import streamlit as st

import streamlit.components.v1 as stc

# File Processing Pkgs

import pandas as pd

import docx2txt

import easyocr as ocr  #OCR

from PIL import Image

from PyPDF2 import PdfFileReader

import pdfplumber

from PyPDF2 import PdfReader

from langchain.embeddings.openai import OpenAIEmbeddings

from langchain.text\_splitter import CharacterTextSplitter

from langchain.vectorstores import FAISS

import cv2

import numpy as np

import pytesseract

import matplotlib.pyplot as plt

import pandas as pd

from PIL import Image

def main():

    st.title("Intelligent Document Processing")

    menu = ["Home", "About"]

    choice = st.sidebar.selectbox("Menu", menu)

    if choice == "Home":

        st.subheader("Home")

        file\_type = st.radio("Select file type:", ("JPEG", "PDF"))

        if file\_type == "JPEG":

            jpeg\_file = st.file\_uploader("Upload JPEG", type=['jpeg', 'jpg'])

            @st.cache

            def load\_model():

              reader = ocr.Reader(['en'],model\_storage\_directory='.')

              return reader

            reader = load\_model() #load model

            if jpeg\_file is not None:

              file\_details = {"Filename": jpeg\_file.name, "FileType": jpeg\_file.type, "FileSize": jpeg\_file.size}

              st.write(file\_details)

              input\_image = Image.open(jpeg\_file) #read image

              st.image(input\_image) #display image

              with st.spinner("🤖 AI is at Work! "):

                result = reader.readtext(np.array(input\_image))

                result\_text = [] #empty list for results

                for text in result:

                  result\_text.append(text[1])

                  result\_string = "\n".join(result\_text)

                st.write(result\_string)

    #st.success("Here you go!")

              st.balloons()

            else:

              st.write("Upload an Image")

            import os

            os.environ["OPENAI\_API\_KEY"]=""

            text\_splitter = CharacterTextSplitter(

                            separator = "\n",

                            chunk\_size=800,

                            chunk\_overlap=200,

                            length\_function=len,

                          )

            texts= text\_splitter.split\_text(result\_string)

            from langchain\_openai import OpenAIEmbeddings

            embeddings= OpenAIEmbeddings()

            document\_search = FAISS.from\_texts(texts, embeddings)

            from langchain.chains.question\_answering import load\_qa\_chain

            from langchain.llms import OpenAI

            chain=load\_qa\_chain(OpenAI(),chain\_type="stuff")

            query=st.text\_input("Ask something:")

            docs=document\_search.similarity\_search(query)

            answer1=chain.run(input\_documents=docs,question=query)

            st.write(answer1)

        elif file\_type == "PDF":

            pdf\_file = st.file\_uploader("Upload PDF", type=['pdf'])

            if pdf\_file is not None:

                file\_details = {"Filename": pdf\_file.name, "FileType": pdf\_file.type, "FileSize": pdf\_file.size}

                st.write(file\_details)

                try:

                    with pdfplumber.open(pdf\_file) as pdf:

                        page = pdf.pages[0]

                        answer=page.extract\_text()

                        import os

                        os.environ["OPENAI\_API\_KEY"]=""

                        text\_splitter = CharacterTextSplitter(

                            separator = "\n",

                            chunk\_size=800,

                            chunk\_overlap=200,

                            length\_function=len,

                          )

                        texts= text\_splitter.split\_text(answer)

                        from langchain\_openai import OpenAIEmbeddings

                        embeddings= OpenAIEmbeddings()

                        document\_search = FAISS.from\_texts(texts, embeddings)

                        from langchain.chains.question\_answering import load\_qa\_chain

                        from langchain.llms import OpenAI

                        chain=load\_qa\_chain(OpenAI(),chain\_type="stuff")

                        query=st.text\_input("Ask something:")

                        docs=document\_search.similarity\_search(query)

                        answer1=chain.run(input\_documents=docs,question=query)

                        st.write(answer1)

                        st.balloons()

                except:

                    st.write("Error reading PDF file.")

    else:

        st.subheader("Team Members")

        st.info("Megavarshini R")

        st.info("Srinivasan Akshaya Dharani")

        st.info("Sujitha S")

if \_\_name\_\_ == '\_\_main\_\_':

    main()