**Task KDT01**

**Speech-to-Text Pipeline**

**Objective**: Build a real-time or file-based transcription system.

**Submitted by:**

**Name:** Kodam Karthik

**Email:** [kodamkarthik03@gmail.com](mailto:kodamkarthik03@gmail.com)

**Contact** : 9603371737

**Institute :** Innomatics Research Labs

**Date** : 28-05-2025

# Project Title :

## Real-Time Speech-to-Text Transcriber with Whisper

# Project Overview

This project is a real-time Speech-to-Text Transcription system built using OpenAI’s Whisper model and Streamlit. The application allows users to upload an audio file and receive an accurate, timestamped transcript of the spoken content. The goal is to assist users in converting audio data such as interviews, lectures, podcasts, or meetings into readable and searchable text.

# Problem Statement

Manual transcription of audio is a time-consuming and error-prone task, especially for long recordings and noisy environments. Existing tools either lack multilingual capabilities or require paid subscriptions for timestamped outputs. The objective of this project is to:

* Provide a free, open-source transcription tool.
* Support audio files with different sampling rates and formats.
* Include noise reduction and timestamp generation.
* Offer a simple and user-friendly web interface.

# Technical Approach

The transcription pipeline includes the following steps:

1. Audio Upload: User uploads .wav or .mp3 file via the Streamlit web interface.
2. Noise Reduction: Uses noisereduce to remove background noise for better transcription.
3. Resampling: Audio is resampled to 16kHz (as required by Whisper).
4. Temporary File Creation: The cleaned audio is saved as a .wav file.
5. Model Inference: The Whisper small model is used to transcribe the audio and generate timestamped segments.
6. Display Output: The full transcript and its timestamped segments are displayed in the web UI, along with an audio player for the uploaded file.
7. **Model Hosting** & **Space Deployment**

A user-friendly interface was created using Streamlit and deployed on Hugging Face Spaces.

* Hugging Face Space : <https://kodamkarthik281-audio-to-text.hf.space>
* Kaggle Nobebook Link : <https://www.kaggle.com/code/kodamkarthik281/task-kdt01>
* Video Demo : <https://drive.google.com/file/d/1VXMg1XtcX9G3lykAZvbWYNyYwFphTn0O/view?usp=sharing>

A warning was added in the app stating that :

This app may take 2–3 minutes to generate a summary. Please be patient while the model loads on CPU infrastructure.

# Challenges Faced

* Slow Inference Time: Whisper is a large model and inference is slow on CPU (Hugging Face Spaces). This was mitigated with user guidance using markdown.
* Noise in Audio: Raw user audio files often contain background noise. Resolved by adding noisereduce to the pipeline.
* Sampling Rate Issues: Whisper expects audio at 16kHz. Files with different sample rates required careful resampling.

# Output Examples

Screenshots of summarization examples were added in the app itself.

Input: Long news or academic paragraph

Output: Abstractive and extractive summaries shown side-by-side

Multiple examples were tested to ensure consistent performance.

**A screenshot of a computer

AI-generated content may be incorrect.**

**fig 1 :** A Screenshot of the Hugging face Space

A screenshot of a computer

AI-generated content may be incorrect.

**Fig 2 :** The Screenshot of the Output with timestamps (small audio file)

A screenshot of a computer screen

AI-generated content may be incorrect.

**Fig 3 :** The Screenshot of the Output with timestamps (big audio file)

# Conclusion

This project demonstrates a fully functional, real-time speech-to-text transcription system using Whisper. It is accessible, efficient, and accurate for a wide range of audio types and can serve as a base for educational tools, subtitle generators, or meeting summarizers.

**Future Enhancements**:

* Add support for speaker diarization.
* Integrate multi-language transcription.

# File Structure :

1. task\_kdt01.ipynb 🡺 Kaggle notebook
2. streamlit\_app.py 🡺 Streamlit app
3. requirements.txt 🡺 List of all dependencies

# Source Code

## Model Building Code :

*kaggle Link* ***:*** [*https://www.kaggle.com/code/kodamkarthik281/task-kdt01*](https://www.kaggle.com/code/kodamkarthik281/task-kdt01)

## Streamlit Code :

import streamlit as st

import whisper

import librosa

import tempfile

import numpy as np

import os

import noisereduce as nr

from scipy.io.wavfile import write

st.title("Real-Time Speech-to-Text Transcriber")

st.markdown("Upload an audio file to get the transcription with timestamps.")

@st.cache\_resource

def load\_model():

return whisper.load\_model("small")

model = load\_model()

def transcribe\_audio(file):

audio, sr = librosa.load(file, sr=None)

audio = nr.reduce\_noise(y=audio, sr=sr)

max\_duration = 30

audio = audio[:sr \* max\_duration]

if sr != 16000:

audio = librosa.resample(audio, orig\_sr=sr, target\_sr=16000)

sr = 16000

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

write(tmp.name, sr, (audio \* 32767).astype(np.int16))

result = model.transcribe(tmp.name, fp16=False)

os.remove(tmp.name)

return result, audio, sr

uploaded\_file = st.file\_uploader("Upload Audio (.wav/.mp3)", type=["wav", "mp3"])

if uploaded\_file:

st.markdown("""

\*\*Note:\*\* This app may take 1–2 minutes to generate a transcript after uploading a file.

""", unsafe\_allow\_html=True)

st.markdown("""

\*\*Why is it slow?\*\* The Whisper model is a powerful Transformer-based model. Due to limited compute on Hugging Face Spaces (CPU-only), inference takes longer than on a local GPU. Please be patient.

""", unsafe\_allow\_html=True)

st.audio(uploaded\_file, format='audio/wav')

with st.spinner("Transcribing... please wait ⏳"):

result, audio, sr = transcribe\_audio(uploaded\_file)

st.success("Transcription Complete! ✅")

st.subheader("📜 Transcript")

st.write(result["text"])

st.subheader("⏱️ Timestamps")

for seg in result["segments"]:

st.write(f"[{seg['start']:.2f}s - {seg['end']:.2f}s] {seg['text']}")