Prepared by group 175

Group Project

AI LANGUAGE TRANSLATOR

Supervisor and Reviewer

Dr. Shiv Shankar Prasad Shukla Sir

Dr. Shweta Saxena Ma’am

Dr. M K Jayanthi Kannan Ma’am

Team Members

Luvkush Singh	(23BAI11229)
Atharva Singh	(23BAI11321)
Rohan Ramdhan Decharwal	(23BAI10930)
Jyotiraditya Chundawat	(23BAI11359)
P. Mohith	(23BAI11382)

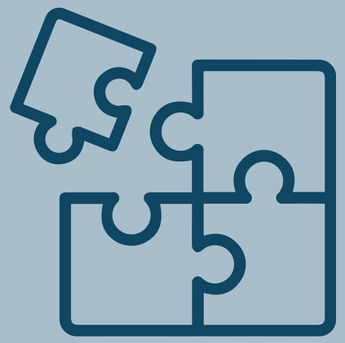
Introduction

- Translation is the process of conveying a written source language text clearly, completely, accurately into a target language. The translation process involves translation, editing and proofreading.
- Translation is an activity, a product, and a process. As an activity, translation is a complex act that requires close reading of a text in the source language, understanding its meaning and creating an equivalent text in the target language.

Project Objectives

Accurate Language Translation

- To provide high-quality and accurate translations between multiple languages, minimizing errors and maintaining the original meaning, tone, and context.



Real-Time Processing

- To enable real-time or near-real-time translation for applications such as live conversations, video subtitles, or online communication platforms.

Wide Language Support

- To support a diverse range of languages, including low-resource and regional languages, enabling inclusivity for global users.

Novelty of the Project

The novelty of the AI Language Translator project lies in its ability to address several limitations of existing translation systems while introducing innovative features:

Context-Aware Translation: Unlike traditional translators, this system incorporates advanced AI models (e.g., Transformers like GPT or T5) that understand context, idiomatic expressions, and cultural nuances to provide more accurate translations.

Real-Time, Multi-Modal Functionality: Offers real-time translation for both text and speech, enabling seamless communication during live conversations. Integrates speech-to-text (STT) and text-to-speech (TTS) technologies for a fully interactive experience.

Novelty of the Project

The novelty of the AI Language Translator project lies in its ability to address several limitations of existing translation systems while introducing innovative features:

Domain-Specific Customization: Can be trained to handle specialized vocabularies, such as technical terms in medicine, law, or business, which most general-purpose translators struggle with.

Offline Capability: Optimized for edge devices to perform translations without requiring internet connectivity, making it ideal for remote or low-resource environments.

Seamless Integration: Combines multiple AI tools into a single pipeline, ensuring high efficiency and low latency for a smooth user experience.

