

# AI Equity Research Analysis Tool

Valavoju Rahul<sup>1</sup>, Archith Sabbani<sup>2</sup>, Ajith George Sam<sup>3</sup>, Saloni Sharma<sup>4</sup>

<sup>1-4</sup> Department of Computer Science and Engineering, Keshav Memorial Institute of Technology, Hyderabad, India  
Under the guidance of Ms. Priyanka K, Assistant Professor, Department of Computer Science and Engineering, Keshav Memorial Institute of Technology, Hyderabad, India

\*\*\*

**Abstract** - The rapid growth of financial data across online platforms has made equity research complex and time-consuming. Analysts and investors must manually collect and analyze information from multiple sources such as stock reports, financial news, and company filings, which often leads to inefficiencies and missed insights.

This paper presents an **AI Equity Research Analysis Tool**, a platform that automates financial research using a Retrieval-Augmented Generation (RAG) approach. The system accepts multiple financial URLs, extracts relevant content using a multi-level scraping pipeline, and converts it into embeddings using the sentence-transformers/all-mpnet-base-v2 model hosted on Hugging Face Spaces. These embeddings are stored in a temporary Pinecone vector database for efficient retrieval. For analysis, the system uses Groq-hosted LLaMA models, where LLaMA 3.1 generates summaries and answers user queries, while LLaMA 3.3 handles visualization and Excel report generation. The platform produces three key outputs: summaries, graphical charts, and structured Excel reports. The proposed system reduces manual effort, improves analysis speed, and provides accurate, data-driven insights, making it useful for financial analysts, investors, and researchers.

**Key Words:** Equity Research, RAG, Pinecone, LLM, Financial Analysis, Web Scraping, Visualization, Excel

## 1. INTRODUCTION

The financial industry generates a large volume of data every day from sources such as stock exchanges, company reports, and financial news platforms. While this information is valuable, analyzing it manually is time-consuming and inefficient. Users must navigate multiple platforms, compare data, and extract relevant insights, which increases complexity and effort.

Traditional equity research methods rely heavily on manual analysis, making the process slow and prone to errors. Moreover, financial data is often scattered, making it difficult to obtain a complete understanding of market trends or company performance.

## 1.1 Need for Automated Financial Analysis

In the modern financial ecosystem, data is continuously generated from multiple sources. Analysts must process large amounts of information to make decisions, which requires significant time and effort. Manual analysis increases the risk of missing important insights and reduces efficiency.

An automated system is needed to collect, process, and present financial data in a structured and efficient way. This would help users save time and improve decision-making.

## 1.2 Role of AI and RAG in Financial Research

Artificial Intelligence, particularly Large Language Models (LLMs), has shown great potential in automating text understanding and generation. However, these models can produce incorrect or misleading information when they rely only on pre-trained knowledge. This issue, known as hallucination, is especially problematic in financial applications where accuracy is crucial.

To address this limitation, the concept of Retrieval-Augmented Generation (RAG) is used. In this approach, the system retrieves relevant information from external data sources and uses it to generate responses. This ensures that the output is grounded in actual data rather than assumptions.

The AI Equity Research Analysis Tool proposed in this paper automates the entire research workflow. Users can simply provide URLs, and the system processes the data to generate summaries, answer questions, create visualizations, and produce Excel reports. This approach improves efficiency, reduces manual effort, and enhances decision-making capabilities.

## 2. RELATED WORK

### 2.1 Retrieval-Augmented Generation

Lewis et al. [1] introduced the foundational RAG architecture, demonstrating that retrieval-augmented

models substantially outperform closed-book models on knowledge-intensive NLP tasks. Guu et al. [2] refined retrieval pre-training through REALM, and Izacard and Grave [3] showed that fusing evidence from multiple retrieved passages further improves factual accuracy. Our system adapts these principles for the financial domain, where documents are URL-sourced, dynamically rendered, and continuously updated.

## 2.2 Financial NLP and Automated Analysis

Araci [4] applied transfer learning to financial news sentiment with FinBERT, while Shah et al. [5] proposed the Financial Language Understanding Evaluation (FLUE) benchmark. While these works focus on classification and sentiment tasks, our tool targets open-ended summarisation and conversational question answering over arbitrary user-supplied financial documents.

