

**SPEECH TO SIGN LANGUAGE CONVERTER**

**SRIHARI M 21602140**

**NAGAPRABHU N 21602144**

**SANJAI RAMASAMY SA 21602159**

**VELS INSTITUTE OF SCIENCE, TECHNOLOGY & ADVANCED STUDIES**

**(VISTAS)**

**FINAL YEAR PROJECT REPORT**

**PHASE – I**

Submitted to the

**SCHOOL OF ENGINEERING**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

In partial fulfilment of the

Requirements for the award of the degree of

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**VELS INSTITUTE OF SCIENCE, TECHNOLOGY AND ADVANCED STUDIES (VISTAS)**

**CHENNAI – 600 117**

**NOVEMBER 2024**

**BONAFIDE CERTIFICATE**

Certified that this project **“Speech to Sign Language Converter”** is the Bonafide work of **SRIHARI M (21602140), NAGAPRABHU N (21602144), and SANJAI RAMASAMY SA (21602159)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported here does not form part of any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE OF THE HOD SIGNATURE OF THE SUPERVISOR

**Dr. K. Kalaivani M.E., Ph.D. Dr. R. Anandan M.E., Ph.D.**

Associate Professor & Head Professor & Head

Department of CSE Department of CSE

School of Engineering School of Engineering

VISTAS VISTAS

Chennai Chennai

Submitted for the Project Work and Viva – Voce examination held on …………..

at Vels Institute of Science, Technology and Advanced Studies (VISTAS), Chennai – 600 117.

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

It is certainly a great delight and proud privilege to acknowledge the help and support I received from the positive minds around me in making this venture a successful one. The infrastructural support with all kinds of lab facilities have been a motivating factor in this completion of project work, all because of **FOUNDER & CHAIRMAN, Shri Dr. ISHARI K. GANESH.,** with great pleasure that I take this opportunity to thank him. I am grateful to our **HEAD OF THE DEPARTMENT, Dr. K. Kalaivani.,** for her continuous support for the project, from initial advice and contacts in the early stages of conceptual inception, and through on-going advice and encouragement to this day. I extend our warmest thanks to my **INTERNAL GUIDE, Dr. R. Anandan,** who has patiently guided and provided us with valuable advice and help. I am thankful to my Project Coordinator **Dr. R. Anandan,** I would like to express my gratitude to our Staff members, Librarian, and Non-teaching staff members of Computer Centre, who have helped and supported me throughout. Last but not the least, I wish to thank my parents and my friends for their individual support and interest who has inspired me and encouraged me to go in my own way, without them I would be unable to complete this project.

**SRIHARI M 21602140**

**NAGAPRABHU N 21602144**

**SANJAI RAMASAMY SA 21602159**

**TABLE OF CONTENTS**

**CONTENTS**

**ABSTRACT**

1. **INTRODUCTION**
   1. **GENERAL**
   2. **PROBLEM DESCRIPTION**
   3. **OBJECTIVE**
   4. **SCOPE OF THE PROJECT**
2. **LITERATURE SURVEY**
   1. **GENERAL**
   2. **LITERATURE REVIEW**
   3. **EXISTING SYSTEM**
   4. **PROPOSED SYSTEM**
   5. **SUMMARY**
3. **SYSTEM DESIGN**
   1. **GENERAL**
   2. **SYSTEM ARCHITECTURE**
   3. **SYSTEM WORKFLOW**
   4. **METHODOLOGY**
   5. **ALGORITHM**
   6. **SUMMARY**
4. **SYSTEM REQUIREMENTS**
   1. **HARDWARE REQUIREMENTS**
   2. **SOFTWARE REQUIREMENTS**
5. **MODULE IMPLEMENTATION**
   1. **LIST OF MODULES**
   2. **SUMMARY**
6. **SYSTEM IMPLEMENTATION**
   1. **GENERAL**
   2. **OVERVIEW OF THE PLATFORM**
   3. **SOURCE CODE**
   4. **RESULTS & DISCUSSIONS**
7. **TESTING**
   1. **GENERAL**
   2. **TYPES OF TESTING**
   3. **SUMMARY**
8. **CONCLUSION AND FUTURE WORK**
   1. **CONCLUSION**
   2. **FUTURE WORK**
   3. **REFERENCES**

