import os

from dotenv import load\_dotenv

import streamlit as st

from langchain\_community.embeddings import OpenAIEmbeddings

from langchain\_community.vectorstores import FAISS

from langchain.chat\_models import ChatOpenAI

from langchain.chains import RetrievalQA

# load API key

load\_dotenv()

OPENAI\_KEY = os.getenv("OPENAI\_API\_KEY")

# initialize vector store

store = FAISS.load\_local(

"sha\_vector\_store",

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

allow\_dangerous\_deserialization=True

)

retriever = store.as\_retriever()

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

llm=ChatOpenAI(openai\_api\_key=OPENAI\_KEY, temperature=0),

chain\_type="stuff",

retriever=retriever

)

# initialize Streamlit

st.title("SHA — Your AI Assistant")

user\_input = st.text\_input("Ask me anything about Bharat")

# persistent miss counter

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

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

# Helper: fun/personal questions

def handle\_fun(q):

if any(w in q for w in ["girlfriend", "relationship", "single", "wife", "crush"]):

return "Haha, that’s classified! Bharat is deeply in love with clean code and Python 🐍."

if "favorite food" in q:

return "He lives on JSON, chai, and weekend biryani—strictly in that order."

if "age" in q:

return "Age is just a number—especially in a data table without a timestamp 😉."

if any(w in q for w in ["hobbies", "free time", "weekend"]):

return "Debugging tricky pipelines, reading AI papers, and sharing memes with fellow engineers."

if "personality" in q or "calm" in q or "composed" in q:

return "Bharat is calm, composed, and tackles challenges one data row at a time."

return None

# Helper: recruiter questions

def handle\_recruiter(q):

if any(w in q for w in ["sponsorship", "visa", "work authorization"]):

return (

"Bharat is on STEM OPT, authorized to work in the U.S., and married while waiting for his H4. "

"Future sponsorship may be needed depending on hire timeline."

)

if "notice period" in q:

return "He can join with about a 2-week notice, flexible for the right opportunity."

if "salary expectation" in q or "current salary" in q:

return (

"Bharat is open and flexible—happy to discuss compensation based on responsibilities, location, and growth potential."

)

if "open to relocation" in q or "relocation" in q:

return "He’s open to relocation, hybrid, or remote roles—whatever best serves the team."

if "remote" in q:

return "Absolutely—Bharat excels in both remote and collaborative office environments."

if "experience" in q and "data engineer" in q:

return "Bharat has 5+ years of data engineering experience across cloud, streaming, and analytics platforms."

return None

# Helper: company timeline questions

def handle\_company(q):

# current company

if "current company" in q or "where is bharat working" in q or "working now" in q:

return "He’s working at KLA as a Data Engineer since May 2024."

# Dentsu: 2020–2022

if "dentsu" in q:

return (

"At Dentsu (May 2020–May 2022), Bharat focused on Azure Data Factory, Spark, Kafka, Hive, and Power BI dashboards. "

"RAG and LLM tech weren’t in use then."

)

# Wichita State / Fagron transition details

if "wichita state" in q or "mscs" in q or "masters" in q:

return (

"Bharat did his Master’s in Computer Science at Wichita State University (Aug 2022–May 2024). "

"He interned as a BI Developer and Post-Production Lead before promotion to Data Engineer at Fagron."

)

# Fagron: 2022–2024

if "fagron" in q:

return (

"At Fagron (Dec 2022–Apr 2024), he built ETL on AWS Glue, Redshift, and Snowflake, "

"supported HIPAA/FDA compliance, and introduced a light-based verification system in post-production."

)

# KLA: 2024–present

if "kla" in q:

return (

"Currently at KLA (May 2024–Present), Bharat builds RAG pipelines with OpenAI and Azure Cognitive Search, "

"implements LLM assistants in Databricks, and designs real-time telemetry analytics for wafer defect detection."

)

return None

# Helper: tech deep-dive questions

def handle\_tech(q):

# RAG or Retrieval-Augmented Generation

if "rag" in q or "retrieval augmented generation" in q:

return (

"Bharat has hands-on RAG experience at KLA, where he set up vector search with OpenAI embeddings "

"and Azure Cognitive Search for semiconductor defect Q&A."

)

# LLM or large language model

if "llm" in q or "large language model" in q:

return (

"He uses LLMs at KLA inside Databricks for code suggestions, pipeline debugging, "

"and contextual Q&A since late 2023."

