Technical Appendix: DeepFinAI – An Agentic End-to-End AI Value Investing System

Jun-Jun Wan, Xiuming Xu, Bin Xu, Ruohan Liu
DeepFinAI.org
deepfinai.official@gmail.com

1. Introduction

DeepFinAI is an advanced end-to-end AI-driven system designed to generate institutional-quality investment analyses and valuation reports tailored specifically for equity analysts, portfolio managers, and institutional investors. It integrates state-of-the-art large language model (LLM) [6, 9, 18] capabilities with established financial modeling, aiming to deliver detailed, actionable investment intelligence reflecting the rigorous, value-oriented investment philosophies of renowned investors such as Warren Buffett, Charlie Munger, Seth Klarman, and Bruce Greenwald [2, 5, 11, 13].

At its core, DeepFinAI employs a two-agent architecture [7]. In the first phase, an information collection and valuation agent operates within a structured Reason-Act-Observe (ReAct) loop [20]. This agent systematically identifies the user's information requirements, invokes specialized tools through the Model Context Protocol (MCP) [1] to retrieve relevant information such as financial statements, company profiles, risk factors, and macroeconomic data, performs valuation calculation, and drafts preliminary report sections, embedding immediate inline citations for each factual assertion. Subsequently, the *BuffAI* thesis synthesis agent consolidates this evidence, producing a coherent, fully structured Markdown report–aligning with value investor styles–that is ready for downstream publishing or PDF conversion. This sequential agent approach ensures data accuracy and narrative coherence.

The MCP framework integrates data adapters connecting to Financial Modeling Prep (FMP) [4], SEC EDGAR [19], and proprietary data sources, providing programmatic access to financial statements, ratios, Management's Discussion & Analysis (MD&A) etc. Tools are LLM-callable functions with structured JSON responses, enforcing a rigorous *Evidence First* principle with immediate data attribution.

To enhance alignment, we plan and have partially implemented advanced post-training techniques, including Reinforcement Learning from Human Feedback (RLHF) [14] using Proximal Policy Optimization (PPO) [16], Direct Preference Optimization (DPO) [15], and parameter-efficient fine-tuning using Low-Rank Adaptation (LoRA) [8]. Using expert-rated analyst reports, this post-training aims to reduce hallucinations and improve report structure and citation. Details are in Section 5.

This technical appendix documents DeepFinAI's architecture, datasets, core modeling engines, safety, alignment and post-training strategies, and benchmarks.

2. System Architecture

DeepFinAI utilizes a modular, multi-layered microservices architecture designed for scalability and efficient data processing, enabling the generation of institutional-quality investment reports. Figure 1 illustrates this system.

2.1. Architectural Layers and Key Components

2.1.1. Frontend Layer

This layer provides the user interface for investment professionals.

- **BuffAI Interactive App**: The core web application for authenticated users to submit natural-language analysis queries, define requirements/risk preferences, manage profiles, and access reports.
- **Integrated Report Viewer**: An in-app module for rendering structured Markdown reports with interactive navigation and download options (e.g., PDF).

2.1.2. Integration Layer

This layer manages external communication, internal request routing, and resource access.

- **API Gateway (APIGW)**: Unified entry point for frontend requests, handling authentication, authorization, rate limiting, and routing to Core Services.
- **Prompt Store**: Central repository for version-controlled system prompts guiding LLM agents, ensuring consistent behavior and facilitating prompt engineering.
- **External API (ExtAPI) Abstraction**: Standardized interface for third-party data consumed by Data Ingestion MCP Servers via adapters.