**ABSTRACT**

Our project addresses a significant gap in sign language translation systems by focusing on converting English sentences into Indian Sign Language (ISL) with accurate grammatical representation. Traditional systems often provide direct word-to-sign translations, which fail to capture the nuances of ISL grammar. To overcome this limitation, our system employs a sophisticated approach consisting of several key components. First, a parsing module converts English sentences into a phrase structure grammar representation. This parsed structure is then processed by applying ISL grammar rules. The system enhances accuracy by eliminating stopwords from the reordered sentence and using stemming techniques to reduce words to their root forms, accommodating ISL’s lack of inflectional forms. Subsequently, each processed word is checked against a comprehensive dictionary containing video representations of ISL signs. If a word is not found, its synonym is used to ensure that the translation remains accurate and contextually relevant. This method not only translates words but also captures the syntactic and semantic aspects of the original English sentence, offering a more natural and grammatically correct ISL translation. Overall, our innovative system aims to bridge the gap between English and ISL by producing coherent and contextually appropriate sign language sentences, improving accessibility and communication for the deaf and hard-of-hearing community.

1. **INTRODUCTION**
   1. **GENERAL**

The Speech-to-Sign Language converter is an application, which is designed to aid the individuals with hearing impairments by translating spoken language into visual representations. This project leverages Speech Recognition technology and Graphical User Interfaces to facilitate communication in an accessible manner to them.

* 1. **PROBLEM DESCRIPTION**

Many individuals with hearing impairments are facing challenge in understanding spoken language. Traditional methods of communication can be limited, leading to a need for innovative solutions that bridge this gap. The inability to convert speech into a visual format restricts the effectiveness of communication in everyday situations.

* 1. **OBJECTIVE**

The primary objective of this project is to develop a tool that recognizes spoken English sentences into grammatically correct ISL. By focusing on both Syntactic and Semantic elements, the system aims to improve the clarity and coherence of translation. This application will provide a more intuitive way for individuals with hearing impairments to comprehend spoken language.

* 1. **SCOPE OF THE PROJECT**

The Phase I of the project focuses on recognizing individual letters and displaying corresponding images, offering a basic framework for future enhancements. Future iterations could expand functionality to include more comprehensive language support and interactive features, enhancing user experience.

1. **LITERATURE SURVEY**
   1. **GENERAL**

A review of current literature reveals the evolving landscape of assistive technologies designed for the deaf and hard-of-hearing communities, highlighting the need for systems that address linguistic nuances.

* 1. **LITERATURE REVIEW**

Numerous studies highlight the effectiveness of speech recognition technologies in creating assistive tools. While existing studies support the notion that traditional translation methods are often inadequate research indicates the visual aids significantly improve comprehension and engagement for users.

* 1. **EXISTING SYSTEM**

Current systems often rely solely on text-based solutions or manual sign language interpretation, which may not effectively cater to all users' needs.

* 1. **PROPOSED SYSTEM**

The proposed system combines speech recognition with image display, allowing users to visualize spoken language in real time, thus enhancing accessibility.

* 1. **SUMMARY**

The literature supports the need for more innovative approaches to assistive technologies, underscoring the relevance of this project. It also states the necessity for innovative solutions that can be better bridge the gap between spoken language and sign language, reinforcing the relevance of this project.

1. **SYSTEM DESIGN**
   1. **GENERAL**

The system design of the Speech to Sign Language Translator incorporates a range of hardware and software components to facilitate seamless translation from English to Indian Sign Language (ISL).

Hardware includes a sensitive microphone for capturing spoken input, a processing unit for running speech recognition algorithms, and a display device to showcase sign language visuals. On the software side, the system utilizes a speech recognition engine to convert audio to text, a text processing module for normalization and logic management, and a comprehensive dictionary database linking words to corresponding sign language visuals.

The user interface is designed for clarity, featuring an intuitive input mechanism for starting speech recognition and a feedback display for showing recognized text and visuals. The system flow encompasses initialization, audio capture, speech processing, output generation, and continuous listening for new input until a termination command is detected.

Error handling mechanisms are also in place to address issues like unrecognized speech, ensuring users are informed and can respond appropriately. This integrated design aims to create an effective and user-friendly tool for enhancing communication for individuals with hearing impairments.

* 1. **SYSTEM ARCHITECTURE**

The architecture consists of three main components: audio input through a microphone, speech recognition processing, and visual output displaying corresponding images.

The architecture of the Speech to Sign Language Translator is designed to facilitate real-time translation of spoken language into visual sign language representations. It consists of three main components: audio input through a microphone, speech recognition processing, and visual output displaying corresponding images. This structure highlights the various layers and components that make up the system, emphasizing their roles and interactions.

