PES UNIVERSITY

EC Campus, Bengaluru

Department of Computer Science & Engineering

COMPUTER NETWORKS - UE21CS252B

5th Semester – E Section

ASSIGNMENT – 2

Title of the Project

LANGUAGE TRANSLATION WEB SERVICE

Submitted to:

Dr. Geetha D

Associate Professor

Submitted By:

Name : Monish D

SRN: PES2UG21CS303

Name: Madhava bhave

SRN: PES2UG21CS265

Name: MV Sanjay

SRN: PES2UG21CS295

Table of Contents

Sl.No	Title	Page No
1.	Abstract and Scope of the Project	03
2.	Project Description	04
3.	Software Requirements	05
4.	SOURCE CODE	06-09
5.	Sample Output	10
6.	Conclusion	11
7.	Refernces	**12**

1. Abstract and Scope of the Project

- This project is a client-server based machine translation service that allows users to translate text from one language to another. The server side of the project uses a Python socket to accept incoming connections and translate text using the Google Translate API, while the client side of the project allows users to input text to be translated and receive the translated text back from the server.

2. Project Description

The project consists of two Python scripts: the server script and the client script. The server script listens for incoming connections on a specified IP address and port number, and uses a separate thread to handle each incoming connection. Upon receiving a connection, the server script receives JSON data containing the text to be translated and the target language, translates the text using the Google Translate API, and sends the translated text back to the client in JSON format.

The client script prompts the user to input text to be translated and the target language, prepares the data as JSON, sends it to the server using a socket connection, and creates a separate thread to handle the socket connection. While waiting for the response from the server, the client script continues to do other work until the response is received and printed to the console.

Software Requirements

To run the project, the following software must be installed:

- Python 3.x
- googletrans library (can be installed using pip)
- An active internet connection for the Google Translate API

Source Code Server:

```
import socket
import json
from googletrans import Translator
import threading

# Function to handle each client connection
def handle_client(client_socket, addr):
    print(f"Connection from {addr} has been established.")

    try:
        # Receive data from the client
        data = client_socket.recv(1024)
        if len(data) > 0:
            # Load JSON data
            data = json.loads(data.decode("utf-8"))
            text = data.get("text", "")
            target_language = data.get("target_language", "en")

            # Translate text using machine translation API
            translator = Translator()
            translated_text = translator.translate(text, dest=target_language).text

            # Prepare response data as JSON
            response_data = {"translated_text": translated_text}
            response = json.dumps(response_data).encode("utf-8")

            # Send translated text back to the client
            client_socket.send(response)
        else:
            # Send error message if no data received
            error_response = json.dumps({"error": "No data received"}).encode("utf-8")
            client_socket.send(error_response)
    except Exception as e:
        # Send error message if an exception occurs
        error_response = json.dumps({"error": str(e)}).encode("utf-8")
        client_socket.send(error_response)
    finally:
        # Close client socket
        client_socket.close()

# Create a socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = "10.14.142.234" # Host IP address
port = 8000 # Port number
server_socket.bind((host, port))
server_socket.listen(5)
```

```
print(f"Translation service is up and running on {host}:{port}...")

while True:
    # Accept incoming client connections
    client_socket, addr = server_socket.accept()

    # Start a new thread for each incoming connection
    threading.Thread(target=handle_client, args=(client_socket, addr)).start()
```

