Project Report: Multi-Level AI Investment Report Generator

---

1. Project Overview

This project aims to build a lightweight, modular AI-powered system that generates detailed investment reports for selected companies based on financial data and textual knowledge retrieval. The system leverages a multi-level agent architecture to organize data fetching, analysis, and report generation. The primary goal is to provide rich, structured, and insightful financial summaries, comparisons, and visualizations, accessible through a user-friendly Streamlit web interface.

---

2. Tech Stack & Methodology

- **Programming Language:** Python 3.10+
- **AI Models:** Google Gemini API (Gemma 3 27B IT) for all natural language generation and summarization tasks, replacing Hugging Face models for improved performance and cost-efficiency.
- **Libraries & Tools:**
    - `streamlit` for the web UI
    - `yfinance` for fetching financial metrics
    - `plotly` for generating interactive financial charts
    - `requests` for API calls
    - `python-dotenv` for environment variable management
- **Architecture:** Modular agent-based design with dedicated components for planning, working, and executive control.

- **Data Retrieval:** Retrieval-Augmented Generation (RAG) pattern using custom knowledge retriever.

---

3. How Exactly Is Your Input Processed

- **User Input:** Users select companies and financial metrics via dropdowns on the Streamlit app.
- **Parsing:** Due to lightweight design constraints, NLP parsing of free-text input is removed; user selection drives report generation.
- **Data Fetching:** The Planning Agent uses `yfinance` to fetch live financial metrics and retrieves company text data via RAG.
- **AI Processing:** Textual summaries and detailed reports are generated via the Gemini API (Gemma model) through the Gemma client wrapper.
- **Report Assembly:** The Worker Agent compiles generated content, including executive summaries, detailed company profiles, financial charts, and interpretations, into a cohesive HTML report.

---

4. Structural Level

- **Frontend:** Streamlit UI providing multi-select dropdowns and report rendering.
- **Executive Agent:** Coordinates parsing, validation, and orchestrates planning and worker agents.
- **Planning Agent:** Handles financial data retrieval and textual summarization.
- **Worker Agent:** Performs language generation for detailed report sections and creates charts.
- **Utility Modules:** Encapsulate chart generation, API clients, and search fallback mechanisms.

- **Gemma Client:** Centralized wrapper managing all calls to the Gemini AI model.

---

5. Role of AI (Gemma Model) in This Project

- Replaces previous Hugging Face models to handle all text generation tasks:
    - Summarizing raw financial and company data
    - Generating executive summaries
    - Producing detailed company profiles
    - Creating comparative financial interpretations
- AI ensures natural, coherent, and comprehensive text outputs, enhancing report quality.
- Offloads complex NLP processing from local resources, enabling lightweight app deployment.

---

6. Development Tools

- **Python IDE:** VS Code / PyCharm
- **API Clients:** `google-genai` Python SDK for Gemini API access
- **Version Control:** Git / GitHub for source management
- **Environment Management:** `python-dotenv` to securely manage API keys
- **Testing:** Local Streamlit app for iterative UI and backend validation
- **Logging:** Python `logging` for error monitoring and debugging

---

7. Key Learnings

- Successfully integrated Google Gemini's Gemma model, improving generation quality and simplifying API usage.
- Learned to architect AI workflows with modular agents to separate concerns clearly.
- Discovered challenges in dependency conflicts, requiring removal of spaCy and simplification of NLP.
- Understood the importance of balancing feature richness with app lightweightness for user experience.
- Enhanced ability to dynamically generate and embed Plotly charts inside generated HTML reports.

