```python
import streamlit as st

from langchain_community.llms import Ollama

from langchain_community.document_loaders import WebBaseLoader

from langchain.document_loaders import PyPDFLoader

from langchain.embeddings import OllamaEmbeddings

from langchain.text_splitter import RecursiveCharacterTextSplitter

from langchain.chains.combine_documents import create_stuff_documents_chain

from langchain_core.prompts import ChatPromptTemplate

from langchain.chains import create_retrieval_chain

from langchain_community.vectorstores import FAISS

import time

st.title("Document-Based Q&A Chatbot")

#  Document Input Method Selection
input_option = st.radio("Select document input method:", ("URL", "Upload PDF"))

docs = None


if input_option == "URL":
    url = st.text_input("Enter URL to load documents from:")
    if url:
        try:
            loader = WebBaseLoader(url)
            docs = loader.load()
            st.success(f"Loaded {len(docs)} documents from URL")
        except Exception as e:
            st.error(f"Failed to load from URL: {e}")


elif input_option == "Upload PDF":
    uploaded_file = st.file_uploader("Upload your PDF document", type=["pdf"])
    if uploaded_file:
        try:
            loader = PyPDFLoader(uploaded_file)
            docs = loader.load()
            st.success(f"Loaded {len(docs)} pages from uploaded PDF")
        except Exception as e:
            st.error(f"Failed to load uploaded PDF: {e}")

# If docs are loaded, process and create vector store
if docs:
    try:
        # Text splitting
        text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
        split_docs = text_splitter.split_documents(docs)

        # Initialize embeddings
        embeddings = OllamaEmbeddings(model="qwen")
        embeddings = OllamaEmbeddings(model="qwen")

        # Create vector store
        vectors = FAISS.from_documents(split_docs, embeddings)

        # Save in session state
        st.session_state.docs = split_docs
        st.session_state.vectors = vectors
        st.session_state.embeddings = embeddings

        st.success("Vector store initialized and ready for queries!")

    except Exception as e:
        st.error(f"Error processing documents: {e}")

# If vector store is ready, allow user to query
if "vectors" in st.session_state:
    # Initialize language model
    llm = Ollama(model="qwen")

    # Define prompt template
    prompt_template = ChatPromptTemplate.from_template("""
Answer the questions based on the provided context only.
Please provide the most accurate response based on the question.
<context>
{context}
</context>
Questions:{input}
""")

    # Create document and retrieval chains
    document_chain = create_stuff_documents_chain(llm, prompt_template)
    retriever = st.session_state.vectors.as_retriever()
    retrieval_chain = create_retrieval_chain(retriever, document_chain)

    user_prompt = st.text_input("Input your prompt here")

    if user_prompt:
        st.write(f"User input received: {user_prompt}")
        start_time = time.time()

        try:
            response = retrieval_chain.invoke({"input": user_prompt})

            st.write(f"Response time: {time.time() - start_time:.2f} seconds")

            if 'answer' in response:
                st.write(response['answer'])
            else:
                st.error("Answer not found in the response.")

            if 'context' in response:
                with st.expander("Document Similarity Search"):
                    for i, doc in enumerate(response["context"]):
                        st.write(doc.page_content)
                        st.write("--------------------------------")
            else:
                st.error("Context not found in the response.")

        except Exception as e:
            st.error(f"An error occurred during the retrieval process: {e}")
```

```
else:
    st.info("Please load documents from a URL or upload a PDF file to start.")
```

Document-Based Q&A Chatbot

Select document input method:

[✓ URL]

Upload PDF

Enter URL to load documents from:

https://www.pinecone.io/learn/vector-database/

Loaded 1 documents from URL

Vector store initialized and ready for queries!

Input your prompt here

what is vector

User input received: what is vector

Response time: 32.62 seconds

Vector refers to a mathematical object with magnitude and direction. In the context of vector databases like Pinecone, vectors typically refer to the high-dimensional embeddings of objects in various domains such as computer vision, natural language processing, and many others.

Document Similarity Search