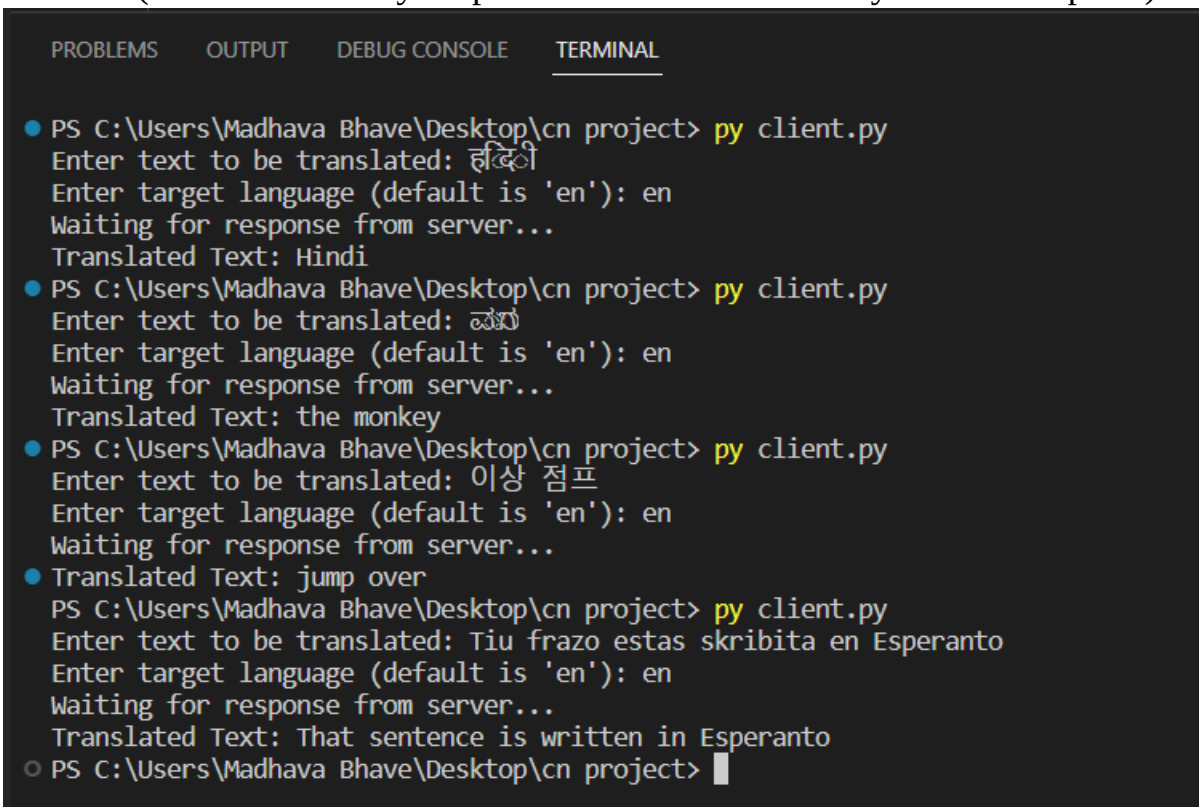
source code client:

```
import json
import socket
import threading

def handle_connection(client_socket):
    # Receive data from the server
    response = client_socket.recv(1024).decode("utf-8")
    if response:
        # Load JSON response
        response_data = json.loads(response)
        translated_text = response_data.get("translated_text", "")
        print("Translated Text:", translated_text)
    else:
        print("Error: No response received from the server.")
    # Close client socket
    client_socket.close()

# Create a socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = "10.14.142.234"  # Host IP address
port = 8000  # Port number

try:
    # Connect to the server
    client_socket.connect((host, port))
    # Input text to be translated
    text = input("Enter text to be translated: ")
    target_language = input("Enter target language (default is 'en'): ")
    # Prepare data as JSON
    data = {"text": text, "target_language": target_language}
    data = json.dumps(data).encode("utf-8")
    # Send data to the server
    client_socket.send(data)
    # Create a thread to handle the socket connection
    connection_thread = threading.Thread(target=handle_connection,
                                         args=(client_socket,))
    connection_thread.start()
    # Do other work while waiting for the response
    print("Waiting for response from server...")
    # Continue working while the thread is running
    connection_thread.join()
except Exception as e:
    print("Error: Could not connect to the server. {}".format(str(e)))
```

```
finally:
    # Close client socket
    client_socket.close()
```

Sample Output

(Include necessary output screenshots followed by brief description)



Conclusion

This project demonstrates the use of sockets and threads in a client-server architecture, as well as the integration of a third-party API (Google Translate) for machine translation. With this project as a foundation, further enhancements can be made to improve the performance, scalability, and reliability of the machine translation service.

References

https://www.geeksforgeeks.org/

https://github.com/ssut/py-googletrans