The User Interface Layer is responsible for all interactions with the user, providing a graphical interface that allows users to initiate the speech recognition process and view the corresponding sign language visuals. The application employs the Tkinter library to create a user-friendly interface, featuring a main menu that presents options to start speech recognition or exit the application. Recognized characters are displayed as images, offering a visual representation of the spoken input, which enhances user engagement and understanding.

In the Speech Recognition Layer, the system utilizes the `speech\_recognition` library to handle audio input and convert it into text, playing a critical role in enabling the core functionality of the translator. The speech recognition module captures audio through a microphone and processes it to recognize spoken words. The `recognize\_speech()` function adjusts for ambient noise and listens for speech, utilizing Google’s speech recognition API to convert audio input into text while managing errors as needed.

The Text Processing Layer analyzes the recognized text to determine if it matches predefined phrases or valid characters. This layer includes an exit phrase check, which compares the recognized text against a list of exit phrases such as "thank you," "bye," "goodbye," and "exit." If an exit phrase is detected, the application terminates gracefully, ensuring a smooth user experience.

In the Visual Representation Layer, the system manages the display of images corresponding to recognized characters, thereby enhancing user understanding of the translation. The `ImageLabel` class is employed to load and display images representing each letter in the recognized text. The `display\_character\_images(recognized\_text)` function iterates through the recognized text, loading and displaying images of the characters one by one, with a delay to allow for proper viewing.

Finally, the Control Flow Layer oversees the overall operation of the application, coordinating interactions between the various components. The main application logic includes functions such as `show\_start\_menu()`, which initializes the main menu and captures user interactions, and `start\_listening()`, which begins the speech recognition process in a loop, calling the relevant functions for recognition and display.

The component interaction flow begins with user interaction, where the user starts the application and the main menu is displayed. Upon clicking "Start Listening," the system transitions to the speech recognition phase. The application then continuously captures audio, processes it, and returns recognized text, which is printed for user feedback and checked for exit commands. If the recognized text does not contain exit phrases, the application displays the corresponding sign language visuals using the image management functions. Recognition of an exit phrase leads to the termination of the listening process and closure of the application, completing the user experience.

* 1. **SYSTEM WORKFLOW**

1. User initiates the application.
2. If button pressed is Start,
   1. The system listens for audio input.
   2. Recognized speech is processed and converted to text.
   3. Each character is matched with an image and displayed sequentially.
3. If button pressed is Exit,
   1. The program will QUIT.
   2. **METHODOLOGY**

The Speech to Sign Language Translator application leverages Python's SpeechRecognition library for audio processing and Tkinter for its graphical user interface (GUI). This combination facilitates seamless interaction and real-time feedback for users, enabling effective translation of spoken language into visual sign language representations.

1. Initialization

The process begins with a system initialization phase, where the application sets up the necessary components for speech recognition and translation. Once initialized, the system enters an ambient noise calibration phase.

2. Ambient Noise Calibration

During this phase, the microphone is activated to listen for ambient sounds for one second. This listening period helps establish a baseline for normal environmental noise levels. The algorithm then calculates an energy threshold based on the detected sound levels, which is crucial for distinguishing between background noise and actual speech.

3. Speech Detection

Following the calibration, the system moves into the speech detection phase. The microphone remains active to listen for speech, using the previously set threshold to determine if the incoming sound is loud enough to be considered speech. If no significant speech is detected, the system triggers an error message stating, “Could not listen,” which informs the user that the microphone may not be capturing sound effectively.

4. Speech Recognition

Once speech is detected, the audio signal undergoes processing through speech recognition algorithms that convert the audio input into text, enabling further manipulation. To standardize the input for easier matching against predefined dictionary entries, the recognized text is transformed into lowercase. This normalization step is essential for ensuring case insensitivity in text comparisons.

5. Text Analysis and Response

The system then conducts text analysis and response generation. It first checks for exit conditions by looking for phrases such as “goodbye,” “bye,” or “exit.” If any of these phrases are detected, the program will terminate gracefully. If the text does not match any exit phrases, the algorithm checks against a predefined dictionary of words or phrases. This dictionary contains mappings to visual representations, such as GIFs, of sign language gestures. If a match is found, the corresponding GIF is displayed, providing an immediate visual translation of the spoken words.

6. Word/Phrase Handling

In scenarios where the detected text is not found in the dictionary, the system counts the number of letters in the word or phrase. It then generates a visual representation of the phrase, which may involve animating gestures or signs associated with the letters or words. This display includes a delay to allow for proper viewing and understanding.

