MULTILINGUAL VOICE MESSAGE TRANSLATION

A DESIGN PROJECT REPORT

Submitted by

MOHAMMED HISHAAM S

MUHAMMAD FAATIN S

SIVA DHARSAN M

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

JUNE, 2025

MULTILINGUAL VOICE MESSAGE TRANSLATION

A DESIGN PROJECT REPORT

Submitted by

MOHAMMED HISHAAM S (811722001032)

MUHAMMAD FAATIN S (811722001034)

SIVA DHARSAN M (811722001048)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

JUNE, 2025

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)
SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this design project report titled **"MULTILINGUAL VOICE MESSAGE TRANSLATION"** is the bonafide work of **MOHAMMED HISHAAM S (811722001032), MUHAMMAD FAATIN S (811722001034), SIVA DHARSAN M (811722001048)** who carried out the design project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other design project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. T.Avudaiappan, M.E., Ph.D.,
HEAD OF THE DEPARTMENT

Department of Artificial Intelligence
K.Ramakrishnan College of Technology
(Autonomous)
Samayapuram – 621 112

SIGNATURE

Mrs. P.Jasmine Jose, M.E.,
SUPERVISOR

ASSISTANT PROFESSOR
Department of Artificial Intelligence
K.Ramakrishnan College of Technology
(Autonomous)
Samayapuram – 621 112

Submitted for the viva-voce examination held on ……

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We jointly declare that the design project report on **"MULTILINGUAL VOICE MESSAGE TRANSLATION"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This design project report is submitted on the partial fulfilment of the requirement of the award of Degree of **BACHELOR OF TECHNOLOGY**.

Signature

_____

MOHAMMED HISHAAM S

_____

MUHAMMAD FAATIN S

_____

SIVA DHARSAN M

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this design project.

We are glad to credit honourable chairman **Dr. K.RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our design project and offering adequate duration in completing our design project.

We would like to thank **Dr. N. VASUDEVAN, M.E., Ph.D.,** Principal, who gave opportunity to frame the design project the full satisfaction.

We whole heartily thank **Dr. T.AVUDAIAPPAN**, **M.E., Ph.D.,** Head of the department, **ARTIFICIAL INTELLIGENCE** for providing his encourage pursuing this design project.

We express our deep and sincere gratitude to our design project guide **Mrs. P. JASMINE JOSE, M.E.,** Department of **ARTIFICIAL INTELLIGENCE,** for her incalculable suggestions, creativity, assistance and patience which motivated us to carry out this design project.

We render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

ABSTRACT

In today's globalized world, communication across different languages remains a significant challenge. This project introduces a Multilingual Voice Message Translation System that bridges language barriers by translating voice messages from one language to another and delivering the output in a natural-sounding voice. The system accepts a voice message in languages such as Tamil, Hindi, Malayalam, Kannada, Arabic, French, or English, detects the spoken language using advanced speech recognition (OpenAI Whisper), translates the transcribed text using a neural machine translation engine (Google Translate API), and finally generates a clear, natural male voice in the user's selected language using Microsoft Neural TTS. Built with a simple and intuitive Gradio-based interface, the system allows users to upload or record voice messages and receive translated audio responses efficiently. The application supports real-world use cases in fields such as education, tourism, customer service, and healthcare, where multilingual communication is essential. By integrating cutting-edge AI technologies into a user-friendly web framework, this project provides a powerful and practical solution for real-time multilingual voice communication.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ASR	-	Automatic Speech Recognition
IDE	-	Integrated Development Environment
LID	-	Language Identification
NLP	-	Natural Language Processing
NMT	-	Natural Machine Translation
SSL	-	Self Supervised Learning
STT	-	Speech To Text
TTS	-	Text To Speech
UI	-	User Interface

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

In a multicultural and multilingual world, the ability to communication across language boundaries has become a critical need in many fields such as education, healthcare, tourism, and customer service. Language translation technology has advanced significantly, but most tools are limited to either text translation or basic voice-based outputs. This project, Multilingual Voice Message Translation System, aims to overcome those limitations by offering a complete voice-to-voice translation platform.

The system allows users to upload or record a voice message in one language and receive an audio output in another selected language. It supports a range of languages including Tamil, Hindi, Malayalam, Kannada, Arabic, French, and English. The system leverages powerful AI models such as OpenAI Whisper for speech recognition, Google Translate (Neural Machine Translation) for text translation, and Microsoft Neural TTS for natural-sounding text-to-speech voice generation.

This project is implemented using Python and hosted through a Gradio-based interface, making it simple for users with minimal technical knowledge. Its modular design ensures high accuracy in transcription, contextual understanding in translation, and human-like voice output. As a result, this system has real-world applicability in situations where live interpreters or multilingual staff may not be available. It simplifies cross-language communication and enhances user accessibility across different regions and language backgrounds.

By integrating these technologies into a single, easy-to-use web interface, the Multilingual Voice Message Translation System not only improves accessibility and inclusivity but also demonstrates how artificial intelligence can be applied meaningfully in everyday interactions.

