```
import numpy as np
from flask import Flask, render_template_string, request, jsonify
from transformers import MarianMTModel, MarianTokenizer
from langdetect import detect

app = Flask(__name__)

# Preload models and tokenizers at startup
model_mapping = {
    'English': "Helsinki-NLP/opus-mt-mul-en",
    'Spanish': "Helsinki-NLP/opus-mt-en-es",
    'French': "Helsinki-NLP/opus-mt-en-fr",
    'German': "Helsinki-NLP/opus-mt-en-de",
    'Chinese': "Helsinki-NLP/opus-mt-en-zh",
    'Hindi': "Helsinki-NLP/opus-mt-en-hi"
}

models = {lang: MarianMTModel.from_pretrained(model_name) for lang, model_name in model_mapping.items()}
tokenizers = {lang: MarianTokenizer.from_pretrained(model_name) for lang, model_name in model_mapping.items()}

# Placeholder language list for UI
languages = list(model_mapping.keys())

def translate_text(input_text, target_language):
    """Translate the input text to the target language."""
    try:
        # Step 1: Detect the source language
        source_language = detect(input_text)

        # Step 2: If the source language is not English, translate to English first
        if source_language != "en":
            intermediate_model = models.get("English")  # Model for translating to English
            intermediate_tokenizer = tokenizers.get("English")
            if not intermediate_model or not intermediate_tokenizer:
                return "Translation model for intermediate step not available."
            inputs = intermediate_tokenizer.encode(input_text, return_tensors="pt", truncation=True)
            outputs = intermediate_model.generate(inputs, max_length=40, num_beams=4, early_stopping=True)
            input_text = intermediate_tokenizer.decode(outputs[0], skip_special_tokens=True)

        # Step 3: Translate from English to the target language
        model = models.get(target_language)
        tokenizer = tokenizers.get(target_language)
        if not model or not tokenizer:
            return "Translation model not available for the selected language."
        inputs = tokenizer.encode(input_text, return_tensors="pt", truncation=True)
        outputs = model.generate(inputs, max_length=40, num_beams=4, early_stopping=True)
        return tokenizer.decode(outputs[0], skip_special_tokens=True)

    except Exception as e:
        return f"Error during translation: {str(e)}"

def translate_text(input_text, target_language):
    """Translate the input text to the target language."""
    try:
        model = models.get(target_language)
        tokenizer = tokenizers.get(target_language)
        if not model or not tokenizer:
            return "Translation model not available for the selected language."
        inputs = tokenizer.encode(input_text, return_tensors="pt", truncation=True)
        outputs = model.generate(inputs, max_length=40, num_beams=4, early_stopping=True)
        return tokenizer.decode(outputs[0], skip_special_tokens=True)
    except Exception as e:
        return f"Error during translation: {str(e)}"

# HTML template with Bootstrap for UI
html_template = """
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Language Translator</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" />
    <style>
        body {
            background: linear-gradient(135deg, #667eea, #764ba2);
            color: white;
            min-height: 100vh;
            display: flex;
            flex-direction: column;
            justify-content: center;
            align-items: center;
            padding: 2rem;
        }
        .translator-container {
            background: rgba(0,0,0,0.6);
            padding: 2rem;
            border-radius: 15px;
            max-width: 600px;
            width: 100%;
            box-shadow: 0 8px 32px 0 rgba(31, 38, 135, 0.37);
        }
        textarea {
            resize: none;
        }
        .result-box {
            background: #222;
            padding: 1rem;
            border-radius: 10px;
            margin-top: 1rem;
            min-height: 100px;
            white-space: pre-wrap;
            font-size: 1.2rem;
        }
        h1 {
            margin-bottom: 1rem;
            font-weight: 700;
            text-align: center;
        }
        .spinner-border {
            display: none;
        }
    </style>
</head>
<body>
    <div class="translator-container">
        <h1>Language Translator</h1>
        <form id="translateForm">
            <div class="mb-3">
                <label for="inputText" class="form-label">Enter text to translate:</label>
                <textarea class="form-control" id="inputText" rows="4" required></textarea>
            </div>
            <div class="mb-3">
                <label for="languageSelect" class="form-label">Select target language:</label>
                <select class="form-select" id="languageSelect" required>
                    <option value="" disabled selected>Select language</option>
                    {% for lang in languages %}
                    <option value="{{ lang }}">{{ lang }}</option>
                    {% endfor %}
                </select>
            </div>
            <button type="submit" class="btn btn-primary w-100">Translate</button>
        </form>
        <div id="sourceLanguage" class="mt-3"></div>
        <div id="loadingSpinner" class="spinner-border text-light mt-3" role="status">
            <span class="visually-hidden">Loading...</span>
        </div>
        <div id="result" class="result-box mt-3"></div>
    </div>

    <script>
        const form = document.getElementById('translateForm');
        const resultBox = document.getElementById('result');
        const sourceLanguageBox = document.getElementById('sourceLanguage');
        const loadingSpinner = document.getElementById('loadingSpinner');

        form.addEventListener('submit', async (e) => {
            e.preventDefault();
            const inputText = document.getElementById('inputText').value.trim();
            const targetLanguage = document.getElementById('languageSelect').value;

            if (!inputText || !targetLanguage) {
                resultBox.textContent = 'Please enter text and select a language.';
                return;
            }

            resultBox.textContent = '';
            sourceLanguageBox.textContent = '';
            loadingSpinner.style.display = 'block';

            try {
                const response = await fetch('/translate', {
                    method: 'POST',
                    headers: { 'Content-Type': 'application/json' },
                    body: JSON.stringify({ text: inputText, language: targetLanguage })
                });
                const data = await response.json();
                loadingSpinner.style.display = 'none';

                if (data.translated_text) {
                    resultBox.textContent = data.translated_text;
                    sourceLanguageBox.textContent = `Detected Source Language: ${data.source_language}`;
                } else {
                    resultBox.textContent = 'Translation failed.';
                }
            } catch (error) {
                loadingSpinner.style.display = 'none';
                resultBox.textContent = 'Error occurred during translation.';
            }
        });
    </script>
</body>
</html>
"""

@app.route('/')
def index():
    return render_template_string(html_template, languages=languages)

@app.route('/translate', methods=['POST'])
def translate():
    data = request.get_json()
    input_text = data.get('text', '')
    target_language = data.get('language', '')
    if not input_text or not target_language:
        return jsonify({'error': 'Invalid input'}), 400

    # Perform translation
    translated_text = translate_text(input_text, target_language)
    return jsonify({'source_language': detect(input_text), 'translated_text': translated_text})

if __name__ == '__main__':
    app.run(debug=True)
```

Analyze files with Gemini