App.js

import { useState } from 'react';

import logo from './logo.png';

import './App.css';

import CodeMirror from '@uiw/react-codemirror';

import { loadLanguage } from '@uiw/codemirror-extensions-langs';

import { githubLight } from '@uiw/codemirror-theme-github';

function App() {

  const [code, setCode] = useState('');

  const [translatedCode, setTranslatedCode] = useState('');

  const [sourceLang, setSourceLang] = useState('python');

  const [targetLang, setTargetLang] = useState('javascript');

  const [loading, setLoading] = useState(false);

  const handleTranslate = async () => {

    if (!code.trim()) {

      alert("Please enter some code to translate.");

      return;

    }

    setLoading(true);

    setTranslatedCode("");

    try {

      const response = await fetch("http://localhost:5000/translate", {

        method: "POST",

        headers: { "Content-Type": "application/json" },

        body: JSON.stringify({ code, sourceLang, targetLang }),

      });

      const data = await response.json();

      setTranslatedCode(data.translatedCode || "// Translation failed.");

    } catch (error) {

      console.error("Error:", error);

      setTranslatedCode("// Error: Unable to translate.\n// Please check the server.");

    } finally {

      setLoading(false);

    }

  };

  const getLanguageExtension = (lang) => {

    try {

      return loadLanguage(lang);

    } catch (e) {

      return [];

    }

  };

  return (

    <div className="App">

      <header className="App-header">

        <img src={logo} className="App-logo" alt="logo" />

        <div className="translator-container">

          <div className="editor-container">

            <div className="editor-box">

              <h2>Enter Code:</h2>

              <CodeMirror

                value={code}

                height="200px"

                theme={githubLight}

                extensions={[getLanguageExtension(sourceLang)]}

                onChange={(value) => setCode(value)}

              />

            </div>

            <div className="editor-box">

              <h2>Translated Code:</h2>

              <CodeMirror

                value={loading ? "// Translating..." : translatedCode}

                height="200px"

                theme={githubLight}

                extensions={[getLanguageExtension(targetLang)]}

                readOnly

              />

            </div>

          </div>

          <div className="lang-selectors">

            <select

              className="lang-dropdown"

              value={sourceLang}

              onChange={(e) => setSourceLang(e.target.value)}

            >

              <option value="python">Python</option>

              <option value="javascript">JavaScript</option>

              <option value="java">Java</option>

              <option value="c">C</option>

              <option value="php">PHP</option>

              <option value="csharp">C#</option>

            </select>

            <button

              onClick={() => {

                const temp = sourceLang;

                setSourceLang(targetLang);

                setTargetLang(temp);

              }}

              className="swap-button"

              title="Swap Languages"

            >

              🔄

            </button>

            <select

              className="lang-dropdown"

              value={targetLang}

              onChange={(e) => setTargetLang(e.target.value)}

            >

              <option value="python">Python</option>

              <option value="javascript">JavaScript</option>

              <option value="java">Java</option>

              <option value="c">C</option>

              <option value="php">PHP</option>

              <option value="csharp">C#</option>

            </select>

          </div>

          <button

            className="translate-button"

            onClick={handleTranslate}

            disabled={loading}

          >

            {loading ? "Translating..." : "Translate Code"}

          </button>

        </div>

      </header>

    </div>

  );

}

export default App;

app.css

.App {

  text-align: center;

}

.App-logo {

  position: absolute;

  top: 20px;

  left: 20px;

  height: 60px;

  pointer-events: none;

}

.App-header {

  background-color: #000000;

  background-image: url('C:/Users/Kanishka Umapathi/Downloads/AI img.jpg');

  background-size: cover;

  background-repeat: no-repeat;

  background-position: center;

  min-height: 100vh;

  display: flex;

  flex-direction: column;

  align-items: center;

  justify-content: center;

  color: white;

  padding: 2rem;

}

.translator-container {

  background: rgba(255, 255, 255, 0.9);

  padding: 2rem;

  border-radius: 12px;

  box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);

  max-width: 90%;

  width: 1000px;

  color: #000;

  display: flex;

  flex-direction: column;

  align-items: center;