Hardware & Software Requirements

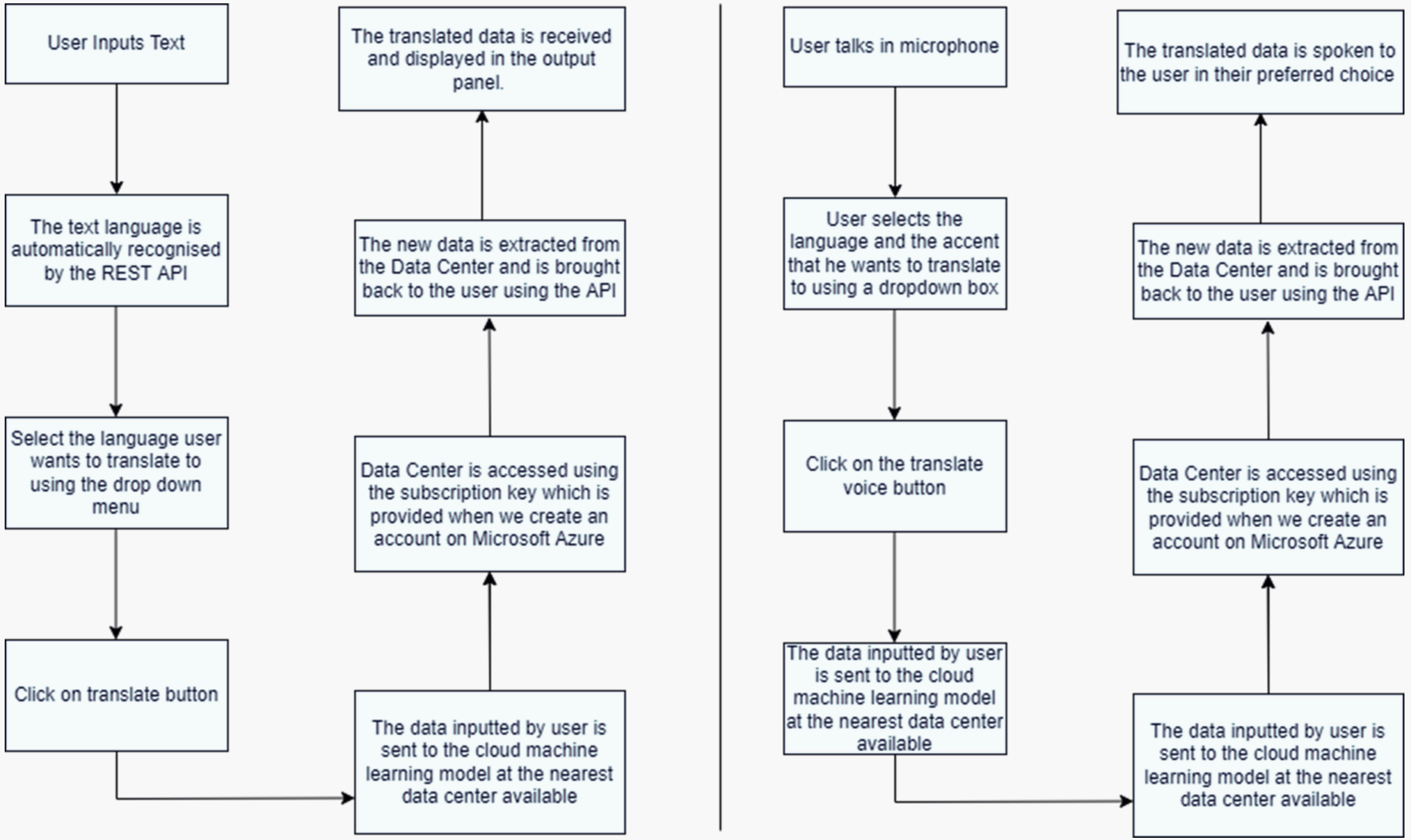
Hardware:

- Microphone and speaker for audio input/output.
- Device (smartphone, tablet, or dedicated hardware).
- GPU or TPU for model training (e.g., NVIDIA Tesla V100).

Software:

- Python programming language.
- ensorFlow or PyTorch for building and training models
- NLP libraries like Hugging Face Transformers.
- API for TTS/STT (e.g., Google Speech-to-Text API or Azure Cognitive Services).

Overall System Architecture and Diagram



Module Description

The Module Description section outlines the key components of your AI Language Translator and provides details about their functionality.

Speech-to-Text (STT) Module:

Purpose: Converts spoken language into text.

Functionality:

- Captures user speech using a microphone.
- Processes audio input using an Automatic Speech Recognition (ASR) model.
- Outputs the spoken words as text.

Technology Used:

- Pre-trained ASR models (e.g., OpenAI Whisper, DeepSpeech, Google Speech-to-Text API).
- Audio processing libraries (e.g., PyDub, Librosa).
- Example: The user says, "Hello, how are you?" → Text output: "Hello, how are you?"

Module Description

Language Detection Module

Purpose: Identifies the input language.

Functionality:

- Analyzes the text or speech input to detect the source language.
- Ensures the system knows what language to translate from.

Technology Used:

- Language detection libraries (e.g., LangDetect, FastText).

Example: Input text: "Bonjour, comment ça va?" → Detected language: French.

Module Description

Translation Module

Purpose: Translates text from the source language to the target language.

Functionality:

- Processes the detected text using a Neural Machine Translation (NMT) model.
- Ensures the translation accounts for context, grammar, and idiomatic expressions.

