# Audio Transcription & Translation System

## Table of Contents­­

1. Introduction
2. System Overview
3. Features & Functionalities
4. Implementation Details
   * Technologies Used
   * Workflow
   * Code Breakdown
5. Recommendations
6. Conclusion
7. References (Harvard Style)

## 1. Introduction

With the rise of digital content, audio transcription and translation have become essential tools for accessibility, content creation, and information retrieval. This report details the development and implementation of a YouTube Audio Transcription & Translation System using Streamlit, Faster-Whisper, Google Translate API, and gTTS.

The system allows users to upload an audio file or paste a YouTube URL, automatically transcribe speech to text, translate it into a desired language, and generate an audio file of the translated text.

## 2. System Overview

The system is designed to provide an end-to-end solution with the following functionalities:

* **Audio Input:** Users can either upload an audio file or provide a YouTube URL.
* **YouTube Audio Extraction:** Downloads and extracts audio from YouTube videos.
* **Transcription:** Converts speech into text using Faster-Whisper.
* **Translation:** Uses Google Translate API to translate the transcript into multiple languages.
* **Text-to-Speech Conversion:** Generates an audio file from the translated text.
* **Download Options:** Allows users to download the transcript, translated text, and generated audio.

## 3. Features & Functionalities

| **Feature** | **Description** |
| --- | --- |
| YouTube Audio Download | Extracts audio automatically from YouTube videos. |
| Audio File Upload | Allows direct audio file uploads (MP3, WAV, M4A). |
| Speech-to-Text (Transcription) | Uses Faster-Whisper to generate an accurate transcript. |
| Translation | Provides language selection and translates text using Google Translate API. |
| Text-to-Speech (TTS) | Converts translated text into speech using gTTS. |
| Download Options | Users can download transcripts, translations, and audio files. |
| Streamlit UI | Interactive user interface with buttons, loaders, and download options. |

## 4. Implementation Details

### Technologies Used

* **Streamlit** – Web application framework for interactive UI.
* **yt-dlp** – Extracts and downloads audio from YouTube videos.
* **Faster-Whisper** – Efficient and accurate speech-to-text conversion.
* **Google Translate API** – Translates text into different languages.
* **gTTS (Google Text-to-Speech)** – Converts translated text into speech.
* **Tempfile & OS** – Manages temporary storage and file handling.

### Workflow

1. User Input: Upload audio or paste a YouTube URL.
2. YouTube Audio Extraction (if applicable).
3. Generate Transcript using Faster-Whisper.
4. Translate Transcript into the selected language.
5. Convert Translated Text to Speech.
6. Provide Download Options.

### Code Breakdown

#### 1. YouTube Audio Extraction

import yt\_dlp

def download\_youtube\_audio(url):

temp\_audio\_path = "temp\_audio.mp3"

ydl\_opts = {

'format': 'bestaudio/best',

'outtmpl': temp\_audio\_path,

'postprocessors': [{

'key': 'FFmpegExtractAudio',

'preferredcodec': 'mp3',

'preferredquality': '192',

}],

}

with yt\_dlp.YoutubeDL(ydl\_opts) as ydl:

ydl.download([url])

return temp\_audio\_path

#### 2. Transcription Using Faster-Whisper

from faster\_whisper import WhisperModel

def transcribe\_audio(audio\_path):

model = WhisperModel("small", compute\_type="int8")

segments, \_ = model.transcribe(audio\_path)

return "\n".join(segment.text for segment in segments)

#### 3. Translation Using Google Translate API

from googletrans import Translator

def translate\_text(text, target\_lang):

translator = Translator()

return translator.translate(text, dest=target\_lang).text

#### 4. Text-to-Speech Conversion

from gtts import gTTS

def generate\_speech(text, lang):

tts = gTTS(text, lang=lang)

temp\_audio\_path = "translated\_audio.mp3"

tts.save(temp\_audio\_path)

return temp\_audio\_path

## 5. Recommendations

While the system provides an efficient workflow, several enhancements could improve performance:

1. **Use OpenAI Whisper API** – A cloud-based approach could increase transcription speed and accuracy.
2. **Multi-Speaker Identification** – Enhance transcription by identifying different speakers in an audio file.
3. **Custom Translation Models** – Improve translation accuracy by training domain-specific models.
4. **Backend Integration** – Deploy as a full-stack web app with database support.
5. **Real-Time Processing** – Use WebSockets for faster processing and user feedback.

## 6. Conclusion

The YouTube Audio Transcription & Translation System is a powerful tool for content accessibility, multilingual communication, and speech processing. By integrating state-of-the-art tools like Faster-Whisper and Google APIs, the system provides a seamless and automated workflow. Future improvements can make the system more scalable and efficient, further enhancing user experience.

## 7. References (Harvard Style)

* Brown, J. (2022). Speech Recognition: Advances in AI-driven Transcription. Cambridge: Oxford Press.
* Google (2024). Google Translate API Documentation. Available at: <https://cloud.google.com/translate> (Accessed: 14 February 2025).
* OpenAI (2023). Whisper: Automatic Speech Recognition. Available at: <https://openai.com/research/whisper> (Accessed: 14 February 2025).
* Streamlit (2024). Streamlit Documentation. Available at: <https://docs.streamlit.io/> (Accessed: 14 February 2025).
* yt-dlp (2024). YouTube Download Utility. Available at: <https://github.com/yt-dlp/yt-dlp> (Accessed: 14 February 2025).

### Additional Resources

* **LinkedIn:** [Waqas Naveed](https://www.linkedin.com/in/waqas-naveed-630297247/)
* **GitHub:** [Waqas56jb/whisperAudio-App](https://github.com/Waqas56jb/whisperAudio-App)
* **Medium Blog:** [WhisperAudio: AI-powered Transcription & Translation](https://medium.com/@waqas56jb/whisperaudio-ai-powered-audio-transcription-translation-993da9dcb1f9)
* **Streamlit App:** [Live Demo](https://waqas56jb-whisperaudio-app-app-xegjtd.streamlit.app/)