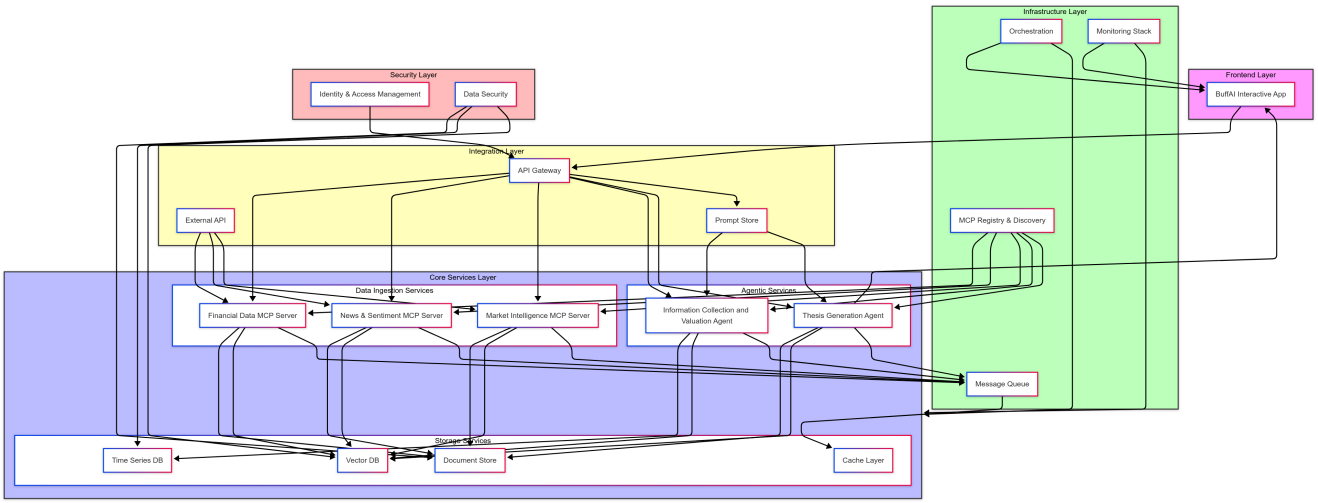

Figure 1. DeepFinAI System Architecture Diagram

2.1.3. Core Services Layer

This layer forms the operational backbone, responsible for data processing, AI-driven analyses, and storage:

- **Agentic Services**: Core analytical AI agents that orchestrate data-driven investment report generation:
  - **Information Collection and Valuation Agent**: Employs a ReAct loop [20], guided by system prompts, to identify information needs, invoke MCP tools for data retrieval (financials, profiles, filings, market context), perform quantitative valuations (e.g., DCF), and draft initial report sections with citations.
  - **Thesis Generation Agent**: Synthesizes data and drafts from the previous agent into a final, coherent Markdown report, embodying the *BuffAI* persona.
  - Agents are designed to be LLM-agnostic. Current implementation and testing utilize Google's Gemini-2.0-flash [18] via the Google Generative AI Python SDK.
- **Data Ingestion MCP Servers**: Microservices that expose data retrieval functionalities via MCP [1] tools:
  - **Financial Data MCP Server (FDS)**: Provides structured financial data, company profiles, valuation inputs, and qualitative SEC filing data via adapters for FMP [4] and SEC EDGAR [19].
- **Storage Services**: Includes Document Store (DS) for reports/profiles, Time Series DB (TSDB) for financial/market data, Vector DB (VDB) for embeddings supporting RAG [12], and a Cache Layer.

2.2. Component Sourcing: Proprietary vs. Open-Source

DeepFinAI combines proprietary innovations with open-source technologies.

- **Proprietary**: Core DeepFinAI system logic; Agentic Services framework; Data Ingestion MCP Server implementations; specialized data adapters; engineered system prompts; BuffAI Interactive App source code; custom Pydantic models, closed models such as Google Gemini [18].
- **Open-Source/Leveraged**: Vue.js, Vite, Ant Design Vue (Frontend); Python, Starlette, aiohttp, pydantic, cachetools (Backend); FAISS (vector database); open models such as DeepSeek R1 [6].

3. Dataset

DeepFinAI integrates data from diverse external sources via dedicated adapters in the Data Ingestion MCP Servers, alongside user and customer-provided information.