)

# Databricks optimization

if "optimize" in q and "databricks" in q:

return (

"He optimized Databricks pipelines by tuning Spark partition sizes, caching hot tables, "

"and refactoring PySpark jobs, reducing runtime by 40%."

)

# Kafka or streaming

if "kafka" in q or "streaming" in q:

return (

"At Dentsu and KLA, he used Kafka for real-time data ingestion, wrote consumers in PySpark Structured Streaming, "

"and ensured exactly-once delivery semantics."

)

# Airflow or orchestration

if "airflow" in q or "etl" in q:

return (

"He built DAGs in Apache Airflow for orchestration, with sensors, retries, and SLA alerts across Fagron and Dentsu pipelines."

)

# CI/CD or DevOps

if "jenkins" in q or "terraform" in q or "docker" in q or "ci/cd" in q:

return (

"He automated deployments using Jenkins pipelines, Terraform for infra-as-code on Azure/AWS, "

"and containerized jobs with Docker."

)

# cloud platforms

if "aws" in q or "azure" in q or "gcp" in q:

return (

"He’s worked with AWS (Glue, S3, Lambda), Azure (Data Factory, Synapse), and GCP (BigQuery) depending on project needs."

)

# BI tools

if "power bi" in q or "tableau" in q or "looker" in q:

return (

"He’s proficient in Power BI, Tableau, and Looker—built many dashboards for KPIs, operations, and compliance metrics."

)

return None

# Helper: education & certifications

def handle\_education(q):

if "master" in q or "wichita state" in q:

return (

"Bharat completed his Master’s in Computer Science at Wichita State University (Aug 2022–May 2024), "

"including a BI Developer internship and Post-Production Lead role."

)

if "bachelor" in q:

return (

"He earned his Bachelor’s in Engineering (Computer Science) and did early projects like Raspberry Pi face recognition in 2018."

)

if "certification" in q or "snowflake" in q or "databricks" in q:

return (

"He holds certifications in AWS Solutions Architect, Snowflake Data Engineering, "

"Databricks Certified Data Engineer, and Python/SQL proficiency."

)

return None

# Helper: project-specific questions

def handle\_projects(q):

if "sawyer" in q or "pybullet" in q or "robotic grasping" in q:

return (

"He built a Sawyer Arm simulation in PyBullet (Jan 2023–Mar 2023), implementing inverse kinematics and reward-based grasp testing."

)

if "face recognition" in q or "raspberry pi" in q:

return (

"He developed a Raspberry Pi face recognition system using OpenCV, Python, and a Pi camera for home security (2018)."

)

return None

# Helper: volunteer & leadership

def handle\_volunteer(q):

if "hackathon" in q or "expense tracker" in q:

return (

"He led a weekend hackathon team building a Python expense tracker with Power BI visuals to alert unusual spending (Mar 2023)."

)

if "guinness" in q or "wheelchair" in q or "coordinator" in q:

return (

"As a coordinator at Vel Tech (May 2019), he helped plan a Guinness World Record wheelchair event raising awareness of disabilities."

)

if "uvh cell" in q or "ethics" in q or "mindfulness" in q:

return (

"He volunteered in the UHV Cell (2018–19) focusing on ethics, empathy, and mindfulness, organizing weekly sessions."

)

if "ieee" in q:

return (

"He was an IEEE member (2017–18), attending tech talks and workshops to stay connected beyond the classroom."

)

return None

# Helper: behavioral / situational

def handle\_behavioral(q):

if "tell me about a time" in q or "example of" in q:

return (

"Sure—tell me the scenario you’d like, like improving pipeline performance or leading a cross-team project, and I’ll share details."

)

if "leadership" in q or "team" in q or "collaborate" in q:

return (

"He’s led cross-functional teams at Fagron and KLA, aligning stakeholders from product, QA, and operations to deliver on time."

)

if "challenge" in q or "problem" in q:

return (

"He faced a challenge optimizing Spark jobs at KLA—rewrote complex joins and introduced partition pruning, cutting runtime by 60%."

)

return None

# Main handler

if user\_input:

q\_lower = user\_input.lower()

# try each category in order

for handler in [

handle\_fun,

handle\_recruiter,

handle\_company,

handle\_tech,

handle\_education,

handle\_projects,

handle\_volunteer,

handle\_behavioral,

]:

result = handler(q\_lower)

if result:

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

break

else:

# resume memory + fallback

docs = 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. Want to try asking something else?"

