**Multi model Rag – Chatbot**

**Data Science Domain:**

**Repository link:** [Gayathri05SK/Rag\_Chatbot](https://github.com/Gayathri05SK/Rag_Chatbot)

**Introduction:**

I have built a **Retrieval-Augmented Generation (RAG)-based chatbot** capable of processing and extracting information from various document types, including PDFs, DOCX, CSV files, images, and videos. The chatbot retrieves relevant content and generates accurate responses based on user queries without relying on the LangChain framework.

The chatbot is designed to enhance information retrieval by integrating multiple data formats into a cohesive knowledge base. By combining natural language processing (NLP), optical character recognition (OCR), speech-to-text conversion, and database management, the chatbot provides comprehensive responses with high accuracy and relevancy with memory Integration of past conversation to understand what the user wants and also it responds with the help of together api **Mistral AI Mixtral-8x7B-Instruct-v0.1.**

**Key Features:**

* **Multi-Modal Input Handling**: Supports text, images, videos, audio, and document formats (PDF, DOCX, CSV).
* **Memory Retention**: Uses SQLite storage for structured CSV data and maintains conversation history.
* **Intelligent Query Processing**: Uses Sentence Transformers for embedding and retrieval.
* **Text Extraction**: Utilizes OCR (EasyOCR) for images and videos.
* **Audio & Video Transcription**: Uses Whisper AI for transcribing speech from audio and video files.
* **Efficient RAG (Retrieval-Augmented Generation) Implementation**: Embeds and retrieves relevant information from stored knowledge.
* **Conversational AI with Together API**: Provides accurate and context-aware responses using **Mistral AI Mixtral-8x7B-Instruct-v0.1** model via the **Together API**.
* **Streamlit-based UI**: Interactive user interface for ease of use.

**Project Architecture:**

1. **Data Extraction**:
   * PDFs: Extracted using pdfplumber.
   * DOCX: Extracted using python-docx.
   * CSV: Loaded into SQLite for structured retrieval.
   * Images: Text extracted using EasyOCR.
   * Videos: Text extracted from frames using EasyOCR, and audio transcribed using Whisper AI.
   * Audio: Transcribed using Whisper AI.
2. **Embedding & Storage**:
   * Text data is preprocessed using spaCy (lemmatization, stopword removal).
   * Sentence embeddings generated using SentenceTransformers (‘all-MiniLM-L6-v2’).
   * SQLite is used for structured storage of CSV data.
   * Embedded text chunks are stored for retrieval.
3. **Retrieval & Processing**:
   * Query embeddings are compared with stored embeddings using cosine similarity.
   * The top-k relevant text chunks are retrieved.
   * Together API is used to generate responses based on retrieved context using **Mistral AI Mixtral-8x7B-Instruct-v0.1** model via the **Together API .**
4. **Response Generation**:
   * Chatbot formulates a context-aware response based on user query and retrieved information.
   * The chatbot retains conversation history for context continuity with the memory Integration of past conversation to understand what the user wants.

**Why Not LangChain?**

Instead of LangChain, a custom implementation was chosen for:

* **Flexibility**: More control over embeddings, retrieval, and chatbot flow.
* **Efficiency**: Optimized retrieval with direct access to stored embeddings.
* **Customization**: Integration of multiple models and APIs tailored to project needs.

**Handling Multiple Formats with libraries used:**

|  |  |  |
| --- | --- | --- |
| **Format** | **Process** | **Libraries used** |
| **Text Query** | Tokenized, preprocessed, embedded, stored in knowledge base. | nltk, spacy, sentence-transformers |
| **CSV Data** | Loaded into SQLite, converted into structured text, embedded, and retrieved as needed. | pandas, sqlite3, sentence-transformers |
| **PDF/DOCX** | Extracted using pdfplumber/python-docx, preprocessed, embedded, and retrieved. | pdfplumber, python-docx, nltk, spacy, sentence-transformers |
| **Image** | OCR (EasyOCR) extracts text, which is processed and embedded. | easyocr, nltk, spacy, sentence-transformers |
| **Video** | Frames analyzed using OCR, and audio transcribed via Whisper AI. | opencv, easyocr, faster-whisper, nltk, spacy, sentence-transformers |
| **Audio** | Transcribed using Whisper AI, processed, embedded, and retrieved. | faster-whisper, nltk, spacy, sentence-transformers |