- **Financial Modeling Prep (FMP)**:
  - **Data Types**: Structured financial statements, stock data, profiles, ratios, growth metrics, DCF inputs.
  - **Access Mechanism**: Via the FMPAdapter within the Financial Data MCP Server (FDS), which utilizes FMP's REST API.
  - **Licensing**: DeepFinAI utilizes a commercial FMP API subscription, adhering to FMP's terms, rate limits, and free-tier awareness.
- **SEC EDGAR Data** :
  - **Data Types**: Qualitative data from 10-K/10-Q filings (Business Description, Risk Factors, MD&A); CIKs.
  - **Access Mechanism**: Via the SecApiAdapter using direct sec.gov
  - **Licensing**: Data sourced directly from sec.gov is considered public domain information provided by the U.S. government.
- **Customer-Provided Data & Knowledge**:
  - **Data Types**: Enterprise customers can optionally link their proprietary data sources, internal research, established investment thesis templates, or preferred analyt-

ical frameworks.
  - **Access Mechanism**: Integration mechanisms (e.g., secure API connections, dedicated MCP tool adapters for customer systems, or document ingestion pipelines for the Vector DB) are provided to incorporate this customer-specific context.
  - **Licensing**: Customer-provided data is governed by customer agreements, ensuring privacy and adherence to their policies.

4. Core Modeling Pipeline

DeepFinAI's core pipeline transforms a user's natural-language request into a polished, institutional-grade investment report via two specialized LLM agents and MCP tools.

4.1. Overview of the Pipeline

1. **Agent 1 – Information Collection & Valuation**: Using a Reason–Act–Observe loop [20], this agent parses the user query, invokes MCP tools to retrieve structured data (financials, profiles, filings) and qualitative excerpts, performs preliminary quantitative analyses (e.g. DCF with LLM-guided assumptions), and drafts each report section in Markdown with inline citations.
2. **Agent 2 – Thesis Generation**: Consumes Agent 1's drafts and data, then synthesizes and refines them into a coherent, fully structured report. Retrieval-augmented inputs from the Vector DB ensure up-to-date context, while prompt templates enforce logical flow, style, and completeness.

4.2. From Query to Drafts

- **Query Ingestion:** Users submit company analyses and risk preferences via the BuffAI App. The API Gateway authenticates the request (IAM) and forwards it to Agent 1.
- **ReAct Iterations:** In each cycle, Agent 1:
  - **Reasons** about missing information for specific report sections.
  - **Acts** by calling MCP tools (via SSE McpClient) to fetch data from FDS, NSS, MIS.
  - **Observes** JSON responses and updates its internal context.
- **Valuation Integration:** Quantitative outputs (e.g. DCF) are merged with LLM-generated narrative, with assumptions (discount rates, growth) justified in text.
- **Drafting & Citation:** Each section is authored in Markdown, immediately attributing all facts to their source tool.

4.3. Synthesis into Final Report

- **Input Aggregation:** Agent 2 ingests Agent 1's drafts, retrieved embeddings (RAG), and prompt templates.
- **Structured Synthesis:** Guided by the BuffAI persona, it reorganizes, rewrites, and enriches content—generating the Executive Summary and ensuring stylistic and analytical consistency.
- **Output Formatting:** The final Markdown report is delivered to the BuffAI App, ready for rendering or PDF export.

5. Safety, Alignment and Post-Training

Ensuring the safety, reliability, and alignment of DeepFinAI's outputs with institutional quality standards and user expectations is a critical design objective. This section details the current safety and alignment measures in place and outlines planned enhancements through advanced post-training techniques.

