#FRONTEND by react

import React, { useState } from 'react';

function App() {

const [image, setImage] = useState(null);

const [description, setDescription] = useState('');

const handleImageUpload = (e) => {

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

};

const handleDescriptionChange = (e) => {

setDescription(e.target.value);

};

const handleSubmit = async () => {

const formData = new FormData();

formData.append('image', image);

formData.append('description', description);

const response = await fetch('/api/upload', {

method: 'POST',

body: formData,

});

const result = await response.json();

// Handle the response with the processed image

};

return (

<div>

<h1>AI Tattoo App</h1>

<input type="file" onChange={handleImageUpload} />

<input type="text" value={description} onChange={handleDescriptionChange} placeholder="Describe your tattoo" />

<button onClick={handleSubmit}>Submit</button>

</div>

);

}

export default App;

#Backend by nodejs with express

const express = require('express');

const multer = require('multer');

const { processImage } = require('./imageProcessor'); // This function will handle image processing

const app = express();

const upload = multer({ dest: 'uploads/' });

app.post('/api/upload', upload.single('image'), async (req, res) => {

const image = req.file;

const description = req.body.description;

try {

const processedImage = await processImage(image.path, description);

// Save processed image to cloud storage and get URL

const imageUrl = await saveToCloud(processedImage);

res.json({ imageUrl });

} catch (error) {

res.status(500).send('Error processing image');

}

});

const saveToCloud = async (file) => {

// Implement cloud storage saving here

};

app.listen(3000, () => {

console.log('Server started on port 3000');

});

#IMAGE PROCESSOR by imageprocessor.js

const { exec } = require('child\_process');

const processImage = (imagePath, description) => {

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

// Use a pre-trained AI model like Pix2Pix to process the image

exec(`python process\_image.py --image ${imagePath} --description "${description}"`, (error, stdout, stderr) => {

if (error) {

reject(error);

} else {

resolve(stdout.trim());

}

});

});

};

module.exports = { processImage };