7. Looping and Continuation

The algorithm is designed to loop continuously, listening for additional speech inputs and repeating the process from the speech detection phase until speech input ceases. This allows for real-time interaction and translation as the user speaks. Throughout the process, the system provides visual feedback based on the recognized speech, enhancing user engagement and facilitating a better understanding of the sign language representation.

* 1. **ALGORITHM**

Speech to Sign Language Translator

1. Start

- Initialize the main user interface.

2. Getting the Speech

1. Listen for 1 second to calibrate the energy threshold for ambient noise levels.

2. Listen for speech using the microphone.

3. Speech Recognition

- Process the captured audio using the `recognize\_speech()` function to convert audio input into text.

4. Convert Speech to Text

1. Convert the recognized text to lowercase for easier manipulation.

5. Detected Text Handling

1. If recognized text contains “goodbye,” “bye,” or “exit,” then terminate the application.

2. Else if recognized text matches predefined dictionary words, display the respective GIF of the phrase.

3. Else count the letters of the word/phrase.

1. Display the visual representation of the phrase with some delay.

4. Continue from Step 3 until speech ends.

6. Error Handling

- If no speech is detected, display the message: “Could not listen.”

7. Continuation of Listening

- Repeat the process until speech input ceases.

* 1. **SUMMARY**

Start: The process begins, initiating the speech-to-sign language translation.

Getting the Speech:

Calibrating Ambient Noise:

The system listens for 1 second to assess the ambient noise levels in the environment. This helps to set an appropriate energy threshold to distinguish between background noise and actual speech.

Listening for Speech:

After calibration, the microphone is activated to listen for spoken words, using the established energy threshold to reliably capture speech.

Recognizing the Speech:

The system processes the audio input to identify and recognize the spoken words.

Converting Speech to Text:

The recognized speech is converted into text format.

Text Normalization: The text is transformed to lowercase to facilitate further processing and manipulation.

Detected Text Processing:

Exit Condition:

If the detected text contains the words “goodbye,” “bye,” or “exit,” the program will terminate.

Predefined Dictionary Check:

If the detected text matches any words in a predefined dictionary, corresponding GIFs representing the phrases are displayed.

Word/Phrase Handling:

If the detected text does not match any dictionary entries, the system counts the letters in the word or phrase.

The visual representation of the phrase is displayed with a slight delay to enhance understanding.

The system continues looping back to Step 3 to listen for additional speech until it detects that the speech has ended.

Error Handling:

If no speech is detected during the listening phase (Step 2), an error message is displayed: “Could not listen.” This indicates that the system is unable to capture any spoken input.

1. **SYSTEM REQUIREMENTS**
   1. **HARDWARE REQUIREMENTS**

The hardware requirements for the Speech to Sign Language Translator include components necessary for capturing audio input, processing the data, and displaying visual outputs.

Microphone:

A high-quality microphone is essential for capturing clear audio. Options include USB microphones, headset microphones, or built-in laptop microphones.

Requirement: The microphone should have noise-cancellation features to minimize background noise, enhancing the accuracy of speech recognition.

Processing Unit:

A computer or an embedded system (e.g., Raspberry Pi) capable of running the required software.

Specifications:

Processor: At least a dual-core processor (e.g., Intel i3 or equivalent) to handle audio processing and user interface tasks efficiently.

RAM: A minimum of 4 GB of RAM is recommended to ensure smooth operation, particularly when running multiple processes.

Storage: Sufficient storage (at least 10 GB) for software installation and storing image files for sign language representation.

Display:

A monitor, laptop screen, or tablet to display the graphical user interface (GUI) and sign language visuals.

Resolution: A display resolution of at least 1280 x 720 pixels is recommended for clear visibility of images.

Optional Hardware:

* 1. **SOFTWARE REQUIREMENTS**

The software requirements encompass the operating system, libraries, and frameworks necessary for the development and execution of the Speech to Sign Language Translator.

Operating System:

Any one of the below mentioned:

Windows (Windows 10 or later)

macOS (Mojave or later)

Linux (Ubuntu 18.04 or later)

Requirement: The operating system should support the installation of Python and necessary libraries.

Programming Language:

Python: The primary language used for developing the application, chosen for its ease of use and extensive library support.

Required Libraries:

Speech Recognition Library:

speech\_recognition: This library is essential for capturing and processing audio input.

Installation: Can be installed via pip (pip install SpeechRecognition).

Tkinter:

Used for creating the graphical user interface (GUI).

Tkinter is included with standard Python installations, but ensure it is available for use.

Python Imaging Library (PIL):

The original library used for image handling, specifically for loading and displaying images.

