# Task: Intelligent Document Processing and Query System

## Objective:

Develop a system that processes technical PDF documents, extracts key information, stores it in a vector database, and provides relevant responses to user queries using Retrieval-Augmented Generation (RAG).

## Requirements:

1. Document Processing:

- Accept 10 PDF files as input.

- Extract text content from each PDF.

- Split each document into logical sections (e.g., paragraphs or pages).

2. Information Extraction and Tagging:

- For each section, extract and tag the following information:

a. Equipment name

b. Domain (e.g., electronics, mechanical, software)

c. Model numbers

d. Manufacturer

3. Vector Database Integration:

- Choose and implement a suitable vector database (e.g., Pinecone, Weaviate, or Milvus, or any other of your choice).

- Convert each tagged section into a vector representation.

- Store the vectors along with their associated metadata (tags) in the database.

4. Query Processing:

- Implement a user interface to accept natural language queries.

- Extract for which equipment, model or manufacture is the query for.

- Convert user queries into vector representations.

- Perform cosine similarity search in the vector database to retrieve the most relevant sections for the matching (equipment, model or manufacturer)

5. Response Generation:

- Utilize a Language Model (e.g., GPT-3, GPT-4) for response generation.

- Use this API key if you do not have your own (key - sk-proj-3NAMKruBiPy16sQr1ixNT3BlbkFJmRPJIl1zNhn7qH2bD1dI). Make sure that activity on this key is monitored so use it only for this task.

- Design an effective prompt that incorporates the retrieved relevant sections and the user's query.

- Generate a coherent and informative response based on the retrieved information.

6. System Integration:

- Develop a Python application that integrates all the above components.

- Ensure smooth data flow from document processing to query response.

7. Performance and Scalability:

- Optimize the system for quick response times.

- Design the system to handle potential scaling to more documents in the future.

## Example Scenario:

Input: 10 PDF files containing technical specifications of various electronic devices.

User Query: "What is the power consumption of the latest XYZ Corp smartphone?"

## Expected System Behaviour:

1. Process and tag all 10 PDFs, storing information in the vector database.

2. Convert the user query to a vector.

3. Retrieve the most relevant section(s) from the database.

4. Generate a response using the LLM, incorporating the retrieved information.

5. Present the answer to the user, e.g., "The latest XYZ Corp smartphone, model ABC123, has a power consumption of 5W in standby mode and up to 15W during peak usage, according to the technical specifications."

## Deliverables:

1. Python code for the entire system.

2. Documentation explaining the architecture, chosen technologies, and how to run the system.

3. A brief report on the system's performance, including response times and accuracy.

## Tip:

Feel free to use LLM to generate code for you.