1.2 OBJECTIVE

The main objective of this project is to design and develop a Multilingual Voice Message Translation System that facilitates seamless communication across different languages through the integration of speech recognition, translation, and voice synthesis technologies. In a world where people speak diverse languages, the need for a system that can understand a spoken message in one language and reproduce it accurately in another language is both practical and essential. This project aims to bridge that gap by providing an intelligent, real-time voice-to-voice translation solution.

The system enables users to upload or record a voice message in any supported language, such as Tamil, Hindi, Malayalam, Kannada, Arabic, French, or English. The voice input is then automatically transcribed using OpenAI's Whisper model, which not only converts speech to text but also detects the language of the audio. This feature eliminates the need for manual language selection and enhances user convenience. Once the text is extracted, it is translated into the user's selected target language using a neural machine translation engine provided by Google Translate. This ensures the translated content is contextually accurate and grammatically correct.

Overall, the objective is to create a complete end-to-end solution that not only supports multiple languages but also delivers accurate, high-quality translations in the form of speech. The system is intended to be useful in various sectors including education, travel, healthcare, customer service, and emergency communication, where overcoming language barriers can significantly enhance understanding and service delivery.

CHAPTER 2

LITERATURE SURVEY

2.1 SELF SUPERVISED LEARNING FOR MULTILINGUAL SPEECH RECOGNITION

Babu, A., Tjandra, A., et al.

This paper presents a self-supervised speech recognition model trained using wav2vec 2.0 on multilingual audio. It performs well across 60+ languages, especially low-resource ones, using unlabeled data. Fine-tuning allows customization for individual languages. The system shares acoustic patterns among languages, improving performance. It shows promising results without needing parallel corpora.

Merits

- Works well for low-resource languages.
- Eliminates the need for labeled data.
- Efficient for multilingual scaling.

Demerits

- Requires high computational power.
- Struggles in noisy environments.
- Needs language-specific fine-tuning.

2.2 SEAMLESSM4T : MULTIMOIDAL MACHINE TRANSLATION

Tang,.Y, Bapna,A., et al.

This paper introduces SeamlessM4T, a unified model for speech-to-text and speech-to-speech translation. It supports over 100 languages and integrates ASR, translation, and TTS. The system preserves speaker traits like gender and accent. It performs strongly in multilingual benchmarks. The model is trained end-to-end using multimodal and multilingual datasets.

Merits

- Handles both text and voice translation.
- Supports over 100 languages.
- Preserves speaker identity.

Demerits

- High memory and compute usage.
- Limited real-time usage.
- Hard to adapt/customize.

2.3 ROBUST SPEECH RECOGNITION VIA LARGE-SCALE WEAK SUPERVISION

Radford, A., et al.

Whisper is a speech recognition model trained on 680,000 hours of multilingual data. It performs language detection and transcription, even under noisy conditions. It supports multiple languages and performs better than many supervised models. Whisper is open-source, making it accessible for developers. Its robustness comes from weak supervision on diverse datasets.

Merits

- High noise tolerance.
- Multilingual and multitask support.
- Open-source and community friendly.

Demerits

- Slower on CPU systems.
- May produce minor translation errors.
- Lacks voice output (TTS not included).

2.4 MASSIVELY MULTILINGUAL SPEECH

Pratap, V., et al

The MMS project trains models on over 1,100 languages using publicly available data. It aims to cover underserved and endangered languages in speech recognition. The model achieves state-of-the-art results with minimal labeled data. It improves language coverage through large-scale multilingual pretraining. The project helps in preserving language diversity.

Merits

- Supports over 1,100 languages.
- Good for low-resource language preservation.
- Minimizes need for transcribed data.

Demerits

- May be less accurate for common languages.
- Requires massive storage.
- Still in early application stage.

2.5 MULTILINGUAL END TO END SPEECH TRANSLATION WITH MODALITY AGNOSTIC META LEARNING

Inaguma, H., et al.

This paper proposes a speech translation system that applies meta-learning to handle multiple source languages in a single model. It performs end-to-end translation without intermediate text, improving latency and efficiency. The model adapts to new languages quickly. Results show high BLEU scores across different language pairs. It’s optimized for real-time use cases.

Merits

- Real-time translation capability.
- Learns fast with few-shot samples.
- Avoids text-based bottlenecks.

Demerits

- Complex training procedure.
- Limited voice output clarity.
- Few supported languages in early version.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The domain of multilingual voice message translation has seen significant advancements in recent years, particularly due to developments in speech recognition, neural machine translation (NMT), and text-to-speech (TTS) synthesis. Various organizations and research groups have developed systems that address portions of the voice translation pipeline—such as converting speech to text, translating text between languages, and converting text back to speech. However, most existing systems are either restricted in language support, depend on internet connectivity for processing, or are fragmented (not integrated end-to-end).

One of the most notable existing systems is Google Translate, which supports speech-to-text and text-to-speech functions in over 100 languages. Although it provides basic voice translation, it often works in a disjointed manner where speech must be first transcribed, then translated, and finally spoken out—each step executed separately. Additionally, Google Translate struggles with preserving speaker characteristics and tone in the output voice. It also relies heavily on internet-based APIs, which may not be ideal for all use cases.

