SMART CHATBOT USING RAG

```
# Install LangChain core and Google Generative AI support
!pip install -qU langchain-core langgraph>=0.2.27 "langchain[google-genai]"

# Install PDF reading library
!pip install -qU PyPDF2

# (Optional) For better PDF extraction, you can also install pdfplumber
!pip install -qU pdfplumber

!pip install langchain-community langchain-core

!pip install -q faiss-cpu

# 🌐 Step 0: Set Google Gemini API Key
import getpass
import os

if not os.environ.get("GOOGLE_API_KEY"):
  os.environ["GOOGLE_API_KEY"] = getpass.getpass("Enter API key for Google Gemini: ")

# 🚀 Step 1: Install Required Packages (if needed)
!pip install -q langchain-core langgraph>=0.2.27 "langchain[google-genai]" faiss-cpu PyPDF2

# ✅ Step 2: Initialize Gemini model
from langchain.chat_models import init_chat_model
model = init_chat_model("gemini-2.0-flash", model_provider="google_genai")
```

```
# 📄 Step 3: Load PDF and chunk it
from PyPDF2 import PdfReader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_google_genai.embeddings import GoogleGenerativeAIEmbeddings
from langchain_community.vectorstores import FAISS

reader = PdfReader(r"about_me.pdf")  # Upload your PDF
raw_text = "".join([page.extract_text() for page in reader.pages])
splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
chunks = splitter.split_text(raw_text)

# 🧠 Step 4: Embed chunks using Gemini Embeddings + store in FAISS
embeddings = GoogleGenerativeAIEmbeddings(model="models/embedding-001")
vectorstore = FAISS.from_texts(chunks, embedding=embeddings)

# 🔍 Step 5: Create retriever from FAISS
retriever = vectorstore.as_retriever()

# 🤖 Step 6: Hybrid RAG function
def hybrid_ask(query: str):
    # Get top relevant chunks from the PDF
    docs = retriever.get_relevant_documents(query)
    context = "\n\n".join([doc.page_content for doc in docs])

    # Combine retrieved text with user question
    prompt = f"""You are a helpful assistant.
Answer the question based on the context below.
If the context is not relevant, answer from your own general knowledge.
```

```
Context:
{context}

Question: {query}
"""
    # Ask Gemini
    response = model.invoke(prompt)
    return response.content

# 💬 Step 7: Chat loop
print("🤖 Chatbot ready! Type your question or 'exit' to quit.\n")
while True:
    user_input = input("You: ")
    if user_input.lower().strip() == "exit":
        print("👋 Exiting chatbot. Goodbye!")
        break
    answer = hybrid_ask(user_input)
    print("AI:", answer, "\n")
```