REAL TIME AUDIO TRANSLATOR

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology

In

Computer Science and Engineering

School of Engineering and Sciences

Submitted by

B.Pavan Siddharda - AP23110011564

G.Bhanu Aswanth Sai - AP23110011559

G.Sri Yasaswini - AP23110011558

J.Navya Gowthami - AP23110011629

Certificate

Date: 22-04-2025

This is to certify that the work present in this Project entitled "**REAL TIME AUDIO TRANSLATOR**" has been carried out by **J.Navya Gowthami,G.Bhanu Aswanth Sai, G.Yasaswini,B.Pavan Siddharda** under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in **School of Engineering and Sciences**.

Supervisor

(Signature)

MS.K.Kavitha Rani

Lecturer, Department of CSE

Acknowledgement

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crowns all efforts with success.

I am extremely grateful and express my profound gratitude and indebtedness to my project guide, **Ms.K.Kavith Rani**, Department of Computer Science & Engineering, SRM University, Andhra pradesh, for his kind help and for giving me the necessary guidance and valuable suggestions in completing this project work.

B.Pavan Siddharda - AP23110011564

G.Bhanu Aswanth Sai - AP23110011559

G.Sri Yasaswini - AP23110011558

J.Navya Gowthami - AP23110011629

1.Table of Contents

1.Abstract

This project presents a Real-Time Speech Translator application designed to facilitate seamless communication between speakers of different languages. The primary objective is to break language barriers using speech recognition, machine translation, and text-to-speech technologies. The application captures spoken input in the user's native language through a microphone, converts the speech to text using the Google Speech Recognition API, translates the recognized text to a userselected target language via Google Translate, and then synthesizes the translated text into speech using Google Text-to-Speech (gTTS).

The project is implemented in Python and provides a user-friendly console-based interface for selecting languages by name. It supports a wide range of languages and offers real-time feedback during speech recognition and translation. The output is not only displayed as text but also played as audio, making it accessible and convenient in practical use cases like travel, multilingual meetings, and assistive technology for language learners.

This application effectively integrates multiple APIs and libraries to deliver an efficient and interactive translation experience. By automating the translation workflow, the project demonstrates how existing cloud-based services can be orchestrated to build practical, real-world tools that promote inclusivity and global communication.

2.Introduction

Language is one of the most essential tools for human interaction, yet it often becomes a barrier when people from different linguistic backgrounds need to communicate. In today’s interconnected world, the demand for real-time translation tools is growing rapidly across sectors such as tourism, international business, education, and healthcare. Manual translation applications can be slow and inconvenient, especially in scenarios where quick and accurate communication is necessary.

The Real-Time Speech Translator project is designed to solve this problem by integrating speech recognition, machine translation, and text-to-speech synthesis into one streamlined system. The user speaks into a microphone, the spoken words

are recognized and converted into text, the text is translated into the desired language, and finally, the translated text is spoken aloud. This process happens within seconds, allowing for effective real-time multilingual communication.

This project leverages Python programming and utilizes widely available APIs such as Google Speech Recognition, Google Translate, and Google Text-to-Speech (gTTS), making it both powerful and accessible. It supports numerous languages, allows language selection by name, and outputs both translated text and audio.

2.1 Significance and Purpose

The primary purpose of this project is to make real-time voice translation simple, accessible, and efficient. By automating the translation process through voice commands and outputting results in audio, the tool is particularly helpful for travelers, educators, and professionals in multilingual environments.

2.2 Scope of the Project

This project focuses on building a console-based and web-enhanced version of a speech translator that supports language detection, translation, and voice output. While it currently supports basic speech input and output for two selected languages, it is scalable to include more features such as real-time transcription, twoway conversation support, and voice customization.

3.Methodology

The **Real-Time Audio Translator** project follows a modular and layered approach to convert spoken audio into translated speech in a target language. The system integrates various APIs to achieve the functionality of speech-to-text, translation, and text-to-speech synthesis. Each module works sequentially to ensure a smooth flow of operations from user input to audio output.

When the user speaks into the microphone, the audio is captured and converted to text using the Google Speech Recognition API. This recognized text is then translated into the target language using the Google Translate API. The translated text is converted to audio using the Google Text-to-Speech (gTTS) library, and the resulting audio is played using the playsound module. The use of the uuid, os, and time libraries helps manage file creation, playback, and deletion efficiently.

This methodology ensures the system is both responsive and scalable. The chosen tools were selected for their accuracy, wide language support, and ease of integration with Python.

