# Case Study: FinSight Al

By Syed Armghan Ahmad - <a href="mailto:syedarmghanahmad.work@gmail.com">syedarmghanahmad.work@gmail.com</a>

Linkedin - GitHub - Repo for this Project

# **Overview**

As a self-taught developer passionate about AI and financial analysis, I developed **FinSight AI**, an AI-powered tool for analyzing financial documents like SEC 10-K filings. Leveraging LlamaIndex for hybrid search, GroqLLM for answer generation, and Plotly for dynamic visualizations, FinSight AI enables users to extract insights from complex financial data via natural language queries. Built with Python, Streamlit, and HuggingFace embeddings, this project showcases my ability to create an end-to-end AI application through self-learning, addressing real-world needs in financial research and investment.

# **Problem Statement**

Financial documents are dense, voluminous, and challenging to navigate, making it difficult for analysts, investors, and researchers to extract specific insights quickly. Traditional keyword searches often miss semantic context, while manual analysis is time-consuming. My goal was to create a tool that:

- Enables natural language queries to extract precise, context-aware insights.
- Combines semantic and keyword search for accurate document retrieval.
- Visualizes financial trends and metrics with dynamic graphs.
- Provides a user-friendly interface for interactive analysis.

# **Approach**

I designed a modular, full-stack application that integrates document processing, hybrid search, AI-driven querying, and interactive visualizations. The system processes uploaded PDFs, indexes them with LlamaIndex, retrieves relevant chunks using vector and BM25 search, and generates answers with GroqLLM, complemented by Plotly graphs.

### **Key Features**

- **Hybrid Search**: Combines vector embeddings (semantic) and BM25 (keyword) search to retrieve the most relevant document chunks.
- **Document Chunking**: Splits PDFs into configurable chunks (default: 512 characters) for efficient processing.
- **AI-Powered Answers**: Uses GroqLLM (Llama 3 70B) to generate concise, formatted responses, including tables and lists.
- **Dynamic Visualizations**: Generates Plotly graphs (line, bar, pie, scatter) based on query intent and extracted numerical data.
- Interactive Interface: Streamlit-based UI with PDF uploads, customizable settings (chunk size, temperature, max tokens), and animated styling.
- **Error Handling:** Includes API key validation and graceful error messages for robust operation.

### **Technical Implementation**

#### Frontend (Streamlit):

- Built an interactive UI with a wide layout, sidebar for PDF uploads, and sliders for chunk size, temperature, and max tokens.
- Applied custom CSS for fade-in animations, styled buttons, and inputs, enhancing user experience.
- Used st.text\_input for queries and st.plotly\_chart for graph rendering.

#### Document Processing:

- Extracted text from PDFs using PyPDF2 and split into chunks with LlamaIndex's SentenceSplitter.
- o Stored metadata (e.g., file name, document type) to maintain context.

### • Hybrid Search:

- Created a VectorStoreIndex with HuggingFace embeddings (BAAI/bge-small-en-v1.5) for semantic search.
- Used BM25Retriever for keyword-based search, merging top-3 results from both for optimal retrieval.

#### Al Engine:

- Integrated GroqLLM via the Groq API for answer generation, with a structured prompt ensuring formatted outputs.
- Extracted numerical data with regex to generate Plotly graphs based on query intent (e.g., trends → line charts).

#### • Core Technologies:

- LlamaIndex: Enabled hybrid search and RAG for document querying.
- o **GroqLLM**: Powered natural language responses with Llama 3 70B.
- o HuggingFace Embeddings: Generated compact, efficient embeddings.
- o **Plotly**: Provided interactive visualizations for financial metrics.
- o PyPDF2: Facilitated PDF text extraction.

