**s**

**ABSTRACT:**

The **DocuQuery: AI-Powered PDF Knowledge Assistant** is an intelligent document processing system that leverages Google PaLM AI to extract, analyze, and summarize content from PDF and Word documents. This project is designed to automate and simplify document-related tasks such as research paper summarization, resume screening, and price list analysis, enabling users to make informed decisions efficiently. Using Natural Language Processing (NLP), vector embeddings, and AI-driven analysis, the system ensures accurate content extraction, structured summarization, and intelligent retrieval of key information.

Developed with Streamlit, the project offers a user-friendly web interface for seamless interaction. It utilizes FAISS-based vector storage for fast and efficient document retrieval and integrates Google AI’s generative models for summarization and contextual understanding. By automating time-consuming manual tasks, DocuQuery enhances productivity and decision-making for businesses, researchers, and professionals. Future improvements may include multi-language support, deeper contextual analysis, and expanded AI capabilities, making it a versatile tool for intelligent document management and knowledge extraction:.

**INTRODUCTION:**

In today’s digital landscape, managing and extracting meaningful insights from vast amounts of textual data is a critical challenge for businesses, researchers, and hiring professionals. DocuQuery: AI-Powered PDF Knowledge Assistant is designed to address this challenge by leveraging Google PaLM AI and Natural Language Processing (NLP) to automate the extraction, summarization, and analysis of document content. The system processes PDF and Word documents to streamline tasks such as price list analysis, research paper summarization, and resume matching, significantly reducing manual effort and improving efficiency.

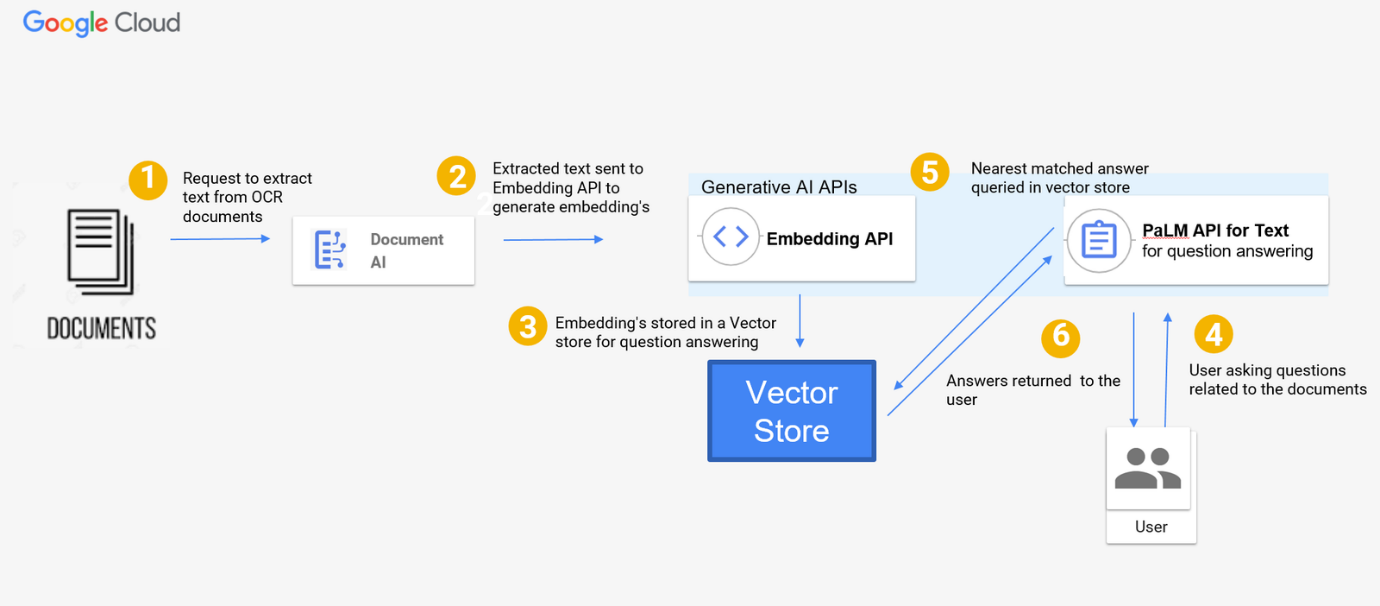
Built using Streamlit, the project provides an intuitive web-based interface, ensuring accessibility and ease of use. It integrates FAISS-based vector storage for fast and efficient information retrieval and employs AI-driven models for contextual understanding and intelligent decision-making. By automating these document-intensive tasks, DocuQuery empowers users to make data-driven decisions quickly and accurately. With its scalable architecture and advanced AI capabilities, the system sets the foundation for future enhancements, including multi-language support, deeper contextual analysis, and expanded document processing functionalities.

