**Description of the Multimodal RAG Application**

**Objective**

The primary objective of this multimodal RAG (Retrieval-Augmented Generation) application is to enable users to upload any document (such as a PDF) and interactively ask questions about its content. The application is designed to provide comprehensive answers that include relevant text and images extracted from the document. This enhances the user's ability to quickly and efficiently retrieve specific information, including visual content, which can be critical for understanding complex documents. The code of application is divided into two sections-

* client.py- For the frontend.
* app.py- For the backend

**Logic**

1. **Document Upload**:

* Users upload a PDF document through the Streamlit front-end interface.
* The document is sent to the backend via the FastAPI file upload API.
* Api endpoint- *http://localhost:8000/file\_upload/*

**2. Image and Text Extraction:**

* Using `pymupdf`, the application extracts all images from the PDF document.
* Text content is also extracted from the document.

3. **Embedding Generation**:

* Images are converted into embeddings using CLIP (Contrastive Language–Image Pre-training) embeddings.
* Text content is converted into embeddings using OpenAI text embeddings.

**4. Storage and Indexing:**

* Both image and text embeddings are stored and indexed in the Qdrant vector database.

**5. Confirmation:**

* Users receive a confirmation that their document has been successfully uploaded and processed.

**6. Question-Answering:**

* Users can ask questions about the uploaded document through the Streamlit interface.
* The document is sent to the backend via the FastAPI ask question API.
* Api endpoint- *http://localhost:8000/*ask question*/*
* LLAVA, a source multimodal LLM model using Ollama, is employed to generate answers that combine text and visual content.
* The query is processed by the query engine using Llama-index for efficient retrieval of relevant content.
* Then query and context are sent to the MultiModal LLM model for the final answer generation.
* The answer is generated, containing:

- The relevant text answer.

- Page numbers where the information was found.

- Relevant images extracted from the document, if any.

**Why This Approach**

**1. Multimodal Retrieval:**

Combining text and image retrieval ensures that users get a comprehensive understanding of the document content, catering to different types of queries.

**2. Efficient Indexing and Retrieval:**

Using Llama-index for indexing and retrieval ensures efficient processing and quick response times, even for large documents.

**3. Powerful Embeddings:**

Utilizing CLIP and OpenAI embeddings for images and text respectively leverages state-of-the-art models for accurate and meaningful content representation.

1. **Advanced Multimodal Model:**

LLAVA(Large Language and Vision Assistant ), using Ollama, enhances the system's ability to understand and generate multimodal content, ensuring high-quality responses that integrate both text and images.

**5. Scalability and Flexibility:**

The use of FastAPI for backend operations and Qdrant as a vector database allows the application to scale efficiently, handling large volumes of data and queries.

**6. User-Friendly Interface:**

Streamlit provides an intuitive and interactive front-end interface, making the application accessible to users with varying levels of technical expertise.

1. **Error Handelling:**

Error Handelling using try and expect blocks so that Users receive meaningful error messages instead of experiencing application crashes or seeing technical error details. The application can handle unexpected situations without crashing, maintaining its stability and reliability.

By managing errors gracefully, the application can recover from certain errors and continue running.

**Additional Features**

**1. Page Number Identification:**

Answers include page numbers to help users locate the original content quickly within the document.

**2. Image Relevance:**

Relevant images are included in the answers, providing a visual context that can be crucial for understanding certain information.

**3. Confirmation and Status Updates:**

Users receive real-time feedback on the status of their document upload and processing, enhancing user experience and trust in the system.

