```python
import docx2txt
from apikey import OPENAI_API_KEY
import openai
import os
import streamlit as st
from langchain.text_splitter import CharacterTextSplitter
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.vectorstores import FAISS
from langchain.chains.question_answering import load_qa_chain
from langchain.llms import OpenAI
from langchain.callbacks import get_openai_callback
from PyPDF2 import PdfReader
from docx2pdf import convert
from langchain.agents import create_csv_agent


os.environ['OPENAI_API_KEY'] = OPENAI_API_KEY
openai.api_base = 'https://api.pawan.krd/v1'


def main():
    st.set_page_config(page_title="Ask your file")
    option = st.selectbox(
        'What is the format of your file?',
        ('.csv', '.pdf', '.docx'))




    if option == '.csv':
        st.header("Ask your csv □")

        # upload file
        csv_file = st.file_uploader("Upload your .csv", type=["csv"])
        if csv_file is not None:

            agent = create_csv_agent(
                OpenAI(temperature=0), csv_file, verbose=True)

            user_question = st.text_input("Ask a question about your CSV: ")

            if user_question is not None and user_question != "":
                with st.spinner(text="In progress..."):
```

```python
                st.write(agent.run(user_question))





if option == '.pdf':
    st.header("Ask your pdf □")

    # upload file
    pdf = st.file_uploader("Upload your PDF", type="pdf")

    # extract the text
    if pdf is not None:
        pdf_reader = PdfReader(pdf)
        text = ""
        for page in pdf_reader.pages:
            text += page.extract_text()

        # split into chunks
        text_splitter = CharacterTextSplitter(
            separator="\n",
            chunk_size=1000,
            chunk_overlap=200,
            length_function=len
        )
        chunks = text_splitter.split_text(text)

        # create embeddings
        embeddings = OpenAIEmbeddings()
        knowledge_base = FAISS.from_texts(chunks, embeddings)
        #
        # show user input
        user_question = st.text_input("Ask a question about your PDF:")
        if user_question:
            docs = knowledge_base.similarity_search(user_question)

            llm = OpenAI()
            chain = load_qa_chain(llm, chain_type="stuff")
            with get_openai_callback() as cb:
                response = chain.run(input_documents=docs, question=user_question)
                print(cb)

                st.write(response)





    if option == '.docx':
        st.header("Ask your docx □")

        # upload file
        docx_file = st.file_uploader("Upload your .docx", type=["docx"])
        if docx_file is not None:
            # Read CSV data into a pandas DataFrame
            text = docx2txt.process(docx_file)

            # # split into chunks
            text_splitter = CharacterTextSplitter(
                separator="\n",
                chunk_size=1000,
                chunk_overlap=200,
                length_function=len
            )
            chunks = text_splitter.split_text(text)
            # create embeddings
            embeddings = OpenAIEmbeddings()
            knowledge_base = FAISS.from_texts(chunks, embeddings)
            #
            # show user input
            user_question = st.text_input("Ask a question about your PDF:")
            if user_question:
                docs = knowledge_base.similarity_search(user_question)

                llm = OpenAI()
                chain = load_qa_chain(llm, chain_type="stuff")
                with get_openai_callback() as cb:
                    response = chain.run(input_documents=docs, question=user_question)
                    print(cb)

                st.write(response)
            #
            #
            #
            #
```

```python
            #
            #
            #






if __name__ == '__main__':
  main()


```