Technology Used:

- Transformer-based models like GPT, BERT, or T5.
- Libraries such as Hugging Face Transformers or Fairseq.

Example: Input text: "Hello, how are you?" → Output: "Hola, ¿cómo estás?" (in Spanish).

Module Description

Text-to-Speech (TTS) Module

Purpose: Converts translated text into spoken output in the target language.

Functionality:

- Synthesizes human-like speech for the translated text.
- Allows users to hear translations instead of reading them.

Technology Used:

- TTS frameworks (e.g., Google Text-to-Speech API, Tacotron 2, or Amazon Polly).

Example: Input text: "Hola, ¿cómo estás?" → Output: Spoken audio of the text in Spanish.

Problem Statement

Language translation is a complex task due to the vast differences in grammar, syntax, semantics, cultural contexts, and idiomatic expressions across languages. Traditional rule-based or statistical translation methods struggle to handle this complexity and often produce translations that are inaccurate, lack fluency, or miss nuances.

Challenges include:

1.Ambiguity in Languages: Words and phrases can have multiple meanings depending on the context.

2.Complex Grammar Structures: Different languages follow different grammar rules and structures.

3.Idiomatic Expressions: These often do not translate directly between languages.

4.Cultural Contexts: The same words may have different connotations in different cultures.

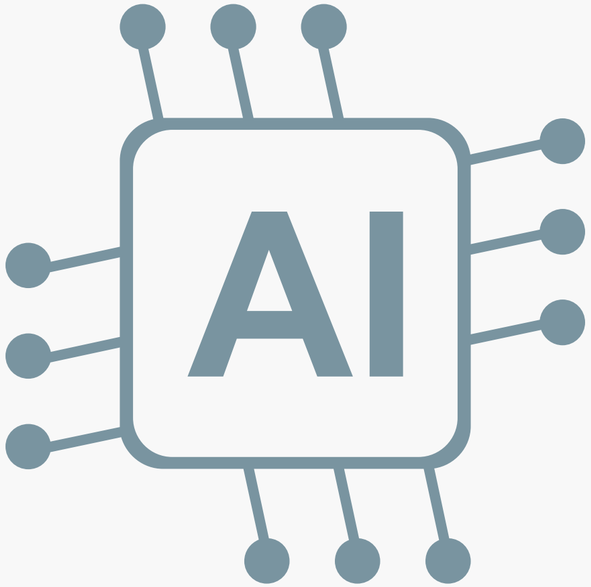
Scalability: With thousands of languages and dialects globally, manually creating translation rules for each is impractical.

The "Limitless" Translator

- We have successfully build a translator with a working GUI and we can translate text and provide the output as speech in the language of your choice.

- It can also function by converting speech into text and translating it into the language of your choice.

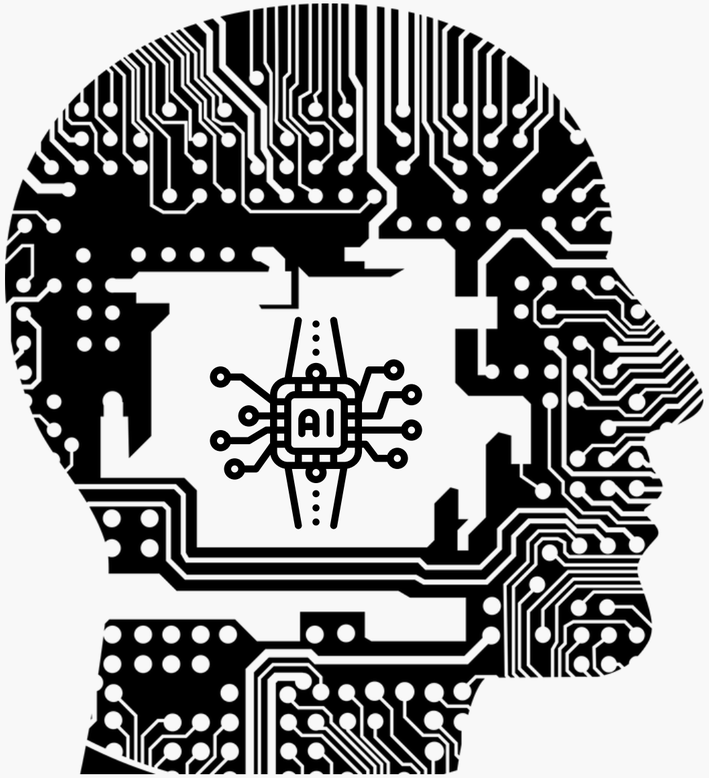
- However with the positives, it still has some limitations. The application sometimes find it hard to understand the accents of some places and therefore translation becomes incorrect. Also the loading time is a bit slow sometimes.



