```
!pip install SpeechRecognition pydub vosk openai-whisper sounddevice numpy
!pip install tkinter  # for file upload GUI
!pip install requests  # needed for some APIs
```

```
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: sounddevice in /usr/local/lib/python3.12/dist-packages (0.5.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (2.0.2)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.12/dist-packages (from SpeechRecognitio
Requirement already satisfied: cffi>=1.0 in /usr/local/lib/python3.12/dist-packages (from vosk) (2.0.0)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from vosk) (2.32.4)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from vosk) (4.67.1)
Requirement already satisfied: srt in /usr/local/lib/python3.12/dist-packages (from vosk) (3.5.3)
Requirement already satisfied: websockets in /usr/local/lib/python3.12/dist-packages (from vosk) (15.0.1)
Requirement already satisfied: more-itertools in /usr/local/lib/python3.12/dist-packages (from openai-whisper) (10
Requirement already satisfied: numba in /usr/local/lib/python3.12/dist-packages (from openai-whisper) (0.60.0)
Requirement already satisfied: tiktoken in /usr/local/lib/python3.12/dist-packages (from openai-whisper) (0.11.0)
Requirement already satisfied: torch in /usr/local/lib/python3.12/dist-packages (from openai-whisper) (2.8.0+cu126
Requirement already satisfied: triton>=2 in /usr/local/lib/python3.12/dist-packages (from openai-whisper) (3.4.0)
Requirement already satisfied: pycparser in /usr/local/lib/python3.12/dist-packages (from cffi>=1.0->vosk) (2.23)
Requirement already satisfied: setuptools>=40.8.0 in /usr/local/lib/python3.12/dist-packages (from triton>=2->open
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib/python3.12/dist-packages (from numba->
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests-
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests->vosk) (3.10
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests->vosk)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests->vosk)
Requirement already satisfied: regex>=2022.1.18 in /usr/local/lib/python3.12/dist-packages (from tiktoken->openai-
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from torch->openai-whisper) (3
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch->openai-whispe
Requirement already satisfied: networkx in /usr/local/lib/python3.12/dist-packages (from torch->openai-whisper) (3
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from torch->openai-whisper) (3.1
Requirement already satisfied: fsspec in /usr/local/lib/python3.12/dist-packages (from torch->openai-whisper) (202
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from to
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in /usr/local/lib/python3.12/dist-packages (from to
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in /usr/local/lib/python3.12/dist-packages (from torch
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in /usr/local/lib/python3.12/dist-packages (from torch
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in /usr/local/lib/python3.12/dist-packages (from torch-
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in /usr/local/lib/python3.12/dist-packages (from torc
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in /usr/local/lib/python3.12/dist-packages (from tor
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in /usr/local/lib/python3.12/dist-packages (from tor
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in /usr/local/lib/python3.12/dist-packages (from torc
Requirement already satisfied: nvidia-nccl-cu12==2.27.3 in /usr/local/lib/python3.12/dist-packages (from torch->op
Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch->o
Requirement already satisfied: nvidia-nvjitlink-cu12==12.6.85 in /usr/local/lib/python3.12/dist-packages (from tor
Requirement already satisfied: nvidia-cufile-cu12==1.11.1.6 in /usr/local/lib/python3.12/dist-packages (from torch
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.12/dist-packages (from sympy>=1.13.3->
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->torch->ope
Building wheels for collected packages: openai-whisper
  Building wheel for openai-whisper (pyproject.toml) ... done
  Created wheel for openai-whisper: filename=openai_whisper-20250625-py3-none-any.whl size=803979 sha256=24bcf8cc5
  Stored in directory: /root/.cache/pip/wheels/61/d2/20/09ec9bef734d126cba375b15898010b6cc28578d8afdde5869
Successfully built openai-whisper
Installing collected packages: openai-whisper
Successfully installed openai-whisper-20250625
ERROR: Could not find a version that satisfies the requirement tkinter (from versions: none)
ERROR: No matching distribution found for tkinter
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (2.32.4)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests) (2.5.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests) (2025
```

```
import os
import sys
import queue
import sounddevice as sd
import numpy as np
import whisper
from vosk import Model, KaldiRecognizer
import speech_recognition as sr
from tkinter import Tk, filedialog
from pydub import AudioSegment
```

```
/usr/local/lib/python3.12/dist-packages/pydub/utils.py:300: SyntaxWarning: invalid escape sequence '\('
  m = re.match('([su]([0-9]{1,2})p?) \(([0-9]{1,2}) bit\)$', token)
/usr/local/lib/python3.12/dist-packages/pydub/utils.py:301: SyntaxWarning: invalid escape sequence '\('
  m2 = re.match('([su]([0-9]{1,2})p?)( \(default\))?$', token)
/usr/local/lib/python3.12/dist-packages/pydub/utils.py:310: SyntaxWarning: invalid escape sequence '\('
  elif re.match('(flt)p?( \(default\))?$', token):
/usr/local/lib/python3.12/dist-packages/pydub/utils.py:314: SyntaxWarning: invalid escape sequence '\('
  elif re.match('(dbl)p?( \(default\))?$', token):
```

```
def upload_audio_cli():
    audio_path = input("Enter the full path of your audio file (e.g., /home/user/audio.wav): ")
    if not os.path.exists(audio_path):
        print("File not found. Please check the path and try again.")