Another leading system is Meta AI's SeamlessM4T, a state-of-the-art multilingual and multimodal translation model introduced in 2023. It supports speech-to-text and speech-to-speech translation for over 100 languages in a unified pipeline. This system is highly accurate and preserves certain speaker features, such as accent and gender. However, SeamlessM4T is computationally expensive, demands significant hardware resources (such as GPUs and TPUs), and is currently not fully open-source for real-time deployment on personal devices.

OpenAI's Whisper, another widely used system, excels in speech recognition and language detection. It can transcribe audio in various languages and works well in noisy environments. While Whisper provides excellent transcription accuracy, it does not include native translation or voice synthesis modules. Developers must manually integrate additional APIs to translate the text and generate audio output, leading to complexity and compatibility challenges.

Additionally, commercial tools such as Amazon Transcribe, Microsoft Azure Speech Services, and IBM Watson Speech-to-Text offer multilingual transcription services but are typically limited in terms of real-time processing, offline support, and pricing accessibility for small-scale or academic use. These services are often tailored for enterprise solutions and lack customization for end-to-end translation workflows in a unified interface.

3.2 PROPOSED SYSTEM

The proposed system is a comprehensive multilingual voice message translation application designed to take a voice input in any language—such as Tamil, Hindi, Malayalam, Kannada, French, Arabic, or English—and convert it into a translated voice output in a user-selected target language. The system follows a sequential process beginning with speech recognition, where the input audio is transcribed and the source language is automatically detected using a deep learning model like OpenAI's Whisper. The resulting text is then translated into the desired target language using a neural machine translation (NMT) model or service. After translation, the text is converted into human-like speech using a natural Text-to-Speech (TTS) engine, with voice characteristics such as gender and accent appropriately matched to the target language.

The entire process is wrapped in a user-friendly interface built with frameworks like Gradio or Streamlit, allowing users to upload audio files and obtain translated voice outputs with minimal technical knowledge. This system

aims to bridge communication barriers across different languages and offers a practical solution for education, healthcare, customer support, and cross-cultural interactions. It is scalable, supports both offline and online deployment modes, and focuses on delivering high accuracy, speed, and naturalness in voice outputs.

3.2.1 Merits

Multilingual Support

Capable of handling voice input and output in multiple languages, including regional and foreign languages like Tamil, Hindi, French, and Arabic.

End-to-End Integration

Combines speech recognition, translation, and voice synthesis into a single, seamless pipeline without requiring multiple tools or platforms.

Natural Voice Output

Produces high-quality, human-like speech in the selected language, improving user experience and understanding.

User-Friendly Interface

Offers an intuitive and clean interface where users can simply upload their voice message and receive translated audio with minimal effort.

Offline and Online Flexibility

Can be configured to work in offline mode using local models, or online using cloud APIs, based on user needs.

CHAPTER 4

SYSTEM SPECIFICATIONS

4.1 HARDWARE SPECIFICATIONS

Processor: Intel i5 or higher / AMD Ryzen 5 or higher (for local processing).

RAM: Minimum 8 GB (16 GB recommended for faster processing).

Storage: At least 10 GB of free disk space.

Graphics Card (Optional): NVIDIA GPU (for faster model inference with Whisper or TTS).

Microphone/Speakers: Required for audio input/output (in live version).

Internet Connection: Required for API-based translation or TTS (optional if offline setup is used).

4.2 SOFTWARE SPECIFICATIONS

Operating System: Windows 10/11, Linux (Ubuntu), or macOS.

Programming Language: Python 3.10+.

Python Libraries: gradio, openai-whisper , googletrans or deep-translator, torch.

Development Tools: VS Code / PyCharm (optional but helpful IDEs),Command Prompt / Terminal, pip (Python package manager).

APIs (Optional for Cloud-Based TTS): Microsoft Azure TTS API, Google Translate API.

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

The proposed system architecture for the multilingual voice message translation application is structured to handle voice input in multiple languages and deliver the translated output as an audio message in a selected target language. The system follows a modular pipeline that integrates several natural language processing and speech technologies, with a user-friendly frontend and a robust backend to ensure smooth functionality.