The "Intelligent" Translator

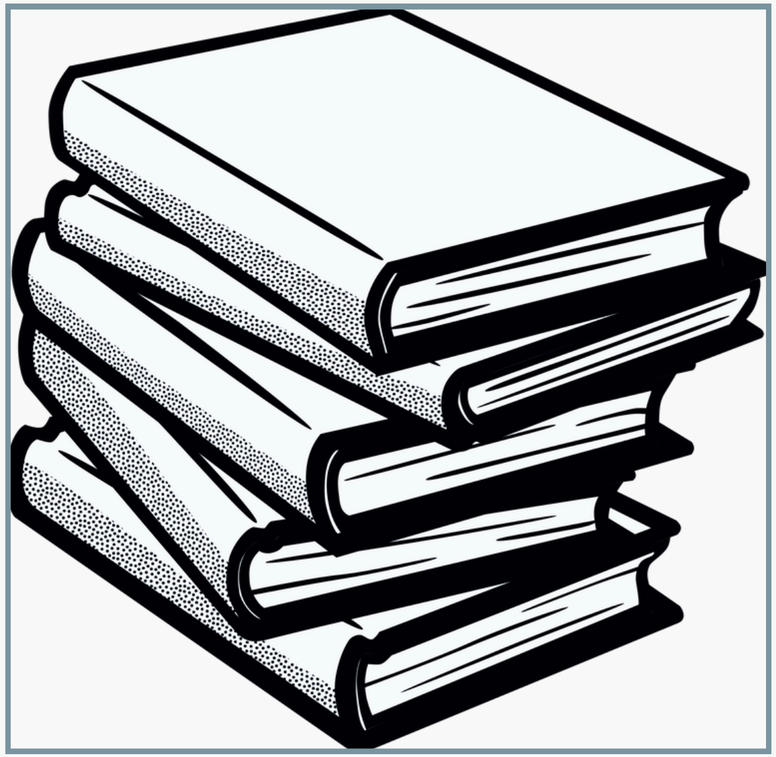
- We have proposed a translator which is easily accessible and is more accurate when working with different dialects.

- It will work on API namely Microsoft Azure Cognitive Services where the translator will have access to different machine learning models already available in the cloud. The translator will access those models and will translate the text into the desired language through API requests. The result will be sent back to the program and will be displayed as the translated text.



The "Novel" Translator

- Our translator is a novel project. Our translator has a ton on features in one single app like text to speech and speech to text translation. We are also looking to integrate more features as we go. Our translator is compact and easy to distribute to the people for them to use.



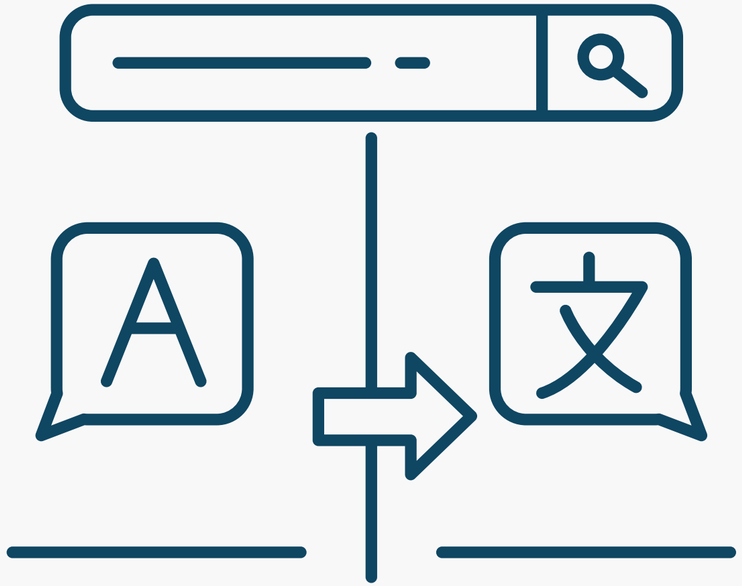
The "Useful" Translator

- If businesses want to access the global audience, they must translate all their technical documents into the languages of their target areas. This is when the technical translation services are required.

