해당 코드를 app.py라는 파이썬 파일을 만들어 입력한 뒤 저장해주시면 됩니다

In [ ]:

```
#!pip install langchain streamlit PyPDF2 langchain-openai

import streamlit as st
from PyPDF2 import PdfReader
from langchain_openai import OpenAIEmbeddings
from langchain_classic.chat_models import ChatOpenAI
from langchain_classic.chains import ConversationalRetrievalChain, RetrievalQA
from langchain_classic.memory import ConversationBufferWindowMemory
from langchain_classic.vectorstores import FAISS
from langchain_classic.document_loaders import PyPDFLoader
from langchain_classic.text_splitter import RecursiveCharacterTextSplitter

import os
from dotenv import load_dotenv
load_dotenv()
OPENAI_API_KEY = os.getenv("OPENAI_API_KEY")

#PDF 문서에서 텍스트를 추출
def get_pdf_text(pdf_docs):
    text = ""
    for pdf in pdf_docs:
        pdf_reader = PdfReader(pdf)
        for page in pdf_reader.pages:
            text += page.extract_text()
    return text

#지정된 조건에 따라 주어진 텍스트를 더 작은 덩어리로 분할
def get_text_chunks(text):
    text_splitter = RecursiveCharacterTextSplitter(
        separators="\\n",
        chunk_size=1000,
        chunk_overlap=200,
        length_function=len
    )
    chunks = text_splitter.split_text(text)
```

```
    return chunks

#주어진 텍스트 청크에 대한 임베딩을 생성하고 FAISS를 사용하여 벡터 저장소를 생성
def get_vectorstore(text_chunks):
    embeddings = OpenAIEmbeddings(model="text-embedding-ada-002")
    vectorstore = FAISS.from_texts(texts=text_chunks, embedding=embeddings)
    return vectorstore

#주어진 벡터 저장소로 대화 체인을 초기화
def get_conversation_chain(vectorstore):
    #ConversationBufferWindowMemory에 이전 대화 저장
    memory = ConversationBufferWindowMemory(memory_key='chat_history', return_message=True)
    conversation_chain = ConversationalRetrievalChain.from_llm(
        llm=ChatOpenAI(temperature=0, model_name='gpt-4.1-mini'),
        retriever=vectorstore.as_retriever(),
        get_chat_history=lambda h: h,
        memory=memory
    ) #ConversationalRetrievalChain을 통해 Langchain 챗봇에 쿼리 전송
    return conversation_chain

user_uploads = st.file_uploader("파일을 업로드해주세요~", accept_multiple_files=True)
if user_uploads is not None:
    if st.button("Upload"):
        with st.spinner("처리중.."):
            # PDF 텍스트 가져오기
            raw_text = get_pdf_text(user_uploads)
            # 텍스트에서 청크 검색
            text_chunks = get_text_chunks(raw_text)
            # PDF 텍스트 저장을 위해 FAISS 벡터 저장소 만들기
            vectorstore = get_vectorstore(text_chunks)
            # 대화 체인 만들기
            st.session_state.conversation = get_conversation_chain(vectorstore)

if user_query := st.chat_input("질문을 입력해주세요~"):
    # 대화 체인을 사용하여 사용자의 메시지를 처리
    if 'conversation' in st.session_state:
        result = st.session_state.conversation({
            "question": user_query,
            "chat_history": st.session_state.get('chat_history', [])
        })
        response = result["answer"]
```

```
    else:
        response = "먼저 문서를 업로드해주세요~."
    with st.chat_message("assistant"):
        st.write(response)
```

만든 뒤 터미널에서 해당 폴더의 위치까지 이동한 뒤 streamlit run app.py 명령어로 streamlit을 실행합니다.