}

.editor-container {

  display: flex;

  gap: 20px;

  justify-content: space-between;

  flex-wrap: wrap;

  width: 100%;

  margin-bottom: 20px;

}

.editor-box {

  flex: 1 1 45%;

  min-width: 300px;

}

.editor-box h2 {

  font-size: 18px;

  margin-bottom: 8px;

}

/\* ✅ Transparent CodeMirror editor with top-left aligned text \*/

.cm-theme {

  background-color: transparent !important;

  border: 1px solid rgba(0, 0, 0, 0.1);

  border-radius: 8px;

  overflow: hidden;

}

.cm-scroller {

  background-color: transparent !important;

  overflow-y: auto;

  min-height: 200px;

}

.cm-content {

  padding: 10px;

  min-height: 200px;

  display: block !important;

  align-items: flex-start !important;

  justify-content: flex-start !important;

  text-align: left;

  white-space: pre;

}

.lang-selectors {

  display: flex;

  justify-content: center;

  align-items: center;

  gap: 10px;

  margin: 15px 0;

}

.lang-dropdown {

  padding: 10px;

  font-size: 14px;

  border-radius: 8px;

  border: 1px solid #ccc;

}

.arrow {

  font-size: 20px;

  font-weight: bold;

}

/\* 🎨 Stylish Translate Button \*/

.translate-button {

  margin-top: 10px;

  padding: 14px 30px;

  background: linear-gradient(135deg, #6e8efb, #a777e3);

  color: white;

  font-size: 16px;

  font-weight: 600;

  border: none;

  border-radius: 40px;

  cursor: pointer;

  transition: all 0.3s ease;

  box-shadow: 0 4px 14px rgba(110, 142, 251, 0.4);

}

.translate-button:hover {

  background: linear-gradient(135deg, #a777e3, #6e8efb);

  transform: scale(1.05);

}

.translate-button:disabled {

  background: #cccccc;

  color: #666;

  cursor: not-allowed;

  box-shadow: none;

}

/\* 🔄 Stylish Swap Button (if you use one in future) \*/

.swap-button {

  background: linear-gradient(135deg, #ff9a9e, #fad0c4);

  border-radius: 50px;

  padding: 10px 14px;

  font-size: 18px;

  font-weight: 600;

  box-shadow: 0 4px 12px rgba(255, 154, 158, 0.4);

  border: none;

  cursor: pointer;

  transition: all 0.3s ease;

}

.swap-button:hover {

  background: linear-gradient(135deg, #fbc2eb, #a6c1ee);

  transform: rotate(180deg);

}

.swap-button:focus {

  outline: none;

  box-shadow: 0 0 0 4px rgba(255, 182, 193, 0.4);

}

Server.js

require('dotenv').config();

const express = require('express');

const axios = require('axios');

const cors = require('cors');

const app = express();

const PORT = process.env.PORT || 5000;

const API\_URL = "https://api.deepinfra.com/v1/openai/chat/completions";

app.use(cors());

app.use(express.json());

app.post('/translate', async (req, res) => {

    const { code, sourceLang, targetLang } = req.body;

    if (!code || !sourceLang || !targetLang) {

        return res.status(400).json({ error: 'Missing required fields' });

    }

    const data = {

        model: "Phind/Phind-CodeLlama-34B-v2",

        messages: [

            { role: "system", content: "You are a helpful assistant." },

            { role: "user", content: `Translate this ${sourceLang} code to ${targetLang}:\n\n${code}` }

        ],

        max\_tokens: 300

    };

    try {

        const response = await axios.post(API\_URL, data, {

            headers: {

                "Authorization": "Bearer Tia27ez7oCAs9vcIrAs8QAkSeXtquYUq"

            }

        });

        const translatedCode = response.data.choices[0].message.content

            .replace(/```.\*?\n/g, "")

            .replace(/```/g, "")

            .trim();

        res.json({ translatedCode });

    } catch (error) {

        console.error("Translation API error:", error.message);

        res.status(500).json({ error: "Translation failed" });

    }

});

app.listen(PORT, () => {

    console.log(`✅ Backend server running on http://localhost:${PORT}`);

});