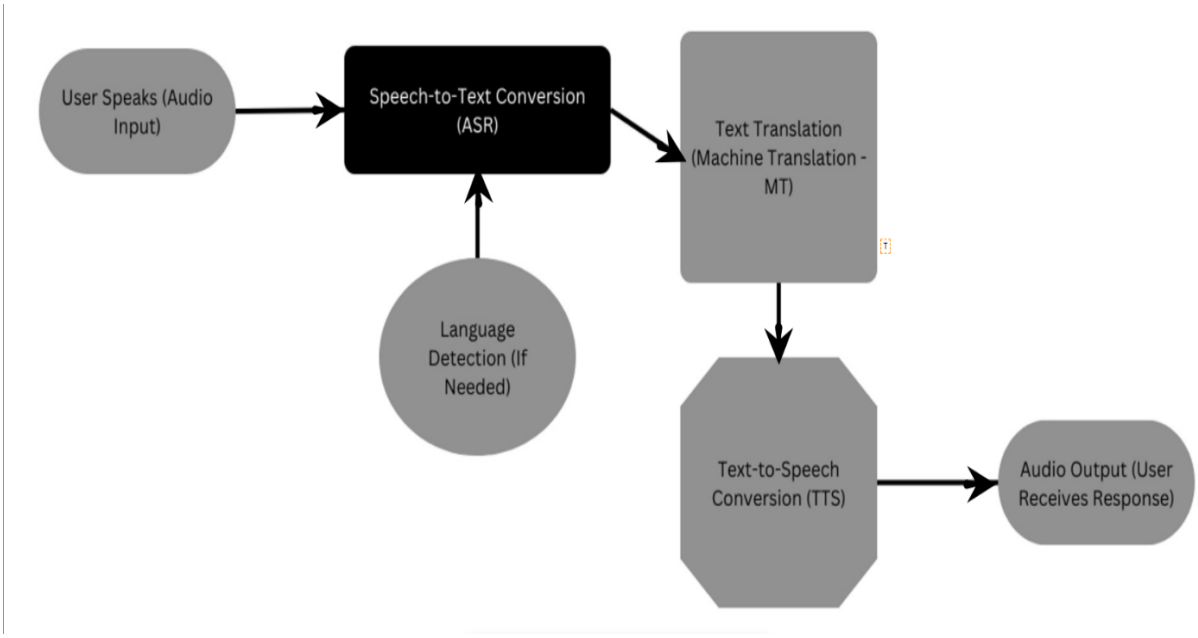

Fig. 5.1 System Architecture

The system architecture of the multilingual voice message translation application consists of a sequence of interconnected modules designed to process and transform voice inputs seamlessly. Initially, the user uploads an audio voice message through a web-based interface. This audio is then passed to the Speech Recognition module, which employs a powerful model like OpenAI's Whisper

to transcribe the spoken content into text while simultaneously detecting the language of the original audio.

The transcribed text is then forwarded to the Translation module, where a neural machine translation engine converts the text from the source language to the target language selected by the user. Following translation, the Text-to-Speech (TTS) module synthesizes the translated text into natural-sounding speech in the desired language, offering options for voice gender and tone. Finally, the translated voice message is delivered back to the user via the interface, allowing them to listen or download the output audio. This modular design ensures a smooth flow from voice input to translated voice output, enabling effective communication across different languages.

The synthesized voice message is delivered back to the frontend, where users can either listen to the audio directly or download it for offline use. This end-to-end system provides a seamless workflow from multilingual voice input to cross-language voice output, making it suitable for various applications including translation assistance, cross-cultural communication, and educational tools.

Overall, this architecture supports scalability, language extensibility, and real-time interaction, while maintaining modularity so that each component can be improved or replaced independently. This makes the system adaptable for future enhancements, such as emotion-aware speech, dialect handling, or offline processing.

CHAPTER 6

MODULES DESCRIPTION

6.1 VOICE INPUT MODULE

The Voice Input Module serves as the entry point of the system, where users upload or record a voice message. This module is designed to support a wide range of languages, including Tamil, Hindi, Malayalam, Kannada, Arabic, French, and English. The quality and clarity of the voice input play a significant role in ensuring the accuracy of downstream processes such as transcription and translation. By integrating Gradio's audio component, this module ensures a user-friendly experience for capturing high-quality voice data.

Components

Gradio Audio Input Component: Allows users to either record live audio or upload a prerecorded file (in formats like .wav, .mp3, or .ogg).

Microphone Access (optional): If users opt for live recording, access to the system microphone is requested.

File Handling Script: Receives and temporarily stores the audio file for further processing.

Working

Audio Input Layer

The module uses the Gradio framework to render an audio interface that accepts both live recordings and uploaded audio files. When a user interacts with this component, the browser either accesses the microphone (WebRTC API) or the file system to load the data.

Preprocessing Logic

Once audio is captured, it undergoes preprocessing to:

- Convert non-standard formats (e.g., .ogg, .m4a) to .wav

- Normalize volume and remove silence (if needed).
- Resample to a standard sampling rate (e.g., 16000 Hz mono).

Validation

The system checks for common issues like:

- Extremely short or empty files.
- Excessive background noise.
- Incorrect format or corrupt data.

Data Transfer

The preprocessed and validated audio file or buffer is then forwarded to the Speech-to-Text module.

Functions

The Voice Input Module handles all tasks related to collecting and preparing audio for processing. Its primary function is to accept voice messages either through direct microphone recording or by uploading an existing audio file. It ensures compatibility by validating formats like .mp3, .wav, .m4a, etc., and performs preprocessing such as trimming silence, resampling (e.g., to 16 kHz mono), and normalizing volume levels. It also ensures that the duration of audio does not exceed preset limits and that the content is sufficiently clear for transcription. After validation, the audio file is temporarily stored and passed securely to the transcription module. This module plays a critical role in ensuring input quality and standardization.

- Accepts user audio in real time or via upload.
- Preprocesses and validates the input.
- Converts and standardizes the audio format.

6.2 SPEECH TO TEXT (TRANSCRIPTION) MODULE

This module is responsible for converting the input voice message into text using Automatic Speech Recognition (ASR). It utilizes the OpenAI Whisper model, which supports multilingual transcription and automatic language

detection. The audio is first transformed into a log-mel spectrogram, which is the input format expected by Whisper. The model's encoder-decoder architecture processes the spectrogram and generates tokenized text in the original language. Whisper also includes punctuation and proper casing, which improve readability. This module is crucial as it transforms unstructured voice data into a structured text format, making it suitable for translation. It supports a variety of accents and noisy environments with good accuracy.