        return None
    return audio_path


def record_audio(duration=5, fs=16000):
    print("Speak something...")
    recording = sd.rec(int(duration * fs), samplerate=fs, channels=1, dtype='int16')
    sd.wait()
    audio_file = "/content/lab3sample.wav"
    # Save the audio
    from scipy.io.wavfile import write
    write(audio_file, fs, recording)
    return audio_file
```

```
def recognize_whisper(audio_path):
    try:
        print("Recognizing with Whisper...")
        model = whisper.load_model("base")  # you can use 'tiny', 'base', 'small', 'medium', 'large'
        result = model.transcribe(audio_path)
        text = result["text"]
        print("Speech successfully converted to text!")
        return text
    except Exception as e:
        return f"Whisper error: {e}"
```

```
def recognize_vosk(audio_path):
    try:
        print("Recognizing with Vosk...")
        if not os.path.exists("vosk-model-small-en-us-0.15"):
            print("Downloading Vosk model...")
            os.system("wget https://alphacephei.com/vosk/models/vosk-model-small-en-us-0.15.zip")
            os.system("unzip vosk-model-small-en-us-0.15.zip")
        model = Model("vosk-model-small-en-us-0.15")
        rec = KaldiRecognizer(model, 16000)

        # Convert audio to proper format if needed
        sound = AudioSegment.from_file(audio_path)
        sound = sound.set_channels(1).set_frame_rate(16000)
        wav_path = "temp.wav"
        sound.export(wav_path, format="wav")

        import wave
        wf = wave.open(wav_path, "rb")
        text = ""
        while True:
            data = wf.readframes(4000)
            if len(data) == 0:
                break
            if rec.AcceptWaveform(data):
                res = rec.Result()
                text += eval(res)['text'] + " "
        text += eval(rec.FinalResult())['text']
        print("Speech successfully converted to text!")
        return text
    except Exception as e:
        return f"Vosk error: {e}"
```

```
def recognize_google(audio_path):
    try:
        print("Recognizing with Google API...")
        r = sr.Recognizer()
        with sr.AudioFile(audio_path) as source:
            audio = r.record(source)
        text = r.recognize_google(audio)
        print("Speech successfully converted to text!")
        return text
    except sr.UnknownValueError:
        return "Google Speech Recognition could not understand audio. Please try speaking more clearly."
    except sr.RequestError:
        return "Google Speech Recognition service is unavailable. Please check your internet connection."
