**COIMBATORE INSTITUTE OF TECHNOLOGY**

BUCKMAN ROUND 1 SUBMISSIONS

****

**SUBMITTED BY:**

**NAME: MUTHUSINGAM N**

**ROLL NO:71762234028**

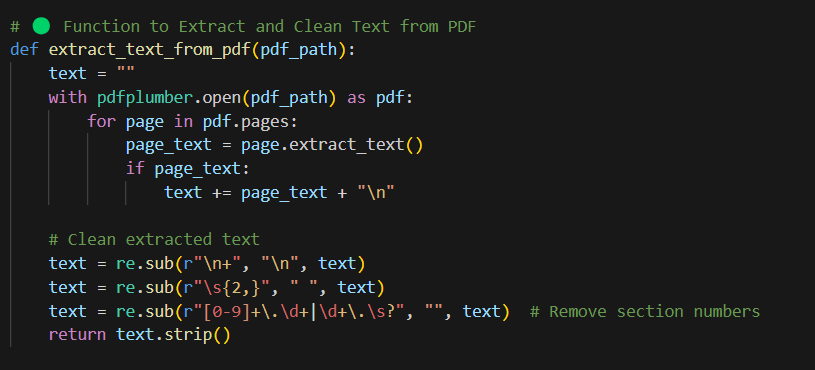
**CLASS: M.SC AIML**

**DATE:25.03.2025**

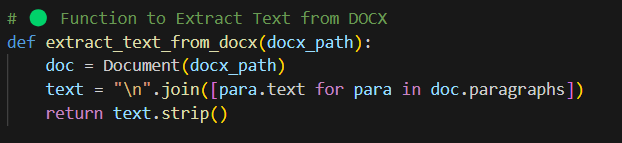
1. **OVERVIEW:**

Retrieval-Augmented Generation (RAG) is an advanced AI technique that combines retrieval-based search with generative AI models to improve response accuracy and relevance. Unlike traditional language models that rely solely on pretrained knowledge, RAG dynamically retrieves external documents to provide up-to-date and factual responses.By implementing RAG without LangChain, developers gain greater flexibility, customization, and efficiency in designing their retrieval and generation pipelines. This approach is particularly useful for applications like chatbots, document search systems, enterprise knowledge assistants, and legal or medical Q&A systems.

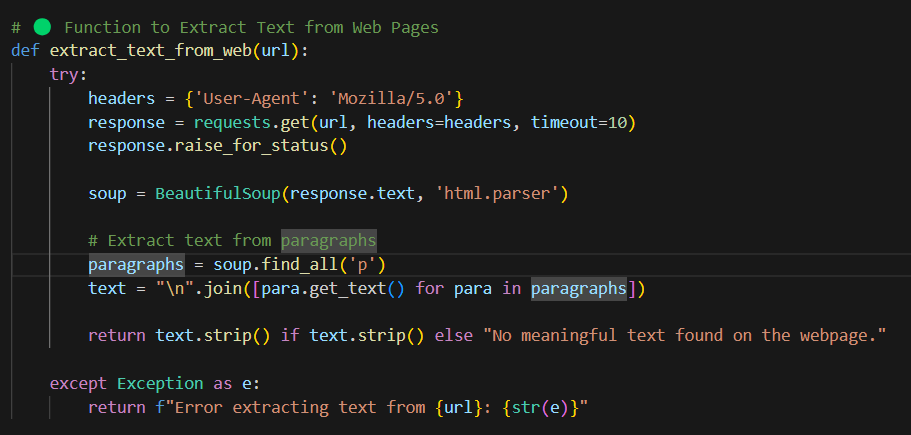
1. **SUPPORTED INPUT FORMATS AND EXTRACTION METHODS:**
   1. **PDF Processing:**

****

1. Open the PDF file using pdfplumber.
2. Extract text from each page and append it.
3. Clean the text by removing extra newlines, spaces, and section numbers.
4. Return the cleaned and formatted text.
   1. **DOCS:**

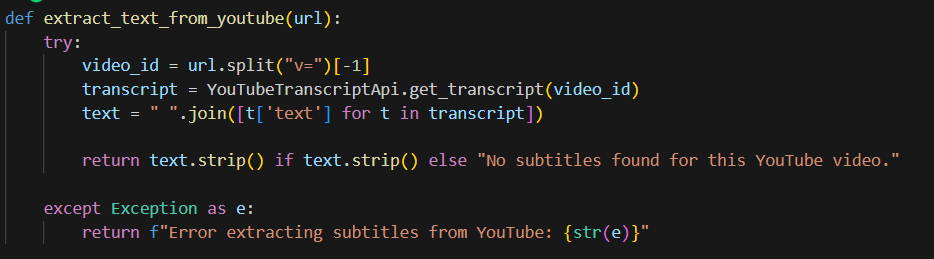
****

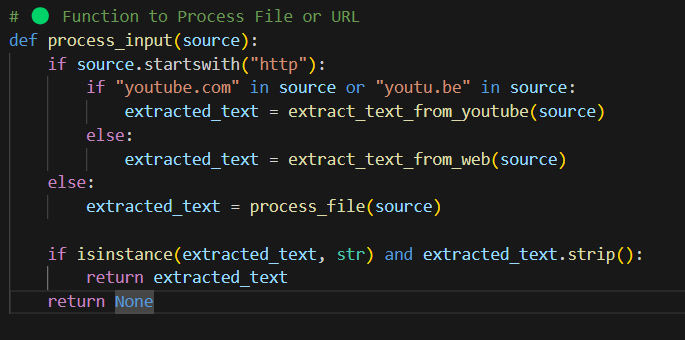
1. Open the DOCX file using Document(docx\_path).
2. Extract text from all paragraphs and join them with newlines.
3. Remove leading and trailing spaces.
4. Return the cleaned text.
   1. **WEBSITE URL:**