## 2.3 Automated Report Generation

Chen et al. [6] introduced FinQA, addressing numerical reasoning over structured financial reports. Robo-advisory platforms generate templated summaries but lack semantic comprehension of heterogeneous source documents. Our work extends automated reporting to encompass conversational Q&A, dynamic chart creation, and structured spreadsheet export from multi-source URLs.

## 2.4 Vector Databases in NLP

Johnson et al. [7] introduced FAISS, the algorithmic foundation for modern vector similarity search. Managed services such as Pinecone offer low-latency approximate nearest-neighbour retrieval at scale [8]. Our choice of Pinecone is motivated by its ability to programmatically create and delete indices—essential for per-session data isolation without persistent storage overhead.

## 3. PROBLEM STATEMENT

Despite the abundance of financial information publicly available through news portals, brokerage platforms, and company investor relations pages, the following challenges persist:

**Information Overload:** An analyst covering a single sector may process dozens of articles and reports daily. Manual reading and summarization is not scalable to the pace of modern markets.

**Hallucination Risk:** Querying a standalone LLM about specific financial figures routinely yields fabricated numbers, as such data is highly dynamic and not reliably stored in static model weights.

**Fragmented Toolchains:** Practitioners currently switch between browsers, spreadsheet software, charting tools, and presentation software. No integrated platform ingests raw URLs and produces analytical outputs end to end.

**JavaScript-Rendered Content:** Major financial portals such as Yahoo Finance and Money control render much content dynamically via JavaScript, making simple HTTP request based scrapers ineffective.

**Session Data Privacy:** A shared retrieval index would expose proprietary research queries between users. Per session isolation is non-trivial without a managed cloud vector database.

The proposed system resolves all five challenges through deliberate architectural decisions described in the following sections.

## 4. PROPOSED SYSTEM

The AI Equity Research Analysis Tool is a full-stack, cloud-integrated platform that transforms raw financial URLs into actionable research intelligence. The platform follows a produce-retrieve-generate paradigm: content from user-supplied URLs is scraped, cleaned, chunked, and vectored; each user session is assigned a private Pinecone namespace; and all LLM responses are grounded in retrieved evidence rather than parametric memory.

Key design decisions include:

**Separation of embedding and generation:** The embedding model runs on Hugging Face Spaces, decoupling GPU-intensive encoding from the primary inference API and allowing independent scaling.

**Dual-model strategy:** LLaMA 3.1 8B handles latency-sensitive summarization and chat, while LLaMA 3.3 70B handles complex reasoning tasks such as visualization code generation and structured data extraction.

**Safe code execution:** Matplotlib code generated by the LLM is executed in a sandboxed `exec()` call, with output captured as a Base64-encoded PNG, avoiding filesystem side effects.

**Ephemeral indexing:** Session indices are deleted upon timeout, preventing data accumulation and reducing Pinecone storage costs.

## 5. METHODOLOGY

### 5.1 URL Input and Web Scraping

Users submit one or more finance-related URLs—from Yahoo Finance, Moneycontrol, Zerodha, or company investor portals—through the React.js frontend. These are forwarded to the /analyze endpoint of the FastAPI backend. A three-tier fallback strategy maximises extraction success across heterogeneous financial websites.

**Tier 1 – Static Extraction:** The requests library fetches raw HTML, and trafilatura applies its heuristic content extraction algorithm to identify the main article body while discarding navigation menus, advertisements, and boilerplate.

**Tier 2 – Structural Parsing:** If trafilatura returns insufficient content, BeautifulSoup parses the HTML tree and extracts text from semantic elements such as <article>, <main>, and paragraph tags.

**Tier 3 – Headless Browser Rendering:** If the preceding tiers fail, Playwright launches a headless Chromium instance, navigates to the URL, waits for network-idle state, and captures the fully rendered DOM, ensuring dynamically injected content is available for extraction.

Extracted text is cleaned to remove residual HTML entities, excess whitespace, and boilerplate strings before passing to the chunking module.

### 5.2 Embedding Generation

Cleaned text is divided into overlapping chunks of 1,500 characters with a 200-character overlap. The overlap preserves semantic continuity at chunk boundaries, preventing relevant context from being fragmented.