Components

Whisper ASR Engine: Pretrained model from OpenAI for speech transcription.

Audio Preprocessor: Converts audio into log-mel spectrograms.

Language Auto-Detector: Whisper includes built-in language identification.

Working

Spectrogram Generation

The audio file is first converted into a log-mel spectrogram. Whisper uses this visual representation of audio frequencies as the model input.

Language Detection

Whisper tries to identify the spoken language from the audio using internal heuristics and a trained language classifier.

Decoding & Transcription

The model uses encoder-decoder architecture with attention mechanisms:

- The encoder processes the spectrogram to extract features.
- The decoder translates these features into tokenized text.
- It maps audio segments to word sequences using CTC (Connectionist Temporal Classification) or transformer-based decoding.

Punctuation & Capitalization

Whisper is capable of predicting punctuation and correct casing, which improves readability.

Output

The resulting transcription in the source language is passed to the Translation Module.

Functions

The main function of the Speech-to-Text Module is to convert spoken content into written text using automatic speech recognition. It processes the input audio into a spectrogram and feeds it to the OpenAI Whisper model, which handles transcription and automatic language detection. It outputs accurately punctuated and capitalized text, improving clarity and readability. Whisper's multilingual support allows it to recognize and transcribe speech in languages such as Tamil, Hindi, Arabic, and more. Additionally, it includes noise-handling capabilities to improve output from low-quality audio. This module ensures a reliable transformation from audio to a machine-readable text format for the next stage.

- Transcribes audio to accurate, punctuated text.
- Auto-detects the spoken language (if enabled).
- Outputs clean, structured text for translation.

6.3 TRANSLATION MODULE

The Translation Module takes the transcribed text and converts it into the target language selected by the user. This is achieved using the Google Translator API accessed via the Deep Translator Python library. It supports many language pairs and ensures contextual and semantic accuracy. The system identifies the language codes (e.g., "en" for English, "ta" for Tamil) and constructs a translation request. Google's Neural Machine Translation (NMT) models

process the request and return fluent, context-aware translations. This module ensures that the message retains its original meaning across languages and prepares the output text for speech synthesis.

Components

Deep Translator Library: Interface for translation APIs.

Google Translator API: Performs language translation at scale.

Error Handler: Manages unsupported languages and fallback scenarios.

Working

Input Validation

Receives transcribed text and the user's selected target language (e.g., French → Tamil).

Language Mapping

Internally converts the language names to ISO 639-1 language codes (e.g., "Tamil" → "ta").

API Request

A translation request is made to Google Translate's API with parameters:

- Source language (if known)
- Target language
- Text content

Translation Processing

Google Translate uses neural machine translation (NMT) models to:

- Tokenize input.
- Map to high-dimensional vector space.
- Generate target language sequence with context preservation.

Output Handling

The translated text is cleaned (if needed) and forwarded to the TTS module.

Functions

This module translates the transcribed text into the target language selected by the user. Using the Google Translate API via Deep Translator, it supports fast and accurate translations between over 100 language pairs. Its core functions include detecting language codes, constructing API requests, and handling text encoding. The translation engine maintains semantic accuracy and adjusts grammar and context based on the target language. The module also includes error-handling functions for unsupported or misrecognized inputs and ensures that the translated text is suitable for TTS processing. It acts as the bridge between text in the original language and the final output language.

- Translates transcribed text into the selected language.
- Maintains meaning, tone, and structure.
- Returns target language text for speech synthesis.

6.4 TEXT TO SPEECH MODULE

The TTS module converts the translated text into a natural-sounding audio file. It uses Edge TTS, which is a wrapper for Microsoft Azure's neural voice models. This module supports voice customization such as gender, accent, and emotional tone. The translated text is fed into the TTS engine, which first converts it into phonemes and then applies prosody (intonation, rhythm) to generate speech. The output is an .mp3 file with human-like articulation. This module completes the voice translation loop by generating a new voice message in the desired language that closely mimics natural human speech.

Components

Edge TTS Python SDK: Generates audio from text using cloud APIs.

Voice Profile Selector: Allows configuration of language, gender, tone.

Audio Output Writer: Saves final voice output as .mp3.

Working

Text Input

Receives translated text from the previous module.

Voice Configuration

Sets voice parameters:

- Language code (e.g., “ta-IN” for Tamil, “fr-FR” for French).
- Voice type (e.g., “male”, “neural”).
- Style (e.g., cheerful, calm).

Synthesis

The system sends the text and configuration to Microsoft’s TTS API, which:

- Converts text to phonemes.
- Applies prosody (pitch, rhythm, stress).
- Generates waveform using neural vocoders (e.g., WaveNet).

Output

The speech waveform is saved as .mp3 or streamed to the frontend.

Functions

The TTS module’s function is to convert the translated text into human-like speech. Using Edge TTS (powered by Microsoft Azure), it transforms text input into natural-sounding voice output in various languages and dialects. It selects the appropriate voice profile based on the chosen language (e.g., female French voice or male Arabic voice), applies emotional tone where possible, and generates an .mp3 or .wav file. It also allows real-time playback and audio file

saving. The module ensures that the synthesized speech is intelligible, expressive, and pleasant to listen to. It finalizes the multilingual voice transformation process.