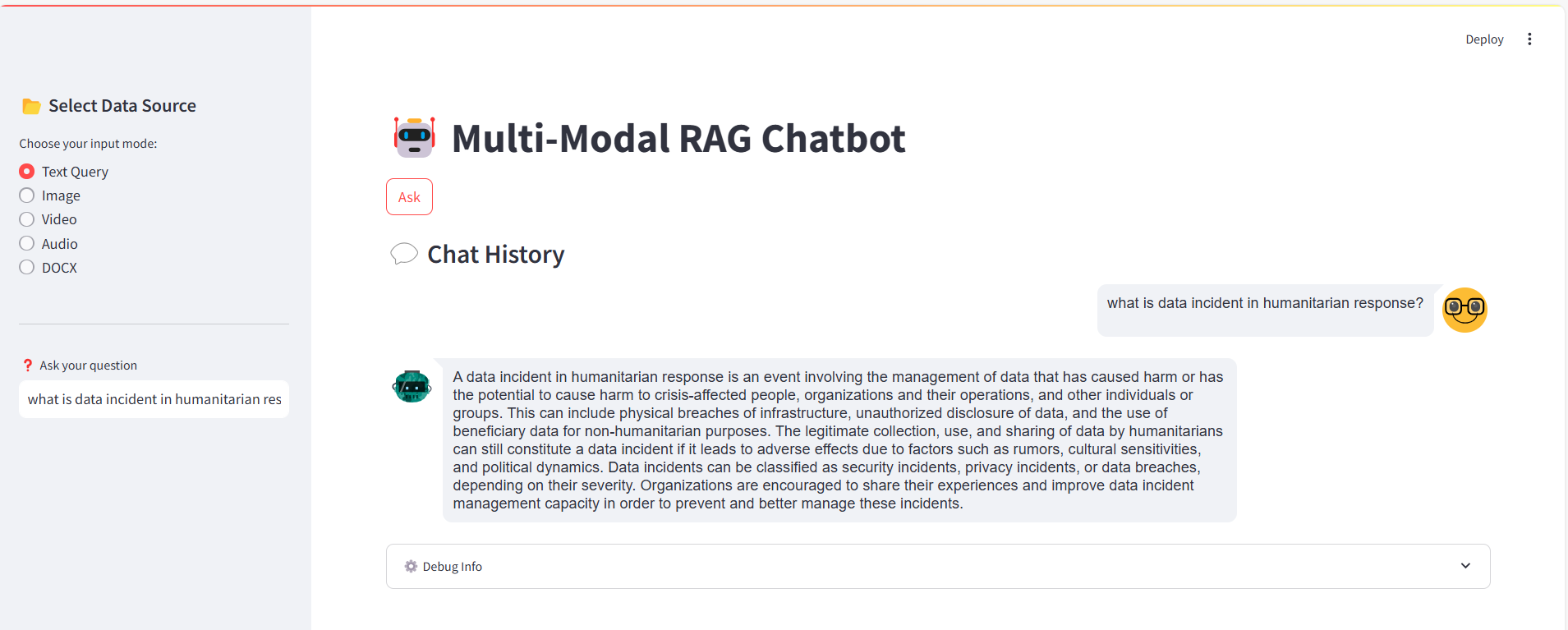
**Further Enhancements:**

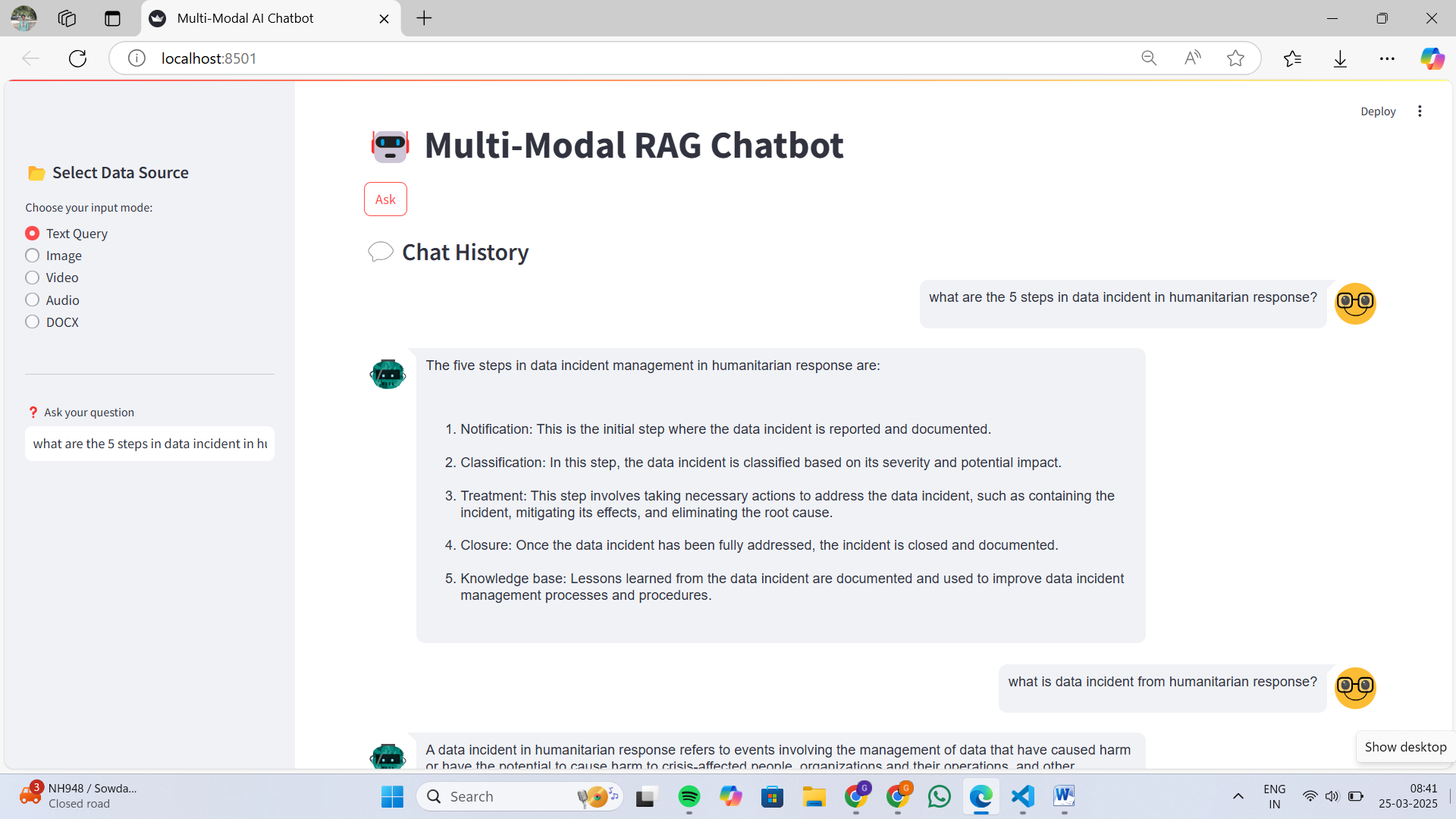
Since I have limited time,I can’t able to implement these concept that I have thought of implementing in this project.