1. Send an HTTP GET request to the given URL with headers.
2. Parse the HTML response using BeautifulSoup.
3. Extract all paragraph (<p>) elements from the page.
4. Join the text content of paragraphs with newlines.
5. Return the cleaned text or an error message if extraction fails.

**2.4 YOUTUBE LINK WITH AND WITHOUT SUBTITLE:**



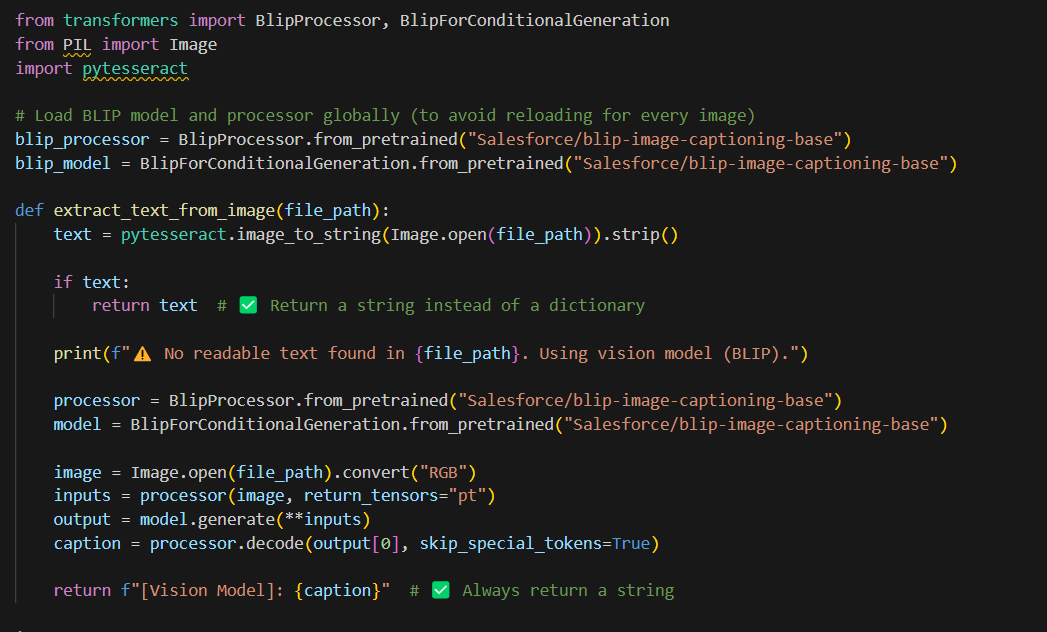


1. **Process Input (File or URL)**

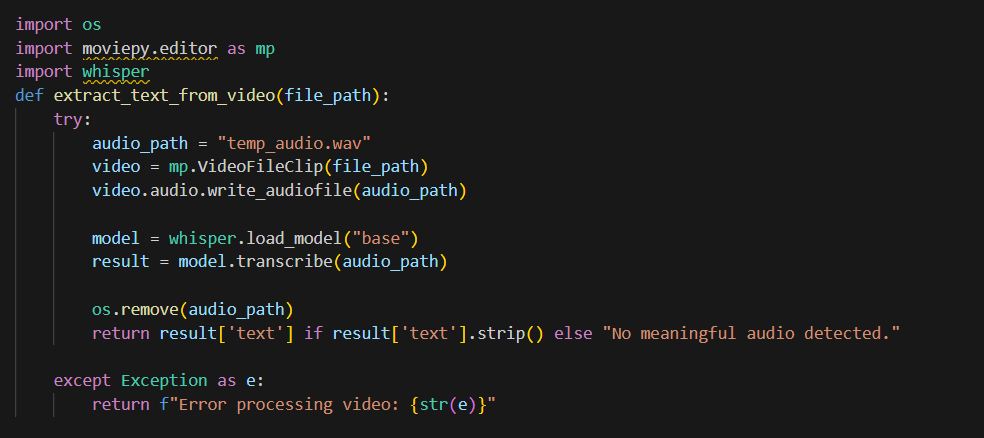
* The process\_input(source) function checks if the input is a URL (starts with "http").
* If the URL is from YouTube, it extracts text using extract\_text\_from\_youtube().
* Otherwise, it processes the web content using extract\_text\_from\_web().
* If the input is not a URL, it is processed as a file using process\_file().
* If the extracted text is valid (not empty), it returns the text; otherwise, it returns None.

