```
# ----------------------  P R O J E C T ----------------------------------------
#                       HR Assistance - Chatbot
#
================================================================================
=
# Platform in use: Linux Mint with 2 GB RAM + 2 GB swap mem
# purpose: Run Llama local from PC/Edge devices with offline model
# Author: Soumen Dey / For CEP Project - GenAi
# Use case: Develope chatbot using llama and vectorDb, context: local pdf file
# Tech stack: llama3, vectorstore as chromadb, gradio for chat interface, Q2_k
quantized model for edge devices.
# platform build: Linux Mint 21.3 Virginia 64-bit, Intel® Celeron(R) CPU N3050 @
1.60GHz × 2 , 2GB RAM
# operating env: Linux arm64 bit with minimum 2 GB RAM + 2 GB swap disk
# Date : May-2025, India

#------------------------- F O L D E R   S T R U C T U R E ---------------------

# folder structure
    # code
    #   |____  bin                        <--- This is your llama.cpp binary
    #   |____  llama-cpp-python           <--- This is llma-cpp-python binary
    #   |____  pdf                        <--- pdf file location to store in
vectordb
    #   |____  chatBotWithGradio.py       <--- python script to run for chatbot

#================================================================================


# ------------------------- I N S T R U C T I O N S ----------------------

# 1: Download the binary: https://github.com/ggml-org/llama.cpp/releases
# 2: Or Build the llama.cpp using make from:

# some important pkg to install
#   pip install fitz, forntend, tools, gradio
#   pip install PyMuPDF --upgrade

#   TMPDIR=/path/to/bigger/temp pip install sentence-transformers

print ('\n\nhello python from llama.cpp \n')

# export LD_LIB_PATH=/bin:$LD_LIB_PATH

import os
os.environ['LD_LIB_PATH'] = '/bin' # this is the llma.cpp binary/executables
from llama_cpp import Llama

print ("Starting app....[llama.cpp found]")

#----------------------------------------
# Builindg llma-cpp-python
# Clone with submodules
    # git clone --recurse-submodules https://github.com/abetlen/llama-cpp-
python.git
    # cd llama-cpp-python
    # git submodule update --init --recursive
```

```
    # If you're not using GPU:  on otherwise
    # CMAKE_ARGS="-DLLAMA_CUBLAS=off" pip install llama-cpp-python --force-
reinstall --no-cache-dir

import sys
sys.path.insert(0, "llama-cpp-python")

# TMPDIR=/path/to/bigger/temp pip install sentence-transformers

#----------------------------------------
# pdffile = "./pdf/the_nestle_hr_policy_pdf_2012.pdf"
pdffile = "./pdf/the_nestle_hr_policy_pdf_2012.pdf"

#model_path = "/home/rimbik/Other_Drive/LlaMa/llama.cpp/models/Llama-3.2-3B-
Instruct-Q4_K_M/Llama-3.2-3B-Instruct-Q4_K_M.gguf"
# Q2 Quantized model to run on 2GB RAM only
#download from
#https://huggingface.co/featherless-ai-quants/EdgerunnersArchive-Llama-3-8B-
Instruct-ortho-baukit-toxic-v2-GGUF/tree/main
model_path = "./model/Llama-3-8b-Q2_Quantized/EdgerunnersArchive-Llama-3-8B-
Instruct-ortho-baukit-toxic-v2-Q2_K.gguf"

print ("pdf loaded")
print ("Model found and loaded......")

# Define path to your persistent directory
persist_path = "static"

# Ensure the directory exists
import os
os.makedirs(persist_path, exist_ok=True)

print ("\nStep:1.....................")


#-------------------------------------------------
    # pip uninstall fitz PyMuPDF
    # pip install PyMuPDF
#-------------------------------------------------
import fitz  # PyMuPDF for PDF reading
from sentence_transformers import SentenceTransformer
import chromadb

print ("\nStep:2 : trying embedding.............")



# 1. Load the embedding model
embed_model = SentenceTransformer("all-MiniLM-L6-v2")

print ("\nStep:3: embedding loaded.............")


# 2. Function to read PDF and extract text
def read_pdf(file_path):
    doc = fitz.open(file_path)
    text = ""
    for page in doc:
        text += page.get_text()
```

```
    return text

print("reading pdf")
# 3. Read local PDF file
pdf_text = read_pdf(pdffile)

print("processing data chunk for vector db")
# pip install --upgrade transformers torch


# 4. Split the text into manageable chunks (adjust based on document size)
chunk_size = 500  # Customize based on your needs
text_chunks = [pdf_text[i:i + chunk_size] for i in range(0, len(pdf_text),
chunk_size)]

print("step:4 over")

# 5. Embed the text chunks using the sentence-transformer model
doc_embeddings = embed_model.encode(text_chunks)

print("step:5 over")

# 6. Initialize ChromaDB client and create a collection (if not already created)
client = chromadb.Client()
collection = client.create_collection("pdf_collection")

# 7. Add the chunks and embeddings to ChromaDB
for i, chunk in enumerate(text_chunks):
    collection.add(
        documents=[chunk],
        metadatas=[{"source": f"chunk_{i}"}],
        embeddings=[doc_embeddings[i]],
        ids=[str(i)]
    )


# Load your LLaMA model with llama-cpp-python
llm = Llama(model_path=model_path)  # Change path to your GGUF model

print("\nmodel loaded sucessfully. Time to execute ...")

# ----- Gradio Interface Function -----
--------------------------------------------------------------------------------
----------

def pause_chat(pause_state, chatbot_history):
  """Toggles the pause state."""
  if pause_state == True:
    return [pause_state, chatbot_history],  # Return True (pause state) and the
current history
  else:
    return [not pause_state, chatbot_history] # Return False (unpause state) and
the current history


def answer_question(prompt, history):
    query_embedding = embed_model.encode([prompt])
    results = collection.query(query_embeddings=query_embedding, n_results=3)
    retrieved_context = "\n".join(results['documents'][0])
```

```
    MAX_CONTEXT_CHARS = 1500
    trimmed_context = retrieved_context[:MAX_CONTEXT_CHARS]


    final_prompt = f"""You are a helpful assistant.

Answer the following question **only** using the context provided.
If the context does not contain the answer, respond with "I am sorry - I do not
know. \nSeems answer not available in the repository!".

Context:
{trimmed_context}

Question: {prompt}
Answer:"""

    response = llm(final_prompt, max_tokens=200)
    return response["choices"][0]["text"].strip()

# ----- Gradio UI -----
import gradio as gr

myInterface = gr.ChatInterface(
    answer_question,
    type="messages",
    chatbot=gr.Chatbot(height=500),
    title="llama: AI-Powered HR Assistant",
    description="model: EdgerunnersArchive-Llama-3-8B-Instruct-ortho-baukit-toxic-
v2-Q2_K.gguf:local, powered by (llama   /hf   ): rimbik",
    theme="ocean",
    cache_examples=True,
    multimodal=True,
    textbox = gr.MultimodalTextbox(placeholder="Ask something based on your Nestle
HR Policy PDF...", container=False, scale=7, sources=["microphone"]),

)

myInterface.launch()

# --------------------------- END ------------------------------
```