**Scenario-1: Price List Extraction & Comparison**  
Businesses can upload multiple price lists from different suppliers, and the tool will extract item prices, descriptions, and quantities. This allows for quick price comparison, helping procurement teams make informed purchasing decisions.

**Scenario-2: Research Paper Summarization & Querying**  
Researchers can simplify complex academic papers by uploading them for AI-generated summaries. The tool highlights key insights and allows users to ask questions for deeper understanding, saving time in research analysis.

**Scenario-3: Resume Screening & Candidate Matching**  
HR teams can automate resume screening by uploading multiple resumes. The tool analyzes qualifications and matches candidates to job requirements, streamlining the hiring process and improving recruitment efficiency.

**ARCHITECTURE**:



**Step 1: User Uploads Documents**

* Users upload **PDF** or **Word** documents.
* Supported formats: **.pdf, .docx**.

**Step 2: Document Processing & Text Extraction**

* Detects document type and extracts text:
  + **PDFs** → Extracted using **PyPDF2**.
  + **Word Docs** → Extracted using **python-docx**.
* Logs extracted text **page-wise** for transparency.

**Step 3: Text Embedding & Storage**

* Converts extracted text into **vector embeddings** using **Google PaLM Embeddings API**.
* Stores embeddings in **FAISS (Vector Store)** for fast retrieval.

**Step 4: User Selects an AI Feature**  
Users can choose from:  
**Price List Analyzer** → Extracts item details & compares prices.  
**Research Paper Summarizer** → Generates **short, medium, or detailed summaries**.  
**Resume Matcher** → Matches resumes to job descriptions & ranks candidates.

**Step 5: AI Processing & Response Generation**

* **For Summarization & Resume Matching:**
  + Uses **Google Gemini API** to generate summaries or rank resumes.
* **For Question Answering:**
  + Converts user queries into **vector embeddings**.
  + Retrieves relevant content using **FAISS** and answers using **Google Gemini API**.

**Step 6: Results Display & Output**

* Displays **extracted text, summaries, or resume matches** on Streamlit UI.
* Users can **download research summaries as PDFs**.

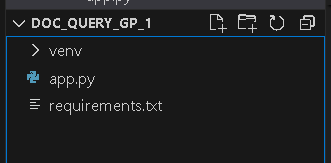
**Prior Knowledge**

 We must have prior knowledge of the following topics to complete this project.

* NLP: <https://www.tutorialspoint.com/natural_language_processing/index.htm>
* RAG: https://www.datacamp.com/blog/what-is-retrieval-augmented-generation-rag
* LLM & PALM: <https://cloud.google.com/vertex-ai/docs/generative-ai/learn-resources>
* Streamlit: Create interactive web applications. https://docs.streamlit.io/
* Google Generative AI: https://aistudio.google.com/
* Embeddings and Vector Stores Facebook AI Similarity Search (FAISS): <https://engineering.fb.com/2017/03/29/data-infrastructure/faiss-a-library-for-efficient-similarity-search/>
* PDF Processing: Extract text from PDF documents. <https://pypi.org/project/PyPDF2/>
* Langchain: https://python.langchain.com/v0.1/docs/get\_started/introduction/

