Week 8 — Track 1: Voice-Interactive LLM (Speech→LLM→Speech)

This notebook scaffolds a simple pipeline:

1. Capture audio → 2) Transcribe (Whisper or SpeechRecognition) → 3) Query an LLM → 4) Synthesize audio reply.

You can swap in your project model (Week 6/7) in the `run_llm()` function.

```
!pip -q install SpeechRecognition pydub gTTS soundfile
!pip -q install openai-whisper
import os, io, time
from typing import Optional
print('Ready.')
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 32.9/32.9 MB 75.7 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.2/98.2 kB 10.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 803.2/803.2 kB 44.2 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
  Building wheel for openai-whisper (pyproject.toml) ... done
Ready.
```

1) Load/record audio

```
AUDIO_PATH = 'sample.wav'  # replace with your file path
print('Set AUDIO_PATH to your file.')
from IPython.display import Audio, display, Javascript
from google.colab import output
import base64, io, soundfile as sf

def record(sec=5):
    print("🎤 Recording... speak for", sec, "seconds.")
    display(Javascript("""
    async function record(sec){
      const stream = await navigator.mediaDevices.getUserMedia({audio:true});
      const recorder = new MediaRecorder(stream);
      const data = [];
      recorder.ondataavailable = e => data.push(e.data);
      recorder.start();
      await new Promise(r => setTimeout(r, sec*1000));
      recorder.stop();
      const blob = await new Promise(r => recorder.onstop = ()=>r(new Blob(data)));
      const arrayBuffer = await blob.arrayBuffer();
      const base64String = btoa(String.fromCharCode(...new Uint8Array(arrayBuffer)));
      google.colab.kernel.invokeFunction('notebook.saveAudio', [base64String], {});
    }
    record(%d)
    """ % sec))

def save_audio(data):
    audio_bytes = base64.b64decode(data)
    with open("voice.wav","wb") as f:
        f.write(audio_bytes)
    print("✅ voice.wav saved.")
    display(Audio("voice.wav"))

output.register_callback('notebook.saveAudio', save_audio)
record(6)  # adjust seconds if needed
```

```
Set AUDIO_PATH to your file.
🎤 Recording... speak for 6 seconds.
✅ voice.wav saved.
```

2) Transcribe (whisper or SpeechRecognition)

```
import whisper
model = whisper.load_model("base")
result = model.transcribe("voice.wav")
query = result["text"]
print("📝 Transcribed:", query)
```

```
📝 Transcribed:  Hello Apple is red sky is blue mango is yellow
```

3) LLM call (replace with your project model)

```
from openai import OpenAI
client = OpenAI(api_key="sk-proj-CXJvAVbx-i78QCCstPZEgA8lmEd_r4XucTs5LVkCAZl5Eqcr0KzpJdZMvs9bfy3WAcEOxO_zInT3BlbkFJtqY4
response = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[{"role": "user", "content": query}]
)
answer = response.choices[0].message.content
print("🤖 LLM:", answer)
```

```
way to express thoughts or create patterns in poetry. Would you like to expand on this idea or create something similar?
```

4) Text-to-Speech (TTS) reply

```
from gtts import gTTS
import IPython.display as ipd

tts = gTTS(answer)
tts.save("response.mp3")
ipd.Audio("response.mp3")
```