- In the travel and tourism sector, language translation is required for a variety of things. For example, the translation of travel documents, brochures, and terms and conditions documents is crucial for interacting with people of different regions.

The "Smart" Translator

- If you’re in the healthcare industry, the need for language translation is even higher because of the involvement of human beings. Any minor error can be a threat to their lives. Therefore, hiring a professional and experienced translator is exceptionally crucial for effective communication between the doctor, nurses, and patients.



HOW AI WORKS IN TRANLATOR ?

AI plays a critical role in voice recognition systems, allowing machines to convert spoken language into text (speech-to-text) and vice versa (text-to-speech) with high accuracy. These systems, also known as automatic speech recognition (ASR) and speech synthesis, are essential in applications like virtual assistants, voice-enabled devices, and translation systems. Here's a detailed breakdown of how AI works in voice recognition and delivery:

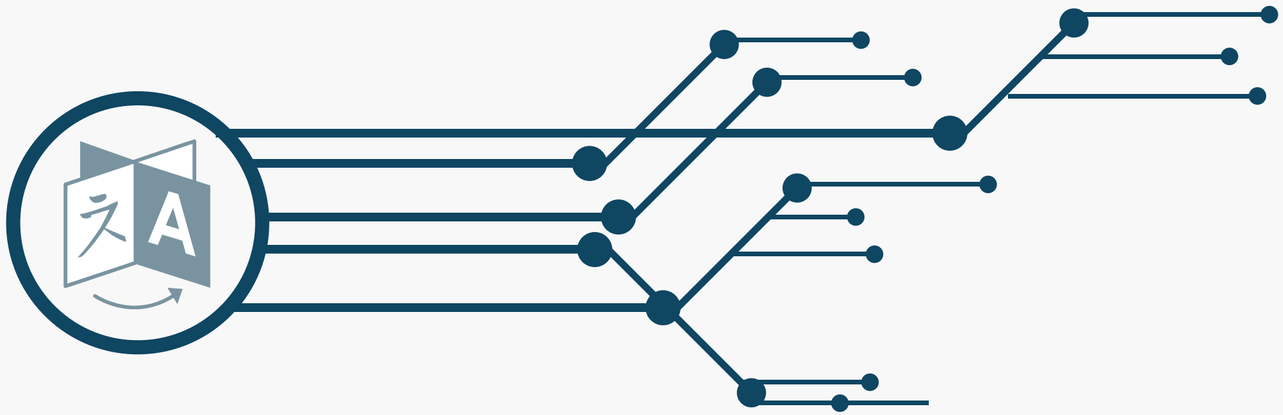
- Voice Recognition (Speech-to-Text) uses deep learning, particularly models like RNNs, LSTMs, and Transformers, to break down speech into sound features, map these to linguistic units, and convert them into text.

- Voice Delivery (Text-to-Speech) utilizes neural networks, such as Tacotron 2 and WaveNet, to analyze the text, predict prosody, and generate natural-sounding speech, including custom voices.

- AI in voice recognition and delivery has led to significant advancements in applications like virtual assistants, real-time language translation, and voice-activated interfaces, enabling seamless human-computer interaction.

BENEFITS OF AI IN TRANSLATOR

- Context Awareness: AI models can understand the meaning of sentences and choose the best translation based on context.
- Speed and Scalability: AI systems can translate languages almost instantly, handling multiple languages and dialects efficiently.
- Personalization: AI can be fine-tuned to specific domains (e.g., medical or legal language) for highly accurate translations.
- Accuracy: AI can produce translations that are much closer to human fluency, especially in popular language pairs.