- Converts translated text to fluent speech.
- Customizes voice for realism and clarity.
- Provides audio output to the UI for playback or download.

6.5 USER INTERFACE MODULE

The User Interface Module ties all other components together and provides a clean, accessible way for users to interact with the system. Developed using Gradio, it includes audio input controls, dropdowns to select source and target languages, and buttons to trigger translation. Once the process is complete, it displays the transcribed and translated text and provides an embedded audio player for playback. Users can also download the output. The UI acts as the bridge between users and the backend, simplifying complex operations into a single click. It ensures usability and enhances the overall user experience.

Components

- Audio Recorder/Uploader.
- Dropdowns for Source & Target Languages.
- Submit Button & Status Indicators.
- Audio Player for Output.

Working

Input Interface

The user is presented with two dropdown menus and an audio input widget.

Event Trigger

When the user uploads audio and clicks “Translate”, the UI sends these inputs to the backend function.

Backend Routing

Gradio internally maps the inputs to a Python function, which coordinates the other modules.

Output Rendering

Once the translated voice file is returned, it is embedded in the interface using an HTML5 audio player. A download button is also available.

Functions

The User Interface Module serves as the interaction hub between the user and the backend modules. It presents inputs such as audio recorders/uploaders, language dropdowns, and action buttons. Its functions include collecting inputs, validating them, passing data to backend functions, and rendering output. It displays the transcribed text, translated text, and provides an embedded audio player for listening to the final voice output. It also offers a download button for saving the result. Gradio simplifies the deployment and use of the system, making it accessible even for non-technical users.

- Collect user inputs and route them to backend.
- Display status and audio outputs.
- Provide user-friendly, zero-setup access to the system.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

The Multilingual Voice Translation System successfully demonstrates the integration of modern AI tools and cloud services to enable seamless voice-to-voice communication across different languages. In a world where linguistic diversity often becomes a barrier to communication, this project provides a practical and accessible solution by converting voice messages from one language into another, both in text and spoken format.

The system was developed using a modular architecture, which includes voice input, automatic speech recognition (ASR), language detection and translation, text-to-speech synthesis (TTS), and an intuitive user interface built with Gradio. The use of the OpenAI Whisper model ensures accurate multilingual transcription, while the Deep Translator API with Google Translate provides fast and reliable language translation. Finally, Edge TTS adds realism and clarity to the output speech, creating a near-human voice that makes the system suitable for everyday use.

Through this project, users are empowered to communicate beyond language boundaries without needing to understand the intermediary steps. This system holds enormous potential for various applications including education, customer service, healthcare, tourism, and cross-cultural communication.

In conclusion, the project achieves its objective of simplifying multilingual communication through a voice-based interface. It reflects the power of combining open-source AI models and cloud-based APIs to solve real-world problems and stands as a scalable foundation for further innovations in the field of speech and language processing.

7.2 FUTURE ENHANCEMENT

The Multilingual Voice Translation System has strong potential for expansion and improvement beyond its current capabilities. One major enhancement is the addition of real-time translation, enabling users to have live multilingual conversations with minimal latency—ideal for video calls, meetings, and customer service bots.

Another enhancement is the development of a mobile or desktop application, allowing users to access the system offline or on-the-go. Incorporating more regional dialects and accents will improve usability in diverse linguistic communities, especially in India and Africa.

Support for emotion-aware TTS, where the system reflects tone and emotion (e.g., happy, sad, angry), can improve the quality of communication. Integration with messaging platforms like WhatsApp or Telegram can also allow users to translate and send voice messages directly within chat apps.

Finally, expanding language coverage, improving model speed with GPU acceleration, and enabling speaker recognition and context-based translation memory will make the system more intelligent and user-friendly in future versions.