1. **Extract Text from YouTube Videos**

* The extract\_text\_from\_youtube(url) function extracts subtitles from a YouTube video.
* It retrieves the video\_id from the URL and fetches the transcript using YouTubeTranscriptApi.get\_transcript().
* It joins the transcript text into a single string and returns it.
* If no subtitles are found or an error occurs, it returns an error message.
  1. **IMAGE:**

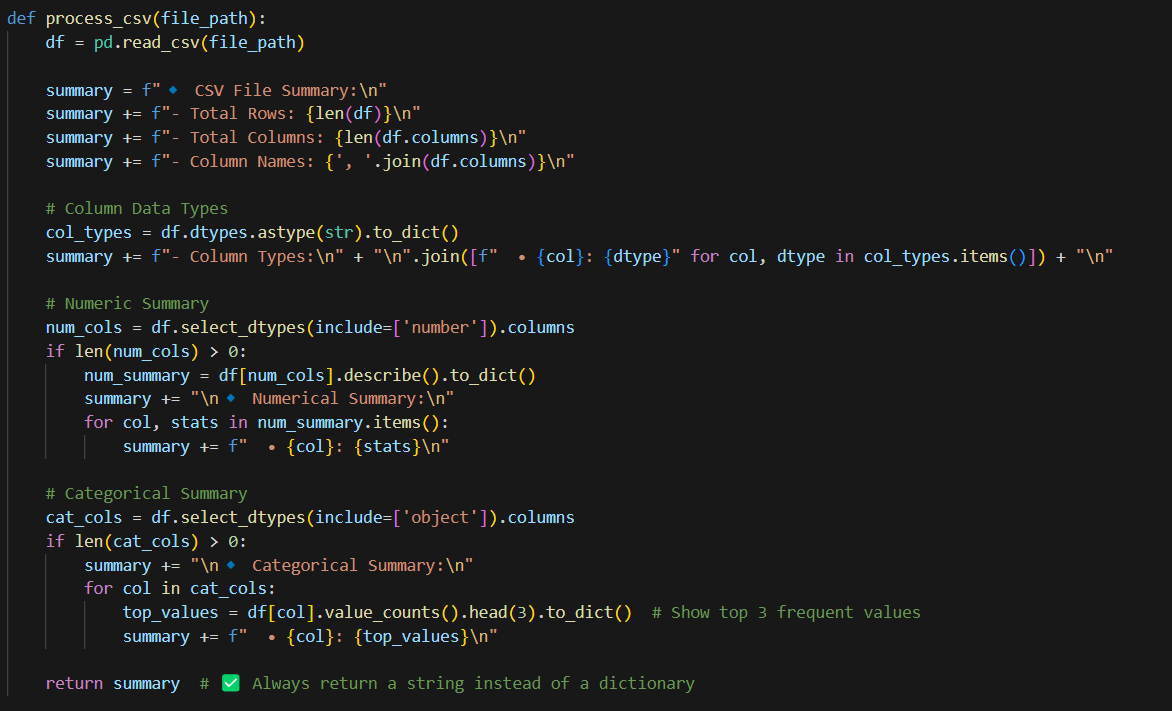


* If text is present in the image → OCR (Tesseract) extracts and returns it.
* If no text is found → BLIP model generates a description of the image.
  1. **VIDEO:**



The function extracts audio from a video file, saves it as a temporary .wav file, and uses the Whisper AI model to transcribe the audio into text. After transcription, the temporary audio file is deleted. If transcription is successful, the extracted text is returned; otherwise, a message indicating no meaningful audio is detected is provided.

* Video is converted to an audio file.
* Whisper AI transcribes the audio into text.
* The extracted text is returned after cleaning up the temporary file.
  1. **CSV FILE:**



* **Reads CSV File** → Loads the CSV into a Pandas DataFrame.
* **Basic Summary** → Computes the total number of **rows** and **columns**, and lists **column names**.
* **Column Data Types** → Identifies and displays **data types** for each column.
* **Numeric Summary** → If numerical columns exist, it generates a **statistical summary** (mean, min, max, standard deviation, etc.).
* **Categorical Summary** → If categorical columns exist, it identifies the **top 3 most frequent values** per column.
* **Returns Summary** → Compiles all the collected information into a **formatted string** for output

**3. PREPROCESSING TASKS IN DETAIL**

**1️⃣ Tokenization (Splitting Text into Meaningful Units)**

✔ Splits text into sentences or words to make processing easier.

✔ Removes unwanted punctuation and special characters.

✔ Helps in text normalization for better vector embeddings.

**2️⃣ Data Cleaning & Normalization**

✔ Removes extra spaces, line breaks, and special characters.

✔ Converts text to lowercase for uniformity.

✔ Eliminates irrelevant metadata (e.g., timestamps from videos, HTML tags from web pages).

**3️⃣ Summarization (Generating Concise Summaries)**

✔ Uses state-of-the-art NLP models to generate summaries.

✔ Helps in faster retrieval by reducing the size of indexed data.

✔ Models like BART (facebook/bart-large-cnn) and T5 (t5-small) are used for text summarization.

**4. MODELS:**

**1️.Summarization Model: BART (facebook/bart-large-cnn)**

✔ Generates concise summaries of long text documents for faster retrieval.

✔ Helps in reducing noise and redundant information from extracted text.

✔ facebook/bart-large-cnn is a fine-tuned version of BART, a transformer-based sequence-to-sequence model.

✔ It is trained on the CNN/DailyMail dataset for abstractive text summarization.

✔ Supports long-form text summarization, making it ideal for processing extracted content from PDFs, DOCX, or scraped web pages.