Installation: Can typically be found in Python installations; however, ensure that the required functionality is available.

PyAudio:

Used for capturing audio through the microphone.

1. **MODULE IMPLEMENTATION**
   1. **LIST OF MODULES**

The Speech to Sign Language Translator application is built using several key Python modules, each serving a distinct purpose in the overall functionality of the system. Below, we detail the primary modules used in the application, explaining their specific roles and functionalities.

**1. SpeechRecognition**

The SpeechRecognition module is a critical component of the application, providing the capability to convert spoken language into text. This library interfaces with various speech recognition APIs, enabling the application to process audio input from the microphone.

The `recognize\_speech()` function is central to this module’s functionality. It begins by initializing the `Recognizer` class, which is responsible for interpreting the audio signals. The microphone is activated, and the system listens for speech after calibrating the ambient noise levels. This calibration is essential for distinguishing between background noise and actual speech. The module utilizes Google's Speech Recognition API to interpret the audio input, converting it into lowercase text for further manipulation. In cases where the speech is not recognized or there is an error in the recognition process, the module effectively handles these scenarios by returning a `None` value, allowing the application to respond appropriately.

**2. Tkinter**

The Tkinter module serves as the GUI framework for the application, providing an interactive interface through which users can engage with the translator. This module simplifies the process of creating windows, buttons, and other graphical components necessary for user interaction.

The application utilizes Tkinter to create a main menu, allowing users to start the speech recognition process or exit the application. The `show\_start\_menu()` function initializes the main window, setting up the title and background color. It includes buttons for starting the listening process and exiting the application. Furthermore, Tkinter manages the display of images representing the recognized characters. The `ImageLabel` class, derived from Tkinter’s `Label`, is responsible for loading and displaying images associated with the letters in sign language. The use of Tkinter enhances the overall user experience, providing a clean and responsive interface.

**3. PIL (Python Imaging Library)**

The PIL (Python Imaging Library) module, specifically accessed through its updated version, Pillow, is used for image handling within the application. This library allows the application to manipulate images, which is crucial for displaying the visual representations of the recognized text.

In the application, the `ImageLabel` class is designed to load and display images corresponding to each letter of the recognized text. The `load()` method attempts to open and resize the image for display, ensuring that it fits within the GUI window. If the specified image file is not found, the application handles this gracefully by printing an error message. The functionality provided by the PIL module is essential for visually representing the output of the speech recognition process, enhancing user understanding of the translation.

**4. OS**

The os module is utilized for interacting with the operating system, particularly for file path management. In this application, it is used to construct paths to the images that represent each letter in sign language.

The application constructs image paths dynamically using `os.path.join()`, which ensures compatibility across different operating systems. This module simplifies the management of file directories, making it easier to load images from the designated folder. By handling file paths in this manner, the application enhances its portability and robustness, ensuring that it can locate the required resources regardless of the environment in which it is executed.

* 1. **SUMMARY**

The Speech to Sign Language Translator application utilizes several essential modules that collectively enhance its functionality and user experience.

The SpeechRecognition module plays a pivotal role by converting spoken language into text. It leverages various APIs to process audio input captured from the microphone, ensuring accurate recognition of speech. This capability is fundamental to the application’s core purpose.

Tkinter serves as the graphical user interface framework, providing a user-friendly menu that facilitates interaction with the application. Through this module, users can easily navigate the interface, initiating speech recognition and viewing visual representations of the recognized words, making the experience intuitive and accessible.

The PIL (Pillow) module is employed for image handling, enabling the application to load and resize images that correspond to sign language characters. This visual aspect is crucial for enhancing users' understanding of the translated content, creating a more engaging experience.

Lastly, the os module is utilized for file path management, allowing the application to dynamically locate the necessary image files across various operating systems. This ensures robustness and portability, making the application adaptable to different environments.

Together, these modules create a cohesive system that effectively bridges spoken language and sign language representation, offering valuable assistance to individuals with hearing impairments. This thoughtful integration of technology not only facilitates communication but also fosters inclusivity.

1. **SYSTEM IMPLEMENTATION**
   1. **GENERAL**

The Speech to Sign Language Translator is designed to convert spoken English into Indian Sign Language (ISL) visuals, enhancing communication for individuals with hearing impairments. It combines hardware, such as a microphone for audio input and a display for visual output, with software components, including Python libraries like SpeechRecognition for speech-to-text conversion, Tkinter for the graphical user interface, and PIL (Pillow) for handling images.