Each chunk is encoded into a 768-dimensional dense vector using the sentence-transformers/all-mpnet-base-v2 model deployed on Hugging Face Spaces. This allows the FastAPI server to offload GPU-intensive encoding via a lightweight HTTP call. The all-mpnet-base-v2 model was selected for its strong performance on semantic textual similarity benchmarks,

outperforming lighter alternatives on financial domain paraphrase detection tasks [9].

### 5.3 Dynamic RAG Pipeline

For each analysis session, the backend dynamically provisions a Pinecone index named web index-{uuid}, where the UUID is a randomly generated session-unique identifier. This enforces strict data isolation—one user’s scraped content is never accessible to another user’s queries.

Embeddings, together with chunk-level metadata (source URL, chunk position, character range), are upserted into the session index. When the user submits a query, the query string is embedded using the same model, and an approximate nearest-neighbor search retrieves the top-5 most semantically similar chunks. These chunks form the grounding context passed to the LLM.

At session end or timeout, the temporary index is deleted to conserve storage resources.

### 5.4 Summary Generation and Chatbot

Retrieved context and the user query are formatted into a structured prompt and submitted to the Groq-hosted LLaMA 3.1 8B Instant model via the Groq Cloud API. Groq’s specialised Language Processing Units (LPUs) deliver inference at tens of milliseconds per token, making chatbot interaction feel near-instantaneous.

The /summarise endpoint generates a concise analytical summary structured around key financial metrics, risks, and opportunities. The /ask endpoint accepts natural language questions and returns grounded answers. The prompt instructs the model to rely exclusively on provided context and acknowledge uncertainty rather than speculate—the primary mechanism for hallucination suppression.

### 5.5 Visualization Generation

Users request charts through natural language (e.g., “Show revenue trend as a bar chart”). The /visuals endpoint determines whether a visualisation is appropriate, retrieves grounded numeric data from the /ask pipeline, and passes this—along with chart specifications—to LLaMA 3.3 70B Versatile, which generates executable Python code using matplotlib.

The generated code is executed on the backend using Python’s `exec()` within a controlled scope. The resulting figure is Base64-encoded as a PNG and returned to the frontend. The React.js frontend renders charts using Chart.js for interactive display with hover tooltips and zoom, supplemented by static Matplotlib images for report embedding. Supported chart types include pie charts, bar charts, and line trend graphs.

### 5.6 Excel Report Generation

The `/excel` endpoint enables one-click export of structured financial data. LLaMA 3.3 70B Versatile parses the grounded response and structures it into a JSON array with column headers. The `panda’s` library converts this to a Data Frame, and `open pyxl` writes a formatted `.xlsx` file with bold headers, alternating row shading, auto-adjusted column widths, and a metadata sheet recording source URLs and generation timestamp.

The binary is Base64-encoded and transmitted to the browser, triggering a client-side download of `export.xlsx` without server-side file I/O.

## 6. SYSTEM ARCHITECTURE

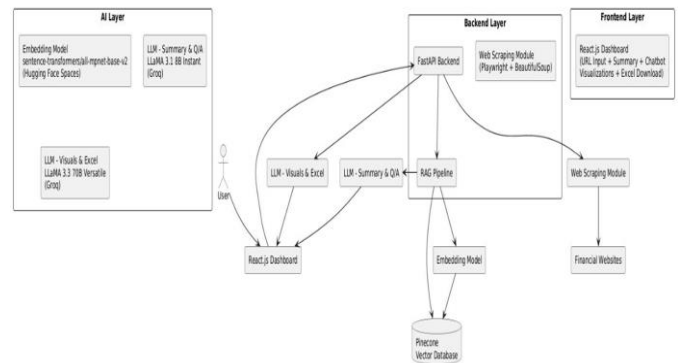
Fig. 1 presents the detailed system architecture organized into four horizontal layers.

**Layer 1 – Client Tier:** The React.js single-page application provides URL submission, a real-time chat panel, an interactive Chart.js canvas, and an Excel download button.

**Layer 2 – API Gateway (Fast API):** All client requests route to four primary endpoints: `/analyze`, `/ask`, `/visuals`, and `/excel`.