**Project Structure:**

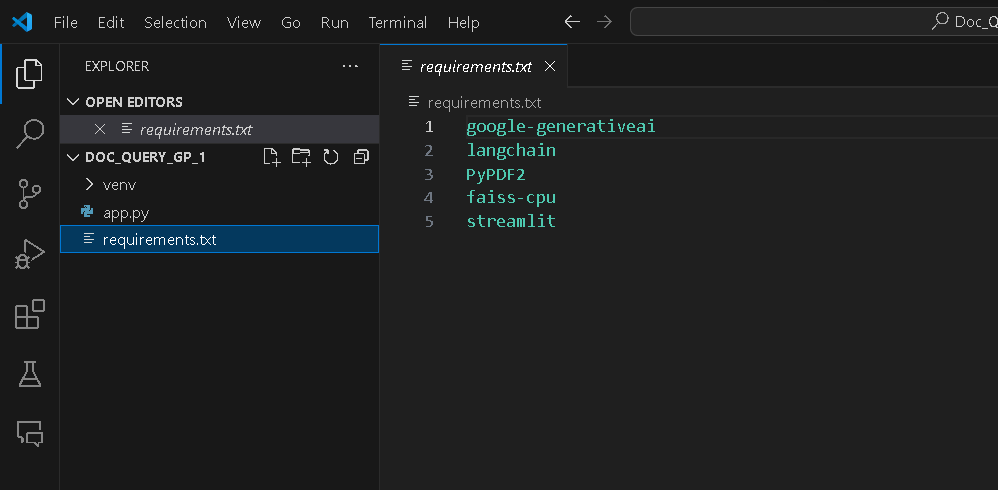
Create the Project folder which contains application file as shown below



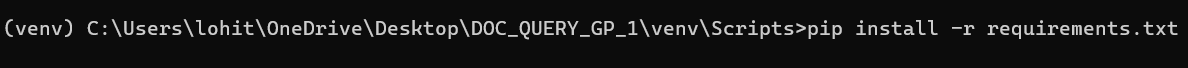
**Milestone 1: Defining Requirements**

Listing the necessary libraries in the **requirements.txt** file facilitates a smooth setup and ensures the project environment can be easily replicated. This enhances reproducibility, allowing others to effortlessly recreate the development setup.