```

```
def main():
    choice = input("Do you want to (1) Upload an audio file or (2) Record audio? Enter 1 or 2: ")

    if choice == "1":
        audio_file = upload_audio_cli()
        if audio_file is None:
            return
    elif choice == "2":
        duration = int(input("Enter recording duration in seconds: "))
        audio_file = record_audio(duration)
    else:
        print("Invalid choice!")
        return

    print(f"Audio file: {audio_file}")

    # Recognize using all three methods
    whisper_text = recognize_whisper(audio_file)
    vosk_text = recognize_vosk(audio_file)
    google_text = recognize_google(audio_file)

    # Display results
    print("\n=== Comparative Analysis ===")
    print(f"Whisper Output: {whisper_text}")
    print(f"Vosk Output: {vosk_text}")
    print(f"Google API Output: {google_text}")
```

```
if __name__ == "__main__":
    main()
```

```
Do you want to (1) Upload an audio file or (2) Record audio? Enter 1 or 2: 1
Enter the full path of your audio file (e.g., /home/user/audio.wav): /content/lab3sample.wav
Audio file: /content/lab3sample.wav
Recognizing with Whisper...
100%|███████████████████████████████████████| 139M/139M [00:00<00:00, 154MiB/s]
/usr/local/lib/python3.12/dist-packages/whisper/transcribe.py:132: UserWarning: FP16 is not supported on CPU; using
  warnings.warn("FP16 is not supported on CPU; using FP32 instead")
Speech successfully converted to text!
Recognizing with Vosk...
Downloading Vosk model...
Speech successfully converted to text!
Recognizing with Google API...
Speech successfully converted to text!

=== Comparative Analysis ===
Whisper Output:  I believe you're just talking nonsense.
Vosk Output: i believe you're just talking nonsense
Google API Output: I believe you are just talking nonsense
```

```
!pip install gtts pydub

from gtts import gTTS
from pydub import AudioSegment
import os

# Directory to save generated audio files
os.makedirs("audio_samples", exist_ok=True)

# Text for the audio
text = "Turn on the lights in the living room."

# 1. Clear Male Voice (gTTS default voice)
tts_male = gTTS(text=text, lang='en', tld='com')
file_male = "audio_samples/clear_male.wav"
tts_male.save(file_male)

# 2. Clear Female Voice (using 'co.uk' TLD for slight variation)
tts_female = gTTS(text=text, lang='en', tld='co.uk')
file_female = "audio_samples/clear_female.wav"
tts_female.save(file_female)

# 3. Fast Speech (speed up audio)
tts_fast = gTTS(text=text, lang='en', tld='com')
tts_fast.save("audio_samples/temp_fast.wav")
audio_fast = AudioSegment.from_file("audio_samples/temp_fast.wav")
audio_fast_fast = audio_fast.speedup(playback_speed=1.5)  # 1.5x faster
audio_fast_fast.export("audio_samples/fast_speech.wav", format="wav")

# 4. Soft Voice (reduce volume)
tts_soft = gTTS(text=text, lang='en', tld='com')
```

```
tts_soft = gTTS(text=text, lang='en', tld='com')
tts_soft.save("audio_samples/temp_soft.wav")
audio_soft = AudioSegment.from_file("audio_samples/temp_soft.wav")
audio_soft_soft = audio_soft - 15  # reduce volume by 15dB
audio_soft_soft.export("audio_samples/soft_voice.wav", format="wav")

# 5. Noisy Background (add white noise)
tts_noise = gTTS(text=text, lang='en', tld='com')
tts_noise.save("audio_samples/temp_noise.wav")
audio_noise = AudioSegment.from_file("audio_samples/temp_noise.wav")
# generate white noise
noise = AudioSegment.silent(duration=len(audio_noise)) + AudioSegment.from_mono_audiosegments(audio_noise)
noisy_audio = audio_noise.overlay(AudioSegment.silent(duration=0) + AudioSegment.from_mono_audiosegments(audio_noise
audio_noise.export("audio_samples/noisy_background.wav", format="wav")

print("All audio samples generated in 'audio_samples/' folder")
```

```
Collecting gtts
  Downloading gTTS-2.5.4-py3-none-any.whl.metadata (4.1 kB)
Requirement already satisfied: pydub in /usr/local/lib/python3.12/dist-packages (0.25.1)
Requirement already satisfied: requests<3,>=2.27 in /usr/local/lib/python3.12/dist-packages (from gtts) (2.32.4)
Collecting click<8.2,>=7.1 (from gtts)
  Downloading click-8.1.8-py3-none-any.whl.metadata (2.3 kB)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<3,
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.27->gtts
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.27
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.27
Downloading gTTS-2.5.4-py3-none-any.whl (29 kB)
Downloading click-8.1.8-py3-none-any.whl (98 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.2/98.2 kB 3.9 MB/s eta 0:00:00
Installing collected packages: click, gtts
  Attempting uninstall: click
    Found existing installation: click 8.3.0
    Uninstalling click-8.3.0:
      Successfully uninstalled click-8.3.0
Successfully installed click-8.1.8 gtts-2.5.4
All audio samples generated in 'audio_samples/' folder
```

```
import os
from vosk import Model, KaldiRecognizer
import whisper
import speech_recognition as sr
from pydub import AudioSegment
import wave

# Reference text for accuracy evaluation
REFERENCE_TEXT = "Turn on the lights in the living room."