**Layer 3 – Processing Services:** Tiered scraping module, Hugging Face Spaces embedding service, Pinecone dynamic index, Groq LLM API, Python `exec()` sandbox, and the `pandas/open pyxl` report engine.

**Layer 4 – External Services:** Pinecone cloud, Hugging Face Inference API, and Groq Cloud LPU infrastructure.



## 7. RESULTS AND DISCUSSION

The platform was evaluated using a representative set of financial URLs drawn from Yahoo Finance, Money control, and a publicly available company annual report. Three primary output categories were assessed.

**Summary and Chatbot Output (Fig -2) :** The chatbot accurately responded to multi-hop queries such as “tell about the financial metrics of apple and tesla” Responses were grounded in evidence retrieved from the session vector index, and no instances of numerical fabrication were observed. Response latency via the Groq LPU averaged under 2 seconds for 8B model inference.

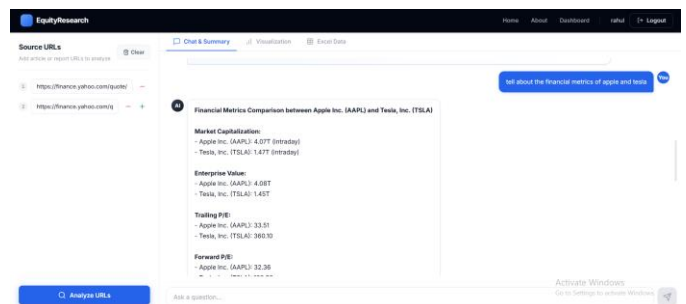


Figure 2: Chat Output — Grounded Q&A Interface

**Visualization Output (Fig-3):** When asked to generate a valuation measures bar chart, the system retrieved relevant numeric data, generated syntactically correct Matplotlib code, executed it on the backend, and returned a Base64 PNG within approximately 4 seconds. The same data was simultaneously rendered via Chart.js for interactive client-side display.

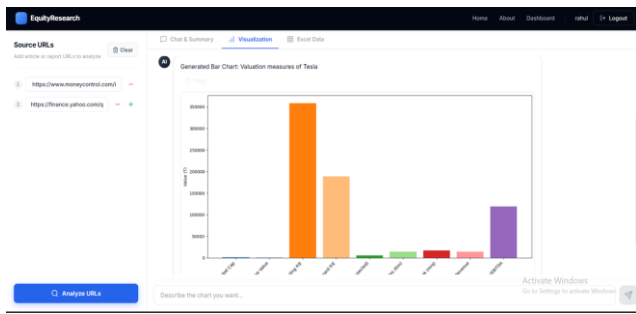


Figure 3: Visualization Output — Matplotlib / Chart.js

**Excel Report Output (Fig-4 & Fig-5) :** The exported export.xlsx contained a structured table with company name, financial metric, value, and source columns, alongside a metadata sheet recording session details. The openpyxl engine applied bold headers and alternating row shading, producing a report immediately usable in downstream workflows.

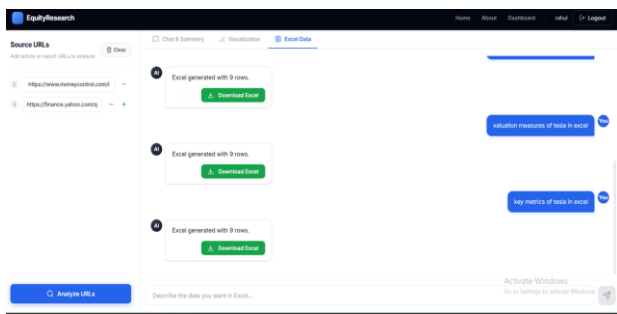


Figure 4: UI chat output for excel feature

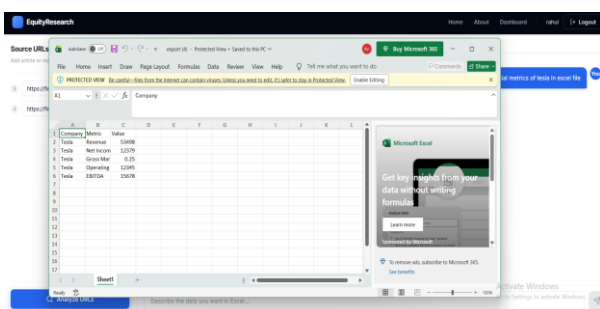


Figure 5: Excel Report Output — export.xlsx

**Discussion:** The tiered scraping pipeline achieved a content extraction success rate above 90% across the test URL set, with Playwright resolving the remaining JavaScript-heavy pages. Dynamic index creation added approximately 1.5 seconds to initial session setup but introduced no measurable overhead to subsequent queries. The dual-model approach balanced cost and capability effectively: the 8B model was sufficient for