APPENDIX A

SOURCE CODE

```
import gradio as gr
import whisper
from deep_translator import GoogleTranslator
import os
import uuid
import asyncio
import edge_tts

# Load the Whisper model for speech-to-text
model = whisper.load_model("base")

# Language and voice settings with correct Edge TTS voice names
language_settings = {
    "English": {"code": "en", "voice": "en-US-GuyNeural"},
    "Tamil": {"code": "ta", "voice": "ta-IN-ValluvarNeural"},
    "Hindi": {"code": "hi", "voice": "hi-IN-MadhurNeural"},
    "Malayalam": {"code": "ml", "voice": "ml-IN-MidhunNeural"},
    "Kannada": {"code": "kn", "voice": "kn-IN-GaganNeural"},
    "French": {"code": "fr", "voice": "fr-FR-HenriNeural"},
    "Arabic": {"code": "ar", "voice": "ar-SA-HamzaNeural"},
}

# Asynchronous TTS audio generator using Edge TTS
```

```
async def generate_edge_tts(text, voice, output_path):
    communicate = edge_tts.Communicate(text=text, voice=voice)
    await communicate.save(output_path)

# Main function to transcribe, translate, and generate speech
def transcribe_translate_speak(audio_path, target_language_name):
    try:
        lang_code = language_settings[target_language_name]["code"]
        voice = language_settings[target_language_name]["voice"]

        # Step 1: Transcribe the audio
        result = model.transcribe(audio_path)
        original_text = result["text"]

        # Step 2: Translate to target language
        translated_text = GoogleTranslator(source='auto',
target=lang_code).translate(original_text)

        # Step 3: Convert translated text to speech using Edge TTS
        output_audio_path = f"translated_{uuid.uuid4().hex}.mp3"
        asyncio.run(generate_edge_tts(translated_text, voice, output_audio_path))

        return f"🗣️ Original Text:\n{original_text}\n\n🌐 Translated to
{target_language_name}:\n{translated_text}", output_audio_path

    except Exception as e:
```

```
        return f"❌ Error: {str(e)}", None


# Gradio UI Interface
interface = gr.Interface(
    fn=transcribe_translate_speak,
    inputs=[
        gr.Audio(type="filepath", label="🎤 Upload Your Voice Message"),
        gr.Dropdown(choices=list(language_settings.keys()), label="🌍 Select Target Language"),
    ],
    outputs=[
        gr.Textbox(label="📄 Transcription & Translation"),
        gr.Audio(label="🔊 Translated Voice Output (Natural Male Voice)"),
    ],
    title="🌐 Multilingual Voice Translator",
    description="Upload an audio file in any language (Tamil, Hindi, Malayalam, Kannada, Arabic, French, English) and get a translated audio response in the selected language with a natural male voice."
)

# Launch the app
interface.launch()
```