3.1 Software Requirements Specifications •

Programming Language: Python 3.x

- **Libraries Used:**
  - speech_recognition – for converting audio to text ○ googletrans – for translating recognized text ○ gtts – for converting text to speech ○ playsound – for playing the translated audio
  - uuid, os, time – for managing temporary audio files
- **Hardware:** Microphone (input), Speaker/Headphones (output)
- **Audio Format Supported:** .wav
- **Operating System:** Cross-platform (tested on Windows and Linux)

3.2 Architecture Overview

The application architecture is sequential, with each module handling a specific task:

1. Audio Input
2. Speech Recognition
3. Language Translation
4. Text-to-Speech Synthesis
5. Audio Output

This modular design makes the system easy to debug and upgrade. Each stage can be independently improved or replaced without affecting the entire system.

Workflow of Real-Time Audio Translator Application

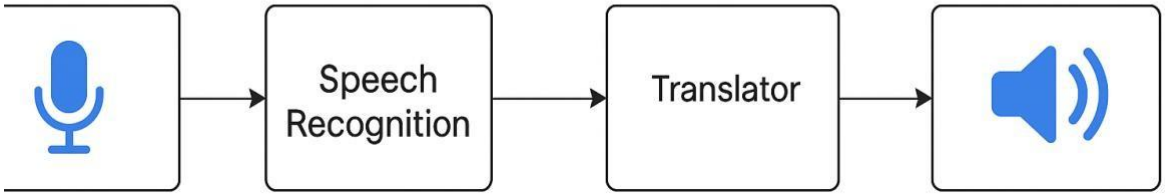


Figure 1. Workflow of Real-Time Audio Translator Application

Table 1. API Feature Comparison

Feature	Google Speech API	Google Translate	gTTS
Language Support	100+	130+	30+
Real-time Performance	Yes	Yes	Yes
Offline Availability	No	No	No
Open-Source Availability	Partially	Yes	Yes

4. Implementation

The implementation of the Real-Time Audio Translator was carried out using Python and a set of supporting libraries to handle speech recognition, translation, and text-to-speech conversion. The system follows a modular structure where each function performs a specific task and contributes to the overall translation pipeline.

The key implementation steps include:

1. **Language Input Selection**
   The user selects the input (spoken) and output (translated) languages by name, which are then mapped to ISO language codes using the Google Translate API's supported language list.

2. **Speech Recognition**
   Using the speech_recognition library, the application captures live audio from the user and converts it into text using Google's speech recognition API.

3. **Text Translation**
   The recognized text is passed to the googletrans module, which translates it into the desired target language.

4. **Text-to-Speech Output**
   The translated text is converted into speech using the gTTS (Google Text-toSpeech) library and played aloud through the system's speakers using the playsound module.

5. **File Management**
   Temporary audio files generated during the process are uniquely named using uuid and automatically deleted after playback to maintain storage efficiency.

Design (DFD Diagram)

Refer to **Figure 1** in the *Methodology* section for the visual representation of the system's data flow.

