Documentation: AI/ML Content Engine

**1. Introduction**

This project aims to build a Content Engine that analyzes and compares three Form 10-K filings: **Google**, **Tesla**, and **Uber**. Using **Retrieval Augmented Generation (RAG)**, the system answers user queries, retrieves relevant information, and highlights differences between the documents.

**2. Objectives**

* Parse and extract content from multiple PDFs.
* Compare information across documents.
* Build a chatbot interface to retrieve insights interactively.
* Ensure local processing for data privacy.

**3. System Design**

The Content Engine is modular and scalable, consisting of:

* **PDF Parser**: Extracts text from documents.
* **Embedding Generator**: Encodes text into vectors.
* **Vector Store**: Manages embeddings and enables efficient search.
* **Query Engine**: Retrieves relevant content using embeddings.
* **LLM**: Provides contextual insights.
* **Streamlit UI**: Serves as the user interface.

**4. Implementation Details**

**4.1 Backend Framework**

I chose **LangChain** for its comprehensive tools and integration capabilities for Retrieval-Augmented Generation (RAG).

**4.2 Frontend Framework**

**Streamlit** was used to create a simple, interactive web application to handle user queries and display responses.

**4.3 Vector Store**

**FAISS** (Facebook AI Similarity Search) was used for:

* Storing document embeddings.
* Performing efficient similarity searches.

**4.4 Embedding Model**

The embedding model used is **SentenceTransformer (all-MiniLM-L6-v2)** for generating sentence-level embeddings locally.

**4.5 Local Language Model (LLM)**

The project uses **deepset/roberta-base-squad2**, a fine-tuned Question-Answering model, to generate responses from retrieved passages.

**5. Workflow**

1. **PDF Parsing**:
   * Extract text using PyPDF2.
   * Tokenize into sentences using NLTK for accurate processing.
2. **Embedding Generation**:
   * Generate sentence embeddings using SentenceTransformer.
3. **Embedding Storage**:
   * Store embeddings in FAISS for fast similarity-based retrieval.
4. **Query Handling**:
   * Accept a user query via Streamlit.
   * Generate the query embedding and retrieve relevant sentences.
5. **Response Generation**:
   * Use the QA model to process the query and retrieved context.
   * Display the result in Streamlit.

**6. Key Features**

* **Multi-Document Comparison**: Highlights similarities and differences between documents.
* **Interactive Chatbot**: Accepts natural language queries and provides insights.
* **Efficient Retrieval**: Uses FAISS for fast and accurate content retrieval.
* **Local Processing**: Ensures data privacy by running all operations locally.

**7. Setup Instructions**

**Prerequisites:**

* Python 3.8+
* Required libraries: streamlit, faiss, PyPDF2, nltk, transformers, sentence-transformers.

**Steps:**

1. Clone the GitHub repository:

git clone https://github.com/Mithraadharshini/Alemeno\_Backend\_LLM.git

cd alemeno.py

1. Install dependencies:

pip install -r requirements.txt

1. Download NLTK data:

import nltk

nltk.download('punkt')

1. Place the PDF files in the data folder:
   * goog-10-k-2023.pdf
   * tsla-20231231-gen.pdf
   * uber-10-k-2023.pdf
2. Run the application:

streamlit run alemeno.py

**8. Testing and Results**

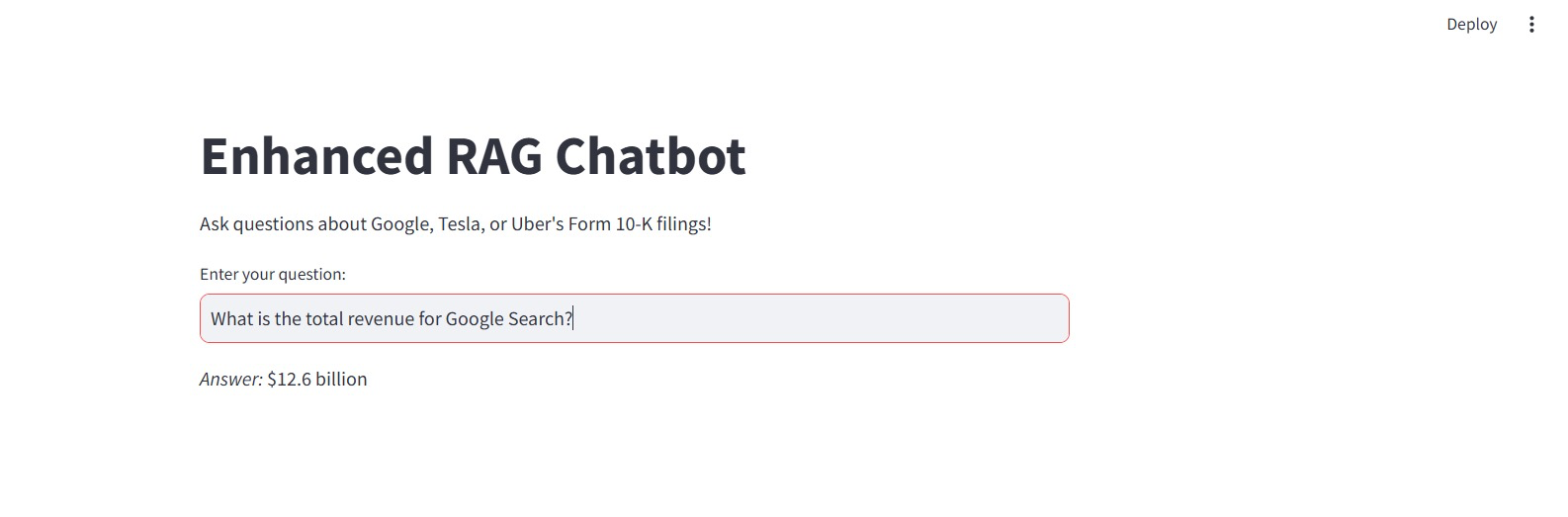
**Sample Questions Tested:**

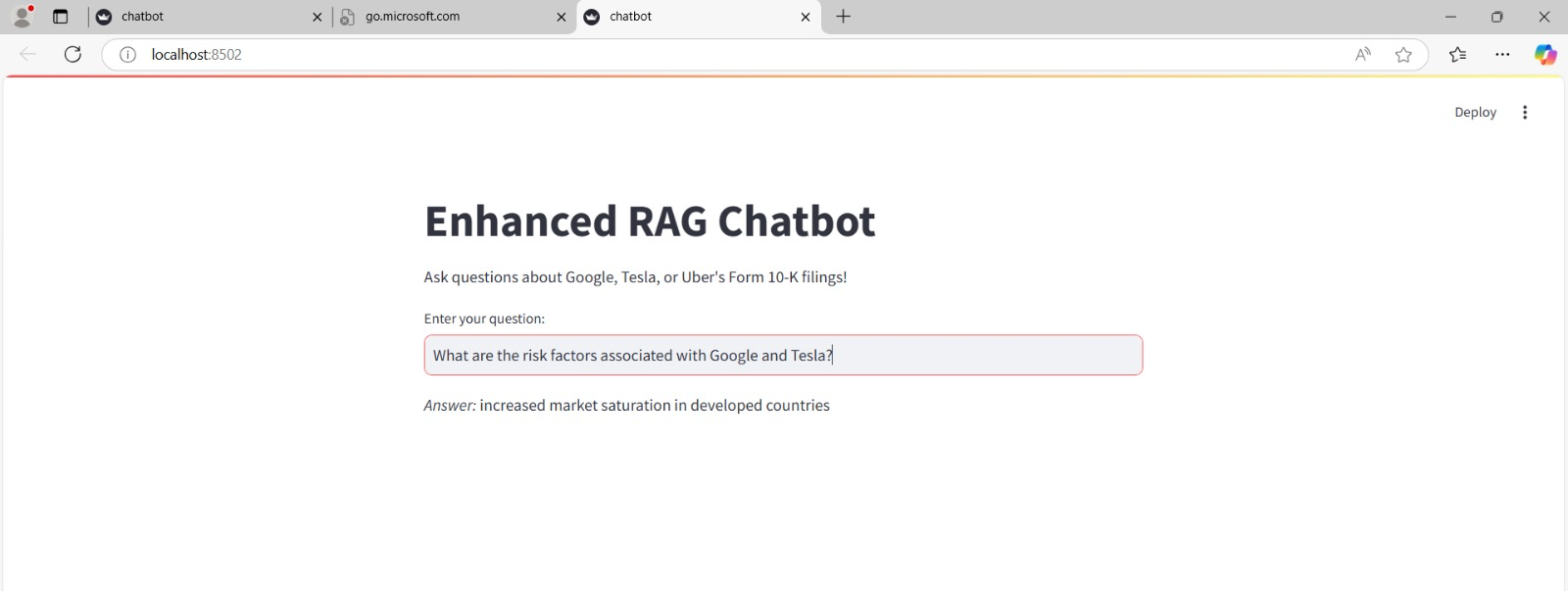
1. **What are the risk factors associated with Google and Tesla?**
   * Retrieved key risks from both documents and summarized differences.
2. **What is the total revenue for Google Search?**
   * Provided accurate figures from the filings.
3. **What are the differences in the business of Tesla and Uber?**
   * Highlighted distinctions between Tesla's automotive focus and Uber's ride-sharing services.