### **Challenges and Solutions**

- Challenge: Learning LlamaIndex and hybrid search as a self-taught developer.
  - Solution: Studied LlamaIndex documentation and experimented with vector and BM25 retrievers to optimize retrieval.
- Challenge: Extracting meaningful numerical data from unstructured text.
  - Solution: Developed regex patterns to capture financial metrics, cleaning and converting them for visualization.
- Challenge: Designing dynamic visualizations for diverse queries.
  - Solution: Created a heuristic-based decide\_graph\_type function to match chart types to query intent.
- Challenge: Ensuring reliable LLM responses.
  - Solution: Crafted a detailed prompt with instructions for formatting, handling missing data, and suggesting graphs.

# **Impact**

FinSight AI streamlines financial document analysis, enabling users to extract insights quickly and visualize trends effectively. Key outcomes include:

- **Enhanced Efficiency**: Users can query complex documents in natural language, reducing analysis time.
- **Improved Accuracy**: Hybrid search ensures relevant document chunks are retrieved, minimizing irrelevant results.
- Actionable Insights: Dynamic graphs visualize financial metrics, aiding decisionmaking for investors and analysts.
- **Skill Development**: Through self-learning, I mastered LlamaIndex, hybrid search, and dynamic visualizations, preparing me for advanced AI projects.
- **Portfolio Strength**: The project showcases my ability to build industry-relevant Al tools, making it a standout addition to my portfolio.

# **Lessons Learned**

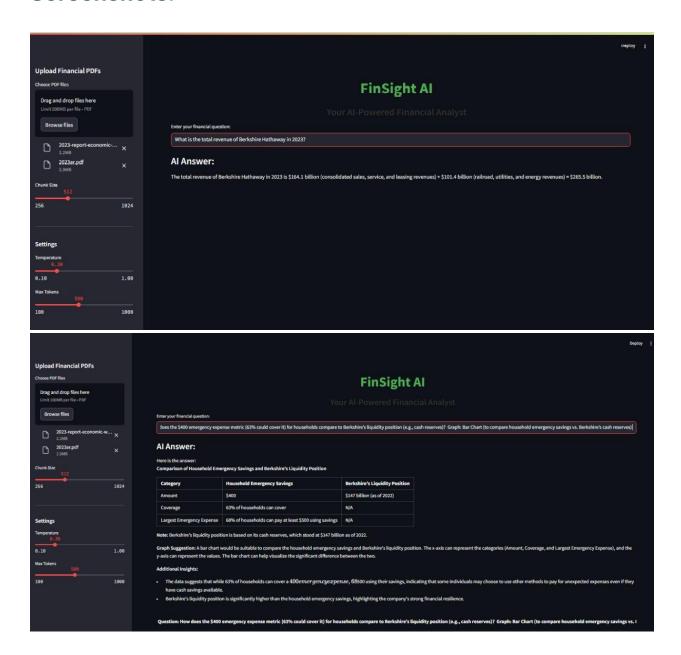
- **Hybrid Search**: Combining vector and BM25 search improves retrieval for diverse queries.
- **Prompt Engineering:** Structured prompts are critical for consistent LLM outputs.
- User-Centric Design: Animated, styled UI enhances engagement and usability.
- Data Extraction: Regex is powerful for parsing unstructured financial data.
- **Self-Learning**: Experimentation and documentation were key to mastering complex tools.