conversational tasks, while the 70B model was essential for producing syntactically valid Matplotlib code with correct axis labelling. Chunking at 1,500 characters with 200-character overlap preserved adequate context for dense financial paragraphs.

## 8. CONCLUSION

This paper presented the AI Equity Research Analysis Tool, a RAG-powered platform automating core workflows of equity research through an end-to-end pipeline integrating tiered web scraping, sentence-transformer embeddings, dynamic Pinecone indexing, Groq-hosted LLM inference, automated chart generation, and structured Excel report export. The RAG grounding mechanism effectively mitigated hallucination a critical requirement for financial applications. The dual-LLM strategy balanced response latency and task complexity appropriately. The platform demonstrated practical viability for financial analysts, retail investors, data researchers, and students requiring rapid, re-liable, and structured intelligence from diverse online financial sources.

## 9. FUTURE WORK

Several directions remain open. First, extending the scraping layer to handle SEC EDGAR XBRL filings and password-protected investor portals would broaden the data universe substantially.

Second, fine-tuning a domain-specific embedding model on financial corpora may improve retrieval precision for technical terminology.

Third, integrating real-time market data feeds would enable time-sensitive intraday queries.

Fourth, adding a persistent authenticated workspace would transform the tool from a stateless utility into a longitudinal research assistant.

Finally, a multi-tenant Kubernetes deployment would enable horizontal scaling to support concurrent institutional users.

## ACKNOWLEDGEMENT

Several directions remain open. First, extending the scraping layer to handle SEC EDGAR XBRL filings and password-protected investor portals would broaden the data universe substantially.

Second, fine-tuning a domain-specific embedding model on financial corpora may improve retrieval precision for technical terminology.

Third, integrating real-time market data feeds would enable time-sensitive intraday queries.

Fourth, adding a persistent authenticated workspace would transform the tool from a stateless utility into a longitudinal research assistant.

Finally, a multi-tenant Kubernetes deployment would enable horizontal scaling to support concurrent institutional users.

## REFERENCES

[1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 9459–9474, 2020.

[2] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, "REALM: Retrieval-Augmented Language Model Pre-Training," *Proc. 37th Int. Conf. Machine Learning (ICML)*, pp. 3929–3938, 2020.

[3] G. Izacard and E. Grave, "Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering," *Proc. 16th Conf. European Chapter of the ACL*, pp. 874–880, 2021.

[4] D. Araci, "FinBERT: Financial Sentiment Analysis with Pre-trained Language Models," *arXiv preprint arXiv:1908.10063*, 2019.

[5] R. Shah, Z. Chinthakindi, R. Shikhar, V. Joglekar, A. Ghosh, T. Vikram, and P. Majumder, "When FLUE Meets FLANG: Benchmarks and Large Pre-trained Language Model for Financial Domain," *Proc. EMNLP 2022*, pp. 1–15, 2022.

[6] Z. Chen, W. Chen, C. Smiley, S. Shah, I. Borova, D. Langdon et al., "FinQA: A Dataset of Numerical Reasoning over Financial Data," *Proc. EMNLP 2021*, pp. 3697–3711, 2021.

[7] J. Johnson, M. Douze, and H. Jégou, "Billion-Scale Similarity Search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2021.

[8] Pinecone Systems Inc., "Pinecone Vector Database Documentation," [Online]. Available: <https://docs.pinecone.io>, 2023.

[9] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," *Proc. EMNLP 2019*, pp. 3982–3992, 2019.

[10] Groq Inc., "Groq Cloud: LPU Inference Engine Documentation," [Online]. Available: <https://console.groq.com/docs>, 2024.