APPENDIX B

SCREENSHOTS

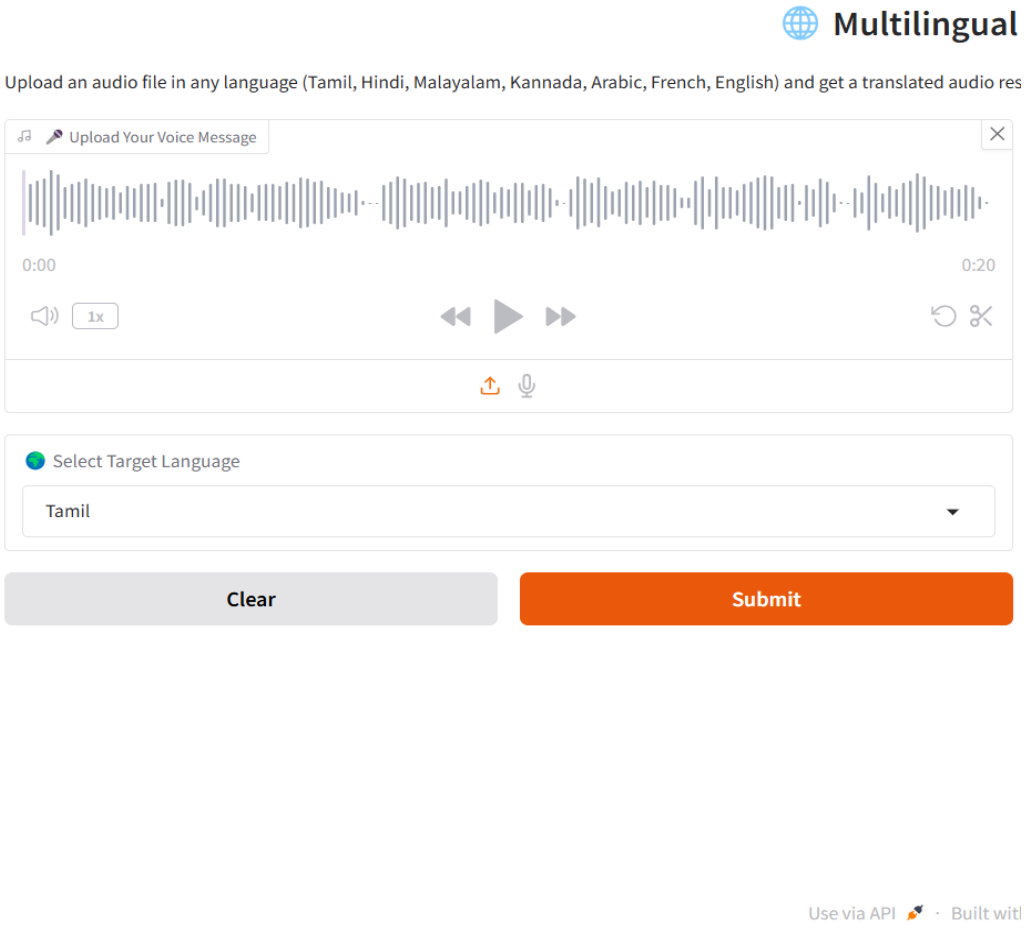

Fig. B.1 Model Input

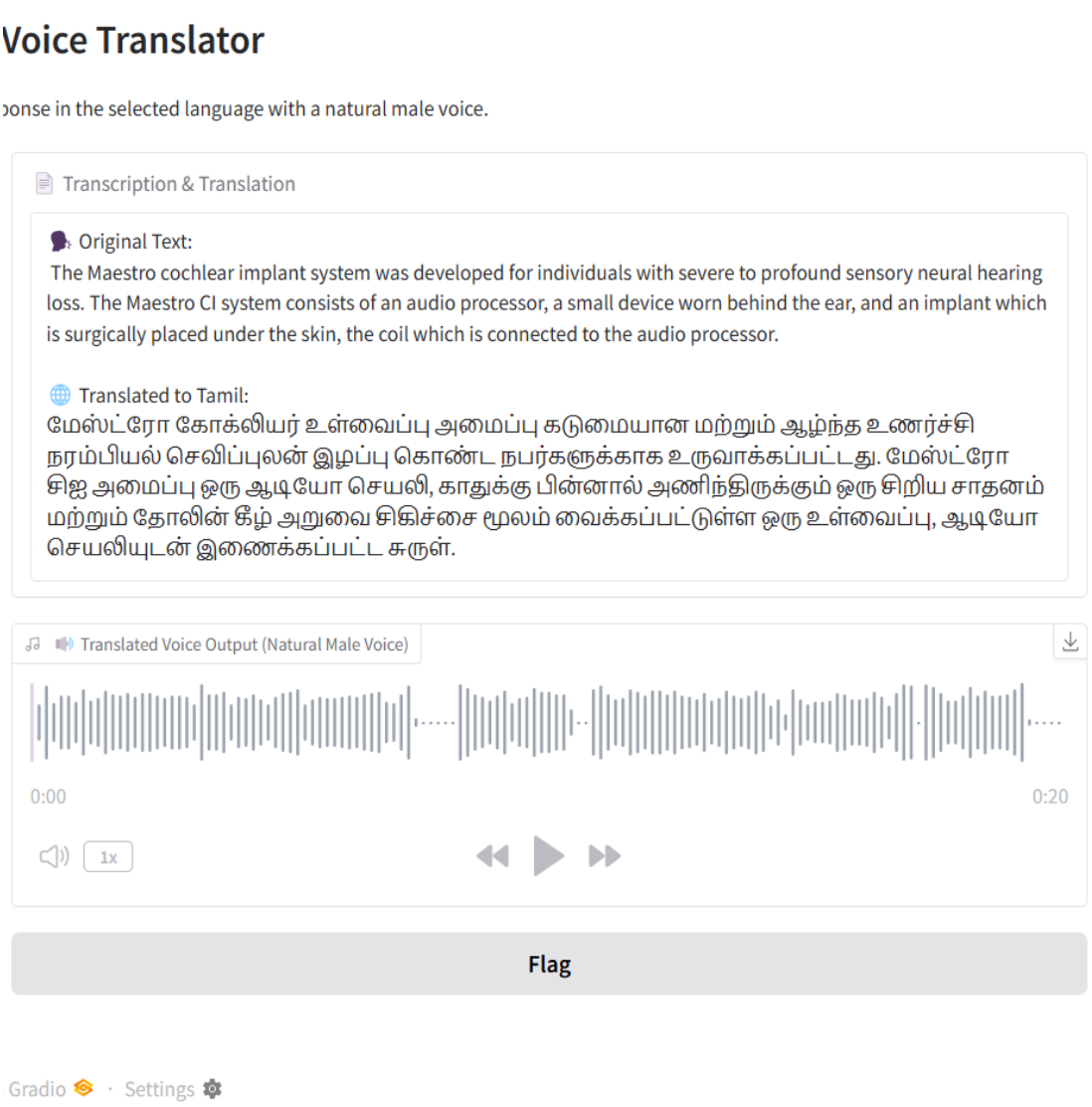

Fig. B.2 Model Output

REFERENCES

1. Sankar, C., Subramanian, S., & Dupoux, E, “Multilingual speech translation using wav2vec 2.0 and transformer-based models. In Proceedings of ACL Findings”, 2023.

2. Guo, J., Zhang, Y., Xu, H., Liu, Z., & Xu, B. “Cross-lingual speech synthesis via knowledge distillation for low-resource languages”. 2023.

3. Jia, Y., Zhang, Y., Weiss, R. J., Wang, “Direct speech-to-speech translation with discrete units”,2022.

4. Zhou, H., Wu, Y., Zhang, W., & Li, M. “Improving end-to-end multilingual speech recognition with language adapters”. In Proceedings of Interspeech, 2022.

5. Jansen, A., Zhen, L., & Ma, J.. “Zero-shot multilingual speech recognition using language-agnostic representations”. 2022.

6. Sato, R., Ueno, Y., Hayashi, T., & Watanabe, S. “Multilingual speech recognition with language-aware Transformer”.2022.

7. Popuri, S., Jain, M., Chuang, Z., Zhang, “Enhanced multilingual speech recognition using self-supervised learning and transfer learning”. In IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), 2021.

8. Chen, N., Sun, X., Xie, L., & Li, “Voice Conversion with Multilingual Phonetic Posteriorgrams”. In IEEE Transactions on Audio, Speech, and Language Processing, 2021.

9. Bahar, P., Iranzo-Sánchez, J., Jaitly, N., & Ney, H. “Multilingual end-to-end speech translation with a shared encoder and language-specific decoders”.2021.

10. Gong, Y., Chung, Y. A., & Glass, J. “AST: Audio Spectrogram Transformer for Audio Classification”. In Proceedings of Interspeech, 2021.