**Activity 1: Create a requirements.txt file to list the required libraries.**



**Activity 2: Install the required libraries.**

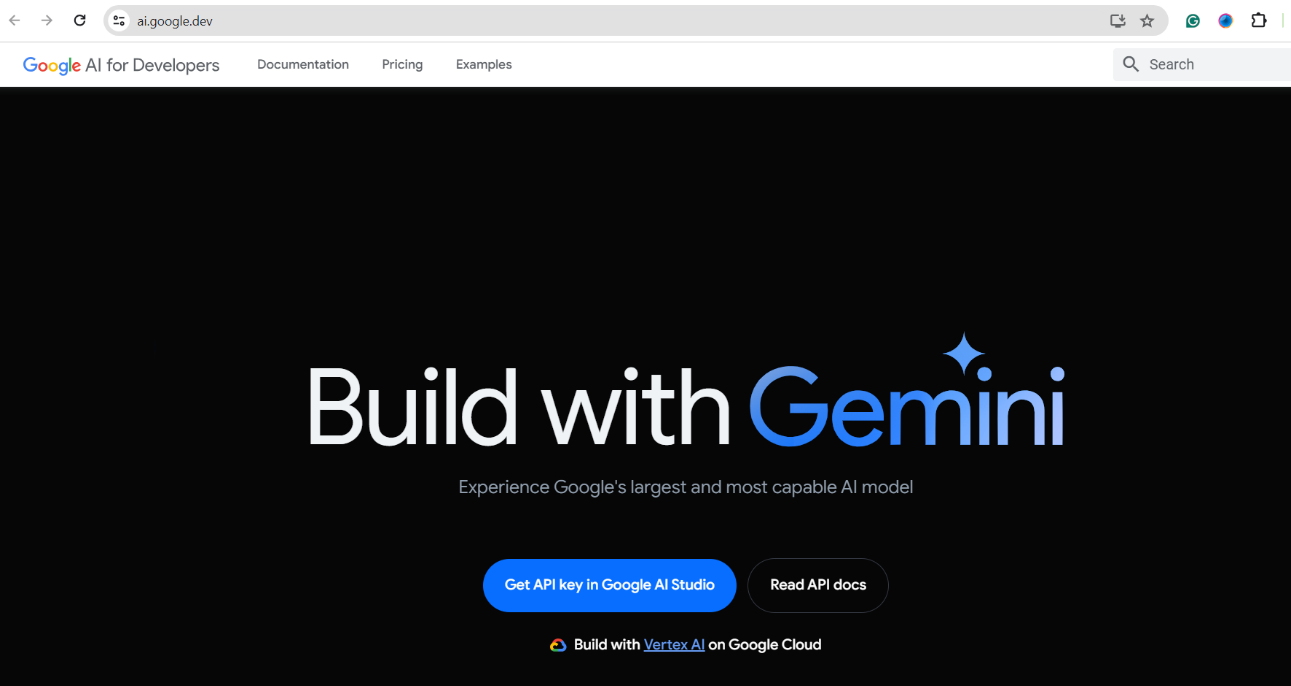
****

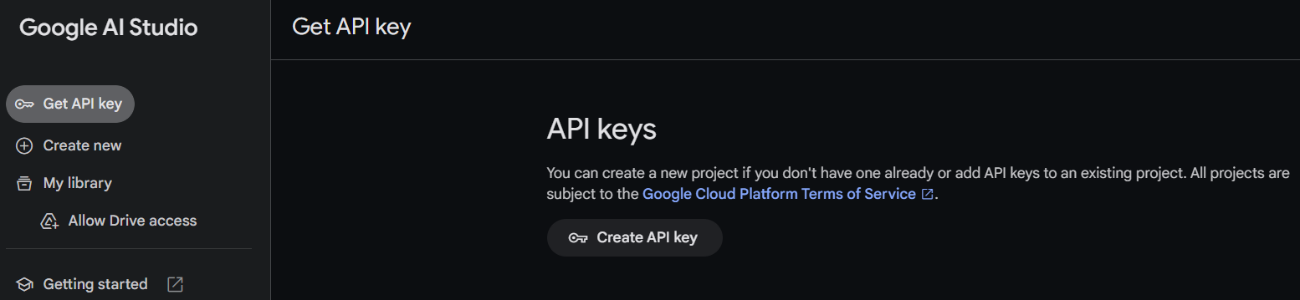
**Milestone 2: Initializing the Model**

For initializing the model we need to generate PALM API.

**Activity 1: Generate PALM API**

* Click on the link (<https://developers.generativeai.google/>).
* Then click on “Get API key in Google AI Studio”.
* Click on “Get API key” from the right navigation menu.
* Now click on “Create API key”. (Refer the below images)
* Copy the API key.





**Milestone 3: Document Processing & Text Extraction**

* Implement **PDF text extraction** using PyPDF2 with page-wise logging.
* Implement **Word document text extraction** using python-docx.
* Extract data from **price lists**, **research papers**, and **resumes** for further processing.

**Milestone 4: AI-Powered Analysis & Summarization**

* Integrate **Google PaLM’s gemini-1.5-pro-latest** model for text summarization.
* Develop a **research paper summarizer** with selectable summary length options (short, medium, detailed).
* Format the summary output with structured key points for better readability.
* Implement a **resume ranking system** that matches resumes to job descriptions using AI.

**Milestone 5: Embedding Generation & Vector Storage**

* Convert extracted text into **embeddings** using **Google's text-embedding-gecko** model.
* Store embeddings in a **FAISS Vector Database** for efficient retrieval.
* Implement a **Conversational AI System** using ConversationalRetrievalChain to enable users to query documents.

**Milestone 6: AI-Based Resume Matching System**

* Extract text from uploaded resumes and job descriptions.
* Use **Google PaLM API** to compare job descriptions with resumes and rank candidates.
* Provide **AI-generated reasoning** for the ranking of candidates based on job requirements.

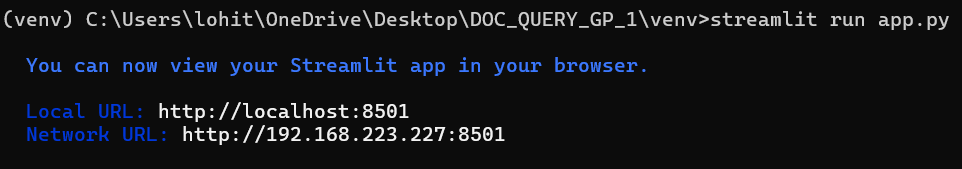
**Milestone 7: Web Application Development with Streamlit**

* Build an **interactive user interface** using Streamlit for document uploads and processing.
* Implement **sidebar navigation** to allow users to select between different functionalities:
  + **Price List Analyzer**
  + **Research Paper Summarizer**
  + **Resume Matcher for Hiring**
* Implement **real-time error handling and user feedback** to improve usability.

