**Project Report**

*on*

**“Interactive Language Translator”**

*Submitted in partial fulfilment of the requirement*

*of*

**University of Mumbai**

*for*

**Skill Base Lab Course: Python Programming**

*Submitted by*

55 Ashita Salis

66 Amisha Verma

63 Merina George Thoppil



***Under the guidance of***

Prof Imran Mirza

Department of Computer Engineering

**DON BOSCO INSTITUTE OF TECHNOLOGY, KURLA**

**Index**

**INTRODUCTION**

**PROGRAM**

**RESULTS**

**CHANGES TO BE IMPLEMENTED**

**Modified OUTPUT SCREENS**

**CONCLUSION**

**BIBILOGRAPHY**

**Introduction**

In our increasingly interconnected world, effective communication across languages is essential for breaking down barriers and fostering collaboration. An interactive language translator serves as a vital tool in facilitating cross-cultural communication by providing users with the ability to translate words, phrases, and even sentences in real-time. This technology not only aids individuals in overcoming language barriers but also promotes cultural exchange and understanding.

**1. Need for Cross-Language Communication:**

* Discuss the importance of overcoming language barriers in today's globalized world.
* Highlight the challenges individuals and businesses face in communicating across languages.
* Emphasize the role of language translators in bridging these communication gaps.

**2. Introduction to Interactive Language Translator:**

* Define what an interactive language translator is and how it differs from traditional translation methods.
* Explain the concept of real-time translation and its significance in facilitating seamless communication.
* Introduce the various features and functionalities typically found in interactive language translators, such as multilingual support, text-to-speech capabilities, and context-based translation.

**3. Technology Behind Interactive Language Translator:**

* Explore the underlying technologies powering interactive language translators, including natural language processing (NLP), machine learning, and neural machine translation (NMT).
* Discuss how these technologies enable the system to understand and translate text accurately and efficiently.
* Provide insights into the challenges and limitations associated with machine translation and ongoing research efforts to improve translation quality.

**4. Implementation and Integration:**

* Detail the process of integrating an interactive language translator into different platforms and applications, such as websites, mobile apps, and messaging services.
* Discuss the role of APIs and SDKs in enabling seamless integration with existing systems.
* Provide examples of how interactive language translators are incorporated into various industries, including travel and tourism, e-commerce, and customer support.

**2) Program**

App.py

from flask import Flask, render\_template, request,jsonify

import re

from gtts import gTTS

import requests

from bs4 import BeautifulSoup

app = Flask(\_\_name\_\_)

languages = {'1': 'Arabic', '2': 'German', '3': 'English', '4': 'Spanish', '5': 'French', '6': 'Hebrew',

             '7': 'Japanese', '8': 'Dutch', '9': 'Polish', '10': 'Portuguese', '11': 'Romanian', '12': 'Russian',

             '13': 'Turkish'}

def response\_function(input\_data):

    user\_agent = 'Mozilla/5.0'

    try:

        response = requests.get(f'https://context.reverso.net/translation/{languages[input\_data[0]].lower()}-{languages[input\_data[1]].lower()}/{input\_data[2]}', headers={'User-Agent': user\_agent})

        soup = BeautifulSoup(response.content, 'html.parser')

        # Find translations and examples

        translations = [word.text.strip() for word in soup.find\_all('a', class\_='translation')]

        examples = [word.text.strip() for word in soup.find\_all('div', class\_='example')]

        # If translations are not found but examples are available, prioritize examples

        if not translations and examples:

            translations = ['Translations not directly available. See examples below.']

        return translations, examples

    except:

        return None, None

@app.route('/')

def index():

    return render\_template('index.html', languages=languages)

@app.route('/translate', methods=['POST'])

def translate():

    if request.method == 'POST':

        # Check if the request contains speech recognition data

        if 'speechToTranslate' in request.form:

            speech\_to\_translate = request.form['speechToTranslate']

