**AI MOCK INTERVIEW SIMULATOR:CODE**

import tkinter as tk

from tkinter import scrolledtext, ttk

import threading

import sounddevice as sd

import queue

import sys

import pyttsx3

import numpy as np

from vosk import Model, KaldiRecognizer

from gpt4all import GPT4All

# --- TTS ---

tts = pyttsx3.init()

def speak(text):

tts.say(text)

tts.runAndWait()

# --- STT ---

q = queue.Queue()

samplerate = 16000

vosk\_model = Model("vosk-model-en-us-0.22")

rec = KaldiRecognizer(vosk\_model, samplerate)

recording = False

user\_answers = [] # Collect user answers for evaluation later

def callback(indata, frames, time, status):

if status:

print(status, file=sys.stderr)

q.put(bytes(indata))

def listen\_until\_stop():

final\_text = ""

try:

with sd.RawInputStream(samplerate=samplerate, blocksize=8000, dtype='int16',

channels=1, callback=callback):

while recording:

if not q.empty():

data = q.get()

if rec.AcceptWaveform(data):

result = rec.Result()

text = eval(result).get("text", "")

print(f"Interim text: {text}")

if text:

final\_text += " " + text

# Get final text too

partial\_result = rec.FinalResult()

partial\_text = eval(partial\_result).get("text", "")

if partial\_text:

print(f"Final text: {partial\_text}")

final\_text += " " + partial\_text

except Exception as e:

print(f"Error during listen: {e}")

return final\_text.strip()

# --- LLM ---

llm = GPT4All("mistral-7b-instruct-v0.1.Q4\_0.gguf", model\_path="model/")

def chat(user\_answer, current\_question, role):

prompt = (

f"You are a mock interviewer for a {role} role.\n"

f"The candidate says: '{user\_answer}'\n"

f"Give a natural, conversational, short follow-up or next question."

)

return llm.generate(prompt, max\_tokens=80).strip()

def evaluate\_answers(answers, role):

answers\_text = "\n".join(f"Answer {i+1}: {ans}" for i, ans in enumerate(answers))

prompt = (

f"You are a professional interviewer evaluating a mock interview for a {role} position.\n"

f"Here are the candidate's answers:\n{answers\_text}\n"

f"Give a final evaluation score out of 10 with 1-2 lines of feedback."

)

return llm.generate(prompt, max\_tokens=100).strip()

# --- GUI App ---

class InterviewApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Mock Interview AI")

self.root.geometry("800x700")

self.root.configure(bg="#121212")

self.interview\_active = False

self.current\_question = "Can you please introduce yourself?"

self.selected\_role = tk.StringVar(value="Data Analyst")

title = tk.Label(root, text="👨‍💼 Mock Interview AI", font=("Helvetica", 20, "bold"), bg="#121212", fg="#ffffff")

title.pack(pady=10)

role\_frame = tk.Frame(root, bg="#121212")

role\_frame.pack(pady=5)

tk.Label(role\_frame, text="Select Role:", bg="#121212", fg="white", font=("Helvetica", 12)).pack(side=tk.LEFT)

self.role\_menu = ttk.Combobox(role\_frame, textvariable=self.selected\_role, values=[

"Data Analyst", "Software Developer", "Product Manager", "Marketing Executive"

], font=("Helvetica", 12), state="readonly", width=30)

self.role\_menu.pack(side=tk.LEFT, padx=10)

self.chat\_box = scrolledtext.ScrolledText(root, wrap=tk.WORD, font=("Helvetica", 12), state="disabled",

bg="#1e1e1e", fg="#ffffff", insertbackground="white", padx=10, pady=10)

self.chat\_box.pack(padx=20, pady=10, fill=tk.BOTH, expand=True)

input\_frame = tk.Frame(root, bg="#121212")

input\_frame.pack(pady=5, fill=tk.X)

self.entry = tk.Entry(input\_frame, font=("Helvetica", 12), bg="#1e1e1e", fg="white", insertbackground="white")

self.entry.pack(side=tk.LEFT, padx=10, pady=5, fill=tk.X, expand=True)

self.send\_btn = tk.Button(input\_frame, text="Send", command=self.send\_text, font=("Helvetica", 12),

bg="#ff0050", fg="white", activebackground="#e60045", width=8)

self.send\_btn.pack(side=tk.RIGHT, padx=10)

controls = tk.Frame(root, bg="#121212")

controls.pack(pady=10)

self.start\_btn = tk.Button(controls, text="🎤 Start Interview", command=self.toggle\_interview,

font=("Helvetica", 12), bg="#4CAF50", fg="white", activebackground="#43a047", width=18)

self.start\_btn.grid(row=0, column=0, padx=10)

self.rec\_btn = tk.Button(controls, text="🎙️ Start Recording", command=self.start\_recording,

font=("Helvetica", 12), bg="#ff9800", fg="white", activebackground="#fb8c00", width=18)

self.rec\_btn.grid(row=0, column=1, padx=10)

self.stop\_rec\_btn = tk.Button(controls, text="⏹️ Stop Recording", command=self.stop\_recording,

font=("Helvetica", 12), bg="#f44336", fg="white", activebackground="#e53935", width=18)

self.stop\_rec\_btn.grid(row=0, column=2, padx=10)

# Greet and ask first question

self.say\_and\_display("Interviewer", self.current\_question)

def say\_and\_display(self, speaker, message):

self.chat\_box.configure(state="normal")

self.chat\_box.insert(tk.END, f"{speaker}: {message}\n\n")

self.chat\_box.configure(state="disabled")

self.chat\_box.yview(tk.END)

speak(message)

def toggle\_interview(self):

global user\_answers

self.interview\_active = not self.interview\_active

if self.interview\_active:

user\_answers = []

self.start\_btn.config(text="⏹ Stop Interview", bg="#f44336")

self.say\_and\_display("Interviewer", "Let's begin the mock interview.")

else:

self.start\_btn.config(text="🎤 Start Interview", bg="#4CAF50")

self.say\_and\_display("Interviewer", "Interview stopped. Thank you!")

role = self.selected\_role.get()

score = evaluate\_answers(user\_answers, role)

self.say\_and\_display("Evaluation", score)

def send\_text(self):

if not self.interview\_active:

return

user\_input = self.entry.get().strip()

if not user\_input:

return

self.entry.delete(0, tk.END)

self.say\_and\_display("You", user\_input)

user\_answers.append(user\_input)

threading.Thread(target=self.respond, args=(user\_input,)).start()

def respond(self, user\_input):

role = self.selected\_role.get()

response = chat(user\_input, self.current\_question, role)

self.current\_question = response

self.say\_and\_display("Interviewer", response)

def start\_recording(self):

global recording

if not self.interview\_active:

return

recording = True

self.say\_and\_display("System", "Recording started. Speak now...")

self.listen\_thread = threading.Thread(target=self.listen\_and\_process)

self.listen\_thread.start()

def stop\_recording(self):

global recording

if not self.interview\_active:

return

recording = False

self.say\_and\_display("System", "Recording stopped. Processing...")

self.root.after(500, lambda: print("Allowing buffer to process..."))

def listen\_and\_process(self):

text = listen\_until\_stop()

if text:

self.say\_and\_display("You", text)

user\_answers.append(text)

self.respond(text)

# --- Run App ---

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = InterviewApp(root)

speak("Welcome to Mock Interview AI. You can type or speak your answers.")

root.mainloop()