**2️. Sentence Embedding Model: all-MiniLM-L6-v2**

✔ Converts text into dense vector embeddings, enabling fast semantic retrieval.

✔ These embeddings are stored in FAISS (Facebook AI Similarity Search) for efficient search.

✔ all-MiniLM-L6-v2 is a lightweight sentence-transformer model optimized for speed and efficiency.

✔ It compresses text into 384-dimensional vectors while maintaining high accuracy.

✔ Used for semantic similarity search in RAG (Retrieval-Augmented Generation).

**3️. Faster Whisper Model: Whisper (base)**

✔ Converts spoken words from videos into text using Whisper, an advanced ASR (Automatic Speech Recognition) model.

✔ Used for transcribing video lectures, podcasts, and meeting recordings.

✔ Extracted text can be summarized or indexed for efficient retrieval.

✔ faster-whisper is an optimized version of OpenAI’s Whisper model, designed for faster inference.

✔ Works well with multiple languages and accents.

✔ Uses beam search decoding to improve transcription accuracy.

**4️. FAISS Index for Efficient Retrieval**

✔ Stores vector embeddings in a highly optimized search index for fast semantic retrieval.

✔ Enables approximate nearest neighbor (ANN) search, which is much faster than traditional keyword-based search.

✔ FAISS (Facebook AI Similarity Search) is an efficient similarity search library for handling large-scale embeddings.

✔ Uses IndexFlatL2, which applies L2 distance for nearest neighbor searches.

**5. WORKFLOW:**

🔹 **Step 1: Understanding Your Input**

* 🌐 **Webpage URL** → I extract text using web scraping.
* 🎥 **YouTube Video** → I use subtitles or convert speech to text.
* 📄 **Document (PDF, CSV, DOCX, etc.)** → I extract text with document parsers.
* 🖼 **Image** → I use OCR for text or AI models for descriptions.
* ✍️ **Raw Text** → I process it directly.

🔹 **Step 2: Extracting the Text**

* **Webpages** → I use BeautifulSoup.
* **Videos** → I use Whisper for transcription.
* **Documents** → I handle PDFs, Word files, and CSVs with specialized tools.
* **Images** → I use Tesseract OCR or BLIP if no text is detected.

🔹 **Step 3: Summarization (Optional)**

* **Extractive summarization** (picking key sentences).
* **Abstractive summarization** (rewording content in a concise way).

🔹 **Step 4: Creating Embeddings**  
I convert text into smart vector embeddings using models like **Sentence-BERT (all-MiniLM-L6-v2)**.

🔹 **Step 5: Storing Information Efficiently**  
I store these embeddings in **FAISS**, a high-speed AI-powered search engine. This helps me retrieve relevant content super fast.

🔹 **Step 6: Querying & Retrieval**  
  
1️.Convert your query into an embedding.  
2️.Search FAISS for the most relevant matches.  
3️.Retrieve and present the most useful information.

**6. INSTALLATION AND SETUP**

The script begins by installing necessary libraries for text extraction, summarization, retrieval, and video processing.

**!pip install torch transformers faiss-cpu sentence-transformers pdfplumber**

**!pip install pymupdf python-docx pytesseract openai-whisper moviepy**

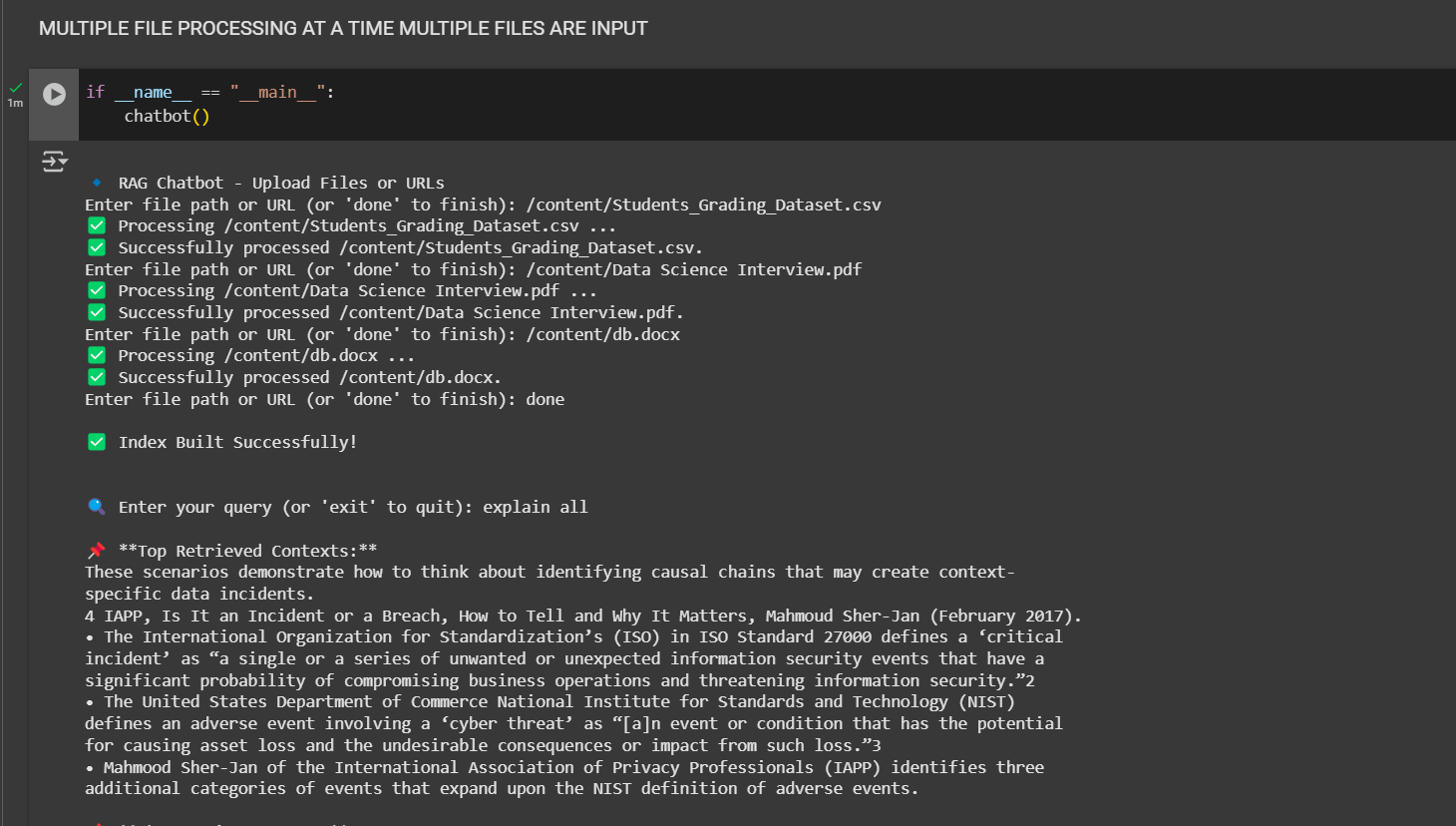
**!apt-get update && apt-get install -y tesseract-ocr**