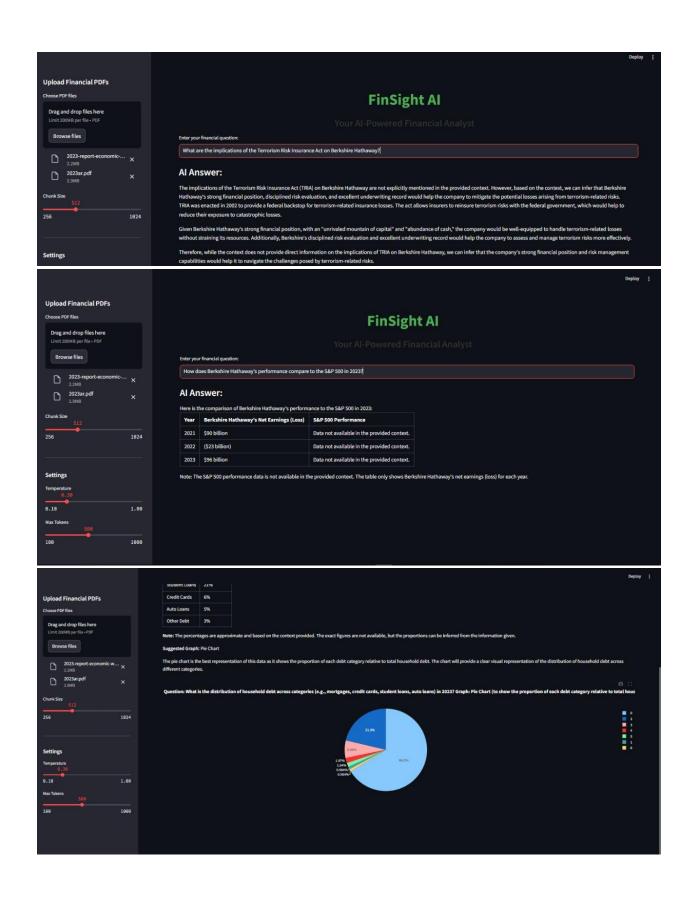
# **Future Enhancements**

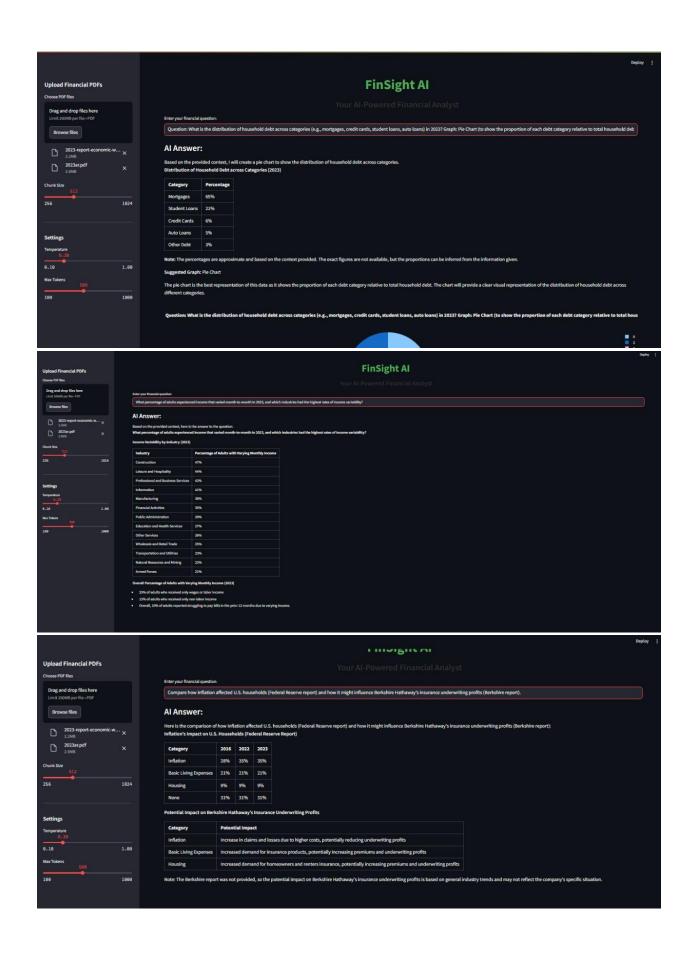
To productionize the application, I would:

