Reports**

* **Analysis.md:** Detailed report with sections, bullet points, and summary.
* **Recommendation.md:** Explicit decision (Buy/Hold/Sell), rationale, and risk notes.
* Include **timestamp of data freshness**.

**5) Non-Functional Requirements**

* **Performance:** End-to-end CLI run ≤ 60 seconds.
* **Resilience:** Retries for MCP failures; degraded outputs if partial data available.
* **Security:** API keys via environment variables; no PII.
* **Portability:** Core system reusable for Streamlit UI in Phase 2.

**6) Data Sources**

* **DuckDuckGo:** Query <TICKER> + company name, filter by last 7–14 days.
* **Yahoo Finance (yfinance wrapper):** Price, financials, company snapshot.

**7) Execution Flow**

1. User runs CLI with ticker.
2. **News Agent** calls MCP search, returns top stories.
3. **Financial Agent** fetches snapshot + statements.
4. **Analyst Agent** merges inputs → generates Analysis.md.
5. **Recommendation Agent** finalizes stance → generates Recommendation.md.
6. Reports saved with timestamp.

**8) Acceptance Criteria**

* CLI produces **two Markdown files** with timestamp.
* Reports include **citations for data sources**.
* LangGraph manages orchestration; no sequential hard-coded calls.
* OpenAI GPT-4.1 used for reasoning (logs confirm model usage).
* Resilient behavior if some tools fail.

**9) Milestones**

* **M1:** Project skeleton + OpenAI integration (3 days).
* **M2:** MCP tools implemented (4–5 days).
* **M3:** Agent prompts + LangGraph orchestration (3 days).
* **M4:** CLI + config + logging (2 days).
* **M5:** Tests (unit + E2E) + docs (3 days).
* **M6:** UAT & sign-off (2 days).

**10) Risks & Mitigations**

* **News relevance/quality** → restrict to recent results, deduplicate.
* **Financial API reliability** → retries, cache fallback.
* **Model cost/latency** → config option for gpt-4o-mini.
* **Compliance** → include disclaimer in outputs.

**11) Open Questions**

1. Should recommendations include a **time horizon** (short/long term)?
2. Locale formatting: US numbers vs. Indian lakh/crore?
3. Do we include **graph visualizations** in analysis, or only text?

**12) Deliverables**

* **Codebase:** LangGraph graph, MCP servers, CLI utility.
* **Docs:** README, architecture diagram, MCP tool schemas.
* **Outputs:** Analysis.md and Recommendation.md.
* **Tests:** unit + integration.

👉 This version is **fully OpenAI-aligned**:

* Orchestration = **LangGraph**
* Reasoning = **GPT-4.1 / GPT-4o-mini**
* Data tools = **MCP (DuckDuckGo, Yahoo Finance)**
* UX = **CLI reports**

Would you like me to also **draft the folder structure + sample LangGraph flow diagram** for Phase 1 so you can directly hand it to devs?

Sources

**You said:**

can we not use GPT 5.0

**ChatGPT said:**

Great question 👍 — yes, you can absolutely specify **GPT-5.0** (or the latest available OpenAI frontier model) instead of GPT-4.1 in your PRD.

Here’s how it would change:

* **LLM Section in PRD**
  + **Primary Model:** gpt-5.0 (latest flagship reasoning model, higher accuracy + efficiency).
  + **Fallback Model:** gpt-4o-mini (for faster, cost-effective summaries/testing).
* **Rationale for GPT-5.0**
  + Better **reasoning and factual grounding** for financial analysis.
  + More efficient **multi-agent coordination** (since MCP + LangGraph agents rely heavily on structured outputs).
  + Potentially **lower latency** and **higher cost efficiency** than GPT-4.1 depending on OpenAI’s release specs.

So yes ✅ — you can replace **GPT-4.1** in the PRD with **GPT-5.0** as the default engine without changing the orchestration (LangGraph, MCP, DuckDuckGo, Yahoo Finance).