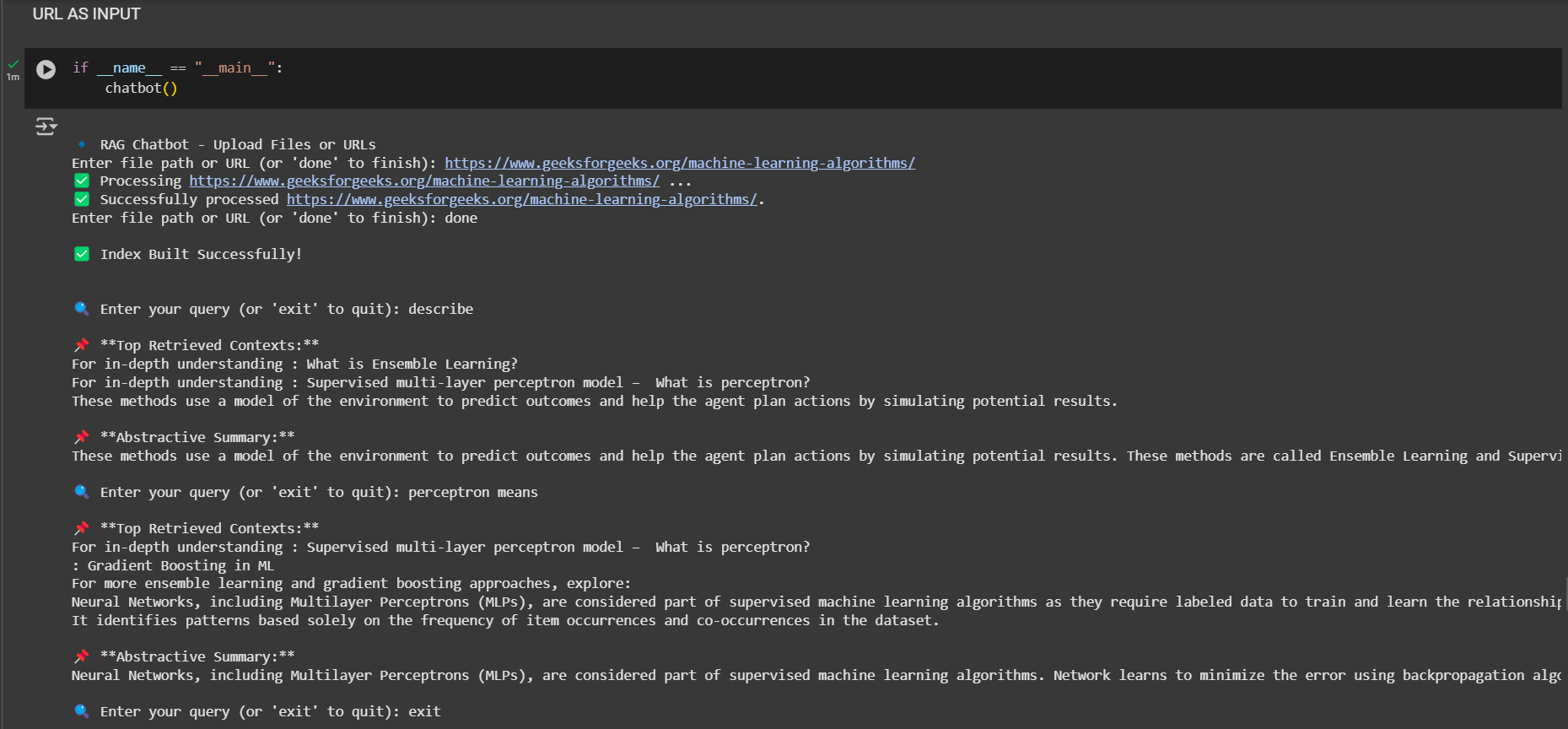
These installations include:

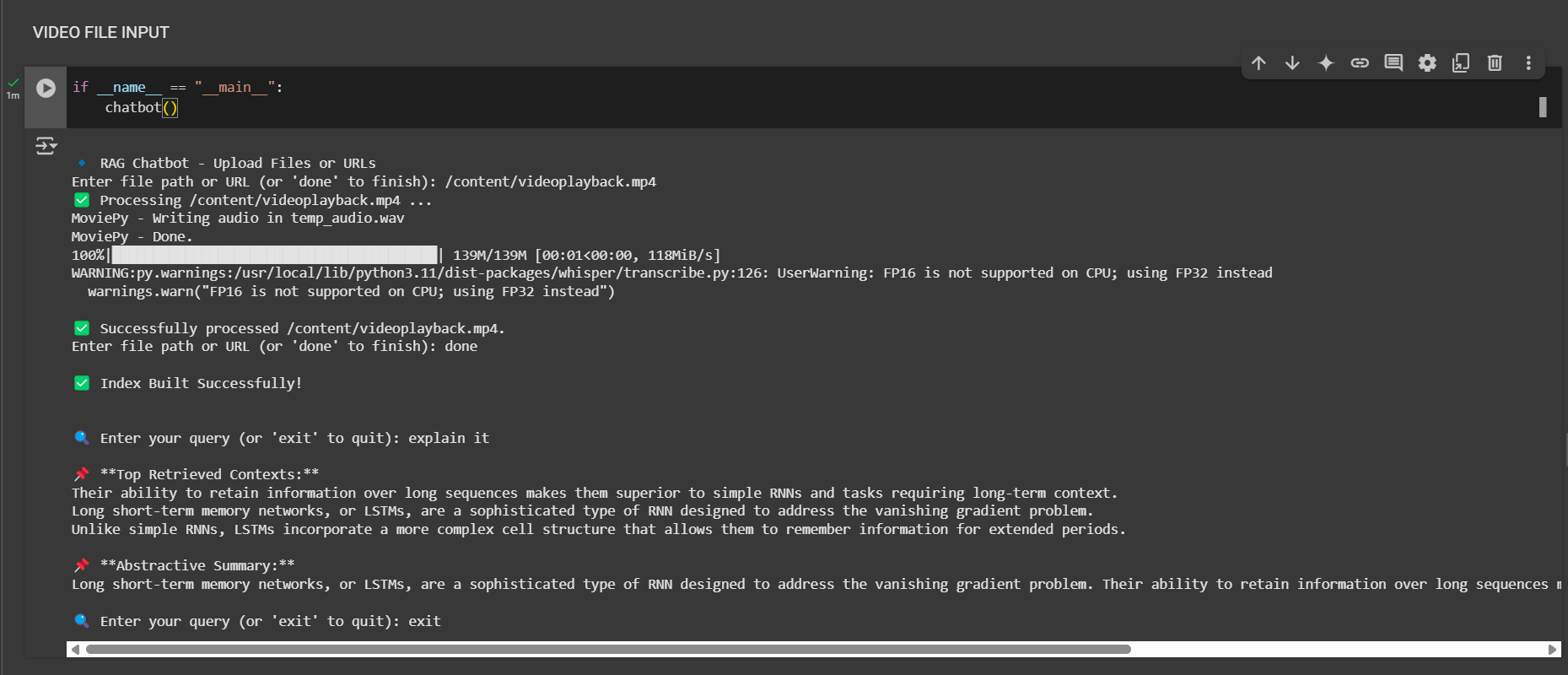
* torch for deep learning models
* transformers for text processing
* faiss-cpu for efficient vector search
* pdfplumber, pytesseract, and docx for document parsing
* openai-whisper for audio transcription