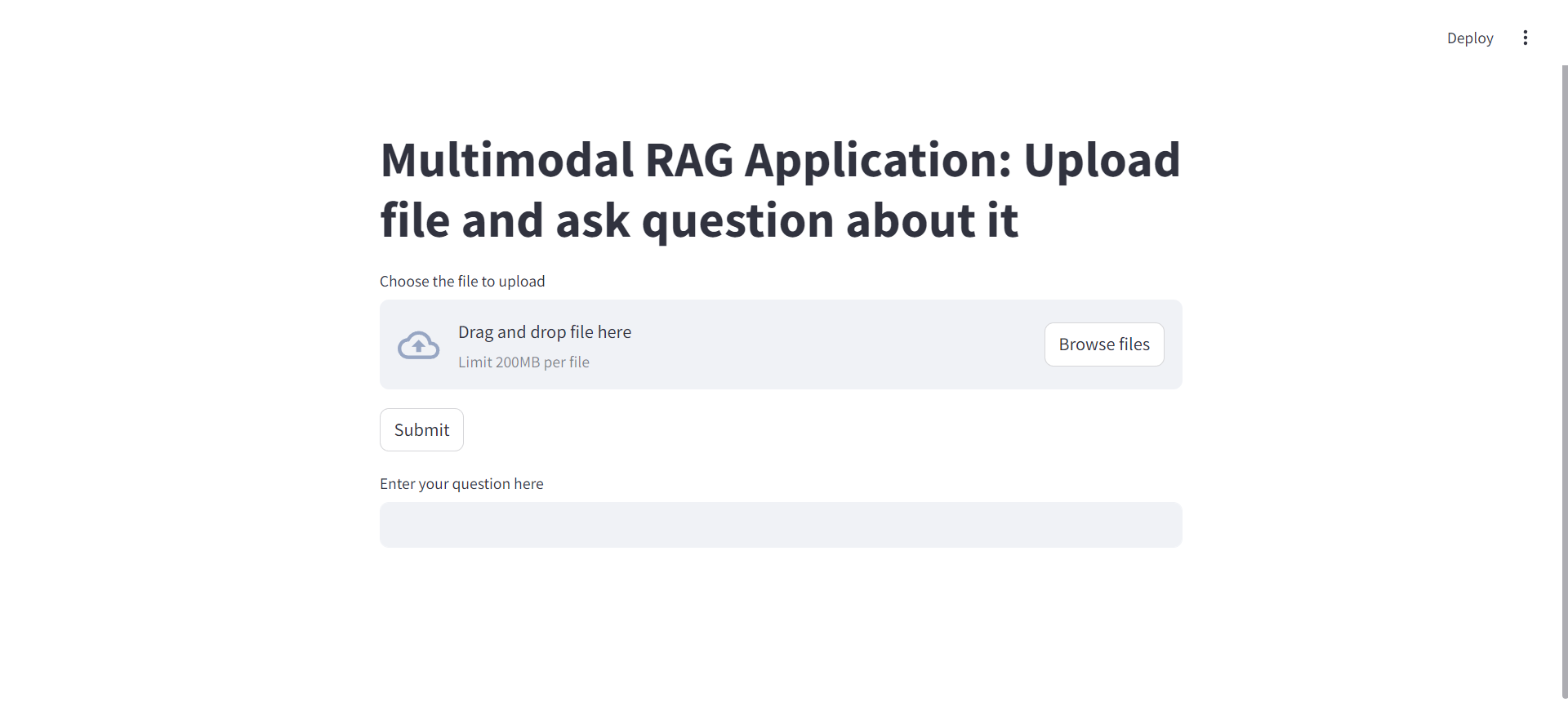
**4. Scalable Architecture:**

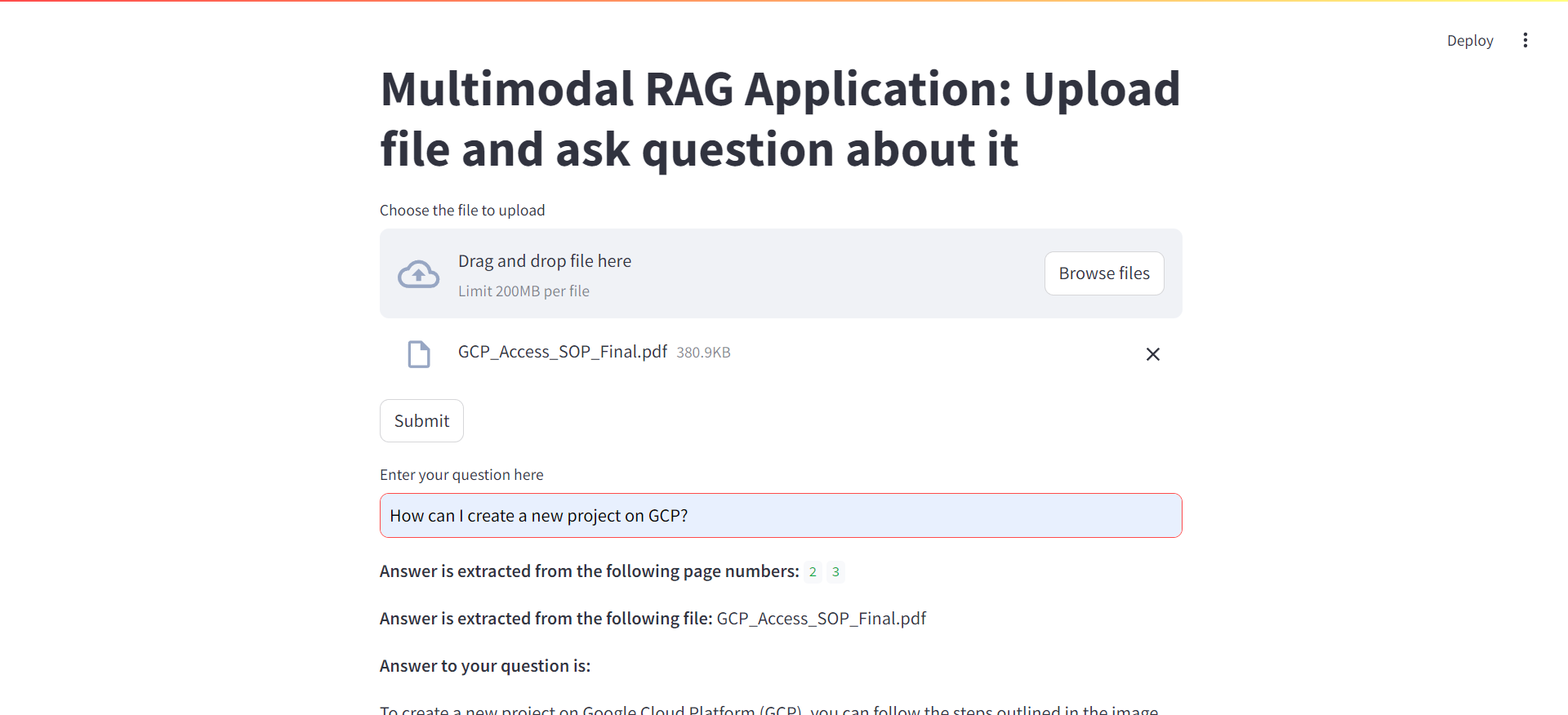
The modular architecture ensures that each component (upload, extraction, embedding, storage, retrieval) can be independently scaled and optimized, ensuring robust performance under heavy load.