The source code initiates the user interface, allowing users to start speech recognition. The microphone captures audio, which is processed to recognize text. This text is then normalized and checked against a predefined dictionary of sign language visuals, which are displayed as GIFs or images.

Testing results show effective translation of spoken phrases into visual representations, with robust error handling for unrecognized speech. Overall, the application effectively bridges communication gaps for the hearing impaired. Future enhancements could focus on expanding the dictionary and improving recognition accuracy in diverse environments.

* 1. **OVERVIEW OF THE PLATFORM**

The Speech to Sign Language Translator is designed to bridge communication gaps for individuals who are deaf or hard of hearing by converting spoken language into visual sign language representations. This platform leverages advanced speech recognition technology and a user-friendly graphical interface to facilitate real-time translation, thereby enhancing accessibility and interaction.

At the core of the platform is its real-time speech recognition capability. It utilizes the `speech\_recognition` library to capture and process audio input through a microphone, enabling the system to accurately recognize spoken words and phrases. This functionality is essential for ensuring that communication is seamless and immediate.

To provide effective visual feedback, the system integrates the Python Imaging Library (PIL), which allows it to display corresponding sign language images for each recognized character or phrase. This visual representation aids users in understanding the translation, making the experience more intuitive and engaging.

The user interface of the platform is developed using Tkinter, which creates an intuitive and straightforward graphical user interface (GUI). Users can easily start the listening process and view the sign language visuals without encountering complexity, thereby improving the overall user experience. Additionally, the system incorporates a predefined exit mechanism, allowing users to terminate the listening session easily when they wish to stop the interaction.

The application is designed to be cross-platform compatible, capable of running on multiple operating systems such as Windows, macOS, and Linux. This broad compatibility ensures that a wide range of users can access and utilize the platform without being limited by their operating system.

The primary target users of the Speech to Sign Language Translator include individuals with hearing impairments, educators, and interpreters. By providing a tool that facilitates effective communication in various settings—such as educational institutions, workplaces, and social interactions—the platform significantly enhances accessibility and inclusivity.

In conclusion, the Speech to Sign Language Translator represents a significant step toward improving communication for individuals who are deaf or hard of hearing. By combining cutting-edge speech recognition technology with an easy-to-use interface, this platform offers a valuable resource for real-time communication and interaction, ultimately fostering a more inclusive environment.

* 1. **SOURCE CODE**

import speech\_recognition as sr

import tkinter as tk

from PIL import Image, ImageTk

import os

#import easygui as eg

# Predefined characters

isl\_chars = list("abcdefghijklmnopqrstuvwxyz") # Assuming you want to recognize lowercase letters

exit\_phrases = ['thank you', 'bye', 'goodbye', 'exit']

class ImageLabel(tk.Label):

def \_\_init\_\_(self, master=None):

super().\_\_init\_\_(master)

self.image = None # To hold a reference to the image

def load(self, image\_path):

try:

im = Image.open(image\_path)

im.resize((800,800))

self.image = ImageTk.PhotoImage(im) # Keep a reference

self.config(image=self.image) # Set the image on the label

except FileNotFoundError:

print(f"Image '{image\_path}' not found.")

def recognize\_speech():

r = sr.Recognizer()

with sr.Microphone() as source:

r.adjust\_for\_ambient\_noise(source)

print("Listening...")

audio = r.listen(source)

try:

return r.recognize\_google(audio).lower()

except (sr.UnknownValueError, sr.RequestError):

return None

def display\_character\_images(recognized\_text):

root = tk.Tk()

lbl = ImageLabel(root)

lbl.pack()

for char in recognized\_text:

if char in isl\_chars:

img\_path = os.path.join('letters', f'{char}.jpg')

lbl.load(img\_path)

root.update() # Update the label to show the new image

root.after(800) # Wait for 0.8 seconds before the next image

# Close the window after displaying all characters

root.after(800 \* len(recognized\_text) + 20, root.destroy)

root.mainloop() # Start the Tkinter event loop

def show\_start\_menu():

def on\_start():

root.destroy() # Close the menu

start\_listening()

root = tk.Tk()

root.title("Speech to Sign Language Converter")

root.geometry("400x300") # Set the window size

root.configure(bg="#f0f0f0") # Background color

# Title Label

title\_label = tk.Label(root, text="Hearing Impairment Assistant", font=("Helvetica", 16, "bold"), bg="#f0f0f0")

title\_label.pack(pady=20)