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

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

else:

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

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}")

import os

import base64

import streamlit as st

# Pull your OpenAI key from Streamlit Cloud’s Secrets

OPENAI\_API\_KEY = st.secrets["OPENAI\_API\_KEY"]

from langchain\_community.embeddings import OpenAIEmbeddings

from langchain\_community.vectorstores import FAISS

from langchain.chat\_models import ChatOpenAI

from langchain.chains import RetrievalQA

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

# Page config

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

st.set\_page\_config(

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

page\_icon="👩‍🚀",

layout="centered",

)

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

# Avatar display

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

def show\_sha\_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 — Your AI Companion</h2>

</div>

""",

unsafe\_allow\_html=True

)

show\_sha\_avatar()

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

# Custom CSS for cosmic theme

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

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;

}

.stMarkdown, .stText {

line-height: 1.6;

}

</style>

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

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

# Auto-generate vector store if missing (for Streamlit deploy)

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

from langchain\_community.document\_loaders import PyPDFLoader

from langchain.text\_splitter import RecursiveCharacterTextSplitter

if not os.path.isdir("sha\_vector\_store"):

loader = PyPDFLoader("resume/bharat\_resume.pdf")

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("sha\_vector\_store")

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

# Load SHA’s memory (vector store) and QA chain

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

store = FAISS.load\_local(

"sha\_vector\_store",

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()

)

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

# Initialize miss\_count for fallback logic

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

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

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

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

# Handler: Fun / Personal Questions

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

def handle\_fun(q):

if any(w in q for w in ["girlfriend", "relationship", "single", "wife", "crush"]):

return "Haha, that’s classified! Bharat is more in love with data pipelines than dating apps."

if "favorite food" in q:

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

if "age" in q:

return "Age is just metadata—especially if there’s no timestamp 😉."

if any(w in q for w in ["hobbies", "free time", "weekend"]):

return "Debugging tricky pipelines, reading AI papers, and sharing memes with fellow engineers."

if any(w in q for w in ["calm", "composed", "personality"]):

return "Bharat is calm, composed, and tackles challenges one data row at a time."

return None

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

# Handler: Recruiter Screening Questions

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

def handle\_recruiter(q):

if any(w in q for w in ["sponsorship", "visa", "work authorization"]):

return (

"Bharat is on STEM OPT, authorized to work in the U.S., and married while waiting for his H4. "

"Future sponsorship may be needed depending on the hire timeline."

)

if "notice period" in q:

return "He can join with about a 2-week notice—flexible for the right opportunity."

if any(w in q for w in ["salary expectation", "current salary", "expected salary"]):

return "Bharat is open and flexible—happy to discuss compensation based on role, location, and growth potential."

if any(w in q for w in ["open to relocation", "relocation"]):

return "He’s open to relocation, hybrid, or remote roles—whatever best serves the team."

if "remote" in q:

return "Absolutely—Bharat excels in both remote and in-office environments."

if "experience" in q and "data engineer" in q:

return "Bharat has 5+ years of data engineering experience across cloud, streaming, and analytics platforms."

return None

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

# Handler: Company Timeline & Roles

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

def handle\_company(q):

if any(t in q for t in ["current company", "present company", "where are you working", "working now"]):

return "Bharat is currently working at \*\*KLA\*\* as a Data Engineer since May 2024."

if "dentsu" in q:

return (

"At Dentsu (May 2020–May 2022), Bharat built data pipelines using Azure Data Factory, Spark, Kafka, "

"and Power BI dashboards. RAG and LLM tech weren't used then."

)

if any(t in q for t in ["wichita state", "master", "mscs"]):

return (

"Bharat completed his Master’s in Computer Science at Wichita State University (Aug 2022–May 2024), "

"including BI Developer and Post-Production Lead internships."

)

if "fagron" in q:

return (

"At Fagron (Dec 2022–Apr 2024), he built ETL on AWS Glue, Redshift, and Snowflake, supported HIPAA/FDA compliance, "

"and introduced a light-based verification system in post-production."

)

if "kla" in q:

return (

"At KLA (May 2024–Present), Bharat builds RAG pipelines using OpenAI and Azure Cognitive Search, "

"implements LLM assistants in Databricks, and designs real-time analytics for wafer defect detection."

)

return None

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

# Handler: Tech Deep-Dive Questions

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

def handle\_tech(q):

if "rag" in q or "retrieval augmented generation" in q:

return (

"Bharat has hands-on RAG experience at KLA, where he set up vector search with OpenAI embeddings "

"and Azure Cognitive Search for semiconductor defect Q&A."