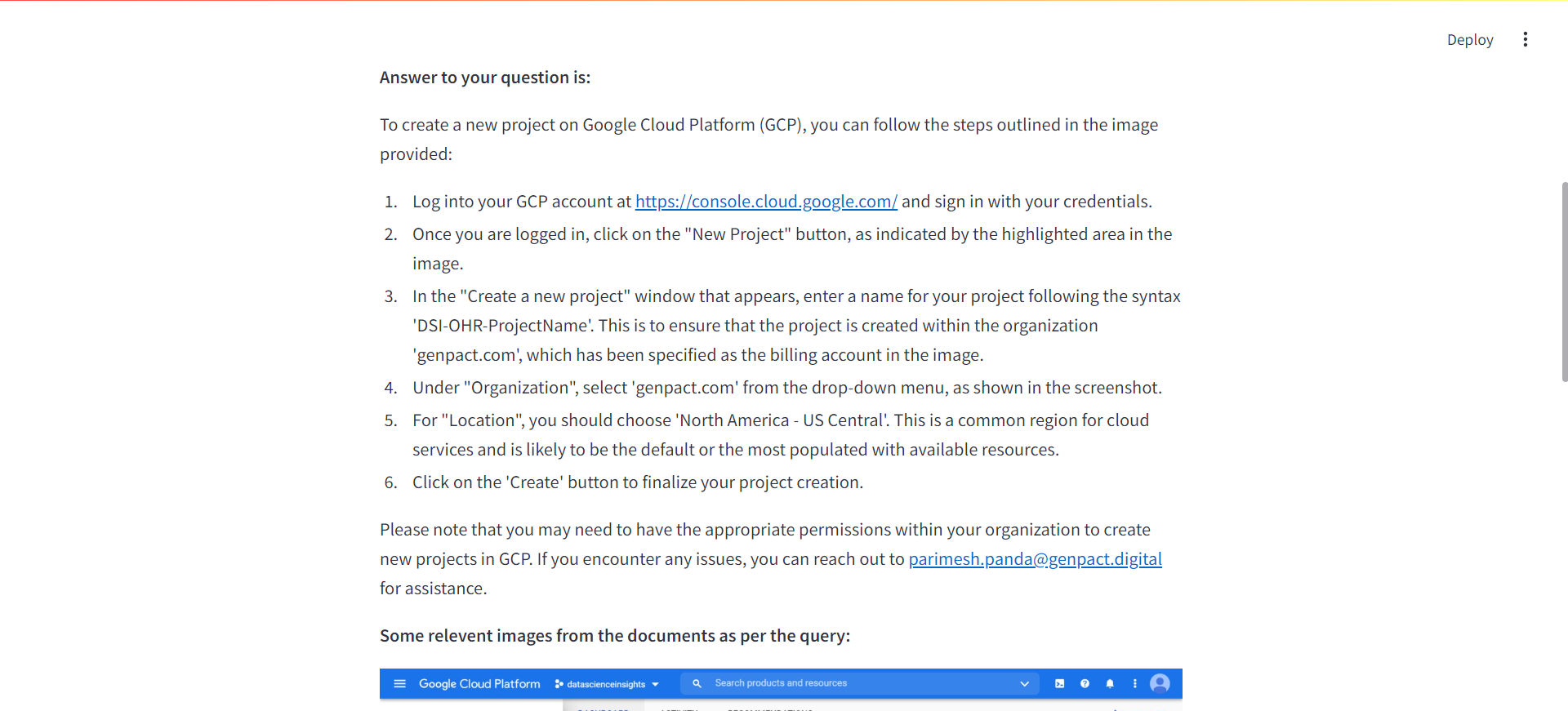
**Guidelines**

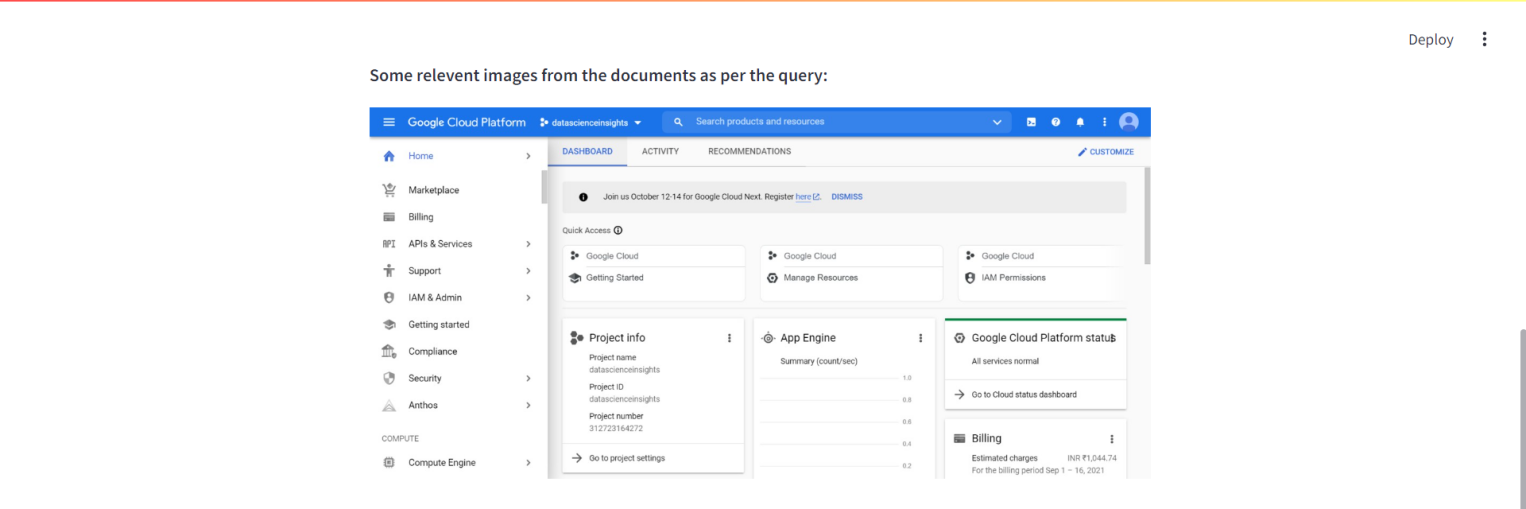
* Open two different terminals.
* First run the backend using the following command- *python app.py*
* The run the frontend in another terminal using the following command- *streamlit run client.py*