Code

```
from tk import *
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
import requests, uuid, json
import azure.cognitiveservices.speech as speechsdk

speech_config = speechsdk.SpeechConfig(subscription="b62e765434e44689b2c1f9ac3958b6f9", region="eastasia")
audio_config = speechsdk.audio.AudioOutputConfig(use_default_speaker=True)


speech_config.speech_synthesis_voice_name='hi-IN-MadhurNeural'

speech_synthesizer = speechsdk.SpeechSynthesizer(speech_config=speech_config, audio_config=audio_config)

root = tk.Tk ()
root.title("Microsoft Language Translator")
root.geometry('590x370')

frame1 = Frame(root, width=590, height=370, relief = RIDGE, borderwidth = 5, bg='#F7DC6F')
frame1.place(x=0,y=0)

Label(root, text="Language Translator", font=("Helvetica 20 bold"), fg="black", bg='#F7DC6F').pack(pady=10)
```

Code

```
def translate():
  lang_1 = text_entry1.get("1.0", "end")
  text_entry2.delete(1.0 , "end")
  cl=choose_language.get()
  try:
    for (key,value) in LANGUAGES.items() :
      if (value==cl):
        lang_key = key

    key = "17930e6220fc4603a5edf1df60745686"
    endpoint = "https://api.cognitive.microsofttranslator.com"

    location = "eastasia"

    path = '/translate'
    constructed_url = endpoint + path

    params = {
        'api-version': '3.0',
        'to': [lang_key],
    }

    headers = {
        'Ocp-Apim-Subscription-Key': key,
```

Code

```
    'Ocp-Apim-Subscription-Region': location,
    'Content-type': 'application/json',
    'X-ClientTraceId': str(uuid.uuid4())
}

body = [{
    'text': lang_1
}]

request = requests.post(constructed_url, params=params, headers=headers, json=body)
response = request.json()
text_entry2.insert('end', response[0]["translations"][0]['text'])
speech_synthesis_result = speech_synthesizer.speak_text_async(response[0]["translations"][0]['text']).get()

if speech_synthesis_result.reason == speechsdk.ResultReason.SynthesizingAudioCompleted:
    print("Speech synthesized for text [{}]".format(response[0]["translations"][0]['text']))
elif speech_synthesis_result.reason == speechsdk.ResultReason.Canceled:
    cancellation_details = speech_synthesis_result.cancellation_details
    print("Speech synthesis canceled: {}".format(cancellation_details.reason))
    if cancellation_details.reason == speechsdk.CancellationReason.Error:
        if cancellation_details.error_details:
            print("Error details: {}".format(cancellation_details.error_details))
            print("Did you set the speech resource key and region values?")
```

Code

```
  except Exception as e :
    messagebox.showerror(e)


def clear():
 text_entry1.delete(1.0,'end')
 text_entry2.delete(1.0, 'end')

a = tk.StringVar()


auto_select = ttk.Combobox(frame1, width=27, textvariable=a, state='randomly', font =('verdana', 10, 'bold'))
auto_select['values'] = (
 'Auto'
)

auto_select.place(x=15, y=60)
auto_select.current(0)

l = tk.StringVar()
choose_language = ttk.Combobox(frame1, width=27, textvariable=l, state='randomly', font=('verdana',10, 'bold'))

LANGUAGES = {
    'af': 'afrikaans',
```

Code

```
'sq': 'albanian',
'am': 'amharic',
'ar': 'arabic',
'hy': 'armenian',
'az': 'azerbaijani',
'eu': 'basque',
'be': 'belarusian',
'bn': 'bengali',
'bs': 'bosnian',
'bg': 'bulgarian',
'ca': 'catalan',
'ceb': 'cebuano',
'ny': 'chichewa',
'zh-cn': 'chinese (simplified)',
'zh-tw': 'chinese (traditional)',
'co': 'corsican',
'hr': 'croatian',
'cs': 'czech',
'da': 'danish',
'nl': 'dutch',
'en': 'english',
'eo': 'esperanto',
'et': 'estonian',
'tl': 'filipino',
```

