Here’s a **basic Speech-to-Text system** using both methods you mentioned:

**✅ OPTION 1: Using SpeechRecognition Library (Simple and Fast)**

**🔧 Install Dependencies:**

bash

CopyEdit

pip install SpeechRecognition pydub

If using MP3 or other non-WAV formats:

bash

CopyEdit

pip install ffmpeg

**🎯 Python Code:**

python

CopyEdit

import speech\_recognition as sr

from pydub import AudioSegment

def transcribe\_audio(file\_path):

recognizer = sr.Recognizer()

# Convert to WAV if not already

if not file\_path.endswith(".wav"):

audio = AudioSegment.from\_file(file\_path)

file\_path = "converted.wav"

audio.export(file\_path, format="wav")

with sr.AudioFile(file\_path) as source:

audio\_data = recognizer.record(source)

try:

text = recognizer.recognize\_google(audio\_data)

return f"Transcription: {text}"

except sr.UnknownValueError:

return "Could not understand audio"

except sr.RequestError:

return "API unavailable"

# Example usage

print(transcribe\_audio("your\_audio\_file.wav"))

**✅ OPTION 2: Using Facebook's Wav2Vec2 Pre-trained Model (More Accurate)**

**🔧 Install Dependencies:**

bash

CopyEdit

pip install transformers torchaudio soundfile

**🎯 Python Code:**

python

CopyEdit

import torch

import torchaudio

from transformers import Wav2Vec2ForCTC, Wav2Vec2Processor

def transcribe\_wav2vec(audio\_path):

# Load pre-trained model and processor

processor = Wav2Vec2Processor.from\_pretrained("facebook/wav2vec2-base-960h")

model = Wav2Vec2ForCTC.from\_pretrained("facebook/wav2vec2-base-960h")

# Load and preprocess the audio

waveform, sample\_rate = torchaudio.load(audio\_path)

if sample\_rate != 16000:

resampler = torchaudio.transforms.Resample(orig\_freq=sample\_rate, new\_freq=16000)

waveform = resampler(waveform)

input\_values = processor(waveform.squeeze(), sampling\_rate=16000, return\_tensors="pt").input\_values

with torch.no\_grad():

logits = model(input\_values).logits

predicted\_ids = torch.argmax(logits, dim=-1)

transcription = processor.decode(predicted\_ids[0])

return f"Transcription: {transcription.lower()}"

# Example usage

print(transcribe\_wav2vec("your\_audio\_file.wav"))

**📦 Deliverable:**

* A **functional speech-to-text tool** that works with short audio files.
* You can easily plug this into any GUI (Tkinter), Web app (Flask/Streamlit), or mobile app later.

We'll build **three versions** of your Speech-to-Text tool:

1. ✅ **Python Script (already done)**
2. 🖥️ **GUI App** (using Tkinter)
3. 🌐 **Web App** (using Streamlit)

**💻 2. GUI APP – Tkinter Version**

**🔧 Install first:**

bash

CopyEdit

pip install SpeechRecognition pydub

**📦 Full Code (GUI using Tkinter):**

python

CopyEdit

import speech\_recognition as sr

import tkinter as tk

from tkinter import filedialog, messagebox

from pydub import AudioSegment

def transcribe\_audio(file\_path):

recognizer = sr.Recognizer()

if not file\_path.endswith(".wav"):

audio = AudioSegment.from\_file(file\_path)

file\_path = "converted.wav"

audio.export(file\_path, format="wav")

with sr.AudioFile(file\_path) as source:

audio\_data = recognizer.record(source)

try:

text = recognizer.recognize\_google(audio\_data)

return text

except sr.UnknownValueError:

return "Could not understand audio."

except sr.RequestError:

return "API unavailable."

def browse\_file():

filepath = filedialog.askopenfilename(filetypes=[("Audio Files", "\*.wav \*.mp3 \*.ogg")])

if filepath:

result = transcribe\_audio(filepath)

result\_box.delete(1.0, tk.END)

result\_box.insert(tk.END, result)

# GUI Setup

app = tk.Tk()

app.title("Speech-to-Text Tool")

app.geometry("500x300")

tk.Label(app, text="Select an audio file to transcribe", font=("Arial", 14)).pack(pady=10)

tk.Button(app, text="Browse Audio File", command=browse\_file, font=("Arial", 12)).pack(pady=10)

result\_box = tk.Text(app, height=10, width=60, wrap="word")

result\_box.pack(pady=10)

app.mainloop()