# ------------------ Helper Functions ------------------

def convert_to_pcm_wav(input_path):
    """Convert any audio file to PCM WAV (16kHz, mono) for Google API."""
    sound = AudioSegment.from_file(input_path)
    sound = sound.set_channels(1).set_frame_rate(16000)
    temp_path = "temp_google.wav"
    sound.export(temp_path, format="wav")
    return temp_path

# ------------------ Recognition Functions ------------------

def recognize_whisper(audio_path):
    try:
        print(f"Whisper recognizing {audio_path} ...")
        model = whisper.load_model("base")  # you can change to tiny/small/medium/large
        result = model.transcribe(audio_path)
        return result["text"]
    except Exception as e:
        return f"Whisper error: {e}"

def recognize_vosk(audio_path):
    try:
        print(f"Vosk recognizing {audio_path} ...")
        if not os.path.exists("vosk-model-small-en-us-0.15"):
            print("Vosk model not found. Please download and unzip it from https://alphacephei.com/vosk/models")
            return "Vosk model missing"
        model = Model("vosk-model-small-en-us-0.15")
        rec = KaldiRecognizer(model, 16000)

        sound = AudioSegment.from_file(audio_path)
        sound = sound.set_channels(1).set_frame_rate(16000)
        wav_path = "temp_vosk.wav"
        sound.export(wav_path, format="wav")

        wf = wave.open(wav_path, "rb")
```

```
            wf = wave.open(wav_path, "rb")
            text = ""
            while True:
                data = wf.readframes(4000)
                if len(data) == 0:
                    break
                if rec.AcceptWaveform(data):
                    res = rec.Result()
                    text += eval(res)['text'] + " "
            text += eval(rec.FinalResult())['text']
            return text
        except Exception as e:
            return f"Vosk error: {e}"

    def recognize_google(audio_path):
        try:
            print(f"Google API recognizing {audio_path} ...")
            pcm_wav = convert_to_pcm_wav(audio_path)

            r = sr.Recognizer()
            with sr.AudioFile(pcm_wav) as source:
                audio = r.record(source)
            text = r.recognize_google(audio)
            return text
        except sr.UnknownValueError:
            return "Could not understand audio. Please speak clearly."
        except sr.RequestError:
            return "Google API unavailable. Check internet connection."