5.1. Current Safety and Alignment Measures

DeepFinAI incorporates several layers of mechanisms to promote factual accuracy, mitigate risks associated with LLM generation, and align outputs with the system's intended purpose:

- **Prompt Engineering**: System prompts (stored in the Prompt Store) define clear personas, enforce the ReAct methodology [20], mandate "Evidence First" citations, and constrain agents to use only vetted data. Prompts include delimiters and explicit instructions to resist injection or role subversion.
- **Few-Shot Prompt Examples**: Snippets from gold-standard investment reports are embedded as few-shot exemplars–showing correct section headings, citation patterns, and value-investor tone. This anchors the LLM's style and structure before any data retrieval or generation takes place.
- **RAG with Gold Reports & Live Data:** The Thesis Generation Agent uses Retrieval-Augmented Generation [12] to fetch relevant passages from both (a) an indexed library of vetted, gold-standard reports and (b) up-to-date embeddings in the Vector DB [3, 10]. This combined RAG approach ensures the model has high-quality reference examples while also accessing the latest data, further minimizing hallucinations and improving narrative consistency.
- **Input Validation**: User queries are sanitized and length-checked before inclusion in LLM prompts.
- **Tool Confinement**: Agents can only invoke predefined MCP tools, each with strict input/output schemas, preventing arbitrary code execution.
- **Response Validation**: LLM outputs–tool-call parameters and draft sections–are parsed and schema-checked; malformed or unexpected content is flagged or rejected.
- **Two-Agent Workflow**: Agent 1 collects, cites, and calculates all data; Agent 2 only reorganizes that vetted content, preventing ungrounded hallucination during synthe-

sis.

- **Adapter-Level Schema Enforcement**: Data from FMP and SEC adapters is validated via Pydantic models, ensuring correct types and formats.
- **Retrieval Grounding**: The Thesis Generation Agent uses RAG (via the Vector DB) to fetch relevant context, further reducing hallucinations.
- **Ongoing Red-Teaming**: Prompts and workflows undergo periodic adversarial testing to uncover and remediate vulnerabilities.

5.2. Post-Training Roadmap

We will refine DeepFinAI through targeted post-training stages, each leveraging specialized data and human feedback:

1. **LoRA Fine-Tuning** [8] We assemble 3,000–5,000 high-quality analyst reports alongside a "value-investor corpus" composed of Berkshire Hathaway letters, Charlie Munger speeches, and seminal value-investing texts known for in-depth qualitative sections—companies' business models, explicit moat analyses, and competitive positioning. Each is converted into structured Markdown with inline tool-call placeholders. A low-rank LoRA adapter [8] is trained on this corpus to internalize section order, citation syntax, and a value-investor narrative voice that emphasizes business quality and moat reasoning.
2. **Preference Alignment via Direct Preference Optimisation (DPO) [15]** Senior analysts use an in-app A/B interface to compare draft sections with respect to qualitative depth, valuation realism, and clarity of moat articulation. We collect $\sim$10 000 pairwise judgments informed by a rater guideline ("Does the analysis identify and justify a moat type?"). Direct Preference Optimization (DPO) fine-tuning then aligns the LLM to these expert preferences, reinforcing contextual reasoning and nuanced business insights.
3. **Group-Relative RLHF [14]** We train a reward model on multi-rater (3-5 analysts) scores for clarity, analytical/qualitative depth, and factual accuracy. Use PPO [16], potentially with a group-relative reward function (inspired by GRPO [17]), to refine style, nuance, and complex value judgments. Explore sparse rewards from comparing historical report valuations with subsequent TSR ($\geq$12 months) for long-term calibration, carefully weighted against expert feedback.
4. **Client-Specific LoRA Adapters** To support bespoke requirements—such as a firm's proprietary style guide, unique valuation assumptions, or thematic overlays–we train lightweight LoRA adapters on a small, client-specific corpus of 100–500 annotated reports or internal style documents. Each adapter captures that client's preferred narrative style, citation conventions, and analytical framework without altering the core model's weights. At inference time, DeepFinAI can "hot-swap" these adapters, ensuring the generated reports adhere precisely to the client's house style and thematic priorities.