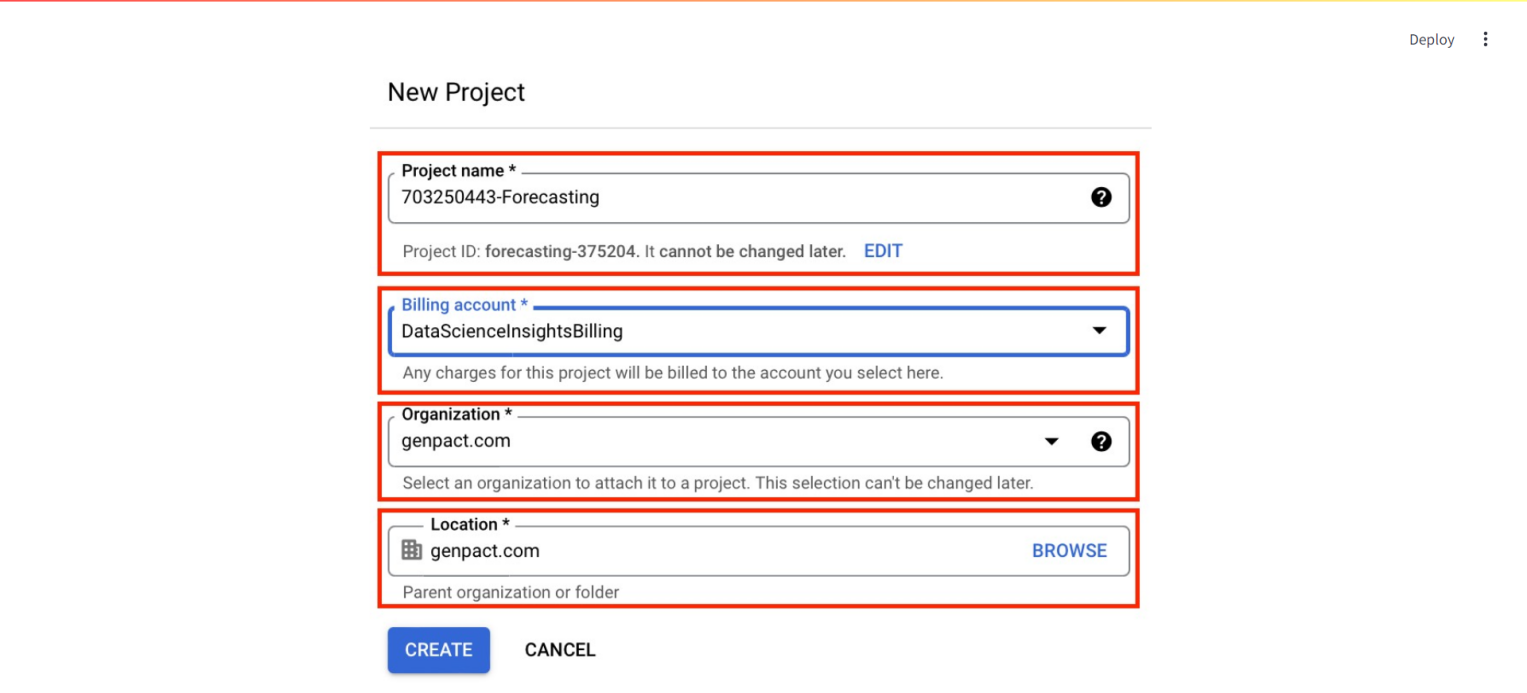
**Application Screenshots**

****

****

****

****

****

**Conclusion**

This multimodal RAG application represents a significant advancement in document interaction technology, providing users with an efficient and effective tool for extracting and understanding information from complex documents. The integration of text and image retrieval, combined with a user-friendly interface, advanced multimodal model (LLAVA using Ollama), and scalable backend, ensures a powerful and versatile solution for various use cases.