            # Process the speech for translation...

            # Example: pass the speech to your existing translation function

            translations, examples = response\_function(['3', '4', speech\_to\_translate])  # Assuming '3' is English and '4' is Spanish

            if translations and examples:

                return jsonify({'translations': translations, 'examples': examples})

            else:

                return jsonify({'error': 'Error occurred while fetching translations.'})

        else:

            # If no speech recognition data is present, proceed with regular text input translation

            language\_from = request.form['language\_from']

            language\_to = request.form['language\_to']

            word = request.form['word'].lower()

            translations, examples = response\_function([language\_from, language\_to, word])

            if translations and examples:

                return render\_template('translation\_result.html', translations=translations, examples=examples)

            else:

                return "Error occurred while fetching translations."

@app.route('/faq')

def faq():

    faq\_content = """

    Frequently Asked Questions

    Q: How accurate is the translation?

    A: The translation provided by the app is based on available online resources and may not always be 100% accurate.

    Q: Can I translate entire sentences?

    A: Yes, you can translate words, phrases, or entire sentences using the app.

    Q: Can I translate between any two languages?

    A: The app supports translation between a variety of languages listed on the home page.

    """

    # Generate speech for FAQ content

    tts\_faq = gTTS(text=faq\_content, lang='en')

    tts\_faq.save("faq.mp3")

    return render\_template('faq.html', faq\_content=faq\_content, faq\_audio="faq.mp3")

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

faq.html

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>FAQ</title>

    <style>

      body {

        background-color: #808080; /\* Set background color to gray \*/

        color: white; /\* Set text color to white for better readability \*/

        font-family: Arial, sans-serif;

        padding: 20px; /\* Add some padding for better spacing \*/

      }

      h1,

      h2 {

        color: #000; /\* Set heading text color to black \*/

      }

      a {

        color: #00f; /\* Set link text color to blue \*/

        text-decoration: none; /\* Remove underline from links \*/

        font-weight: bold; /\* Make links bold \*/

        margin-top: 20px; /\* Add space between paragraphs and links \*/

        display: block; /\* Make links block elements \*/

      }

      a:hover {

        color: #f00; /\* Change link color on hover to red \*/

      }

    </style>

  </head>

  <body>

    <h1>Frequently Asked Questions</h1>

    <h2>Q: How accurate is the translation?</h2>

    <p>

      A: The translation provided by the app is based on available online

      resources and may not always be 100% accurate.

    </p>

    <h2>Q: Can I translate entire sentences?</h2>

    <p>

      A: Yes, you can translate words, phrases, or entire sentences using the

      app.

    </p>

    <h2>Q: Can I translate between any two languages?</h2>

    <p>

      A: The app supports translation between a variety of languages listed on

      the home page.

    </p>

    <audio controls>

      <source

        src="{{ url\_for('static', filename='faq.mp3') }}"

        type="audio/mpeg"

      />

      Your browser does not support the audio element.

    </audio>

    <a href="/">Back to Home</a>

  </body>

</html>

Index.html

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>Language Translator</title>

    <style>

      body {

        background-color: gray; /\* Set background color to blue \*/

        color: white; /\* Set text color to white for better readability \*/

        font-family: Arial, sans-serif;

        font-family: Arial, sans-serif;

        text-align: center; /\* Align content to center \*/

      }

      h1 {

        font-size: 2em; /\* Increase font size \*/

      }

      form {

        display: flex;

        flex-direction: column;

        align-items: center;

        margin-top: 20px; /\* Add some space between heading and form \*/

      }

      select,

      input,

      button {

        font-size: 1.2em; /\* Increase font size for form elements \*/

        margin-bottom: 10px; /\* Add some space between form elements \*/

        width: calc(100% - 20px); /\* Set width to 100% minus padding \*/

        max-width: 400px; /\* Limit maximum width \*/

        padding: 10px; /\* Add padding to form elements \*/

      }

      button {

        width: calc(

          2 / 3 \* 100% - 20px

        ); /\* Button occupies 2/3 of the screen width \*/

      }

      button,

      a {

        display: inline-block; /\* Ensure button and link are displayed inline \*/

      }

      select#font\_size {

    margin-bottom: 10px;