---

8. Reflections & Takeaways

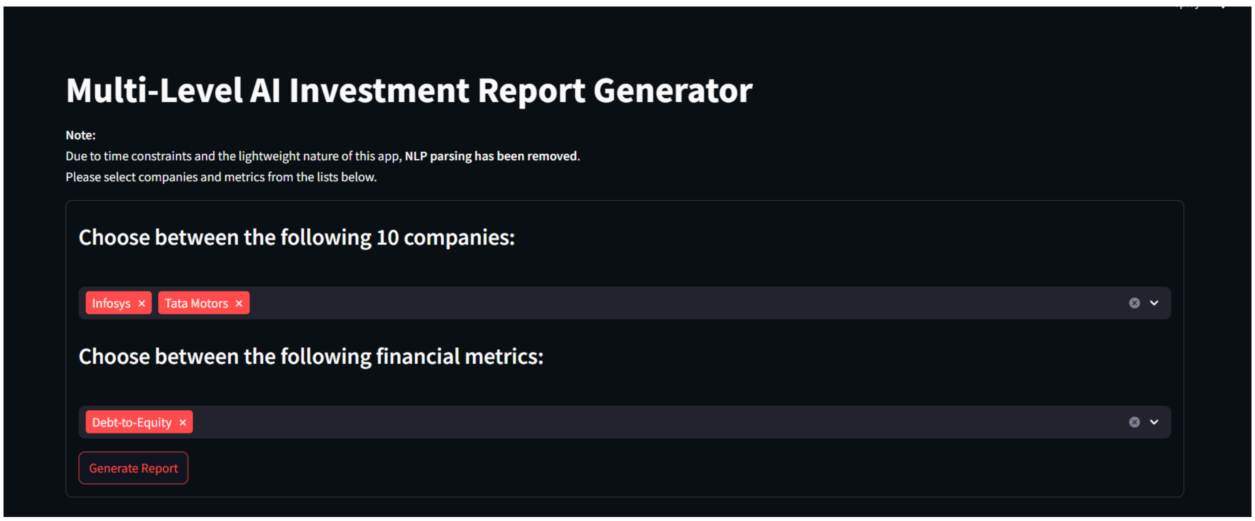
- Removing heavy NLP dependencies like spaCy in favor of lightweight parsing preserves system simplicity without sacrificing core functionality.
- Leveraging large AI models via cloud APIs reduces local compute needs but necessitates robust error handling and rate limiting.
- Modular design allowed seamless swapping of underlying AI models (Hugging Face → Gemma) with minimal disruption.
- Streamlit proves to be an excellent framework for rapid prototyping of interactive data-driven AI applications.
- Future improvements could include reintroducing NLP parsing via lightweight models or GPT itself, improving UI/UX, and enabling PDF export functionality.

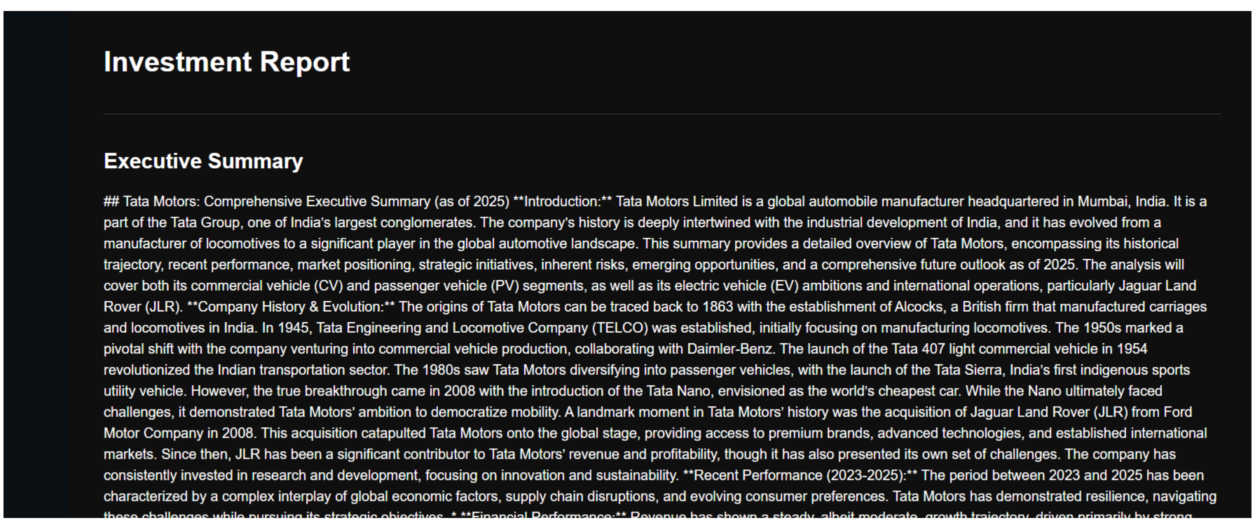
---

9. Conclusion

This project demonstrates how to build a practical, modular AI system for financial report generation by combining live financial data, retrieval-augmented textual knowledge, and state-of-the-art large language models accessible via API. The migration to Google Gemini's Gemma model has resulted in improved quality and maintainability. The lightweight Streamlit frontend ensures easy accessibility, while the backend's multi-agent structure provides flexibility and extensibility for future enhancements. Overall, the system offers a scalable foundation for AI-assisted financial analysis and reporting.

Snippets:





Project Note

Due to concurrent academic commitments, including an important exam during the project period, the time available for development and refinement was limited. Given additional time, further improvements could be made, such as enhancing the natural language processing capabilities, integrating more robust error handling, and expanding the range of financial data sources to improve report comprehensiveness. This project lays a solid foundation, and with more time, it could be extended to deliver even greater value and sophistication.