* **Enhanced Memory Storage**: Improve long-term memory retention for user interactions but I can store the memory of the present session that is live.
* **Hybrid Cloud Deployment**: Optimize chatbot performance with cloud-based models.
* **Voice Interaction**: Enable real-time voice input and response.
* **Image generation :** Generate chart and images based on user query.

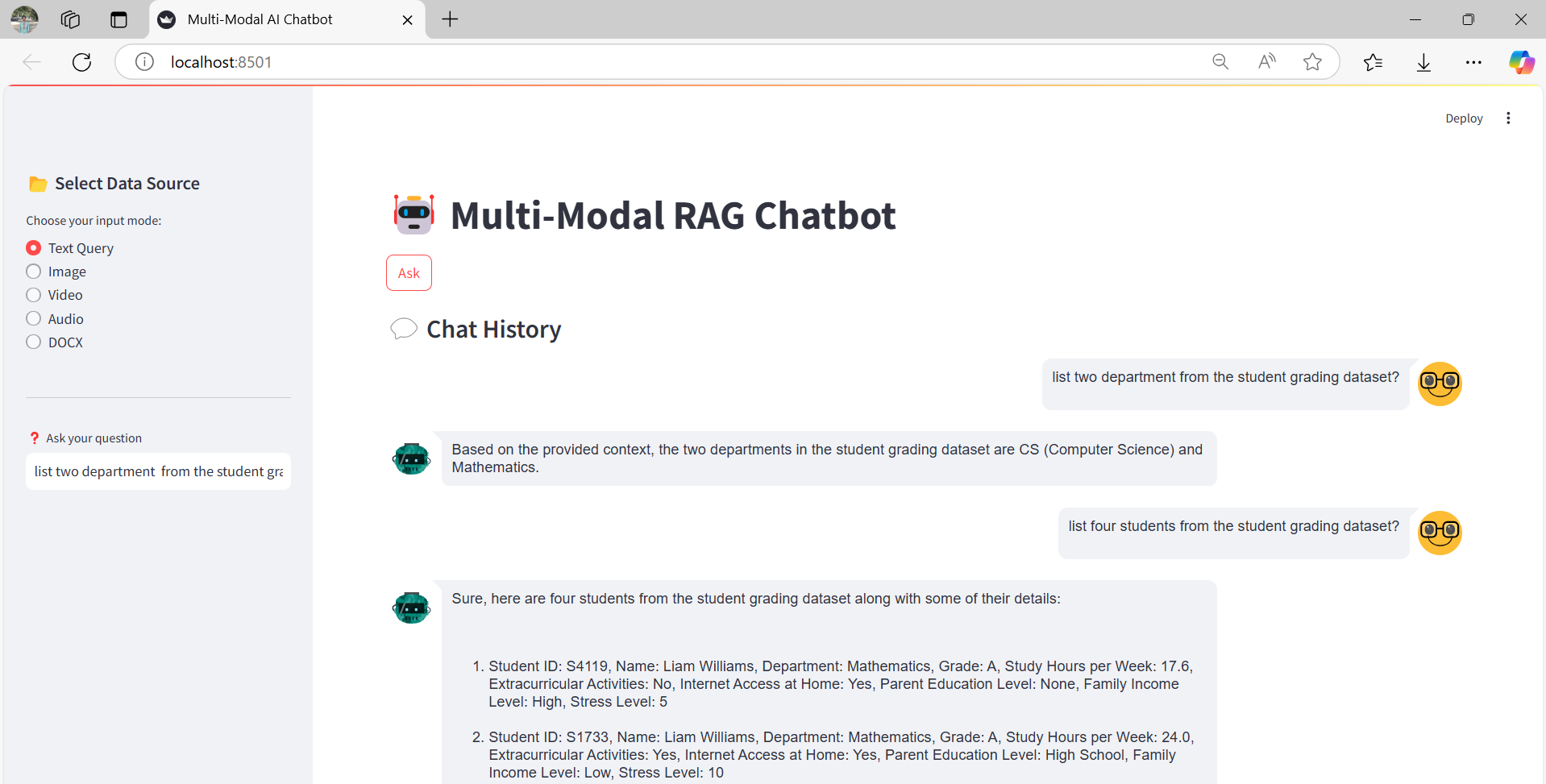
**Output:**

* **Pdf format:**

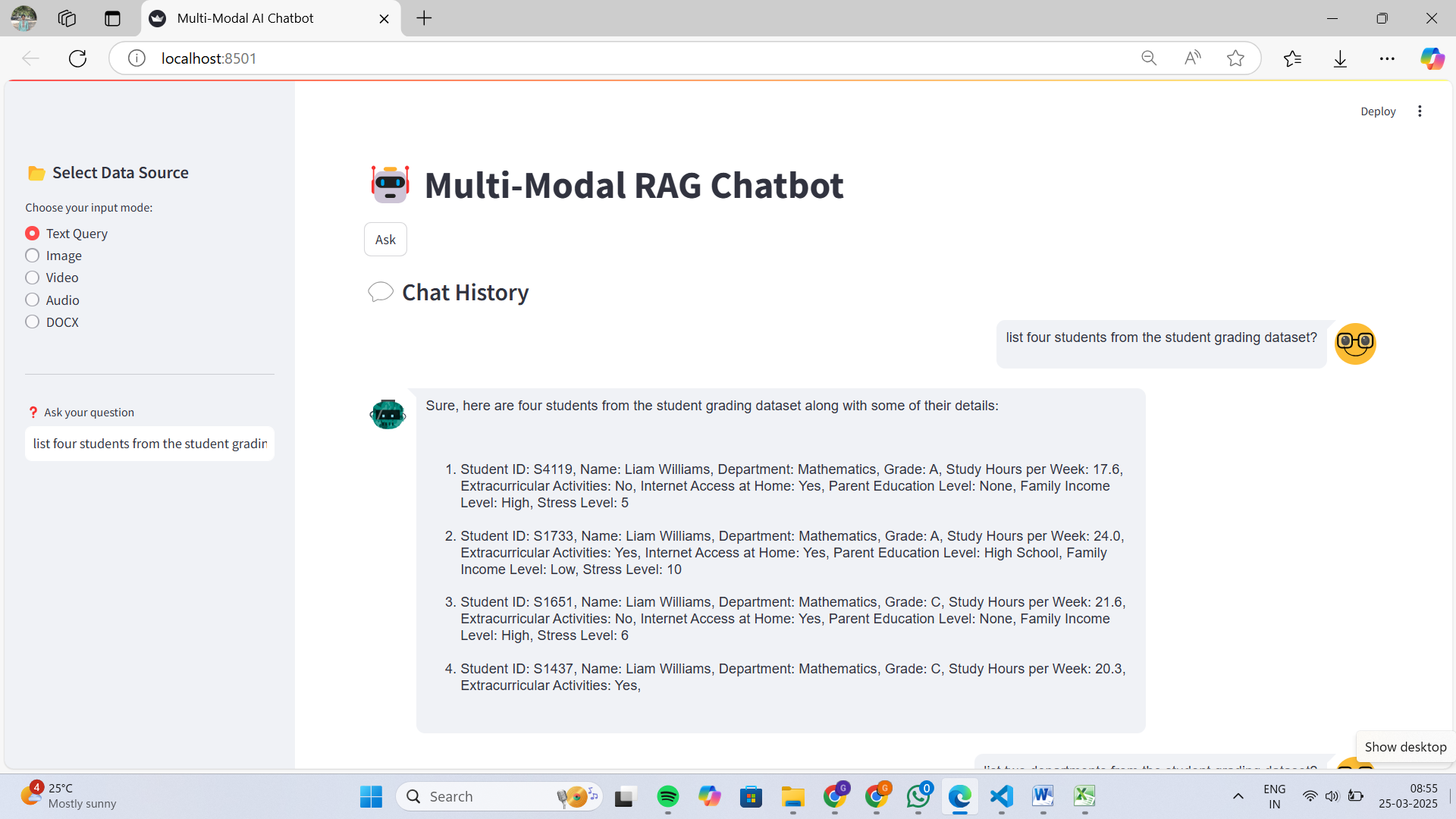




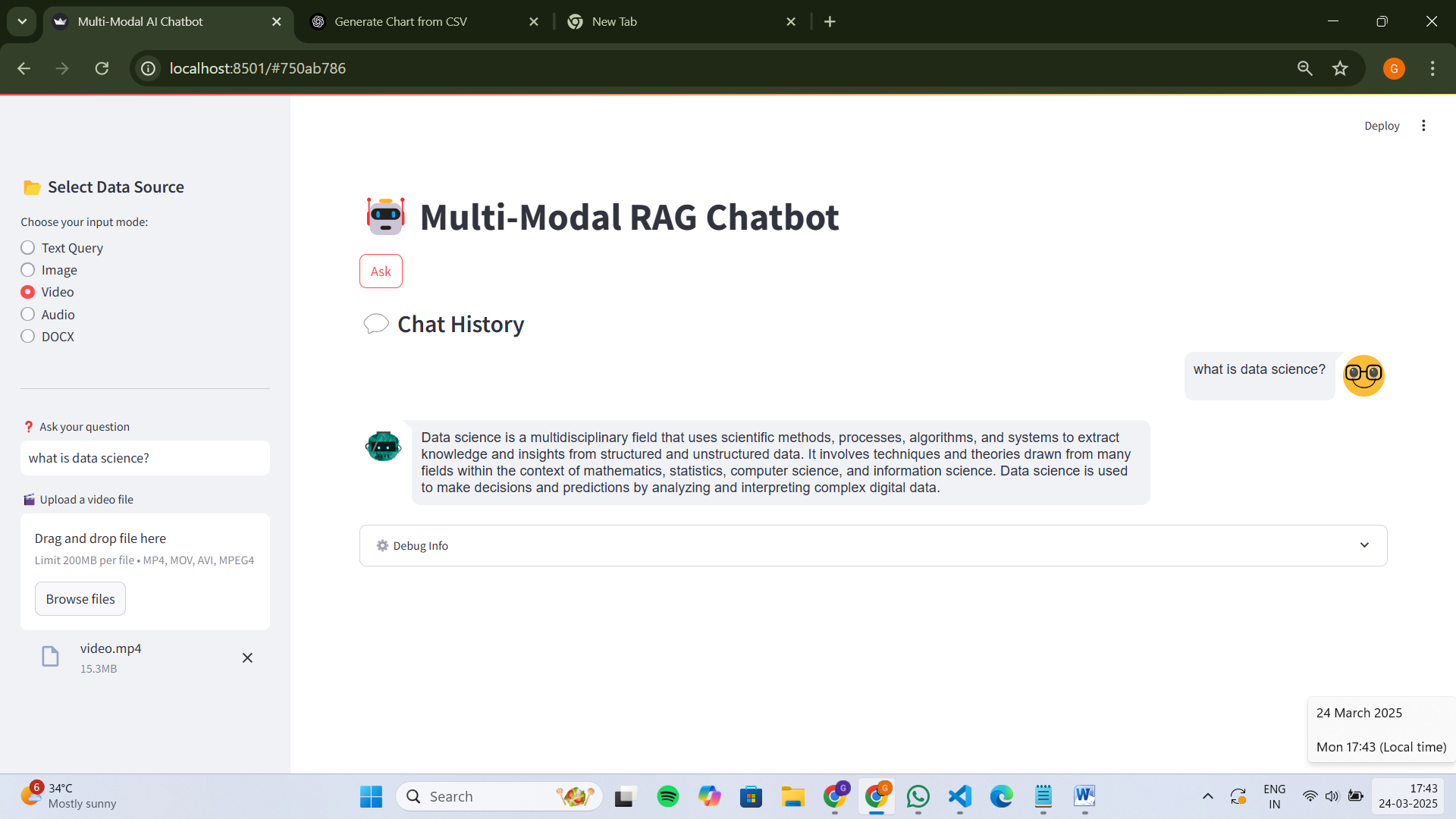
* **CSV format :**
* **With user query type with specific condition:**



