CODE

app.py

```python
from flask import Flask, request
from twilio.twiml.messaging_response import MessagingResponse
from langchain_groq import ChatGroq
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
from langchain.docstore import InMemoryDocstore
from langchain_google_genai import GoogleGenerativeAIEmbeddings
from dotenv import load_dotenv
import os
import pickle
import faiss

# Load environment variables
load_dotenv()

# Load the GROQ and OpenAI API keys
groq_api_key = os.getenv('GROQ_API_KEY')
os.environ["GOOGLE_API_KEY"] = os.getenv("GOOGLE_API_KEY")

# Initialize Flask app
app = Flask(__name__)

# Initialize ChatGroq
llm = ChatGroq(groq_api_key=groq_api_key, model_name="Llama3-8b-8192")

# Create ChatPromptTemplate
prompt = 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}
   """
)

# Load vectors and documents
index = faiss.read_index("vectors.index")
with open("documents.pkl", "rb") as f:
   documents = pickle.load(f)

# Create the embeddings and docstore objects
```

```python
embedding_function =
GoogleGenerativeAIEmbeddings(model="models/embedding-001")
docstore = InMemoryDocstore({i: doc for i, doc in enumerate(documents)})

# Create the FAISS vector store
vectors = FAISS(embedding_function=embedding_function, docstore=docstore,
index=index, index_to_docstore_id={i: i for i in range(len(documents))})

# Handle incoming messages from Twilio webhook
@app.route('/webhook', methods=['POST'])
def webhook():
   incoming_msg = request.values.get('Body', '').lower()
   response = generate_response(incoming_msg)
   twilio_response = MessagingResponse()
   twilio_response.message(response)
   return str(twilio_response)

# Generate response to user's question
def generate_response(question):
   document_chain = create_stuff_documents_chain(llm, prompt)
   retriever = vectors.as_retriever()
   retrieval_chain = create_retrieval_chain(retriever, document_chain)
   response = retrieval_chain.invoke({'input': question})
   return response['answer']

# Main function to run the Flask app
if __name__ == '__main__':
   app.run(debug=True)
```

embed_pdf.py

```python
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_community.vectorstores import FAISS
from langchain_community.document_loaders import PyPDFDirectoryLoader
from langchain_google_genai import GoogleGenerativeAIEmbeddings
from dotenv import load_dotenv
import os
import faiss
import pickle

# Load environment variables
load_dotenv()

# Load the Google API key
os.environ["GOOGLE_API_KEY"] = os.getenv("GOOGLE_API_KEY")

# Function to perform vector embedding
def embed_pdfs():
   embeddings = GoogleGenerativeAIEmbeddings(model="models/embedding-001")
   loader = PyPDFDirectoryLoader("./constitution")  # Data Ingestion
   docs = loader.load()  # Document Loading
   text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000,
chunk_overlap=200)  # Chunk Creation
   final_documents = text_splitter.split_documents(docs[:20])  # Splitting
   vectors = FAISS.from_documents(final_documents, embeddings)  # Vector
OpenAI embeddings
   return vectors, final_documents

# Embed PDFs and save vectors
vectors, final_documents = embed_pdfs()
faiss.write_index(vectors.index, "vectors.index")

# Save the documents separately
with open("documents.pkl", "wb") as f:
   pickle.dump(final_documents, f)
```

1.first run python **embed_pdfs.py**

2. Then enter your ngrok token **./ngrok config add-authtoken 2grJiZv1WRzAfBi9ofwtq3xf47x_3GX3AWnAXXKj7GkvLoVgK**

3. Then run .\ngrok http 5000

4. Then copy e.g this https://b6b7-2c0f-fe38-2100-6a1e-70eb-2325-28bc-66b8.ngrok-free.app/webhook

To twilio sandbox settings by accessing twilio then pasting the link as shown above

5.Test it out