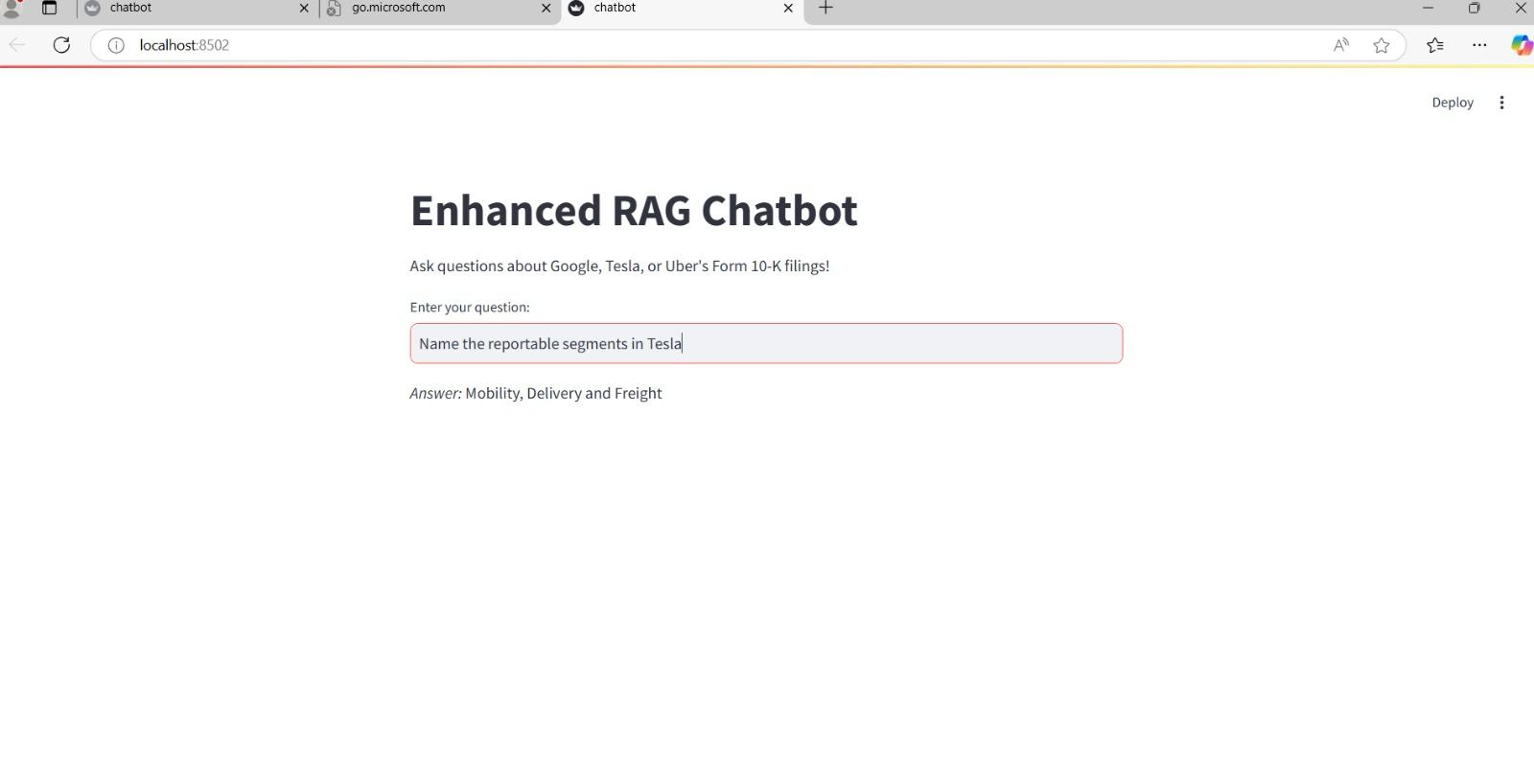
**Performance:**

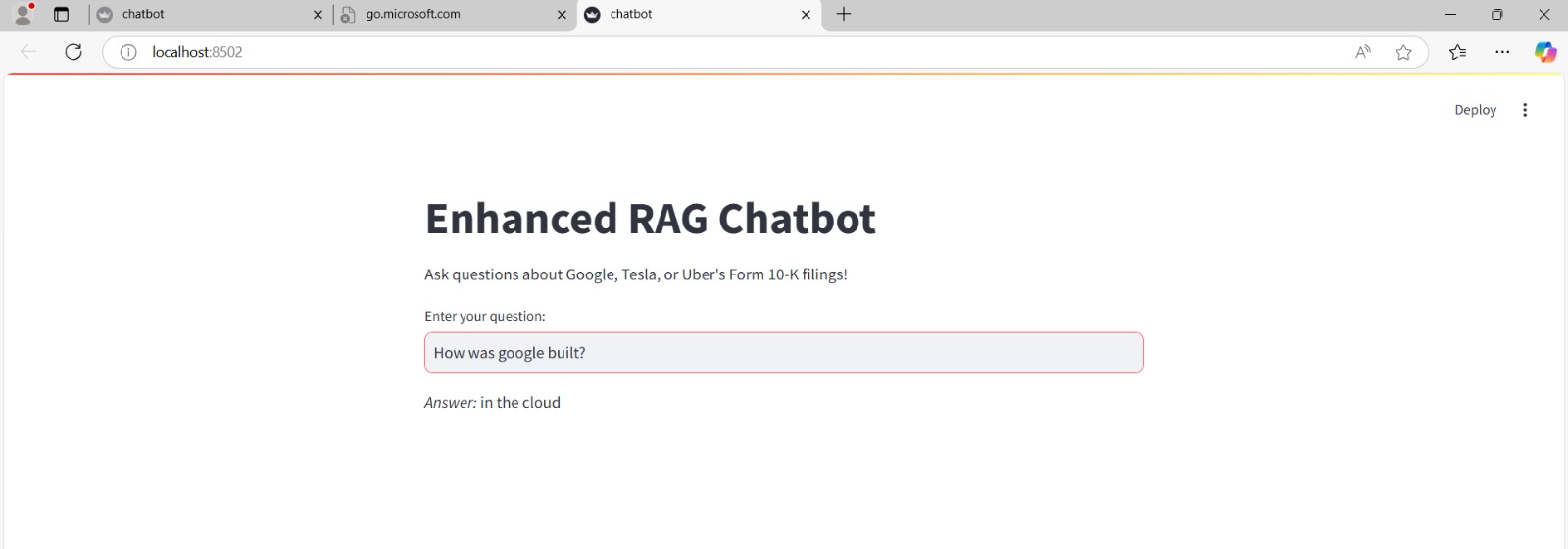
* Fast query response times.
* Accurate retrieval and summarization of content.