6. Evaluation and Benchmarks

To evaluate DeepFinAI's multi-agent, ReAct-based architecture, we conducted a benchmark study comparing its outputs against those from a simpler, single-agent, non-ReAct baseline. All evaluations relied on qualitative assessments from human financial experts.

6.1. Experimental Design

- **Report Generation Approaches**:
  1. **Single-Agent (No ReAct)**: Single LLM pass to generate the full report, with MCP tool use instructed but without iterative observation/reflection.
  2. **Multi-Agent (With ReAct)**: DeepFinAI's two-agent pipeline.
- **Company Sample**: We selected a few publicly traded companies across diverse sectors.
- **Human Expert Panel**: Three senior equity analysts (5+ years experience each), blinded to report generation method.
- **Evaluation Criteria**: (1) Section Completeness, (2) Qualitative Depth (business model, moats, competition), (3) Valuation Justification (assumptions, DCF connection), (4) Overall Usefulness for investment decisions.

6.2. Results

Table 1 summarizes the average scores and accompanying expert observations. As shown, the multi-agent (ReAct-based) approach substantially outperforms the single-agent baseline across all metrics, with experts noting much stronger section coverage, deeper qualitative analysis, clearer valuation rationale, and greater overall utility.

6.3. Discussion

The benchmark indicates that the multi-agent ReAct architecture significantly improves report quality across all dimensions compared to a single-pass baseline. This validates the current architectural design and underscores the potential of the planned post-training methods (Section 5.2) for further enhancing investment-grade insights.

References

[1] Anthropic. Introducing the model context protocol. https://www.anthropic.com/news/model-context-protocol, 2024. Accessed: 2025-05-30. 1, 2

[2] Warren E Buffett. The superinvestors of graham-and-doddsville. *Hermes*, 17, 1984. 1

Criterion	Single-Agent	Multi-Agent	Key Expert Observations
Section Completeness	1.2 / 5.0	4.0 / 5.0	Single-agent often omitted/superficially covered sections; Multi-agent consistently included all with citations.
Qualitative Depth	1.8 / 5.0	3.4 / 5.0	Multi-agent showed richer business/moat discussions due to iterative retrieval.
Valuation Justification	1.0 / 5.0	3.2 / 5.0	Multi-agent explicitly stated assumptions (discount rates, risk premiums); Single-agent often lacked input explanations.
Overall Usefulness	0.9 / 5.0	3.5 / 5.0	Multi-agent reports deemed a credible introduction; Single-agent outputs largely unusable without major revision.

Table 1. Average Expert Ratings and Key Observations for Single-Agent vs. Multi-Agent Reports

[3] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *arXiv preprint arXiv:2401.08281*, 2024. 3

[4] Financial Modeling Prep. Financial modeling prep api. https://financialmodelingprep.com/developer/docs/. Accessed: 2025-05-30. 1, 2

[5] Bruce C Greenwald, Judd Kahn, Erin Bellissimo, Mark A Cooper, and Tano Santos. *Value investing: from Graham to Buffett and beyond*. John Wiley & Sons, 2020. 1

[6] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. 1, 2

[7] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*, 2024. 1

[8] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022. 1, 4

[9] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024. 1

[10] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2019. 3

[11] Seth A. Klarman. *Margin of Safety: Risk-Averse Value Investing Strategies for the Thoughtful Investor*. HarperCollins, New York, 1991. 1

[12] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020. 2, 3

[13] Charles T Munger. *Poor Charlie's Almanack: The Wit and Wisdom of Charles T. Munger*. Donning Company, 2005. 1

[14] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022. 1, 4

[15] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023. 1, 4

[16] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 1, 4

[17] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. 4

[18] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. 1, 2

[19] U.S. Securities and Exchange Commission. Edgar — electronic data gathering, analysis, and retrieval. https://www.sec.gov/edgar/search/. Accessed: 2025-05-30. 1, 2

[20] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. 1, 2, 3