)

if "llm" in q or "large language model" in q:

return (

"He uses LLMs at KLA inside Databricks for code suggestions, pipeline debugging, "

"and contextual Q&A since late 2023."

)

if "optimize" in q and "databricks" in q:

return (

"He optimized Databricks pipelines by tuning Spark partition sizes, caching hot tables, "

"and refactoring PySpark jobs, reducing runtime by 40%."

)

if "kafka" in q or "streaming" in q:

return (

"At Dentsu and KLA, he used Kafka for real-time data ingestion, wrote consumers in "

"PySpark Structured Streaming, and ensured exactly-once delivery semantics."

)

if "airflow" in q or "etl" in q:

return (

"He built DAGs in Apache Airflow for orchestration, with sensors, retries, and SLA alerts "

"across Fagron and Dentsu pipelines."

)

if any(w in q for w in ["jenkins", "terraform", "docker", "ci/cd"]):

return (

"He automated deployments using Jenkins, Terraform for infra-as-code, and containerized jobs with Docker."

)

if any(w in q for w in ["aws", "azure", "gcp"]):

return (

"He’s worked with AWS (Glue, S3, Lambda), Azure (Data Factory, Synapse), and GCP (BigQuery) "

"depending on project needs."

)

if any(w in q for w in ["power bi", "tableau", "looker"]):

return (

"He’s proficient in Power BI, Tableau, and Looker—built many dashboards for KPIs, operations, and compliance."

)

return None

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

# Handler: Education & Certifications

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

def handle\_education(q):

if any(w in q for w in ["master", "wichita state"]):

return (

"Bharat completed his Master’s in Computer Science at Wichita State University (Aug 2022–May 2024)."

)

if "bachelor" in q:

return (

"He earned his Bachelor’s in Engineering (Computer Science) in 2018 and built early projects like Raspberry Pi face recognition."

)

if any(w in q for w in ["certification", "certified"]):

return (

"He holds certifications: AWS Solutions Architect, Databricks Certified Data Engineer, "

"Snowflake Data Engineer, and Python & SQL certifications."

)

return None

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

# Handler: Projects

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

def handle\_projects(q):

if any(w in q for w in ["sawyer", "pybullet", "grasping"]):

return (

"He built a Sawyer Arm simulation in PyBullet (Jan 2023–Mar 2023), implementing inverse kinematics and reward-based grasp testing."

)

if any(w in q for w in ["face recognition", "raspberry pi"]):

return (

"He developed a Raspberry Pi face recognition system using Python and OpenCV for home security (2018)."

)

if "expense tracker" in q or "hackathon" in q:

return (

"He led a weekend hackathon building a Python expense tracker with Power BI visuals to alert on unusual spending (Mar 2023)."

)

return None

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

# Handler: Volunteer & Leadership

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

def handle\_volunteer(q):

if any(w in q for w in ["guinness", "wheelchair", "coordinator"]):

return (

"As a coordinator at Vel Tech (May 2019), he helped plan a Guinness World Record wheelchair event raising disability awareness."

)

if any(w in q for w in ["uvh", "ethics", "mindfulness"]):

return (

"He volunteered in the UHV Cell (2018–19), organizing sessions on ethics, empathy, and mindfulness."

)

if "ieee" in q:

return (

"He was an IEEE member (2017–18), attending workshops and tech talks to stay connected beyond class."

)

return None

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

# Handler: Behavioral / Situational

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

def handle\_behavioral(q):

if any(w in q for w in ["tell me about a time", "example of", "how did you"]):

return (

"Sure—tell me which scenario you'd like, such as optimizing pipeline performance or leading a project, and I'll share the details."

)

if any(w in q for w in ["leadership", "team", "collaborate"]):

return (

"He’s led cross-functional teams at Fagron and KLA, aligning product, QA, and engineering to deliver on time."

)

if any(w in q for w in ["challenge", "problem"]):

return (

"He faced a challenge optimizing Spark jobs at KLA—rewrote complex joins and used partition pruning to cut runtime by 60%."

)

return None

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

# Main interaction

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

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

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

if user\_input:

q\_lower = user\_input.lower()

for handler in [

handle\_fun,

handle\_recruiter,

handle\_company,

handle\_tech,

handle\_education,

handle\_projects,

handle\_volunteer,

handle\_behavioral,

]:

resp = handler(q\_lower)

if resp:

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

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. Want to try asking something else?"

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

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

else:

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

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}")