Code

```
'fi': 'finnish',
'fr': 'french',
'fy': 'frisian',
'gl': 'galician',
'ka': 'georgian',
'de': 'german',
'el': 'greek',
'gu': 'gujarati',
'ht': 'haitian creole',
'ha': 'hausa',
'haw': 'hawaiian',
'iw': 'hebrew',
'he': 'hebrew',
'hi': 'hindi',
'hmn': 'hmong',
'hu': 'hungarian',
'is': 'icelandic',
'ig': 'igbo',
'id': 'indonesian',
'ga': 'irish',
'it': 'italian',
'ja': 'japanese',
'jw': 'javanese',
'kn': 'kannada',
```

Code

```
'kk': 'kazakh',
'km': 'khmer',
'ko': 'korean',
'ku': 'kurdish (kurmanji)',
'ky': 'kyrgyz',
'lo': 'lao',
'la': 'latin',
'lv': 'latvian',
'lt': 'lithuanian',
'lb': 'luxembourgish',
'mk': 'macedonian',
'mg': 'malagasy',
'ms': 'malay',
'ml': 'malayalam',
'mt': 'maltese',
'mi': 'maori',
'mr': 'marathi',
'mn': 'mongolian',
'my': 'myanmar (burmese)',
'ne': 'nepali',
'no': 'norwegian',
'or': 'odia',
'ps': 'pashto',
'fa': 'persian',
```

Code

'pl': 'polish',
'pt': 'portuguese',
'pa': 'punjabi',
'ro': 'romanian',
'ru': 'russian',
'sm': 'samoan',
'gd': 'scots gaelic',
'sr': 'serbian',
'st': 'sesotho',
'sn': 'shona',
'sd': 'sindhi',
'si': 'sinhala',
'sk': 'slovak',
'sl': 'slovenian',
'so': 'somali',
'es': 'spanish',
'su': 'sundanese',
'sw': 'swahili',
'sv': 'swedish',
'tg': 'tajik',
'ta': 'tamil',
'te': 'telugu',
'th': 'thai',
'tr': 'turkish',

Code

```
    'uk': 'ukrainian',
    'ur': 'urdu',
    'ug': 'uyghur',
    'uz': 'uzbek',
    'vi': 'vietnamese',
    'cy': 'welsh',
    'xh': 'xhosa',
    'yi': 'yiddish',
    'yo': 'yoruba',
    'zu': 'zulu',
}
def speak():
    speech_config = speechsdk.SpeechConfig(subscription="b62e765434e44689b2c1f9ac3958b6f9", region="eastasia")
    speech_config.speech_recognition_language="en-IN"

    audio_config = speechsdk.audio.AudioConfig(use_default_microphone=True)
    speech_recognizer = speechsdk.SpeechRecognizer(speech_config=speech_config, audio_config=audio_config)

    print("Speak into your microphone.")
    speech_recognition_result = speech_recognizer.recognize_once_async().get()

    if speech_recognition_result.reason == speechsdk.ResultReason.RecognizedSpeech:
        text_entry1.insert("end" , speech_recognition_result.text)
    elif speech_recognition_result.reason == speechsdk.ResultReason.NoMatch:
```

Code

```
        print("No speech could be recognized: {}".format(speech_recognition_result.no_match_details))
    elif speech_recognition_result.reason == speechsdk.ResultReason.Canceled:
        cancellation_details = speech_recognition_result.cancellation_details
        print("Speech Recognition canceled: {}".format(cancellation_details.reason))
        if cancellation_details.reason == speechsdk.CancellationReason.Error:
            print("Error details: {}".format(cancellation_details.error_details))
            print("Did you set the speech resource key and region values?")


choose_language['values'] = list(LANGUAGES.values())
choose_language.place(x=305,y=60)
choose_language.current(0)


text_entry1 = Text (frame1, width=20,height=7,borderwidth=5,relief=RIDGE, font=('verdana', 15))
text_entry1.place(x=10,y=100)

text_entry2 = Text (frame1, width=20,height=7,borderwidth=5,relief=RIDGE, font=('verdana', 15))
text_entry2.place(x=300,y=100)

btn1 = Button(frame1, command = translate, text="Translate", relief=RAISED, borderwidth=2, font=('verdana',10,'bold'), bg='#248aa2', fg="white", cursor="hand2")
btn1.place(x=150, y=300)

btn2 = Button(frame1, command = clear, text="Clear", relief=RAISED, borderwidth=2, font=('verdana',10,'bold'), bg='#248aa2', fg="white", cursor="hand2")
```