}

select#font\_size {

    font-size: 1.2em;

  }  </style>

  </head>

  <body>

    <h1>Welcome to Language Translator</h1>

    <form action="/translate" method="post">

      <label for="language\_from">From:</label>

      <select name="language\_from" id="language\_from">

        {% for key, value in languages.items() %}

        <option value="{{ key }}">{{ value }}</option>

        {% endfor %}

      </select>

      <label for="language\_to">To:</label>

      <select name="language\_to" id="language\_to">

        {% for key, value in languages.items() %}

        <option value="{{ key }}">{{ value }}</option>

        {% endfor %}

      </select>

      <label for="word">Word to Translate:</label>

      <input type="text" id="word" name="word" required />

      <!-- Add this button for speech recognition -->

<button type="button" id="startSpeechRecognition">Start Speech Recognition</button>

      <select name="font\_size" id="font\_size">

        <option value="1.2em">Small</option>

        <option value="1.5em">Medium</option>

        <option value="2em">Large</option>

      </select>

      <button type="submit">Translate</button>

    </form>

    <a href="/faq">FAQ</a>

    <!-- JavaScript to adjust font size -->

    <script>

      document.getElementById('font\_size').addEventListener('change', function() {

        var fontSize = this.value;

        document.body.style.fontSize = fontSize;

      });

      document.getElementById('startSpeechRecognition').addEventListener('click', function() {

    // Start speech recognition

    var recognition = new webkitSpeechRecognition();

    recognition.lang = 'en-US'; // Set the language for recognition

    recognition.onresult = function(event) {

      var speechToTranslate = event.results[0][0].transcript;

      document.getElementById('word').value = speechToTranslate; // Set the recognized speech as the input value

    };

    recognition.onerror = function(event) {

      console.error('Speech recognition error:', event.error);

    };

    recognition.start();

  });

    </script>

  </body>

</html>

Translation\_result.html

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>Translation Result</title>

    <style>

      body {

        background-color: gray; /\* Set background color to blue \*/

        color: white; /\* Set text color to white for better readability \*/

        font-family: Arial, sans-serif;

      }

      .container {

        max-width: 800px;

        margin: 0 auto;

        padding: 20px;

      }

      h1 {

        font-size: 2em;

        text-align: center;

      }

      .translations {

        margin-bottom: 16px;

      }

      .example {

        margin-bottom: 16px;

      }

      select#font\_size {

    margin-bottom: 10px;

}

    </style>

  </head>

  <body>

    <div class="container">

      <h1>Translation Result</h1>

      <div class="translations">

        <h2>Translations</h2>

        <ul>

          {% for translation in translations %}

          <li>{{ translation }}</li>

          {% endfor %}

        </ul>

      </div>

      <div class="examples">

        <h2>Examples</h2>

        <ul>

          {% for example in examples %}

          <li>{{ example }}</li>

          {% endfor %}

        </ul>

      </div>

      <select name="font\_size" id="font\_size">

        <option value="1.2em">Small</option>

        <option value="1.5em">Medium</option>

        <option value="2em">Large</option>

      </select>

    </form>

    <!-- JavaScript to adjust font size -->

    <script>

      document.getElementById('font\_size').addEventListener('change', function() {

        var fontSize = this.value;

        document.body.style.fontSize = fontSize;

      });