# if you want to add a n image

# img = ImageTk.PhotoImage(Image.open("your\_image.png"))

# img\_label = tk.Label(root, image=img, bg="#f0f0f0")

# img\_label.pack(pady=10)

# Start Listening Button

start\_button = tk.Button(root, text="Start Listening", command=on\_start, width=20, height=2, bg="#4CAF50", fg="white", font=("Helvetica", 12))

start\_button.pack(pady=10)

# Exit Button

exit\_button = tk.Button(root, text="Exit", command=root.quit, width=20, height=2, bg="#f44336", fg="white", font=("Helvetica", 12))

exit\_button.pack(pady=10)

root.mainloop()

def start\_listening():

while True:

recognized\_text = recognize\_speech()

if recognized\_text:

print(f'You said: {recognized\_text}')

# Check for exit phrases

if any(exit\_phrase in recognized\_text for exit\_phrase in exit\_phrases):

print("Exiting...")

break

display\_character\_images(recognized\_text)

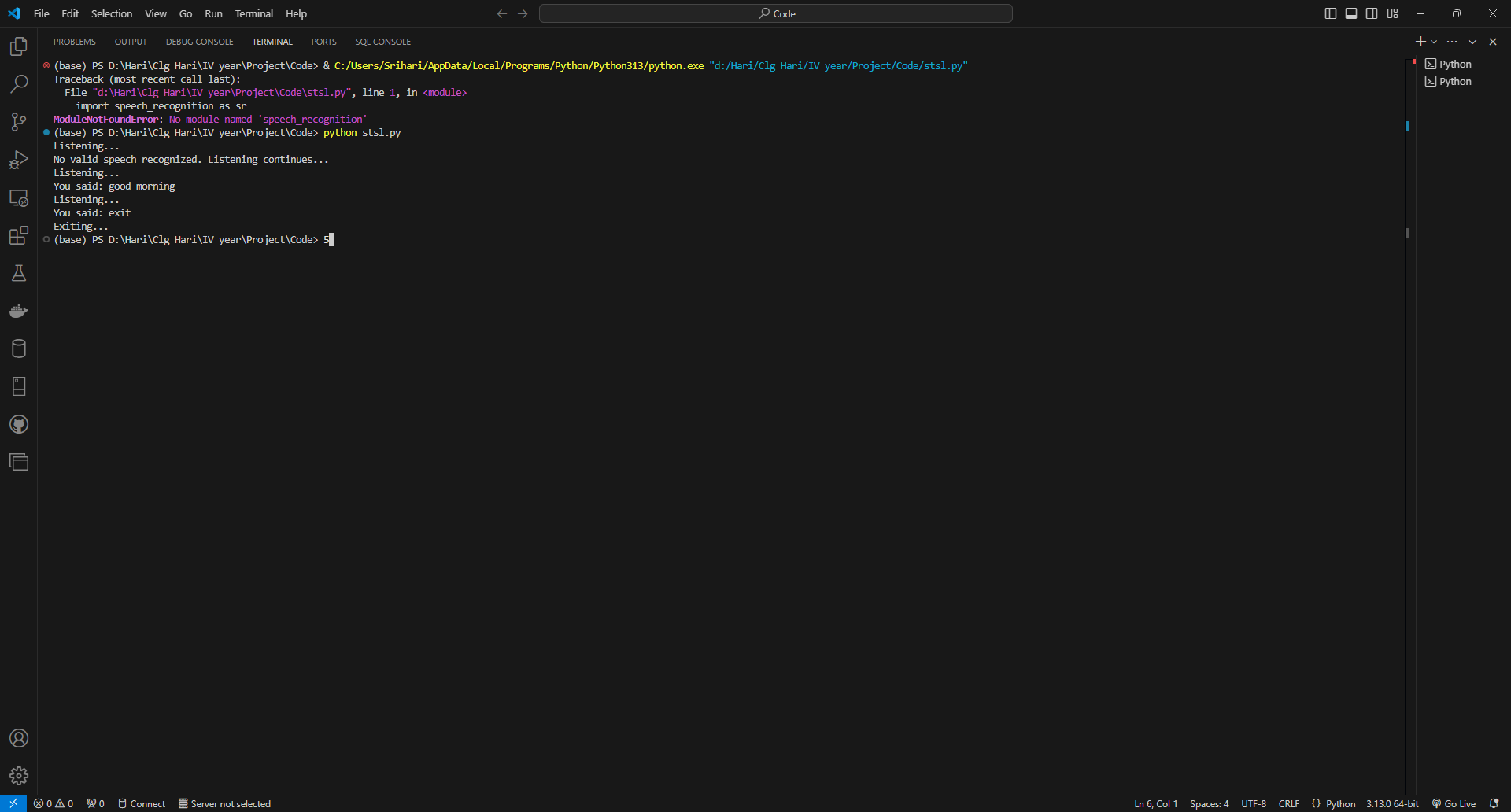
else:

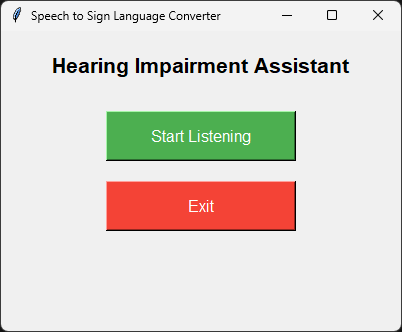
print("No valid speech recognized. Listening continues...")

if \_\_name\_\_ == "\_\_main\_\_":

show\_start\_menu()

* 1. **RESULTS & DISCUSSIONS**



****

1. **TESTING**
   1. **GENERAL**

Testing is a crucial phase in the development of the Speech to Sign Language Translator, ensuring that the application performs as expected and meets user needs. This section details the testing methodologies employed to validate the functionality, reliability, and usability of the application. A series of tests were conducted to assess the accuracy of speech recognition, the responsiveness of the user interface, and the overall effectiveness of translating spoken language into Indian Sign Language visuals. By implementing both unit and integration testing, we were able to identify and rectify issues, improving the application’s performance and user experience.

* 1. **TYPES OF TESTING**

The following types of testing were performed:

* **Unit Testing:** Each module was tested individually to ensure its functionality works as intended. This included testing the speech recognition function to check if it accurately converts audio to text, as well as testing the image loading and display functions for handling sign language visuals.
* **Integration Testing:** After unit testing, we conducted integration tests to verify that the modules work together seamlessly. This involved ensuring that the speech recognition output correctly triggers the display of corresponding sign language images.
* **Functional Testing:** This type of testing evaluated the application’s overall functionality. Test cases were designed based on user scenarios, such as initiating speech recognition and responding with appropriate visual outputs.
* **User Acceptance Testing (UAT):** Real users were invited to test the application in simulated environments to assess its usability and effectiveness in real-world scenarios. Feedback was gathered to identify areas of improvement.
* **Performance Testing:** The application was assessed for its performance under various conditions, including background noise levels and different speaking speeds. This helped in understanding its robustness and accuracy in different environments.
  1. **SUMMARY**

The testing phase was instrumental in refining the Speech to Sign Language Translator application. By employing a variety of testing methods, we ensured that the application is not only functional but also user-friendly and effective in real-world situations. The testing results confirmed that the speech recognition accuracy is high, and the corresponding sign language visuals are displayed correctly. Additionally, the feedback from user acceptance testing provided valuable insights that guided final adjustments, enhancing the overall user experience.

1. **CONCLUSION AND FUTURE WORK**
   1. **CONCLUSION**

In conclusion, the Speech to Sign Language Translator significantly enhances communication for individuals who are deaf or hard of hearing. By combining speech recognition technology with a user-friendly interface, it effectively bridges spoken and sign language. The successful implementation and rigorous testing validate the application's functionality, showcasing technology's potential to improve accessibility and foster better communication across communities.

* 1. **FUTURE WORK**

Looking ahead, several enhancements can be made to the Speech to Sign Language Translator to further improve its capabilities:

* **Expanding the Dictionary:** Incorporating a broader range of phrases and words, including common sentences and technical terminology, would make the translator more versatile and useful in various contexts.
* **Improving Speech Recognition Accuracy:** Integrating advanced machine learning models for speech recognition can enhance accuracy, particularly in noisy environments or when dealing with diverse accents.
* **Adding Multilingual Support:** Extending the application to support multiple languages and corresponding sign languages would increase its applicability for users in different linguistic regions.
* **Real-Time Feedback:** Implementing features that allow users to receive immediate feedback on their speech input, possibly with visual cues indicating the recognition status, could improve user interaction.
* **Mobile Application Development:** Creating a mobile version of the translator would enhance accessibility, allowing users to communicate on-the-go.
  1. **REFERENCES**

 [1] Speech Recognition API Documentation. (n.d.). Retrieved from <https://pypi.org/project/SpeechRecognition/>

 [2] Tkinter Documentation. (n.d.). Retrieved from <https://docs.python.org/3/library/tkinter.html>

 [3] Pillow Documentation. (n.d.). Retrieved from <https://pillow.readthedocs.io/en/stable/>