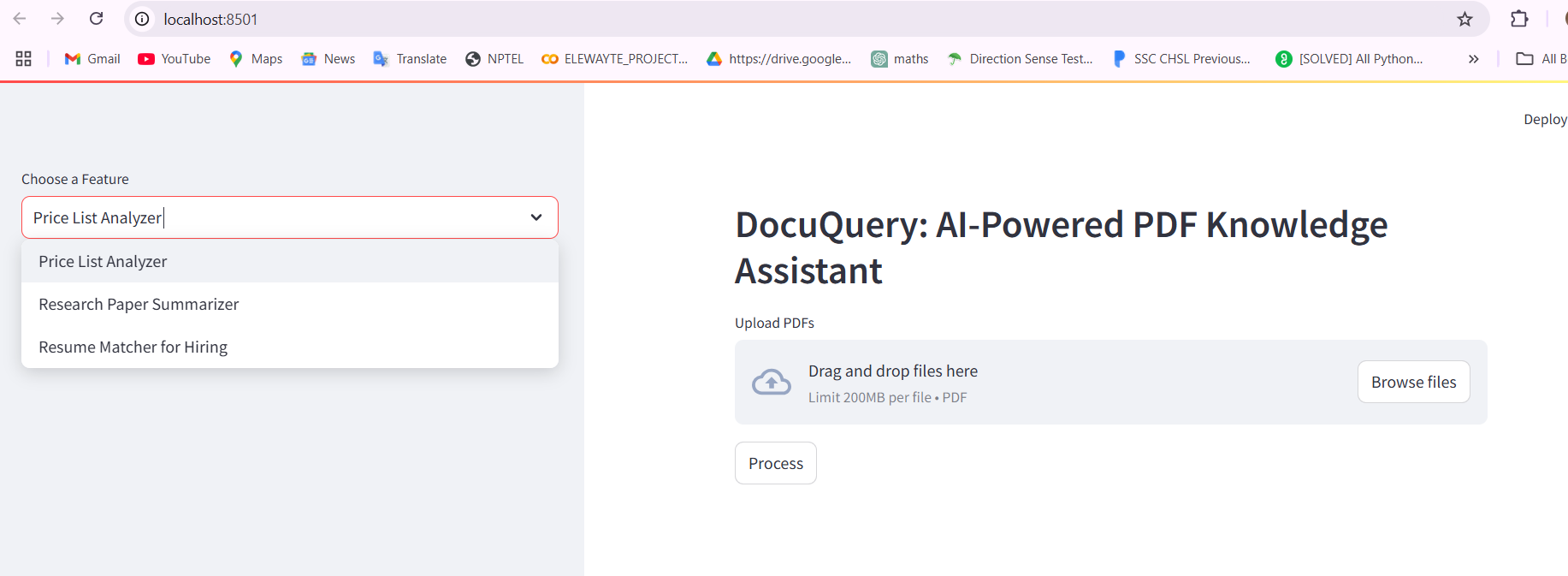
**Milestone 7: Model Deployment & PDF Report Generation**

* Deploy the **Streamlit Web App** locally for real-time document analysis.
* Implement a feature to **download summarized research papers as PDFs** using FPDF.
* Finalize the **user experience and design improvements** for better interaction.
* Provide clear **instructions for users** on how to interact with the application.