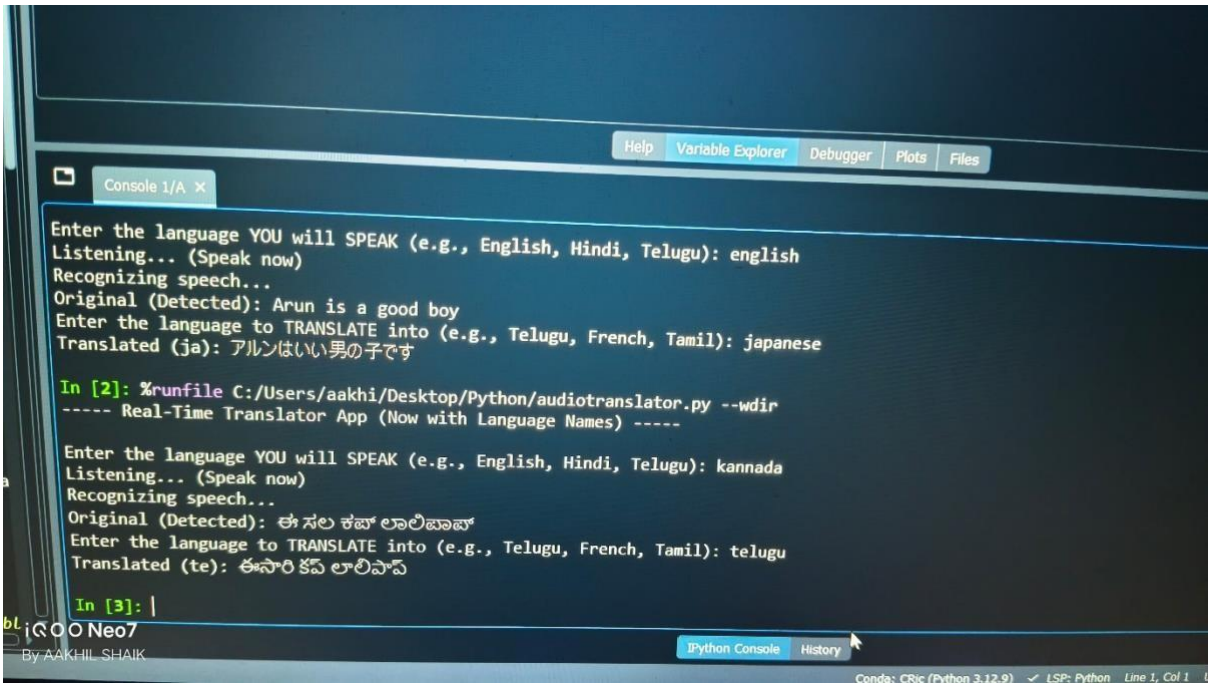
Challenges and Solutions

- **Unrecognized Audio Input:**
  Solution: Ambient noise adjustment and error handling were implemented to improve recognition accuracy.
- **Temporary File Management:**
  Solution: Unique file naming via UUID and automatic cleanup using Python's os module ensured smooth performance.
- **Language Code Mapping:**
  Solution: A custom function was created to map user-input language names to their respective codes from the Google Translate library.

