**Multimodal RAG App for Medical Use Cases**

In our application, we are developing a multimodal Retrieval-Augmented Generation (RAG) system using LangChain, specifically designed for the medical field. The application is structured around a knowledge graph built from various medical documents, often containing Information in multiple formats. This structure allows our application to provide precise answers and retrieve relevant images from these documents in response to user queries.

To enhance the application's core functionality, we have integrated GPT-4o, a powerful model capable of handling complex and diverse tasks. Additionally, we have incorporated an embedding model and a vector database. These components work together to improve the application's understanding of the content and enhance the accuracy of its responses. By combining these technologies, our RAG system is better equipped to deliver reliable and relevant Information in the medical domain.

**Step-by-step Process:**

1. **Extracting Information and Images from Documents:**

We used the unstructured library, a vital tool for handling tables and images in PDF documents, to extract pieces of Information and pictures from the papers.

1. **Classify Texts and Summarize:**

This processes a list of elements extracted from a PDF, identifying whether each component is text or a table. It then uses an AI model (GPT-3.5-turbo) to generate summaries for each text and table element. The original elements and their summaries are stored in separate lists for further use

1. **Process and Encode Images:**

This processes image files in a specified directory, encoding each image in base64 format. It then uses an AI model (GPT-4o) to generate summaries describing the contents of each image, explicitly focusing on images related to a dog's health. The encoded images and their corresponding summaries are stored in separate lists for further use.

1. **Create documents from text, table, and image elements:**

This process creates a list of documents from text, table, and image elements, along with their summaries. Each document is assigned a unique ID and metadata that includes the original content and its type (text, table, or image). These documents are then stored in a vector store using FAISS, a library for efficient similarity search. The vector store is built using embeddings generated by an AI model, enabling efficient retrieval and search of the documents based on their content. Vectors are then stored in local folders.

1. **Query and Answers:**

When a question is asked, the system searches for relevant documents using a similarity search. It then compiles the content of these documents into a context string, differentiating between text, tables, and images. The context and question are fed into an AI model (GPT-4o) using a predefined prompt, instructing the model to provide a detailed answer. If the model cannot answer confidently, it will decline the question. The final answer and any relevant images are returned.