    # ------------------ Accuracy Evaluation ------------------

    def evaluate_accuracy(recognized_text):
        ref = REFERENCE_TEXT.lower()
        rec = recognized_text.lower()
        if rec == ref:
            return "High"
        elif any(word in rec for word in ref.split()):
            return "Medium"
        else:
            return "Low"

    # ------------------ Main Comparative Analysis ------------------

    audio_files = {
        "Clear Male Voice": "audio_samples/clear_male.wav",
        "Clear Female Voice": "audio_samples/clear_female.wav",
        "Fast Speech": "audio_samples/fast_speech.wav",
        "Soft Voice": "audio_samples/soft_voice.wav",
        "Noisy Background": "audio_samples/noisy_background.wav"
    }

    comparison_results = []

    for audio_type, path in audio_files.items():
        print(f"\n--- Processing: {audio_type} ---")
        whisper_text = recognize_whisper(path)
        vosk_text = recognize_vosk(path)
        google_text = recognize_google(path)

        comparison_results.append({
            "Audio Type": audio_type,
            "Whisper Output": whisper_text,
            "Vosk Output": vosk_text,
            "Google API Output": google_text,
            "Whisper Accuracy": evaluate_accuracy(whisper_text),
            "Vosk Accuracy": evaluate_accuracy(vosk_text),
            "Google Accuracy": evaluate_accuracy(google_text)
        })

    # ------------------ Display Results ------------------

    print("\n\n=== Comparative Analysis Table ===")
    for row in comparison_results:
        print(f"\nAudio Type: {row['Audio Type']}")
        print(f"Whisper Output: {row['Whisper Output']} (Accuracy: {row['Whisper Accuracy']})")
        print(f"Vosk Output: {row['Vosk Output']} (Accuracy: {row['Vosk Accuracy']})")
        print(f"Google API Output: {row['Google API Output']} (Accuracy: {row['Google Accuracy']})")
```

```
--- Processing: Clear Male Voice ---
Whisper recognizing audio_samples/clear_male.wav ...
/usr/local/lib/python3.12/dist-packages/whisper/transcribe.py:132: UserWarning: FP16 is not supported on CPU; usin
```

```
  warnings.warn("FP16 is not supported on CPU; using FP32 instead")
Vosk recognizing audio_samples/clear_male.wav ...
Google API recognizing audio_samples/clear_male.wav ...

--- Processing: Clear Female Voice ---
Whisper recognizing audio_samples/clear_female.wav ...
/usr/local/lib/python3.12/dist-packages/whisper/transcribe.py:132: UserWarning: FP16 is not supported on CPU; usin
  warnings.warn("FP16 is not supported on CPU; using FP32 instead")
Vosk recognizing audio_samples/clear_female.wav ...
Google API recognizing audio_samples/clear_female.wav ...

--- Processing: Fast Speech ---
Whisper recognizing audio_samples/fast_speech.wav ...
/usr/local/lib/python3.12/dist-packages/whisper/transcribe.py:132: UserWarning: FP16 is not supported on CPU; usin
  warnings.warn("FP16 is not supported on CPU; using FP32 instead")
Vosk recognizing audio_samples/fast_speech.wav ...
Google API recognizing audio_samples/fast_speech.wav ...

--- Processing: Soft Voice ---
Whisper recognizing audio_samples/soft_voice.wav ...
/usr/local/lib/python3.12/dist-packages/whisper/transcribe.py:132: UserWarning: FP16 is not supported on CPU; usin
  warnings.warn("FP16 is not supported on CPU; using FP32 instead")
Vosk recognizing audio_samples/soft_voice.wav ...
Google API recognizing audio_samples/soft_voice.wav ...

--- Processing: Noisy Background ---
Whisper recognizing audio_samples/noisy_background.wav ...
/usr/local/lib/python3.12/dist-packages/whisper/transcribe.py:132: UserWarning: FP16 is not supported on CPU; usin
  warnings.warn("FP16 is not supported on CPU; using FP32 instead")
Vosk recognizing audio_samples/noisy_background.wav ...
Google API recognizing audio_samples/noisy_background.wav ...


=== Comparative Analysis Table ===

Audio Type: Clear Male Voice
Whisper Output:  Turn on the lights in the living room. (Accuracy: Medium)
Vosk Output: turn on the lights in the living room (Accuracy: Medium)
Google API Output: turn on the lights in the living room (Accuracy: Medium)

Audio Type: Clear Female Voice
Whisper Output:  Turn on the lights in the living room. (Accuracy: Medium)
Vosk Output: turn on the lights in the living room (Accuracy: Medium)
Google API Output: turn on the lights in the living room (Accuracy: Medium)

Audio Type: Fast Speech
Whisper Output:  Turn on the lights in the living room. (Accuracy: Medium)
Vosk Output: turn on the lights in the living room (Accuracy: Medium)
Google API Output: turn on the lights in the living room (Accuracy: Medium)

Audio Type: Soft Voice
Whisper Output:  Turn on the lights in the living room. (Accuracy: Medium)
Vosk Output: turn on the lights in the living room (Accuracy: Medium)
```