import os

import base64

import streamlit as st

from dotenv import load\_dotenv

from langchain\_community.embeddings import OpenAIEmbeddings

from langchain\_community.vectorstores import FAISS

from langchain.chat\_models import ChatOpenAI

from langchain.chains import RetrievalQA

from langchain\_community.document\_loaders import PyPDFLoader

from langchain.text\_splitter import RecursiveCharacterTextSplitter

# ─────────────────────────────────────────────────────────────────────────────

# Setup

# ─────────────────────────────────────────────────────────────────────────────

st.set\_page\_config(

page\_title="SHA — Bharat’s Full Stack Assistant",

page\_icon="💻",

layout="centered",

)

load\_dotenv()

OPENAI\_API\_KEY = st.secrets.get("OPENAI\_API\_KEY") or os.getenv("OPENAI\_API\_KEY")

# ─────────────────────────────────────────────────────────────────────────────

# Avatar display

# ─────────────────────────────────────────────────────────────────────────────

def show\_avatar():

file\_path = "shaavatar.png"

if os.path.exists(file\_path):

with open(file\_path, "rb") as f:

encoded = base64.b64encode(f.read()).decode()

st.markdown(f"""

<div style='text-align:center; margin-bottom:15px;'>

<img src="data:image/png;base64,{encoded}" width="120"

style="border-radius:50%; box-shadow:0 0 15px #7F5AF0;">

<h2 style='color:#E0E0E0; margin-top:10px;'>SHA — Bharat’s Full Stack Assistant</h2>

</div>

""", unsafe\_allow\_html=True)

show\_avatar()

# ─────────────────────────────────────────────────────────────────────────────

# Style

# ─────────────────────────────────────────────────────────────────────────────

st.markdown("""

<link href="https://fonts.googleapis.com/css2?family=Poppins&display=swap" rel="stylesheet">

<style>

html, body, [class\*="css"] {

background: linear-gradient(135deg, #0A0F2C 0%, #1B0033 100%);

color: #E0E0E0;

font-family: 'Poppins', sans-serif;

}

.stTextInput > div > div > input {

background-color: #1B1B2F;

color: #E0E0E0;

border: 1px solid #7F5AF0;

border-radius: 8px;

padding: 12px;

}

.stButton>button, button[kind="primary"] {

background-color: #7F5AF0 !important;

color: #FFFFFF !important;

border-radius: 8px;

padding: 8px 16px;

}

</style>

""", unsafe\_allow\_html=True)

# ─────────────────────────────────────────────────────────────────────────────

# Vector Store (build if not exists)

# ─────────────────────────────────────────────────────────────────────────────

VECTOR\_DIR = "sha\_vector\_store"

RESUME\_PATH = "resume/bharat\_resume.pdf"

if not os.path.exists(VECTOR\_DIR):

loader = PyPDFLoader(RESUME\_PATH)

pages = loader.load()

splitter = RecursiveCharacterTextSplitter(chunk\_size=500, chunk\_overlap=50)

docs = splitter.split\_documents(pages)

embeddings = OpenAIEmbeddings(openai\_api\_key=OPENAI\_API\_KEY)

FAISS.from\_documents(docs, embeddings).save\_local(VECTOR\_DIR)

# ─────────────────────────────────────────────────────────────────────────────

# Load QA chain

# ─────────────────────────────────────────────────────────────────────────────

store = FAISS.load\_local(

VECTOR\_DIR,

OpenAIEmbeddings(openai\_api\_key=OPENAI\_API\_KEY),

allow\_dangerous\_deserialization=True

)

qa\_chain = RetrievalQA.from\_chain\_type(

llm=ChatOpenAI(openai\_api\_key=OPENAI\_API\_KEY, temperature=0.1),

chain\_type="stuff",

retriever=store.as\_retriever()

)

# ─────────────────────────────────────────────────────────────────────────────

# Handlers (example: update/add your logic here as needed)

# ─────────────────────────────────────────────────────────────────────────────

def handle\_fun(q):

if "food" in q:

return "He runs on Java, JSON, and weekend biryani—strictly in that order."

if "hobbies" in q:

return "Building side projects, contributing to GitHub, and exploring cloud tools."

if "emoji" in q:

return "💻—because clean code is his love language."

return None

def handle\_company(q):

if "comcast" in q or "current job" in q:

return "Bharat is currently working at Comcast as a Full Stack Developer (June 2024–Present)."

if "jpmorgan" in q or "jpmc" in q:

return "Previously at JPMorgan Chase, he built microservices and SPAs using Spring Boot and ReactJS."

if "dentsu" in q:

return "At Dentsu, he worked on cloud-based healthcare dashboards using Angular and Azure."

return None

def handle\_education(q):

if "master" in q:

return "Bharat earned his Master’s in Computer Science from Wichita State University (2024)."

if "certification" in q:

return "Certifications include: AWS Developer, Oracle Java Programmer, and Power BI Analyst."

return None

# ─────────────────────────────────────────────────────────────────────────────

# Main Q&A

# ─────────────────────────────────────────────────────────────────────────────

if "miss\_count" not in st.session\_state:

st.session\_state["miss\_count"] = 0

st.markdown("### 🤖 Ask SHA anything about Bharat:")

user\_input = st.text\_input("Your Question:")

if user\_input:

q = user\_input.lower()

for fn in [handle\_fun, handle\_company, handle\_education]:

reply = fn(q)

if reply:

st.markdown(f"\*\*SHA:\*\* {reply}")

break

else:

docs = store.as\_retriever().get\_relevant\_documents(user\_input)

if not docs:

st.session\_state["miss\_count"] += 1

if st.session\_state["miss\_count"] == 1:

msg = "Hmm, that’s not in my memory yet. Try another question?"

elif st.session\_state["miss\_count"] == 2:

msg = "Still not finding anything—maybe Bharat didn’t include it in his resume."

else:

msg = "Alright, here’s my best guess… but you might want to ask Bharat directly 😄"

st.markdown(f"\*\*SHA:\*\* {msg}")

else:

st.session\_state["miss\_count"] = 0

with st.spinner("SHA is thinking..."):

answer = qa\_chain.run(user\_input)

st.markdown(f"\*\*SHA:\*\* {answer}")