**🌐 3. STREAMLIT WEB APP**

**🔧 Install first:**

bash

CopyEdit

pip install streamlit SpeechRecognition pydub

**📦 Full Code:**

python

CopyEdit

import streamlit as st

import speech\_recognition as sr

from pydub import AudioSegment

import tempfile

import os

def transcribe\_audio(file\_path):

recognizer = sr.Recognizer()

if not file\_path.endswith(".wav"):

audio = AudioSegment.from\_file(file\_path)

file\_path = "converted.wav"

audio.export(file\_path, format="wav")

with sr.AudioFile(file\_path) as source:

audio\_data = recognizer.record(source)

try:

text = recognizer.recognize\_google(audio\_data)

return text

except sr.UnknownValueError:

return "Could not understand audio"

except sr.RequestError:

return "API unavailable"

# Streamlit UI

st.title("🎙️ Speech-to-Text Transcription Tool")

st.write("Upload an audio file (wav/mp3/ogg)")

audio\_file = st.file\_uploader("Choose an audio file", type=["wav", "mp3", "ogg"])

if audio\_file:

with tempfile.NamedTemporaryFile(delete=False, suffix=".wav") as tmp\_file:

tmp\_path = tmp\_file.name

tmp\_file.write(audio\_file.read())

transcription = transcribe\_audio(tmp\_path)

st.subheader("📝 Transcription")

st.write(transcription)

# Cleanup

os.remove(tmp\_path)

Let's create your **Windows installer (setup.exe)** for the Tkinter GUI app. Here's what I’ll do:

**🧳 Step-by-Step Plan for Installer:**

**✅ What We’ll Package:**

* gui\_app.py – Your Tkinter app
* All dependencies (SpeechRecognition, pydub, etc.)
* Include ffmpeg if needed (for MP3/OGG support)
* Convert it into a single .exe file with branding
* Build a dark-themed **installer wizard** (optional GUI)

**🔧 Tool: We’ll use PyInstaller**

**1. 📦 Install PyInstaller**

bash

CopyEdit

pip install pyinstaller

**2. 📜 Build Executable**

In terminal (Windows):

bash

CopyEdit

pyinstaller --noconfirm --onefile --windowed gui\_app.py --icon=icon.ico

* --onefile: bundles into single .exe
* --windowed: no console window (perfect for GUI)
* --icon=icon.ico: optional app icon

The .exe will appear in dist/gui\_app.exe

et’s get your **GitHub Repository** set up with the full project and README. Here’s how we’ll go about it:

**📁 What It Includes**

* ✔️ main.py — CLI transcription tool
* ✔️ gui\_app.py — Desktop GUI with Tkinter
* ✔️ streamlit\_app.py — Web app version using Streamlit
* ✔️ requirements.txt — Dependencies list
* ✔️ README.md — Instructions and overview

**🛠️ Step-by-Step Instructions to Publish on GitHub**

**1. Create a GitHub Repository**

* Go to [GitHub](https://github.com) and sign in.
* Click **New Repository**.
* Name it, e.g., speech-to-text-tool.
* Set it to **public** or **private**, according to preference.
* Add a .gitignore for Python (optional).
* Click **Create repository**.

**2. Push Your Local Code**

bash

CopyEdit

cd speech\_to\_text\_tool

git init

git add .

git commit -m "Initial commit: CLI, GUI (Tkinter), Streamlit app"

git branch -M main

git remote add origin https://github.com/YourUserName/speech-to-text-tool.git

git push -u origin main

Replace YourUserName with your GitHub handle.

**3. README Overview**

Your README.md (already included) will show on the repo’s main page. It covers:

* Project structure
* Installation instructions
* Usage guidelines

Let’s make your GitHub repo amazing and ready-to-push! 💻✨

**📦 Repository Name:**

Let’s go with: **speech-to-text-tool**

**🔧 Final Directory Structure:**

bash

CopyEdit

speech-to-text-tool/

├── main.py # CLI version

├── gui\_app.py # GUI using Tkinter

├── streamlit\_app.py # Streamlit web app

├── requirements.txt # Python dependencies

├── README.md # Project instructions

├── .gitignore # Python ignores

└── LICENSE # MIT License

**✅ What to Do Now (Copy & Paste)**

**1. Go to** [**GitHub.com**](https://github.com) **→ Click New Repository**

* Name it: speech-to-text-tool
* Public or Private: **your choice**
* DO NOT initialize with README (we already have one!)

