```
import os
print("DEBUG: GOOGLE_API_KEY =", os.getenv("AIzaSyDrqsCWeZSLqD3M_m-6Ut48N8jg9Yrmnws"))  # Ensure key is present
import PyPDF2

def get_pdf_text(pdf_docs):
   text = ""
   for pdf in pdf_docs:
      pdf_reader = PyPDF2.PdfReader(pdf)
      for page in pdf_reader.pages:
         page_text = page.extract_text()
         if page_text:
          text += page_text
   return text or "no valid text found in the uploaded pdf."



from PyPDF2 import PdfReader

def get_pdf_text(pdf_files):
   text = ""
   for pdf in pdf_files:
      pdf_reader = PdfReader(pdf)
      for page in pdf_reader.pages:
         text += page.extract_text() + "\n"
   return text

from langchain.text_splitter import RecursiveCharacterTextSplitter

def get_text_chunks(text, chunk_size=1000, overlap=20):
   splitter = RecursiveCharacterTextSplitter(chunk_size=chunk_size, chunk_overlap=overlap)
   return splitter.split_text(text)

from langchain_community.embeddings import GooglePalmEmbeddings
from langchain_community.vectorstores import FAISS
```

```
def store_embeddings(embeddings):
    print("DEBUG: Embeddings Type:", type(embeddings))  # Should be list
    print("DEBUG: Embeddings Value:", embeddings)       # Should display embedding
data

    if embeddings is None or len(embeddings) == 0:
        raise ValueError("Embedding generation failed. Check PDF content or API key.")

    dim = len(embeddings[0])  # Safeguard added
    vector_store = FAISS.from_texts(embeddings, dimension=dim)
    return vector_store
def get_vector_store(text_chunks):
 def get_text_embeddings(text_chunks):
     embedding_model = GooglePalmEmbeddings(model_name='models/embedding-
001',google_api_key='AIzaSyDrqsCWeZSLqD3M_m-6Ut48N8jg9Yrmnws')
     embeddings = embedding_model.embed_documents(text_chunks)

     embeddings = GooglePalmEmbeddings()
     vector_store = FAISS.from_texts(text_chunks, embeddings)
     return vector_store
     return embeddings



import faiss
import numpy as np

def store_embeddings(embeddings):
    print(embeddings)
    dim = len(embeddings[0])  # DimensionAIzaSyDrqsCWeZSLqD3M_m-6Ut48N8jg9
Yrmnws of embeddings
    index = faiss.IndexFlatL2(dim)  # Create FAISS index
    index.add(np.array(embeddings))  # Store embeddings
    return index

from langchain_community.llms import GooglePalm
from langchain.chains import ConversationalRetrievalChain
```

```
from langchain.memory import ConversationBufferMemory

def get_conversational_chain(vector_store):
    llm = GooglePalm()
    memory = ConversationBufferMemory(memory_key="chat_history", return_mess
ages=True)
    return ConversationalRetrievalChain(llm=llm, retriever=vector_store, memory=me
mory)

import streamlit as st

def main():
    st.set_page_config(page_title="DocuQuery: AI-Powered PDF Knowledge Assistan
t")
    st.header("DocuQuery: AI-Powered PDF Assistant")

    uploaded_files = st.sidebar.file_uploader("Upload PDFs", accept_multiple_files=Tr
ue)
    process_button = st.sidebar.button("Process")

    if process_button and uploaded_files:
        with st.spinner("Processing..."):
            text = get_pdf_text(uploaded_files)
            chunks = get_text_chunks(text)
            embeddings = get_vector_store(chunks)
            vector_store = store_embeddings(embeddings)
            conversation_chain = get_conversational_chain(vector_store)
        st.success("Processing complete! Ask questions below.")

        user_question = st.text_input("Ask a question:")
        if user_question:
            response = conversation_chain.run(user_question)
            st.write("Bot:", response)



if __name__ == "__main__":
```

main()