# DocuQuery: AI-Powered PDF Knowledge Assistant Using Google PALM

### Skills Required:

- Python

- NLP (Natural Language Processing)

- Streamlit for UI Development

- Google Gemini

## Project Description

DocuQuery is an AI-powered tool that enhances document processing by leveraging Google PALM for natural language understanding. The tool enables users to interact with PDF documents through intelligent querying, automated summaries, and knowledge extraction.

## Use Cases

### Price List Analyzer

Businesses dealing with multiple supplier price lists can use DocuQuery to extract and compare item prices efficiently. Users can upload multiple price lists, and the tool will extract relevant details, facilitating better procurement decisions.

### Research Paper Summarizer

Researchers can upload academic papers, and DocuQuery will generate concise summaries of key findings. Users can also pose questions related to the paper’s content, improving research efficiency.

### Resume Matcher for Hiring

Recruiters can upload resumes and specify job criteria. The tool will analyze and rank candidates based on their qualifications and experience, streamlining the hiring process.

## Technical Implementation

### Dependencies

- Streamlit for UI interaction

- PyPDF2 for extracting text from PDFs

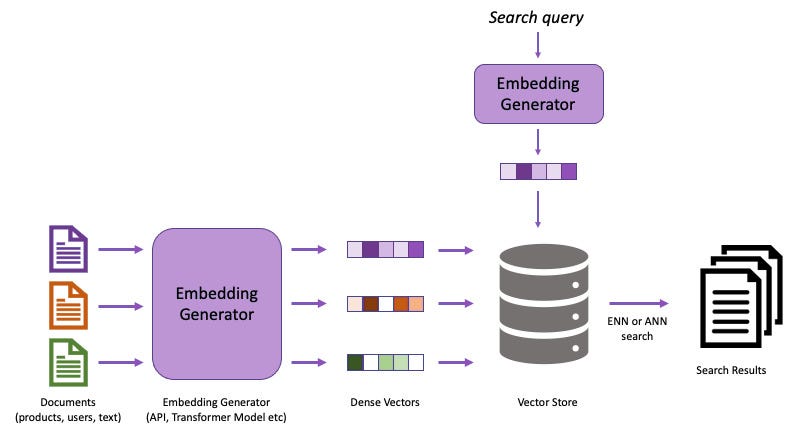
- Langchain for text processing and embedding

- Google Generative AI (PALM) for embeddings and conversational AI

- FAISS for vector storage and similarity search

- dotenv for environment variable management

### Architecture



**Project Structure:**

Create the Project folder which contains application file as shown below

A screenshot of a computer

AI-generated content may be incorrect.

**Milestone 1: Requirements Specification**

Specifying the required libraries in the requirements.txt file ensures seamless setup and reproducibility of the project environment, making it easier for others to replicate the development environment.

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

**A screen shot of a computer

AI-generated content may be incorrect.**

**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.

A screenshot of a computer

AI-generated content may be incorrect.

### A screenshot of a black screen AI-generated content may be incorrect.

### Activity 2: Define App logic and components

### Extracting PDF Text

from PyPDF2 import PdfReader  
  
A computer screen with text and images

AI-generated content may be incorrect.

### Splitting Text into Chunks

from langchain.text\_splitter import RecursiveCharacterTextSplitter  
  
A screenshot of a computer

AI-generated content may be incorrect.

### Creating a Vector Store

from langchain\_google\_genai import GoogleGenerativeAIEmbeddings  
from langchain\_community.vectorstores import FAISS  
  
A screen shot of text

AI-generated content may be incorrect.

### Milestone 3: Interfacing with Pre-trained Model

### In this milestone, we will build a prompt template to generate feedback based on the project details entered by the user.

### Conversational AI with Gemini

from langchain\_google\_genai import ChatGoogleGenerativeAI  
from langchain.chains.question\_answering import load\_qa\_chain  
from langchain.prompts import PromptTemplate  
  
A screen shot of a computer

AI-generated content may be incorrect.

### Handling User Input

import streamlit as st  
  
A computer screen with text on it

AI-generated content may be incorrect.

### Milestone 4: Model Deployment

### In this milestone, we are deploying the created model using streamlit. Model deployment using Streamlit involves creating a user-friendly web interface, enabling users to interact with the model through a browser. Streamlit provides easy-to-use tools for developing and deploying data-driven applications, allowing for seamless integration of models into web-based applications.

### Main Application

import streamlit as st  
  
A computer screen with many colorful text

AI-generated content may be incorrect.

**Run the web application**

* Open the anaconda prompt from the start menu
* Navigate to the folder where your Python script is.
* Now type “streamlit run app.py” command
* Navigate to the localhost where you can view your web page

A screen shot of a computer

AI-generated content may be incorrect.

Now, the application will open in the web browser,

A screenshot of a computer

AI-generated content may be incorrect.

Upload the document and click on submit and process to create embedding and store in FAISS.

A screenshot of a menu

AI-generated content may be incorrect.

Now ask questions like retrieve info

A screenshot of a computer

AI-generated content may be incorrect.

This will retirive the information from the embeedings stored in the vectore store and give the relevant information.

## Conclusion

## DocuQuery is a smart tool that makes reading PDFs easy. Whether finding prices, summarizing papers, or choosing the best resume, it helps users get quick answers. With future improvements, it will become even more powerful. DocuQuery is like a helpful reading buddy for everyone!