Npx create-next-app@latest ai-image-detection

Cd ai-image-detection

Npm install axios

Npm install -D tailwindcss postcss autoprefixer

Npx tailwindcss init -p

/\*\* @type {import(‘tailwindcss’).Config} \*/

Module.exports = {

Content: [

“./pages/\*\*/\*.{js,ts,jsx,tsx}”,

“./components/\*\*/\*.{js,ts,jsx,tsx}”,

],

Theme: {

Extend: {},

},

Plugins: [],

}

HUGGINGFACE\_API\_KEY=your\_huggingface\_token\_here

// pages/api/analyze-image.ts

Import axios from ‘axios’;

Export default async function handler(req, res) {

If (req.method !== ‘POST’) {

Return res.status(405).json({ message: ‘Method Not Allowed’ });

}

Const { imageBase64 } = req.body;

Try {

Const response = await axios.post(

‘https://api-inference.huggingface.co/models/google/vit-base-patch16-224’,

imageBase64,

{

Headers: {

Authorization: `Bearer ${process.env.HUGGINGFACE\_API\_KEY}`,

‘Content-Type’: ‘application/octet-stream’,

},

}

);

Res.status(200).json({ result: response.data });

} catch (error) {

Res.status(500).json({ m// pages/ai-image.tsx

Import { useState } from “react”;

Export default function AiImage() {

Const [image, setImage] = useState<File | null>(null);

Const [result, setResult] = useState<any>(null);

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

Const handleFileChange = (e: React.ChangeEvent<HTMLInputElement>) => {

If (e.target.files && e.target.files[0]) {

setImage(e.target.files[0]);

}

};

Const convertToBase64 = (file: File): Promise<string | ArrayBuffer | null> => {

Return new Promise((resolve, reject) => {

Const reader = new FileReader();

Reader.readAsArrayBuffer(file);

Reader.onload = () => resolve(reader.result);

Reader.onerror = (error) => reject(error);

});

};

Const handleAnalyze = async () => {

If (!image) return;

setLoading(true);

const base64Image = await convertToBase64(image);

const response = await fetch(“/api/analyze-image”, {

method: “POST”,

headers: { “Content-Type”: “application/json” },

body: JSON.stringify({ imageBase64: base64Image }),

});

Const data = await response.json();

setResult(data.result);

setLoading(false);

};

Return (

<div className=”min-h-screen bg-gray-100 p-8”>

<h1 className=”text-2xl font-bold mb-4”>AI Image Detector</h1>

<input type=”file” accept=”image/\*” onChange={handleFileChange} />

<button

onClick={handleAnalyze}

className=”mt-4 bg-blue-600 text-white px-4 py-2 rounded hover:bg-blue-700”

>

{loading ? “Analyzing…” : “Analyze Image”}

</button>

{result && (

<div className=”mt-6 bg-white shadow p-4 rounded”>

<h2 className=”text-lg font-semibold mb-2”>Result:</h2>

<ul className=”list-disc ml-6”>

{result.map((item: any, index: number) => (

<li key={index}>

<strong>{item.label}</strong>: {Math.round(item.score \* 100)}%

</li>

))}

</ul>

</div>

)}

</div>

);

}essage: ‘API Error’, error: error.message });

}

}import Link from ‘next/link’;

Export default function Home() {

Return (

<div className=”flex flex-col items-center justify-center min-h-screen p-8 bg-gray-100”>

<h1 className=”text-3xl font-bold mb-4”>Welcome to Smart Farming AI</h1>

<Link href=”/ai-image”>

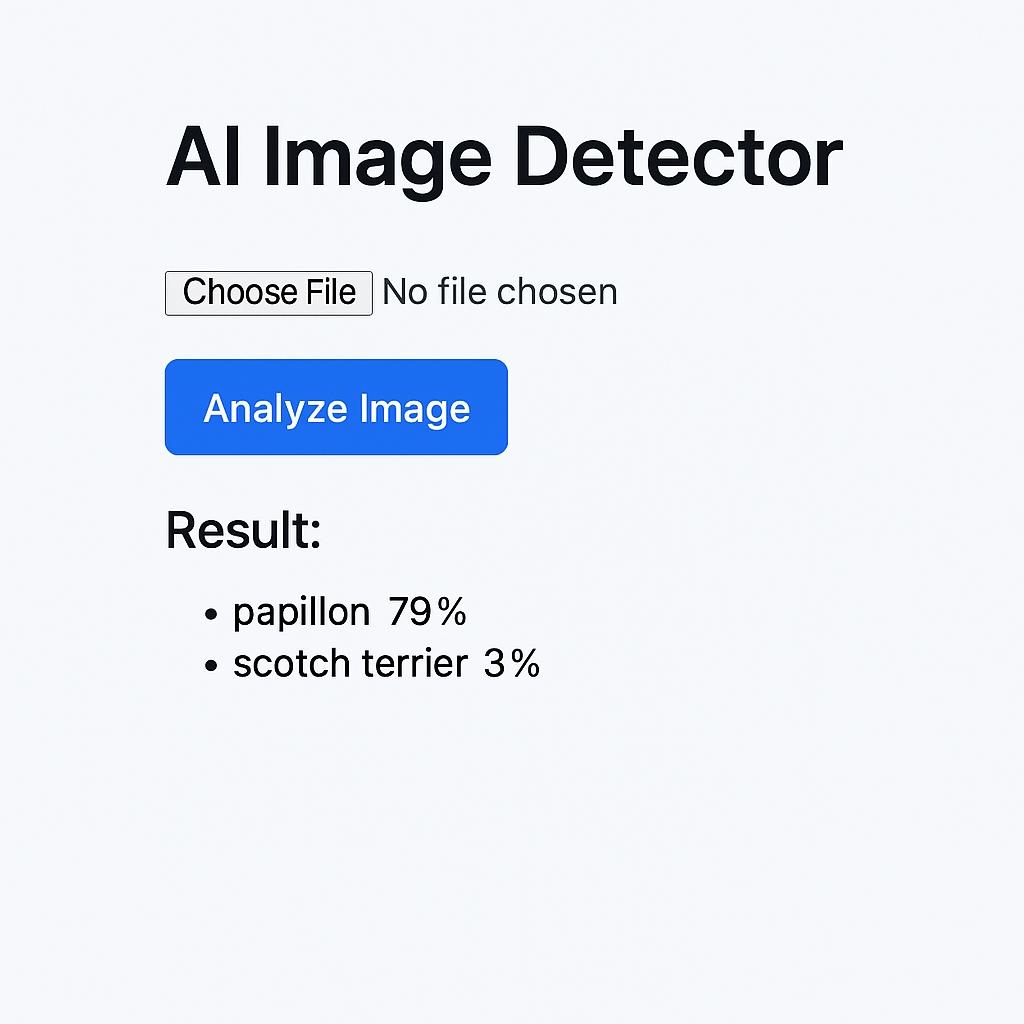
<a className=”text-white bg-green-600 px-4 py-2 rounded hover:bg-green-700”>Go to AI Image Detection</a>

</Link>

</div>

);

}



Github:<https://github.com/ShreenithiE/MVP-internship>

Versel link:<https://mvp-internship-theta.vercel.app/>