**7.OUTPUT:**

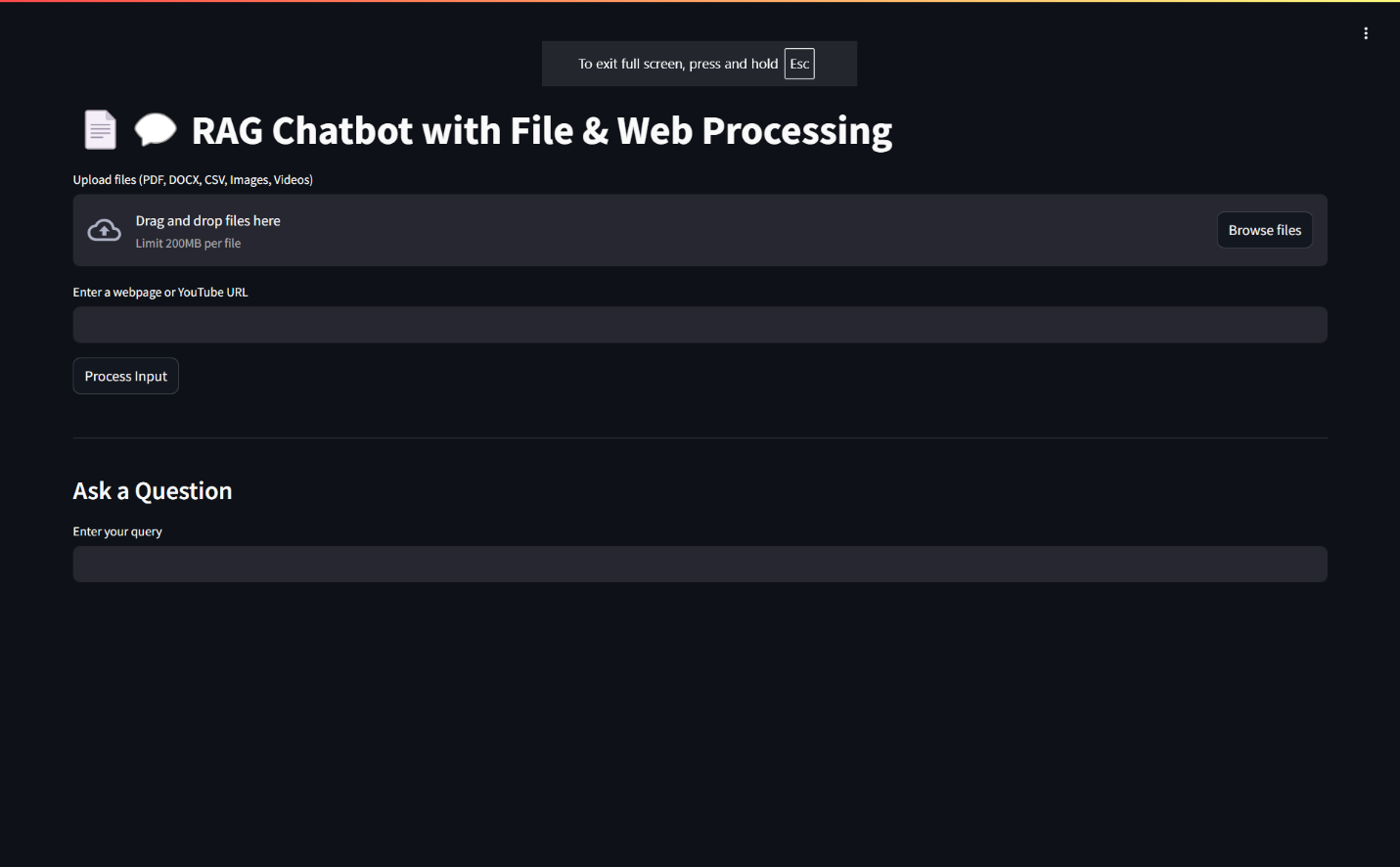
IN COLOB:

