```
# 1. Install Flask and the special Tunnel tool
!pip install flask deep-translator flask-cloudflared

# 2. Import libraries
from flask import Flask, render_template_string, request, jsonify
from deep_translator import GoogleTranslator
from flask_cloudflared import run_with_cloudflared

app = Flask(__name__)
# This line creates the reliable public tunnel automatically
run_with_cloudflared(app)

# --- HTML Frontend ---
HTML_TEMPLATE = """
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Translator App</title>
    <style>
        body { font-family: sans-serif; background: #eef2f3; display: flex; justify-content: center; align-items: center; mi
        .container { background: white; padding: 2rem; border-radius: 15px; box-shadow: 0 10px 25px rgba(0,0,0,0.1); width:
        h1 { text-align: center; color: #333; margin-bottom: 20px; }
        textarea { width: 100%; height: 120px; padding: 15px; border: 1px solid #ddd; border-radius: 10px; resize: none; fon
        .controls { display: flex; gap: 10px; margin: 20px 0; }
        select, button { flex: 1; padding: 12px; border-radius: 8px; border: 1px solid #ddd; font-size: 16px; height: 50px;
        button { background: #007bff; color: white; border: none; cursor: pointer; font-weight: bold; transition: background
        button:hover { background: #0056b3; }
        button:disabled { background: #ccc; }
        .result-box { background: #f8f9fa; color: #333; font-weight: 500; }
        label { font-weight: bold; margin-bottom: 5px; display: block; color: #555; }
    </style>
</head>
<body>
    <div class="container">
        <h1>Global Translator</h1>

        <label>Input Text:</label>
        <textarea id="inputText" placeholder="Type something to translate..."></textarea>

        <div class="controls">
            <select id="langSelect">
                <option value="es">Spanish</option>
                <option value="fr">French</option>
                <option value="de">German</option>
                <option value="it">Italian</option>
                <option value="ja">Japanese</option>
                <option value="ko">Korean</option>
                <option value="ru">Russian</option>
                <option value="zh-CN">Chinese</option>
                <option value="hi">Hindi</option>
            </select>
            <button id="transBtn" onclick="translateText()">Translate</button>
        </div>

        <label>Translation:</label>
        <textarea id="outputText" class="result-box" readonly placeholder="Result will appear here..."></textarea>
    </div>

    <script>
        async function translateText() {
            const text = document.getElementById("inputText").value;
            const targetLang = document.getElementById("langSelect").value;
            const outputBox = document.getElementById("outputText");
            const btn = document.getElementById("transBtn");

            if (!text) { alert("Please enter text!"); return; }

            outputBox.value = "Translating...";
            btn.disabled = true;
            btn.innerText = "Working...";

            try {
                const response = await fetch('/translate', {
                    method: 'POST',
                    headers: { 'Content-Type': 'application/json' },
                    body: JSON.stringify({ text: text, target: targetLang })
                });

                const data = await response.json();
                const data = await response.json();
                outputBox.value = data.translated_text || "Error: " + data.error;
            } catch (err) {
                outputBox.value = "Network Error: " + err;
            } finally {
                btn.disabled = false;
                btn.innerText = "Translate";
            }
        }
    </script>
</body>
</html>
"""

# --- Backend Logic ---
@app.route('/')
def home():
    return render_template_string(HTML_TEMPLATE)

@app.route('/translate', methods=['POST'])
def translate():
    try:
        data = request.get_json()
        translator = GoogleTranslator(source='auto', target=data.get('target', 'en'))
        return jsonify({'translated_text': translator.translate(data.get('text', ''))})
    except Exception as e:
        return jsonify({'error': str(e)}), 500

if __name__ == '__main__':
    # This runs the app on the special Cloudflare tunnel
    print("Starting server... Please wait for the 'trycloudflare.com' link below.")
    app.run()
```

```
Requirement already satisfied: flask in /usr/local/lib/python3.12/dist-packages (3.1.2)
Requirement already satisfied: deep-translator in /usr/local/lib/python3.12/dist-packages (1.11.4)
Collecting flask-cloudflared
  Downloading flask_cloudflared-0.0.14-py3-none-any.whl.metadata (4.6 kB)
Requirement already satisfied: blinker>=1.9.0 in /usr/local/lib/python3.12/dist-packages (from flask) (1.9.0)
Requirement already satisfied: click>=8.1.3 in /usr/local/lib/python3.12/dist-packages (from flask) (8.3.1)
Requirement already satisfied: itsdangerous>=2.2.0 in /usr/local/lib/python3.12/dist-packages (from flask) (2.2.0)
Requirement already satisfied: jinja2>=3.1.2 in /usr/local/lib/python3.12/dist-packages (from flask) (3.1.6)
Requirement already satisfied: markupsafe>=2.1.1 in /usr/local/lib/python3.12/dist-packages (from flask) (3.0.3)
Requirement already satisfied: werkzeug>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from flask) (3.1.5)
Requirement already satisfied: beautifulsoup4<5.0.0,>=4.9.1 in /usr/local/lib/python3.12/dist-packages (from deep-translator
Requirement already satisfied: requests<3.0.0,>=2.23.0 in /usr/local/lib/python3.12/dist-packages (from deep-translator) (2.
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.12/dist-packages (from beautifulsoup4<5.0.0,>=4.9.1->
Requirement already satisfied: typing-extensions>=4.0.0 in /usr/local/lib/python3.12/dist-packages (from beautifulsoup4<5.0.
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.23.0->deep-t
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.23.0->
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.23.0->
Downloading flask_cloudflared-0.0.14-py3-none-any.whl (6.4 kB)
Installing collected packages: flask-cloudflared
Successfully installed flask-cloudflared-0.0.14
Starting server... Please wait for the 'trycloudflare.com' link below.
 * Serving Flask app '__main__'
 * Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
 * Downloading cloudflared for Linux x86_64...
 * Running on https://hindu-humidity-sunday-panels.trycloudflare.com
 * Traffic stats available on http://127.0.0.1:8924/metrics
INFO:werkzeug:127.0.0.1 - - [16/Jan/2026 16:53:36] "GET / HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [16/Jan/2026 16:53:42] "GET /favicon.ico HTTP/1.1" 404 -
INFO:werkzeug:127.0.0.1 - - [16/Jan/2026 16:53:53] "POST /translate HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [16/Jan/2026 16:53:58] "POST /translate HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [16/Jan/2026 16:54:01] "GET / HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [16/Jan/2026 16:55:54] "POST /translate HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [16/Jan/2026 17:11:30] "GET / HTTP/1.1" 200 -
```