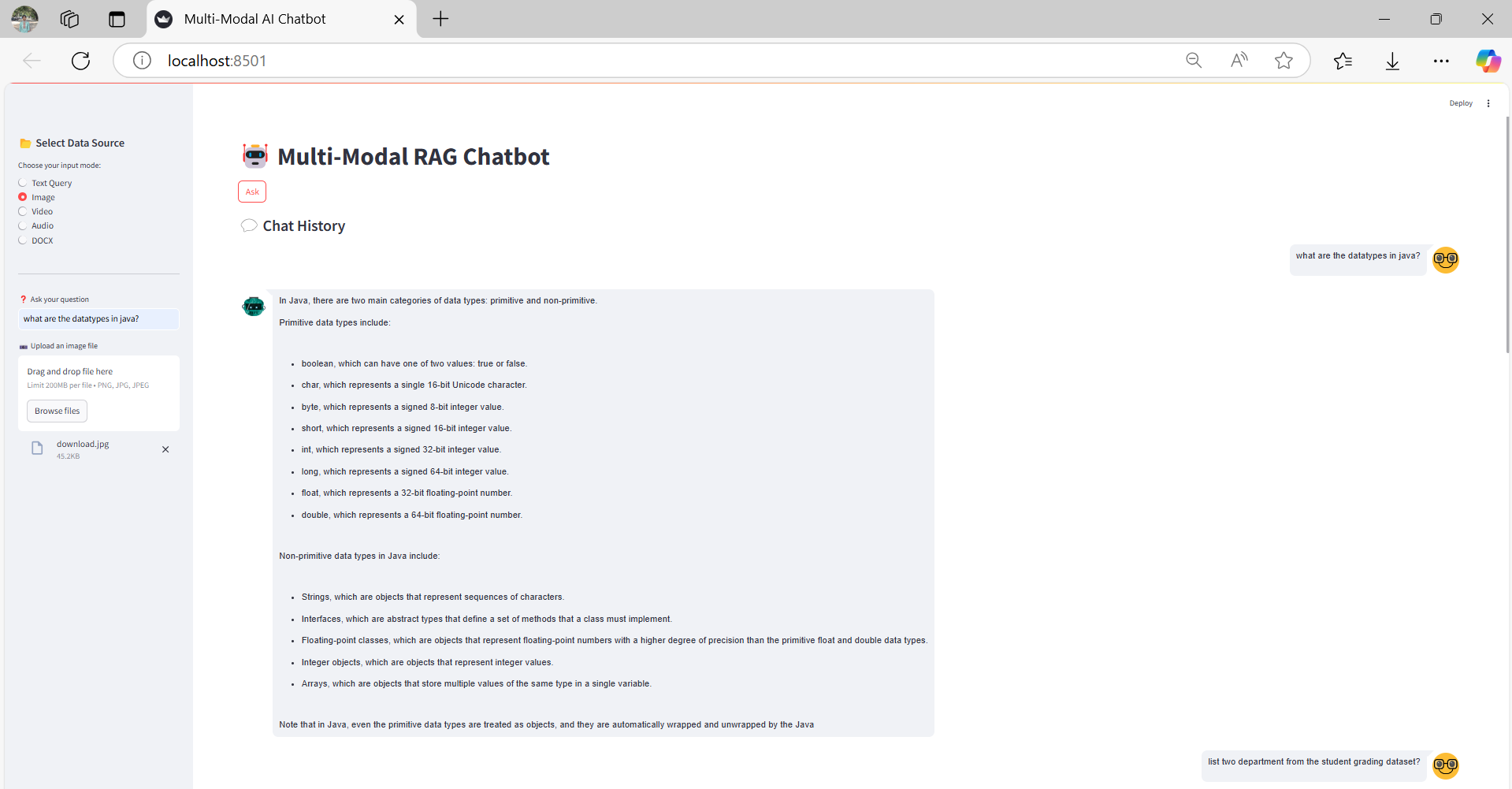
* **Query with the retrieval from SQLite storage:**



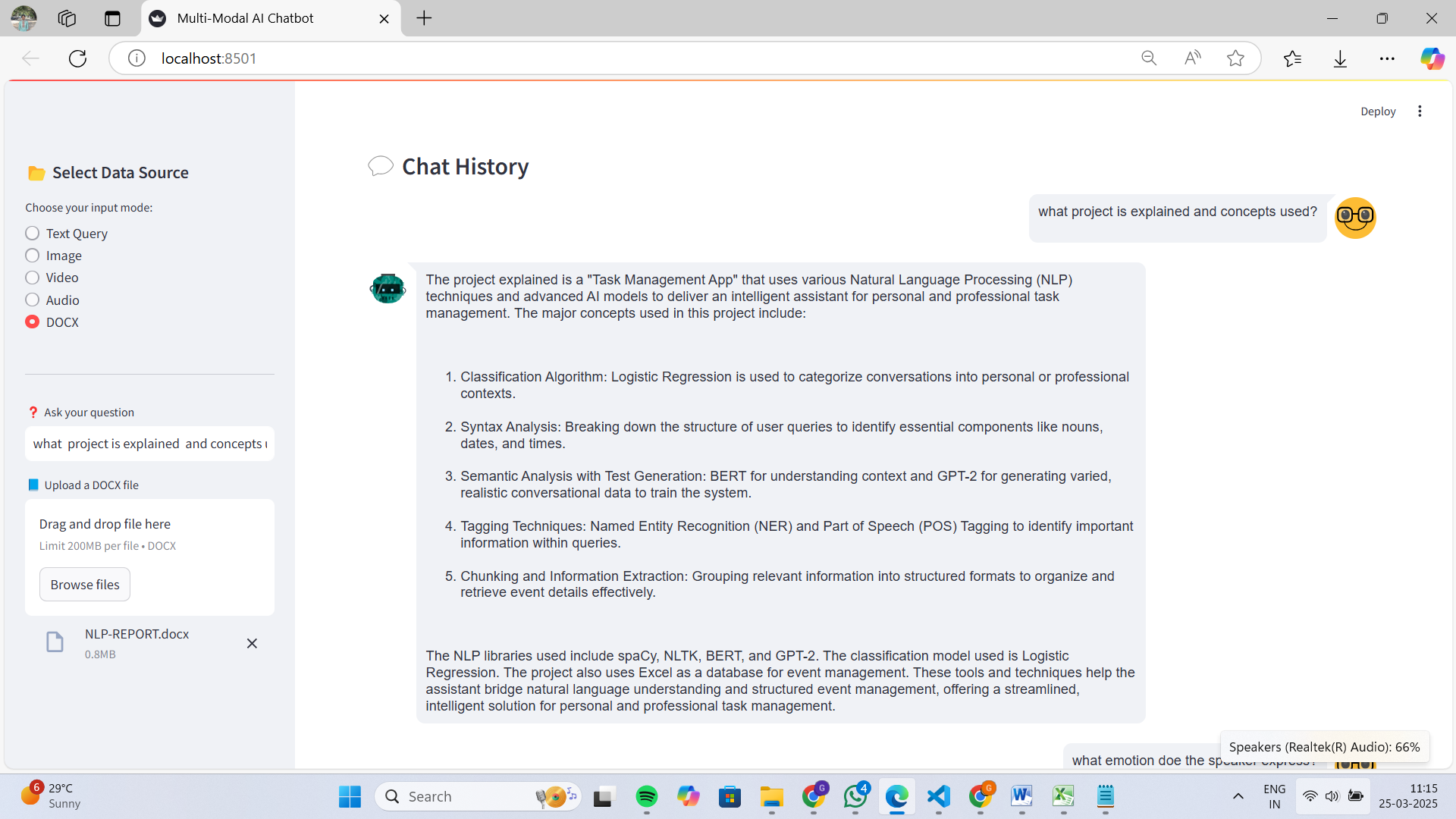
* **Video Format:**
* **Using speech to text it responds to the user query from the video attached:**