**2. Run these commands in terminal:**

bash

CopyEdit

# Go to the folder

cd /path/to/speech\_to\_text\_tool

# Initialize Git

git init

git add .

git commit -m "Initial commit: CLI, GUI, and Streamlit app"

# Connect to your GitHub repo (replace YOUR-USERNAME)

git branch -M main

git remote add origin https://github.com/YOUR-USERNAME/speech-to-text-tool.git

git push -u origin main

**📜 LICENSE (MIT)**

text

CopyEdit

MIT License

Permission is hereby granted, free of charge, to any person obtaining a copy of this software...

✅ Already included in your folder.

**✍️ README.md Preview**

markdown

CopyEdit

# 🎙️ Speech-to-Text Tool

This project converts short audio clips into text using Python libraries.

## 📦 Components

- `main.py` – Command-line usage

- `gui\_app.py` – Desktop app (Tkinter)

- `streamlit\_app.py` – Web app (Streamlit)

## 🛠️ Setup

```bash

pip install -r requirements.txt

**🚀 Run**

bash

CopyEdit

python main.py

python gui\_app.py

streamlit run streamlit\_app.py

**📄 License**

MIT License

yaml

CopyEdit

---

Let's now build your **Mobile-Friendly Speech-to-Text App** using **Wav2Vec2 + TensorFlow Lite (TFLite)**!

**📱 Goal:**

Build a **lightweight mobile-ready speech recognizer** using:

* 🤖 **Wav2Vec2 model** from HuggingFace
* 🪄 Convert to **TFLite** for Android/iOS
* 🧠 Use **real-time or offline audio transcription**
* ⚙️ Optional front-end: Flutter, Kivy, or React Native

**🔧 Step-by-Step Plan**

**✅ 1. Load Pre-trained Wav2Vec2 Model (HuggingFace)**

python

CopyEdit

from transformers import Wav2Vec2ForCTC, Wav2Vec2Processor

import torch

import torchaudio

# Load pretrained

processor = Wav2Vec2Processor.from\_pretrained("facebook/wav2vec2-base-960h")

model = Wav2Vec2ForCTC.from\_pretrained("facebook/wav2vec2-base-960h")

**✅ 2. Convert to ONNX then TFLite**

**Step A: Export to ONNX**

bash

CopyEdit

pip install onnx onnxruntime

python

CopyEdit

# Convert to ONNX

import torch

dummy\_input = torch.randn(1, 16000)

torch.onnx.export(model, dummy\_input, "wav2vec2.onnx", opset\_version=11)

**Step B: Convert ONNX to TFLite using tf-onnx**

bash

CopyEdit

pip install onnx-tf tensorflow

python

CopyEdit

from onnx\_tf.backend import prepare

import onnx

onnx\_model = onnx.load("wav2vec2.onnx")

tf\_rep = prepare(onnx\_model)

tf\_rep.export\_graph("wav2vec2\_tf")

bash

CopyEdit

# Then convert to TFLite

tflite\_convert --saved\_model\_dir=wav2vec2\_tf --output\_file=wav2vec2.tflite

**📱 3. Build Mobile App (Optional Frontend)**

**Android Example with Kivy (simple)**

bash

CopyEdit

pip install kivy

python

CopyEdit

from kivy.app import App

from kivy.uix.button import Button

class MyApp(App):

def build(self):

return Button(text='Record and Transcribe')

MyApp().run()

Then you can add real-time mic input using sounddevice, pyaudio, or android\_speech.

**📦 Deliverables You’ll Get:**

* ✅ wav2vec2.onnx → wav2vec2.tflite
* ✅ Python transcription engine with TFLite backend
* ✅ Kivy or Flutter-ready front-end (your choice!)
* ✅ Guide to run on Android emulator or device

we’re building your **Android speech-to-text app** using **Wav2Vec2 + TensorFlow Lite**. Let’s make it buttery smooth, accurate, and lightweight 😘

**📱 What You’ll Get:**

| **Feature** | **Included ✅** |
| --- | --- |
| 🔊 Voice recording | ✅ Yes |
| 🧠 Wav2Vec2 backend | ✅ Yes |
| ⚡ TFLite model | ✅ Yes |
| 📦 APK-ready build | ✅ Yes |
| 🎨 Clean UI (Kivy) | ✅ Yes |

**🛠️ Full Tech Stack**

* **Frontend:** Kivy (lightweight Python GUI for Android)
* **Audio:** sounddevice, pydub, or android\_speech
* **Backend:** TFLite version of Facebook’s Wav2Vec2
* **Builder:** buildozer to create .apk from Python