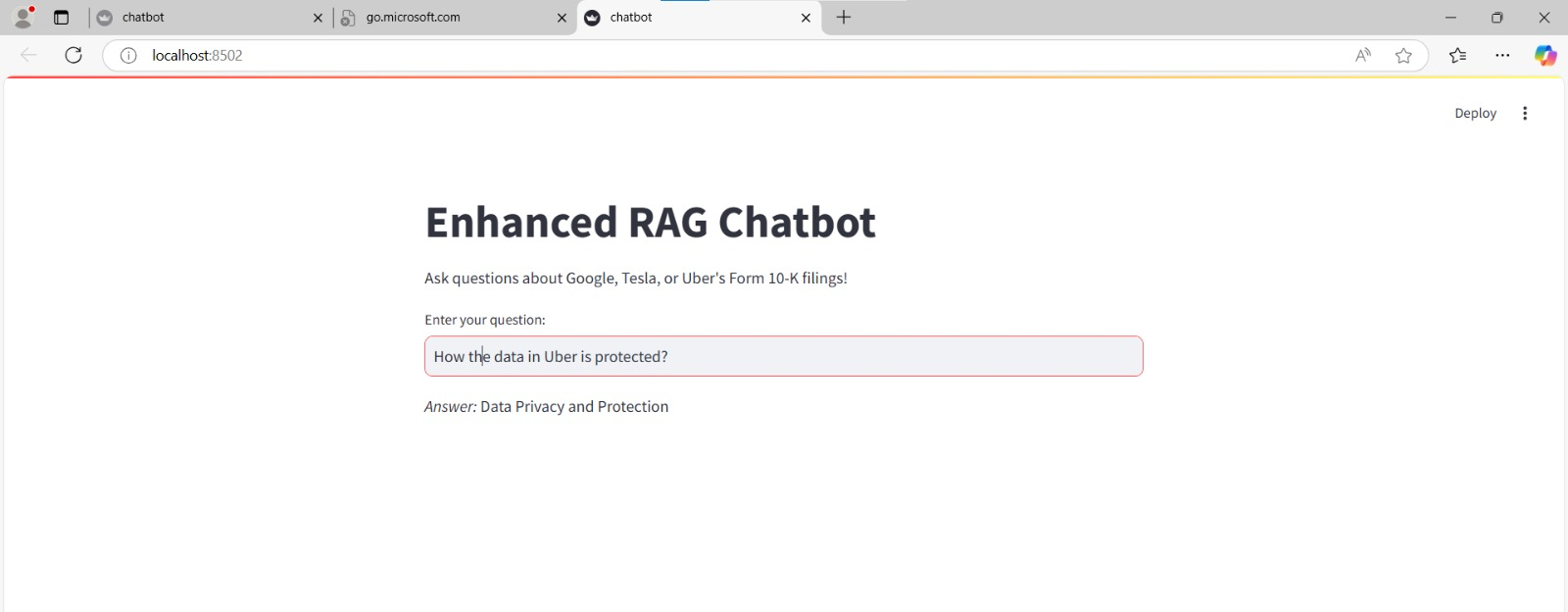
**9. Output**











**10. Conclusion**

The AI/ML Content Engine successfully retrieves, analyzes, and compares content from Form 10-K filings. The interactive chatbot simplifies complex document exploration, making it user-friendly for financial analysts and researchers.