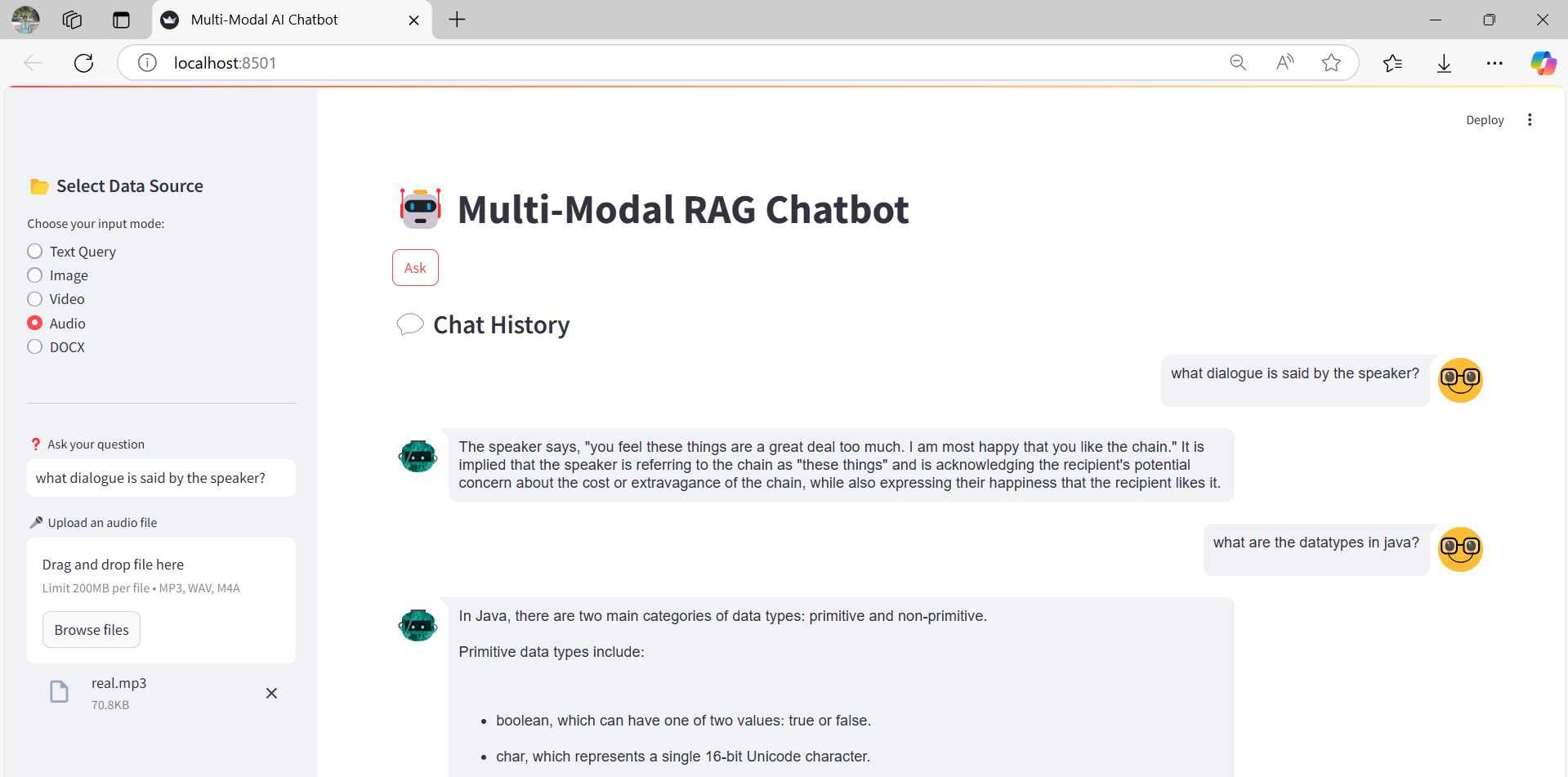
* **Image format:**
* **Extracts text from the image and responds :**