Code

```
btn2.place(x=270,y=300)

btn3 = Button(frame1, command = speak, text="Speak Something", relief=RAISED, borderwidth=2, font=('verdana',10,'bold'), bg='#248aa2', fg="white", cursor="hand2")
btn3.place(x=350,y=300)



root.mainloop()
```

Code Explanation

Imports

1. `from tk import *`: Provides GUI components (deprecated; `tkinter` preferred).
2. `import tkinter as tk`: Imports Python's standard GUI toolkit.
3. `from tkinter import ttk`: Adds themed widgets like dropdowns.
4. `from tkinter import messagebox`: Allows showing pop-up messages.
5. `import requests`: A library for HTTP requests (used for APIs).
6. `import uuid`: Generates unique IDs for requests.
7. `import json`: Handles JSON data for requests and responses.
8. `import azure.cognitiveservices.speech as speechsdk`: Azure SDK for speech synthesis and recognition.

Azure Speech Services Configuration

1. `speech_config = speechsdk.SpeechConfig(subscription="...", region="...")`: Configures Azure Speech SDK with your subscription key and region.
2. `audio_config = speechsdk.audio.AudioOutputConfig(use_default_speaker=True)`: Sets up audio output to use the default speaker.
3. `speech_config.speech_synthesis_voice_name = 'hi-IN-MadhurNeural'`: Sets the voice for text-to-speech in Hindi.
4. `speech_synthesizer = speechsdk.SpeechSynthesizer(...)`: Creates a synthesizer for converting text to speech.

Code Explanation

GUI Setup

1. `root = tk.Tk()`: Creates the main application window.
2. `root.title("...")`: Sets the title of the window.
3. `root.geometry('590x370')`: Sets the size of the window to 590x370 pixels.

Main Frame and Label

1. `frame1 = Frame(...)`: Creates a container for widgets with a specific size, style, and background color.
2. `frame1.place(x=0, y=0)`: Places the frame at the top-left corner of the window.
3. `Label(root, ...)`: Adds a title label to the app.

Translation Function

1. `lang_1 = text_entry1.get("1.0", "end")`: Gets the user input from the first text box.
2. `text_entry2.delete(1.0, "end")`: Clears any text in the second text box.
3. `cl = choose_language.get()`: Retrieves the selected target language from the dropdown.
4. Language Code Loop: Finds the language code corresponding to the selected language.
5. Azure API Configuration: Sets up the `key`, `endpoint`, `params`, `headers`, and `body` for the Translator API request.
6. API Call: Sends a POST request using `requests.post` and retrieves the response.
7. Display Result: Inserts the translated text into the output text box.

Code Explanation

Speech Synthesis

1. Converts the translated text to speech asynchronously using Azure.
2. Checks if synthesis is completed or canceled and logs messages accordingly.

Error Handling

Displays an error message using a pop-up dialog (`messagebox.showerror`) if an exception occurs.

Clear Function

Clears both the input (`text_entry1`) and output (`text_entry2`) text boxes when called.

Dropdown for Language Selection

1. `auto_select`: Dropdown for selecting the source language (not fully implemented).
2. `choose_language`: Dropdown populated with supported languages from the `LANGUAGES` dictionary.

Code Explanation

Text Boxes for Input and Output

1. `text_entry1`: A multi-line text box for user input.
2. `text_entry2`: A multi-line text box for displaying translated output.

Buttons for Actions

1. `btn1`: Button to trigger the `translate` function.
2. `btn2`: Button to clear both text boxes using the `clear` function.
3. `btn3`: Button to start speech recognition via the `speak` function.

Speech Recognition

1. Captures audio input using Azure’s Speech SDK and transcribes it into text.
2. Inserts the recognized text into the input text box (`text_entry1`).
3. Handles errors for no speech or recognition cancellations.

Thank you