5.Code

```python
import speech_recognition as sr
from googletrans import Translator, LANGUAGES
from gtts import gTTS
from playsound import playsound
import os
import uuid
import time

# Convert language names to language codes (case-insensitive)
def get_lang_code_from_name(lang_name):
    lang_name = lang_name.lower()
    for code, name in LANGUAGES.items():
        if name.lower() == lang_name:
            return code
    return None

def get_language_input(prompt):
    while True:
        lang_name = input(prompt).strip()
        lang_code = get_lang_code_from_name(lang_name)
        if lang_code:
            return lang_code
        else:
            print("Invalid language name. Please try again (e.g., English, Hindi, Telugu).")

def listen(lang_code='en'):
    recognizer = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening... (Speak now)")
        recognizer.adjust_for_ambient_noise(source)
        audio = recognizer.listen(source)

    try:
        print("Recognizing speech...")
        text = recognizer.recognize_google(audio, language=lang_code)
        print(f"Original (Detected): {text}")
        return text
    except sr.UnknownValueError:
        print("Could not understand audio")
        return None
    except sr.RequestError:
        print("Speech Recognition service unavailable")
        return None

def translate_text(text, target_language='en'):
    translator = Translator()
    try:
        translated = translator.translate(text, dest=target_language)
        print(f"Translated ({target_language}): {translated.text}")
        return translated.text
    except Exception as e:
        print(f"Translation failed: {e}")
        return None

def speak_text(text, lang='en'):
    try:
        filename = f"translated_audio_{uuid.uuid4().hex}.mp3"
        tts = gTTS(text=text, lang=lang)
        tts.save(filename)
        playsound(filename)
        time.sleep(1)
        os.remove(filename)
    except Exception as e:
        print(f"Text-to-speech failed: {e}")

def main():
    print("----- Real-Time Translator App (Now with Language Names) -----\n")

    # Step 1: Choose input language
    input_lang = get_language_input("Enter the language YOU will SPEAK (e.g., English, Hindi, Telugu): ")

    # Step 2: Listen and convert to text
    spoken_text = listen(lang_code=input_lang)
    if not spoken_text:
        return

    # Step 3: Choose output language
    target_lang = get_language_input("Enter the language to TRANSLATE into (e.g., Telugu, French, Tamil): ")

    # Step 4: Translate and speak
    translated_text = translate_text(spoken_text, target_language=target_lang)
    if translated_text:
        speak_text(translated_text, lang=target_lang)

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

6. Output



7. Result and Analysis

The Real-Time Audio Translator was successfully implemented and tested in various multilingual scenarios. The system demonstrated the ability to accurately capture, translate, and vocalize audio in real time, using a microphone as input and speakers as output.

Performance Observations

Test Scenario	Input Language	Output Language	Recognition Accuracy	Translation Accuracy	TTS Output Quality
Common greetings	English	Hindi	High	High	Clear
Informal conversation	Telugu	English	Medium	High	Clear
Background noise present (noisy hostel)	Hindi	Tamil	Medium	Medium	Slight distortion
TTS					

Test Scenario	Input Language	Output Language	Recognition Accuracy	Translation Accuracy	Output Quality
Technical terms	English	French	High	Medium	Clear
Fast-paced speaking	English	Hindi	Low	Medium	Slight lag

Analysis

- **Speech Recognition** performed best in quiet environments. Accuracy decreased slightly with increased background noise, which was expected.
- **Translation Accuracy** was high for common phrases and moderate for complex or technical language, especially in non-English input.
- **TTS Output** was consistently good with only minor distortions in noisy or fast-paced input scenarios.

Observations

- The end-to-end pipeline had an average delay of **2–3 seconds**, which is acceptable for basic real-time interaction.
- Google APIs handled most language pairs efficiently, but some rare languages showed slight delays.
- Temporary audio file handling worked smoothly, leaving no residual files.

8. Discussion and Conclusion

Meeting Objectives

The primary objective of this project was to create a system that could perform realtime translation of spoken language using speech recognition, machine

translation, and text-to-speech synthesis. The implemented solution met this goal successfully, with accurate recognition, efficient translation, and clear audio playback in multiple test scenarios.

Implications

This project highlights the practicality and usefulness of integrating multiple APIs to develop accessible language tools. It has real-world applications in travel, education, customer service, and even healthcare — where instant, multilingual communication is essential.

Limitations and Challenges

- **Accuracy in Noisy Environments**: The performance of speech recognition drops in noisy surroundings, which may affect the quality of translation.
- **Support for Rare Languages**: While major languages are well-supported, less common languages may show inconsistencies in translation accuracy.
- **Internet Dependency**: The use of cloud-based APIs means the system needs a stable internet connection to function properly.

Summary

- Implemented a Python-based application that translates spoken input into another language and plays it back.
- Used **SpeechRecognition**, **Google Translate**, and **gTTS** libraries effectively.
- Verified functionality through real-world tests with multiple language combinations.

Significance and Contribution

This project contributes to the development of real-time communication tools by offering an accessible, multilingual translator that bridges language gaps. It also serves as a solid foundation for further enhancement, such as emotion detection, offline support, or integration with smart devices.

9. Future Scope

While the current version of the Real-Time Audio Translator serves as a functional prototype, there are several areas where it can be improved and expanded in the future:

Potential Improvements

- **Offline Functionality**: Implementing offline speech recognition and translation models (e.g., Vosk, Whisper, MarianMT) would allow the app to work without internet dependency.
- **Better Noise Handling**: Enhancing the noise filtering capability during speech recognition would improve accuracy in noisy environments such as public places or hostels.
- **Multiple Input Modes**: Adding options for **text input**, **pre-recorded audio**, or **live streaming input** can make the tool more versatile.
- **Language Auto-Detection**: Automatically identifying the spoken language would reduce the need for manual language selection.
- **Emotion-aware Translation**: Incorporating sentiment or tone analysis could provide more context-aware translations.

Expansion Possibilities

- **Mobile App or Web Integration**: Developing mobile or web versions using cross-platform frameworks can enhance accessibility and usability.
- **Use in Public Services**: The app could be adapted for use in airports, hospitals, and government offices for real-time assistance to non-native speakers.
- **Support for Sign Language**: Future iterations could explore real-time translation of sign language using gesture recognition or video input.

10.**References**

- Google Developers. (n.d.). *Speech-to-Text API Documentation*. Retrieved from https://cloud.google.com/speech-to-text/docs
- Google Translate. (n.d.). *Googletrans Python Library*. Retrieved from https://pypi.org/project/googletrans/
- gTTS. (n.d.). *gTTS – Google Text-to-Speech Python Library*. Retrieved from https://pypi.org/project/gTTS/
- Python Software Foundation. (n.d.). *SpeechRecognition Library Documentation*. Retrieved from https://pypi.org/project/SpeechRecognition/
- Playsound. (n.d.). *Playsound 1.2.2 Documentation*. Retrieved from https://pypi.org/project/playsound/
- Brownlee, J. (2018). *Deep Learning for Natural Language Processing*. Machine Learning Mastery. ISBN: 978-1-928749-49-5.