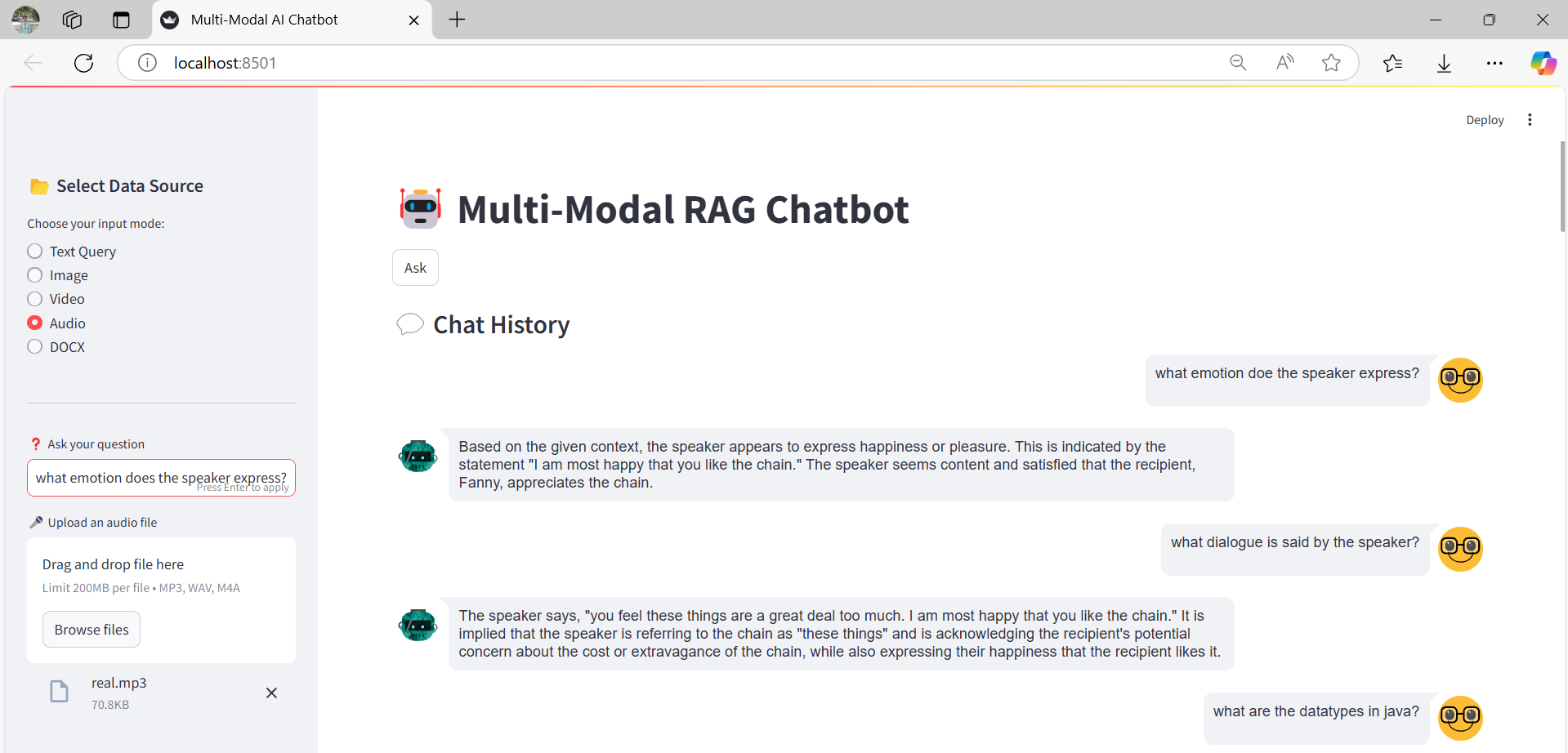
* **Docx format:**
* **Respond by extracting text from docx file:**



* **Audio format:**
* **Audio can be extracted from both audio files and video file format:**



* **Detects the emotion from that audio attached:**



* **Memory Integration of Past conversation to understand what the user wants:**

