A MAJOR PROJECT SUMMARY

ON

‘LANGUAGE TRANSLATION USING PYTHON’

Submitted in the partial fulfillment for award of degree in

BACHELOR OF TECHNOLOGY (CSE-AIML)

to

Suresh Gyan Vihar University, Jaipur

Submitted by

Name- Anwar Alam

SID- 100846

SEM- 7th

Under the guidance of

Ms Priyanka Gupta

Assistant Professor-CEIT

Department of Computer Engineering and Information Technology

Suresh Gyan Vihar University, Jaipur

2024-2025

TABLE OF CONTENT

1 . Introduction

Purpose of the Application

The Language Translator application serves as a tool to translate text from one language to another using Google's Translation API. It is designed to provide a user-friendly interface for seamless language translation with additional features like word and character count, live digital clock, and responsiveness to enhance the user experience.

Key Features

- Multilingual text translation using Google's Translation API.
- Automatic source language detection.
- Real-time word and character count display.
- Live digital clock for user convenience.
- Interactive and responsive user interface for ease of use.

2. Architecture Overview

Backend: Flask Application

The backend is developed using Flask, a lightweight Python web framework. It manages API requests, text processing, and interactions with Google's Translation API.

Frontend: HTML, CSS, and JavaScript

The frontend is built with HTML for structure, CSS for styling and layout, and JavaScript for interactivity. It provides a visually appealing and functional interface for users.

Integration with Jupyter Notebook

The application is integrated into Jupyter Notebook using Flask and ngrok, allowing users to access it directly within a notebook environment.

3. Backend Implementation

Flask Application Setup

The Flask application is initialized with required configurations. The flask_ngrok library is used to expose the local app to the internet via ngrok.

Routes

- **Homepage (/)**
  The homepage serves the main HTML interface where users input text, select languages, and view translations.
- **Translation Endpoint (/translate)**
  This endpoint receives user input via POST requests, translates the text using Google's Translation API, and returns the translated text as a JSON response.

Threading for Jupyter Notebook Compatibility

To enable seamless integration with Jupyter Notebook, the Flask app is run on a separate thread, ensuring compatibility and uninterrupted functionality.

4. Frontend Implementation

HTML Structure
The HTML provides a structured layout comprising input and output sections. Users can select source and target languages, input text, and view translated results.

CSS for Styling and Layout
CSS is used to create a visually appealing layout with responsive design elements. The input and output sections are styled to enhance readability and usability.

JavaScript for Interactivity

Digital Clock
A live digital clock updates every second, displaying the current time for user convenience.

- **Word and Character Count**
  JavaScript functions dynamically calculate and display the number of words and characters in the input text.
- **Text Translation**
  An asynchronous function sends user input to the backend and displays the translated text in the output section.
- **Clear Functionality**
  A button allows users to clear the input and output fields, resetting the application for new translations.

5. Application Features

Translation with Googletrans API
The application utilizes Google’s Translation API to provide accurate and fast translations across multiple languages.

Real-Time Word and Character Count
Word and character counts update dynamically as the user types, offering real-time feedback.

Live Digital Clock
A real-time clock displayed on the interface adds a functional element for users.

Responsive User Interface
The application’s design ensures compatibility across devices, adapting to various screen sizes.

6. Integration with Jupyter Notebook

Embedding Flask App with IFrame
The application is displayed within Jupyter Notebook using an IFrame, offering an embedded, interactive experience.

Displaying the App in the Notebook
Ngrok exposes the local Flask application, allowing it to be accessed through a generated URL and displayed in the notebook.

7. Libraries and Tools Used

Python Libraries

- Flask: Web framework for the backend.
- Googletrans: Translation API for language translation.
- flask_ngrok: Exposes the local Flask app via ngrok.

Frontend Technologies

- HTML: Markup language for structuring the interface.
- CSS: Styling and layout for the frontend.
- JavaScript: Adds interactivity to the application.

8. How the Application Works

Steps to Use the Application

1. Open the application in the browser or Jupyter Notebook.
2. Select the source and target languages.
3. Input text into the provided field.
4. Click the "Translate" button to generate the translated text.
5. View the translation in the output section.

Key Functionality Flow

1. User inputs text and selects languages.
2. Text and language preferences are sent to the Flask backend.
3. The backend processes the request using Google's Translation API.
4. Translated text is returned and displayed on the frontend.

9. Conclusion

Summary of Features and Benefits

The Language Translator application combines powerful translation capabilities with a user-friendly interface, offering real-time features like word count, character count, and a live clock. Its integration with Jupyter Notebook makes it accessible for developers and researchers.

Learning Outcomes

This project demonstrates the integration of Flask with frontend technologies and third-party APIs, highlighting best practices in web development and real-time application features.