RAG-Based Document Chatbot Documentation

1. Source Code (app.py)

```
import streamlit as st
from langchain.document_loaders import PyMuPDFLoader
from langchain.text_splitter import CharacterTextSplitter
from langchain.vectorstores import FAISS
from langchain.embeddings import HuggingFaceEmbeddings
from langchain.chains import RetrievalQA
from transformers import AutoTokenizer, AutoModelForCausalLM, pipeline
from langchain.llms import HuggingFacePipeline
import tempfile
import os

st.set_page_config(page_title="Gemma Document Chatbot")
st.title("Ask Questions About Your PDF")

pdf_file = st.file_uploader("Upload a PDF", type="pdf")

if pdf_file:
    with tempfile.NamedTemporaryFile(delete=False, suffix=".pdf") as tmp_file:
        tmp_file.write(pdf_file.read())
        file_path = tmp_file.name
    st.success("PDF uploaded successfully!")

    loader = PyMuPDFLoader(file_path)
    documents = loader.load()
    text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
    docs = text_splitter.split_documents(documents)

    embedding = HuggingFaceEmbeddings(model_name="all-MiniLM-L6-v2")
    vectordb = FAISS.from_documents(docs, embedding)
    retriever = vectordb.as_retriever()

    st.info("Loading Gemma model...")
    model_id = "google/gemma-2b-it"
    os.environ["HUGGINGFACEHUB_API_TOKEN"] = "your_hf_token_here"

    tokenizer = AutoTokenizer.from_pretrained(model_id, use_auth_token=True)
    model = AutoModelForCausalLM.from_pretrained(model_id, device_map="auto", torch_dtype="auto", use_auth_token=True)

    pipe = pipeline("text-generation", model=model, tokenizer=tokenizer, max_new_tokens=512)
    llm = HuggingFacePipeline(pipeline=pipe)

    qa_chain = RetrievalQA.from_chain_type(llm=llm, retriever=retriever, return_source_documents=True)

    query = st.text_input("Ask a question about your PDF:")
    if query:
        with st.spinner("Retrieving answer..."):
            response = qa_chain.invoke({"query": query})
            st.markdown("Answer")
            st.write(response["result"])
```

2. Instructions to Run the Chatbot

1. Install Required Packages:

```
pip install streamlit pyngrok langchain transformers accelerate sentence-transformers faiss-cpu huggingface_hub pymupdf
```

2. Get a Hugging Face Access Token from:
   https://huggingface.co/settings/tokens

3. Replace "your_hf_token_here" in the code with your token.

4. Run the Streamlit App:

```
streamlit run app.py
```

5. (Optional) If you're using Colab, install and set up ngrok:

```
!pip install pyngrok
!ngrok config add-authtoken YOUR_NGROK_TOKEN
from pyngrok import ngrok
ngrok.connect(8501)
```

3. Tech Stack Used

- Frontend: Streamlit for interactive UI
- LLM: google/gemma-2b-it via Hugging Face Transformers
- Vector Store: FAISS for semantic search
- Embeddings: all-MiniLM-L6-v2 from Sentence Transformers
- Document Loader: PyMuPDF via LangChain
- Text Chunking: CharacterTextSplitter from LangChain

4. Response Structuring Approach

- PDF is split into 1000-character chunks with 200-character overlap
- Chunks are embedded using MiniLM
- FAISS retrieves the most relevant chunks based on user query
- The context and query are passed to the Gemma model for grounded answers

5. Challenges Faced and Solutions

1. Gated Model Access:

   Solved using Hugging Face access tokens and selecting open models like gemma-2b-it

2. LangChain API Changes:

   Updated deprecated .run() to .invoke()

3. Ngrok Authentication:

   Fixed by registering on ngrok.com and adding an authtoken

4. Colab Resource Limits:

   Used the lighter Gemma 2B model instead of Mistral 7B