    </script>

      <a href="/">Back to Home</a>

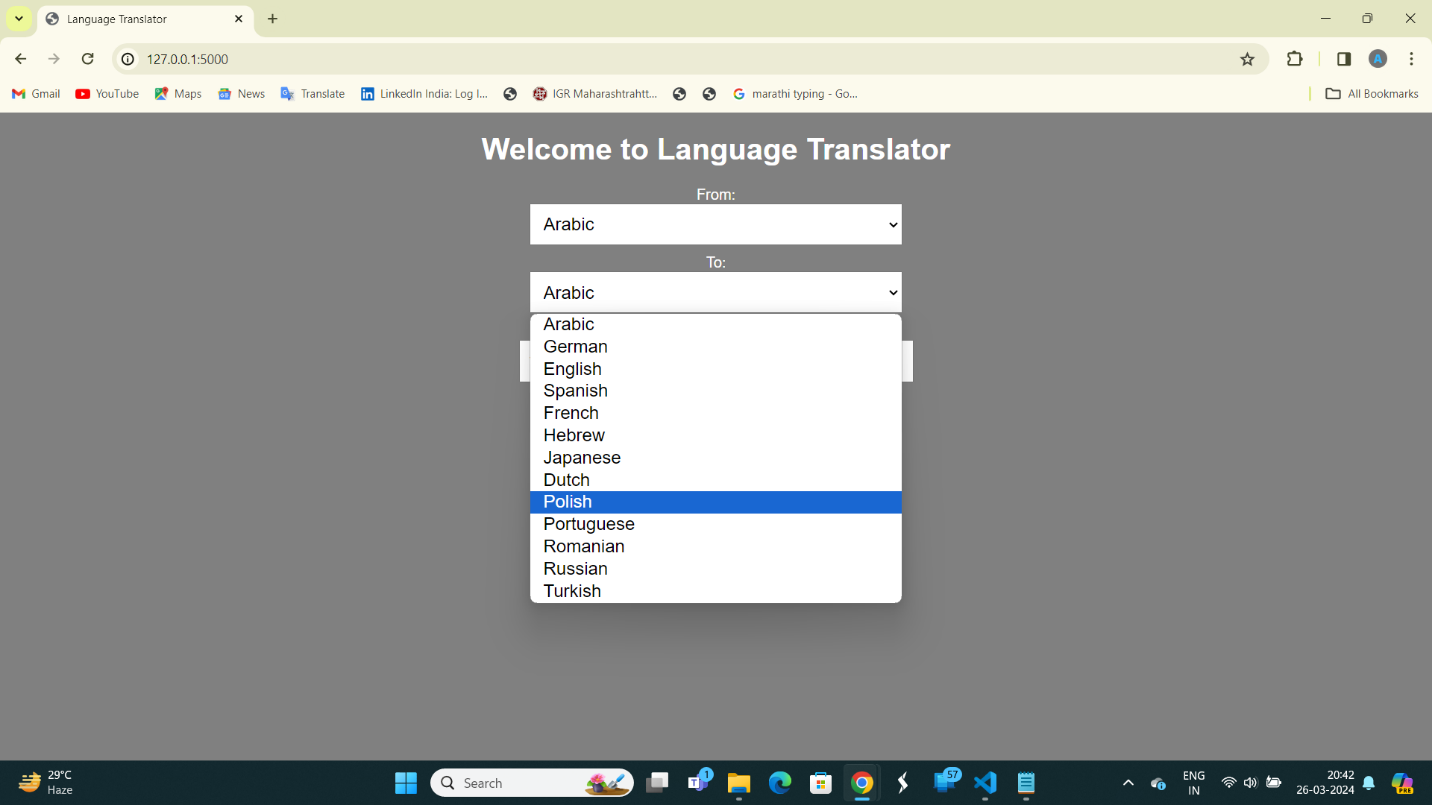
      <a href="/faq">FAQ</a>

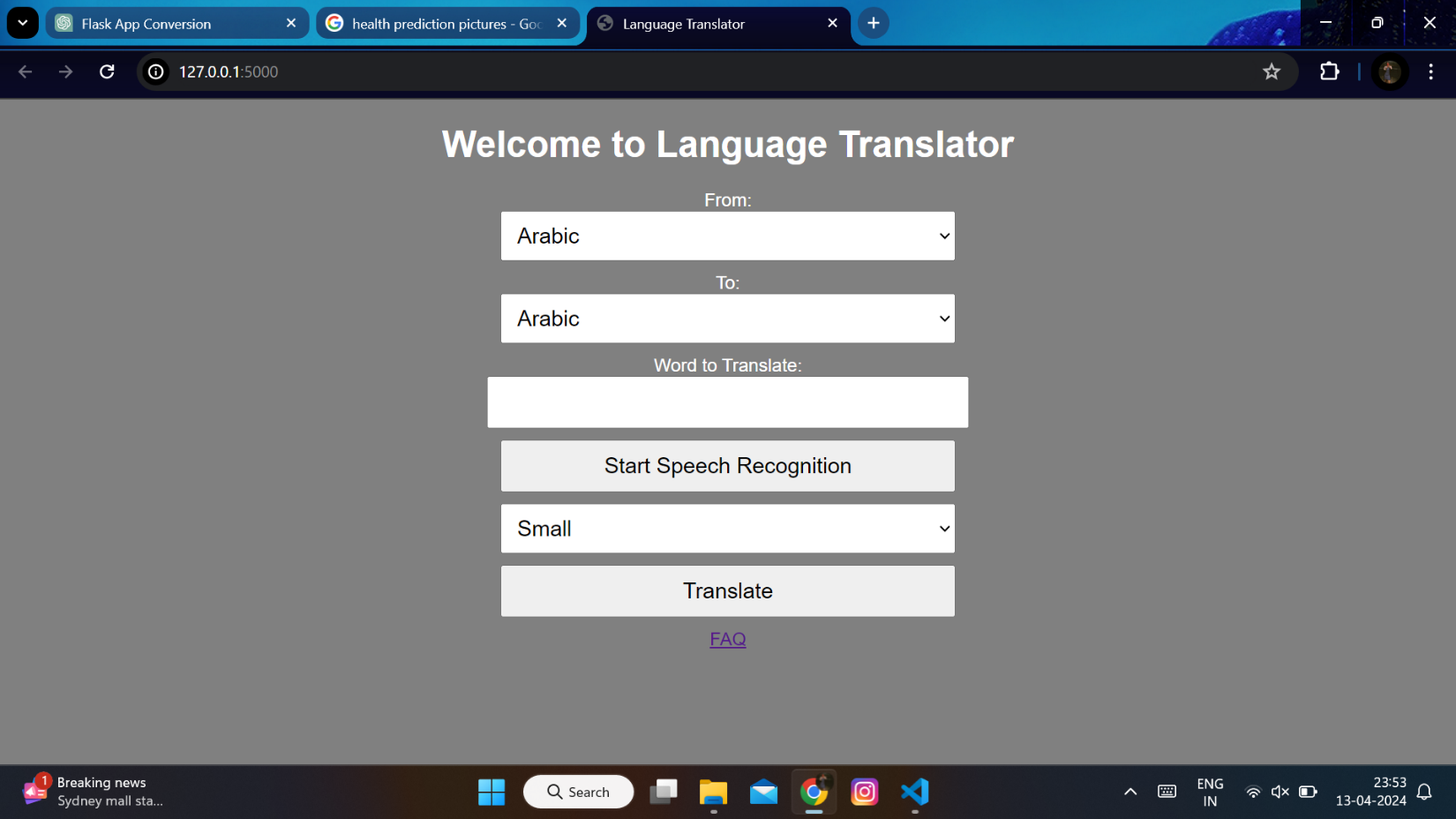
    </div>

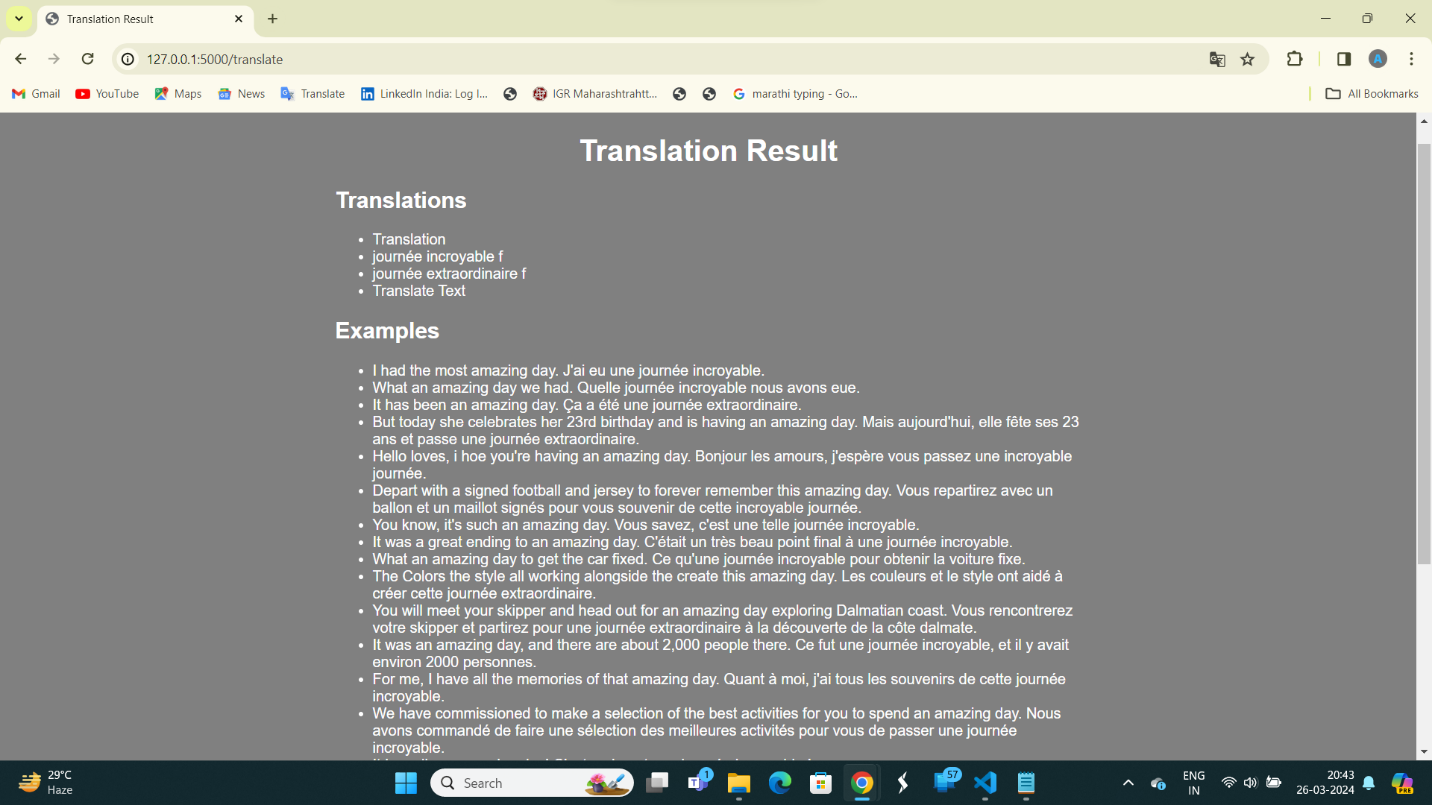
  </body>

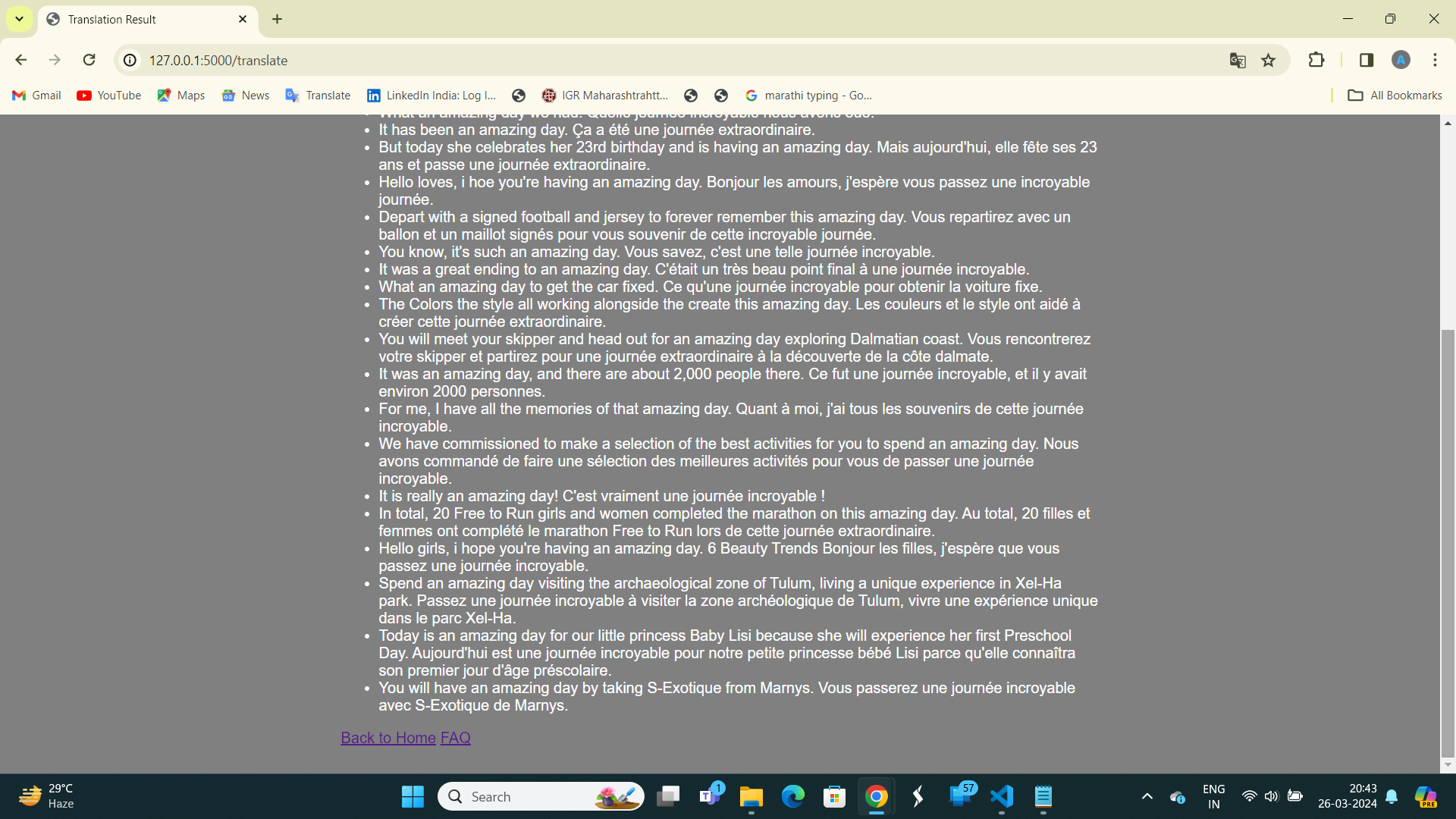
</html>

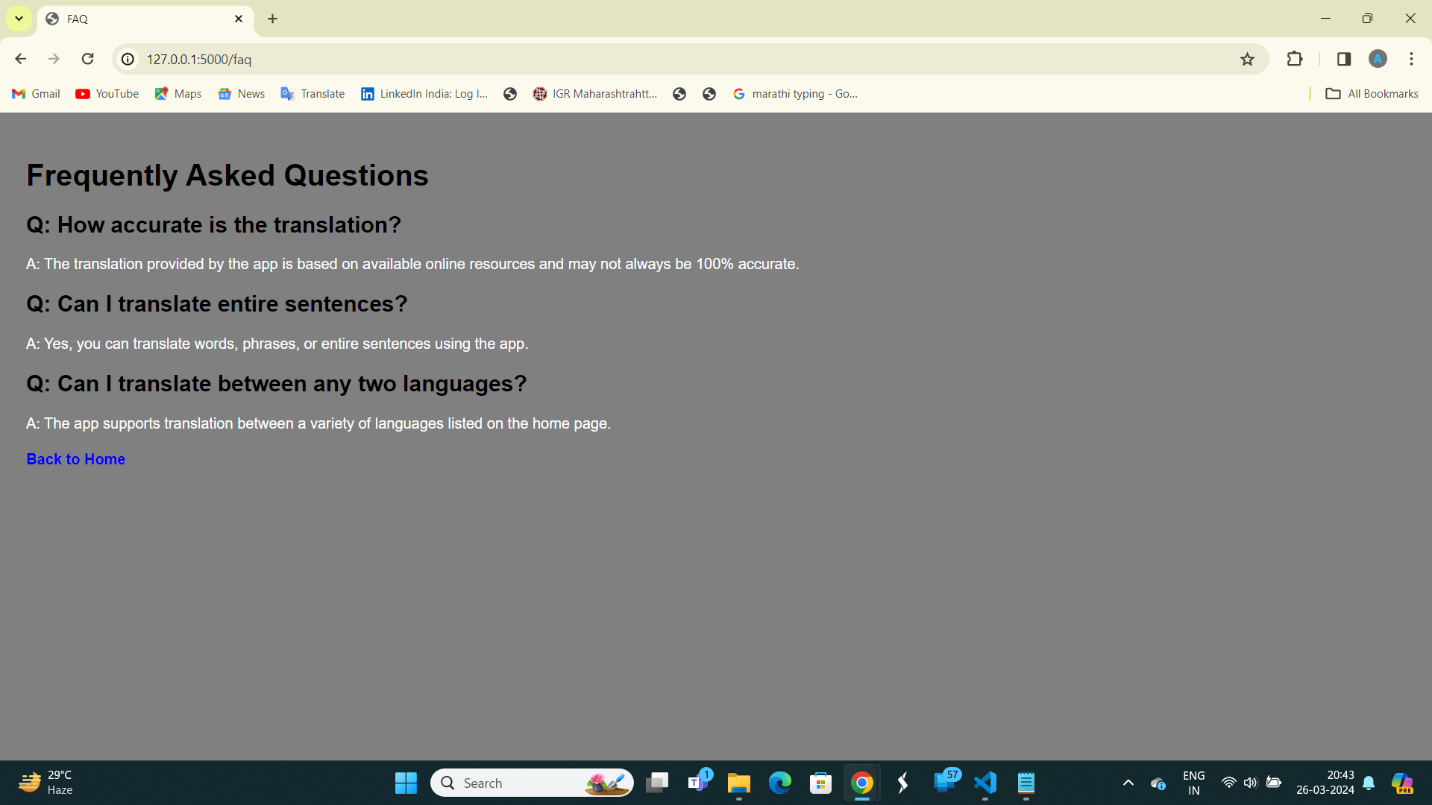
**3) Results:**











**4)** **Changes to be Implemented**

Some potential improvements and changes that could be implemented in the project include:

Error handling: Implement better error handling to gracefully handle cases where translation data cannot be fetched.

User authentication: Add user authentication to allow registered users to save their translation history or preferences.

Performance optimization: Optimize the code for better performance, especially when dealing with large amounts of translation data

**5)** **Modified Output Screens**

The output screens include:

index.html: The homepage where users can select the source and target languages and enter the word to translate.

faq.html: Frequently Asked Questions page providing information about the translation accuracy and usage of the application.

translation\_result.html: The translation result page displaying the translated word or phrase along with examples.

**6) Conclusion**

The interactive language translator project provides a simple yet effective tool for translating words or phrases between multiple languages. While the current implementation works well, there is room for improvement in terms of error handling, user authentication, and performance optimization.

**7)** **Bibliography**

Flask Documentation: <https://flask.palletsprojects.com/en/3.0.x/>

Requests Documentation: <https://requests.readthedocs.io/en/latest/>

BeautifulSoup Documentation: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>