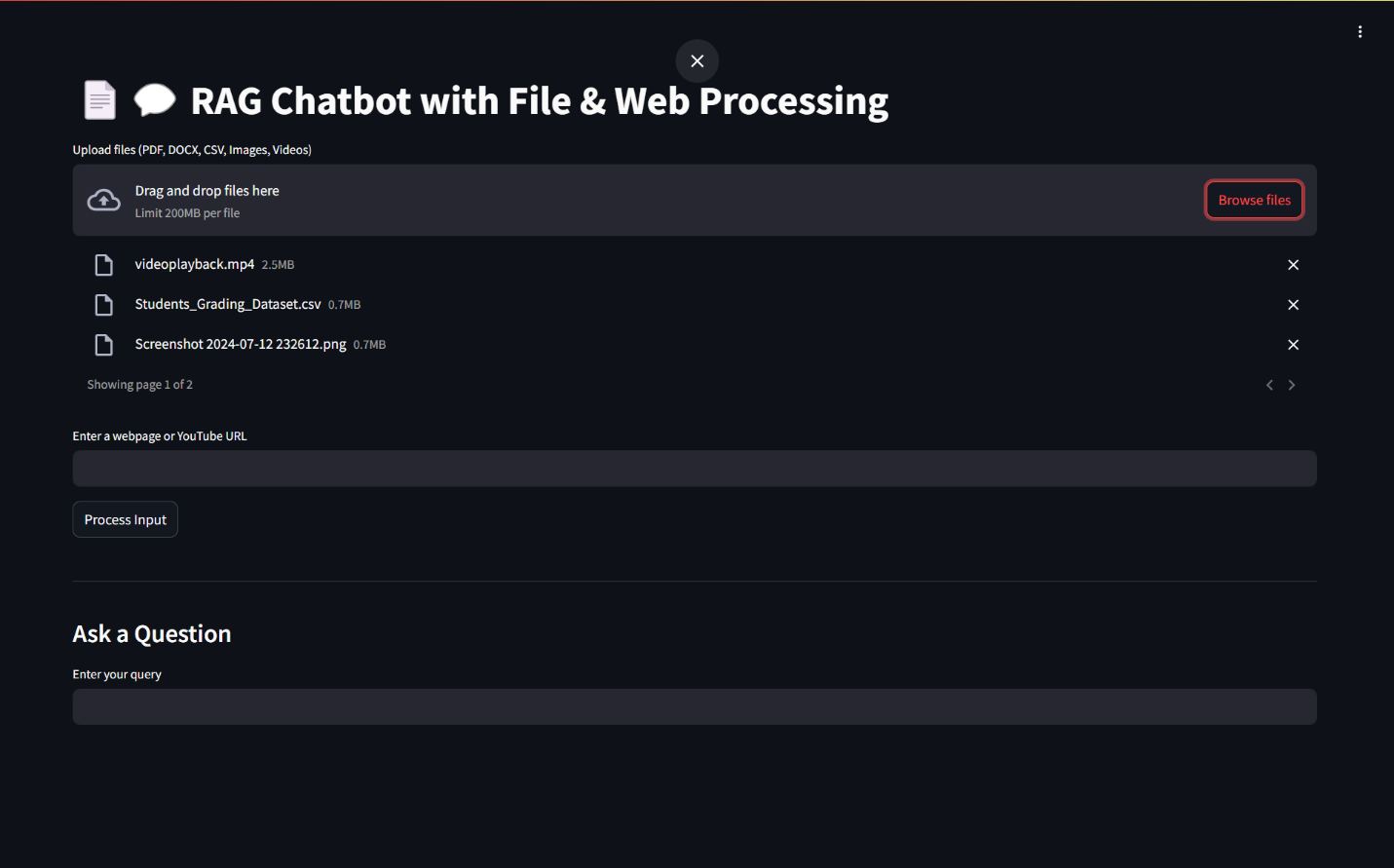


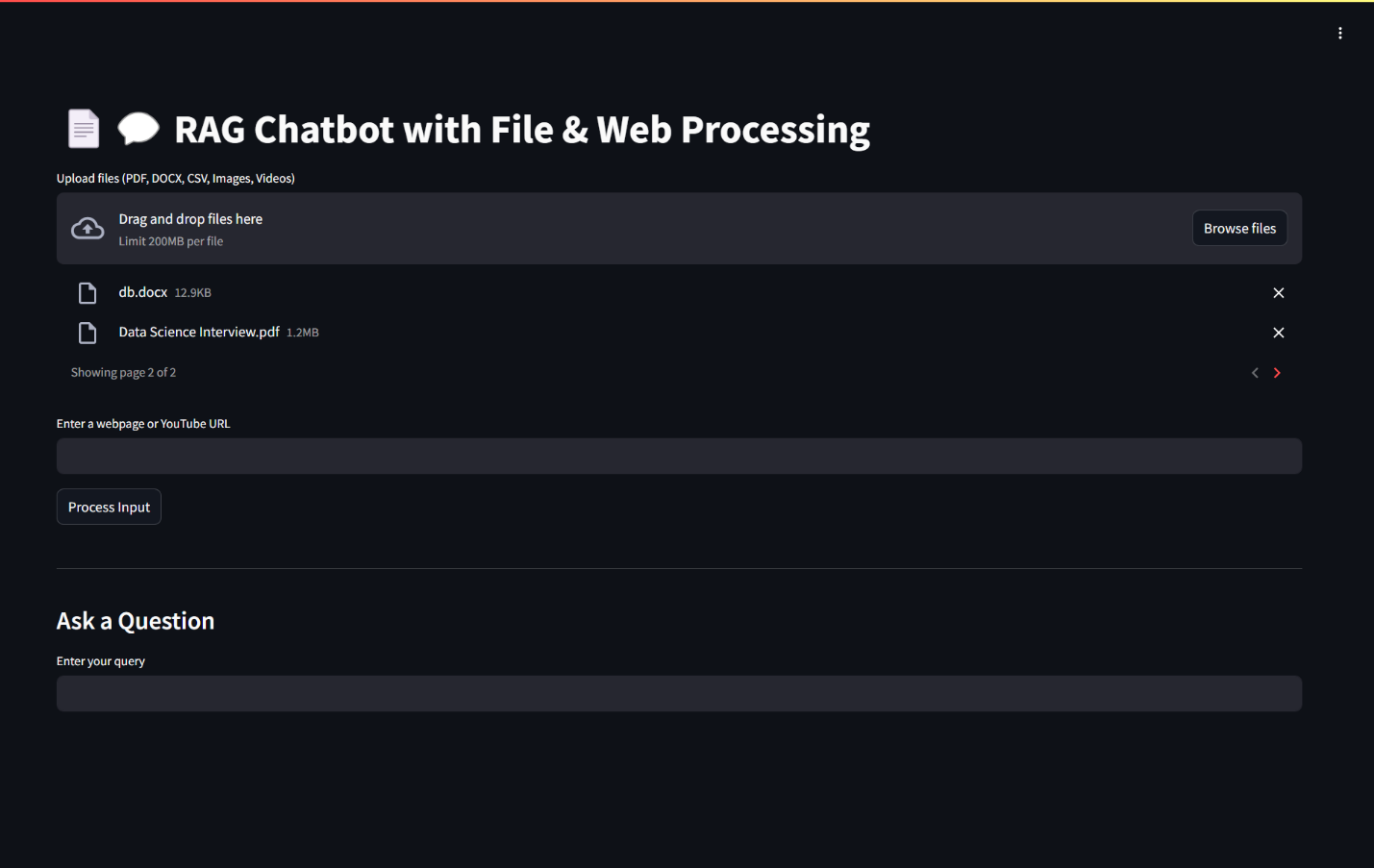




**STREAMLIT INTERFACE:**







**8. DIAGRAM:**