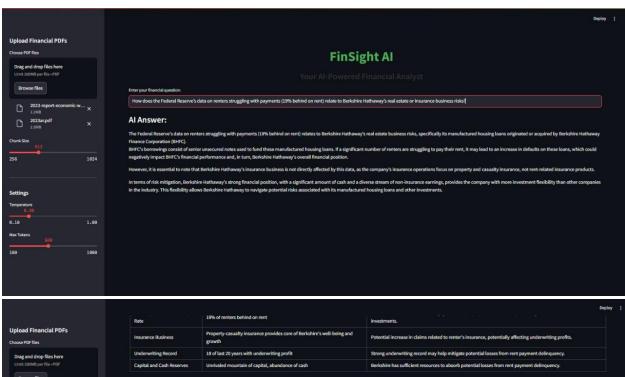
- Store processed documents in MongoDB and raw PDFs in AWS S3 for persistence.
- Replace in-memory VectorStoreIndex with Pinecone for scalable vector search.
- Implement a FastAPI backend with async endpoints for query handling and rate limiting.
- Add user authentication (OAuth2) and role-based access for security.
- Precompute graphs for common queries, storing in PostgreSQL for performance.

# **Screenshots:**

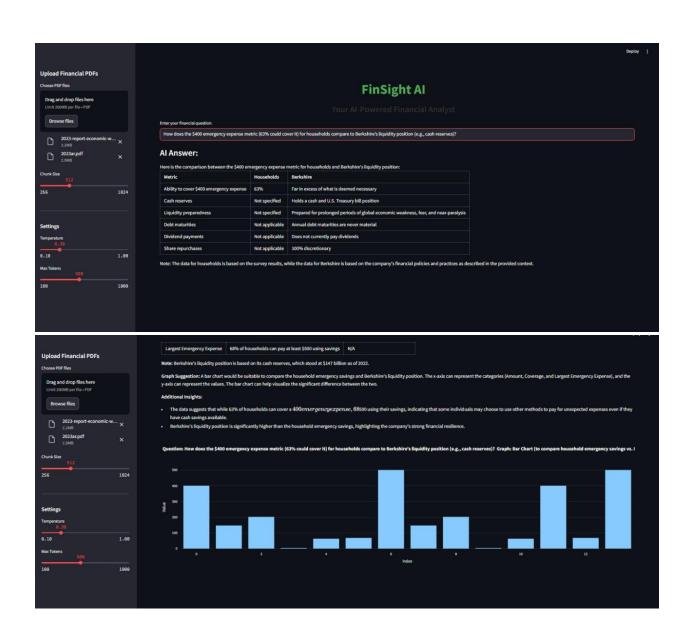


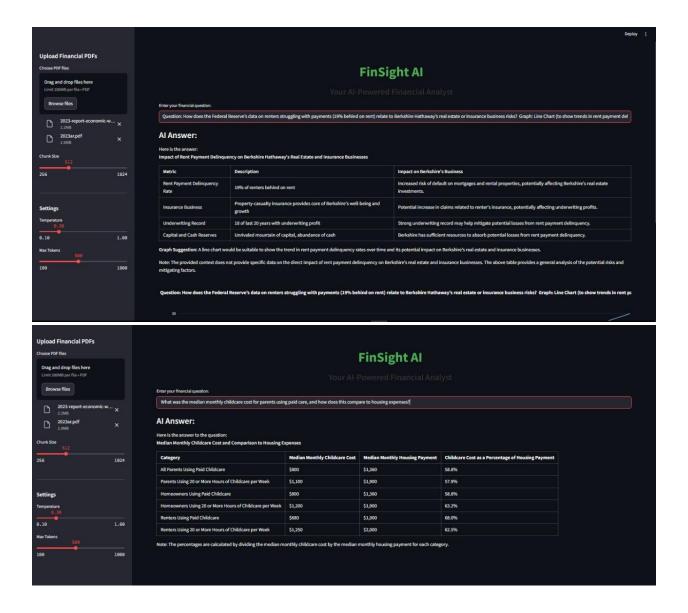












# **Conclusion**

FinSight AI is a testament to my self-learning journey and passion for AI-driven financial analysis. By building a tool that combines hybrid search, RAG, and dynamic visualizations, I addressed a real-world challenge in financial research. This project highlights my skills in document processing, AI integration, and user interface design, preparing me for impactful contributions in professional settings.