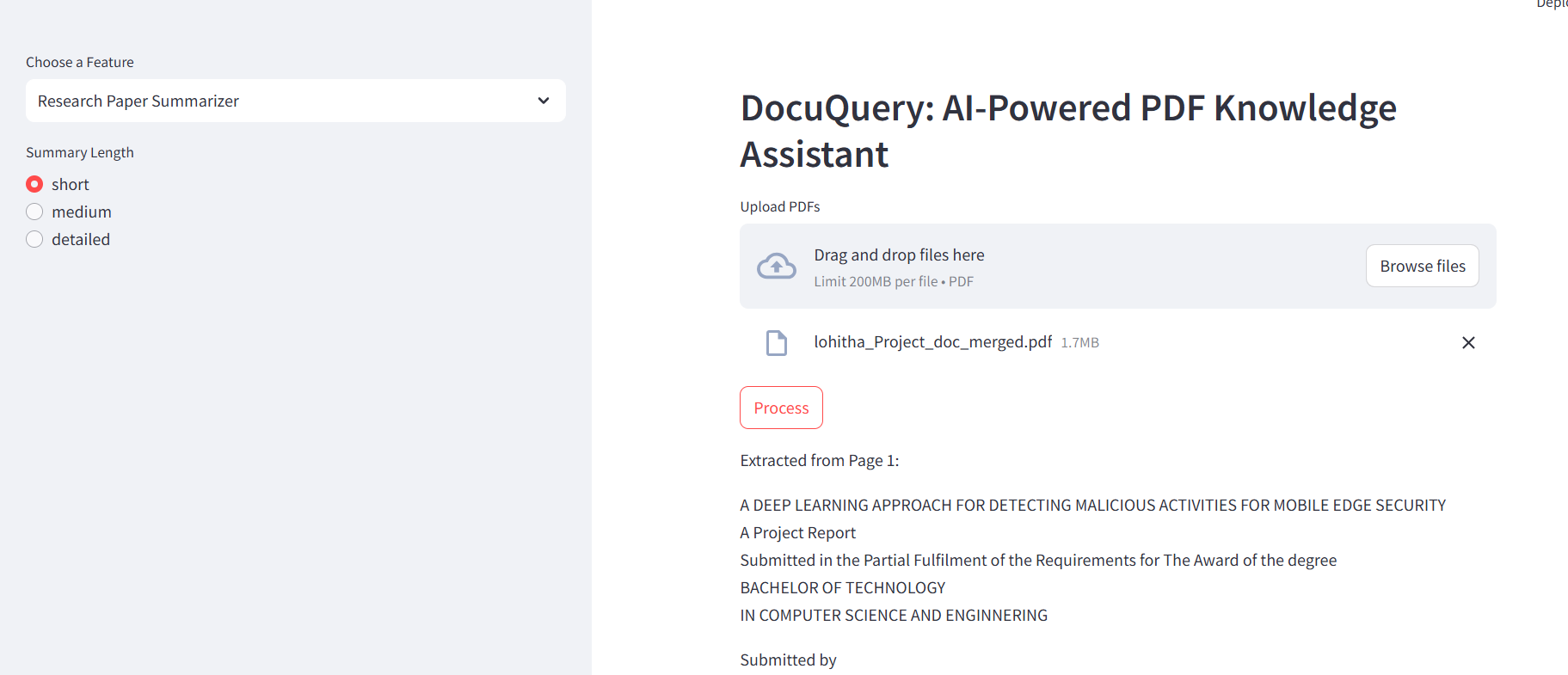
By running the application, it looks like below:



Now, the application will open in the web browser,



**After giving the input:**



**CONCLUSION**:

The **DocuQuery: AI-Powered PDF Knowledge Assistant** is an advanced document processing solution that integrates Google PaLM AI to extract, analyze, and summarize content from PDFs and Word documents. By leveraging Natural Language Processing (NLP), vector embeddings, and AI-driven analysis, the system enables users to efficiently handle research paper summarization, resume matching, and price list analysis. Built with a Streamlit-based web interface, it offers an intuitive and interactive experience for users to process documents, retrieve insights, and generate AI-powered summaries. Additionally, the integration of FAISS for vector storage enhances fast and efficient document retrieval, making it a powerful tool for professionals and researchers.

With automated document analysis, AI-driven resume screening, and structured PDF report generation, DocuQuery enhances productivity and simplifies decision-making for businesses and individuals. Its ability to process large volumes of data accurately and efficiently reduces manual workload and improves workflow efficiency. As a scalable and adaptable AI-powered tool, future enhancements could include multi-language support, deeper document classification, and more advanced AI insights, making it a valuable asset for modern knowledge management and intelligent document processing.