import React, { useState } from 'react';

import axios from 'axios';

import { OpenAI } from 'openai';

import './App.css';

function App() {

const [naturalLanguage, setNaturalLanguage] = useState('');

const [generatedPrompt, setGeneratedPrompt] = useState('');

const [imageBlobUrl, setImageBlobUrl] = useState('');

const [loading, setLoading] = useState(false); // 加载状态

const [inputError, setInputError] = useState(false);

const openai = new OpenAI({

apiKey: 'sk-n3AVBXLlQq6NXOlgEQT9bhtHEgGFYZ8vc7L45cYds9KBItHI',

dangerouslyAllowBrowser: true,

baseURL: 'https://oneapi.daidr.me/v1'

});

const generateSDPrompt = async () => {

try {

const response = await openai.chat.completions.create({

model: 'gpt-4o-mini',

messages: [{ role: 'system', content: `Convert the following natural language to a Stable Diffusion prompt: ${naturalLanguage}` }],

});

const sdPrompt = response.choices[0].message.content.trim();

setGeneratedPrompt(sdPrompt);

return sdPrompt;

} catch (error) {

console.error('Error generating prompt:', error);

alert('生成 SD prompt 时出现错误');

}

};

const generateImage = async () => {

if (!naturalLanguage.trim()) {

setInputError(true);

return;

}

setInputError(false);

setLoading(true);

const apiUrl = 'http://sd-eb8afe--proxy.fcv3.1279009797310410.cn-hangzhou.fc.devsapp.net/txt2img';

const sdPrompt = await generateSDPrompt();

if (!sdPrompt) {

setLoading(false);

return;

}

const positivePrompt = " ,(full-length portrait: 1.5), (8k, RAW photo, best quality, masterpiece:1.2), (realistic, photo-realistic:1.37), (male:1.3), studio light, white backgrouond, smile，fashion costume design sheet, three views, multi-angle display, clothing design, blueprint"

const finalPrompt = `${sdPrompt}${positivePrompt}`;

const payload = {

prompt: finalPrompt,

steps: 20,

width: 512,

height: 512,

sampler\_name: 'sampler\_v2',

batch\_size: 1,

n\_iter: 1,

send\_images: true,

save\_images: true,

stable\_diffusion\_model: 'chilloutmix\_NiPrunedFp32Fix',

negative\_prompt: "(NSFW:2.0),EasyNegative, paintings, sketches, (worst quality:2), (low quality:2), (normal quality:2), lowres, normal quality, ((monochrome)), ((grayscale)), skin spots, acnes, skin blemishes, age spot, ,extra fingers,fewer fingers, strange fingers, bad hand, fat ass, hole, naked, fat thigh,6 fingers, underwear, nsfw, nude,leg open, fat",

};

try {

const response = await axios.post(apiUrl, payload, {

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

});

if (response.data.ossUrl && response.data.ossUrl.length > 0) {

const imageResponse = await axios.get(response.data.ossUrl[0], { responseType: 'blob' });

const imageBlob = imageResponse.data;

const imageBlobUrl = URL.createObjectURL(imageBlob);

setImageBlobUrl(imageBlobUrl);

} else {

alert('图像生成失败');

}

} catch (error) {

console.error('Error generating image:', error);

alert('生成图像时出现错误');

} finally {

setLoading(false);

}

};

const downloadImage = () => {

const link = document.createElement('a');

link.href = imageBlobUrl;

link.download = 'generated\_image.png';

document.body.appendChild(link);

link.click();

document.body.removeChild(link);

};

return (

<div className="app-container">

<div className="input-container">

<input

type="text"

value={naturalLanguage}

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

placeholder="请输入自然语言描述"

className="input-box"

/>

{inputError && <p style={{color: 'red'}}>输入框不能为空！</p>}

<button onClick={generateImage} className="generate-button">

{loading ? '生成中...' : '生成图像'}

</button>

<button className="generate-button">

<a href="http://sd-eb8afe--sd.fcv3.1279009797310410.cn-hangzhou.fc.devsapp.net" target="\_blank"

rel="noopener noreferrer">

跳转至专业版

</a>

</button>

</div>

{loading && (

<div className="loading-message">

<h3>请稍候，正在生成图像...</h3>

</div>

)}

{generatedPrompt && (

<div className="prompt-container">

<h3>生成的 SD Prompt:</h3>

<p>{generatedPrompt}</p>

</div>

)}

{imageBlobUrl && (

<div className="image-container">

<h2>生成的图像：</h2>

<img src={imageBlobUrl} alt="Generated" className="generated-image" />

<br />

<button onClick={downloadImage} className="download-button">

下载图像

</button>

</div>

)}

</div>

);

}

export default App;

import React, { useState } from 'react';

import axios from 'axios';

import { OpenAI } from 'openai';

import './App.css';

function App() {

const [naturalLanguage, setNaturalLanguage] = useState('');

const [generatedPrompt, setGeneratedPrompt] = useState('');

const [imageBlobUrl, setImageBlobUrl] = useState('');

const [loading, setLoading] = useState(false); // 加载状态

const [inputError, setInputError] = useState(false);

const openai = new OpenAI({

apiKey: 'sk-n3AVBXLlQq6NXOlgEQT9bhtHEgGFYZ8vc7L45cYds9KBItHI',

dangerouslyAllowBrowser: true,

baseURL: 'https://oneapi.daidr.me/v1'

});

const generateSDPrompt = async () => {

try {

const response = await openai.chat.completions.create({

model: 'gpt-4o-mini',

messages: [{ role: 'system', content: `Convert the following natural language to a Stable Diffusion prompt: ${naturalLanguage}` }],

});

const sdPrompt = response.choices[0].message.content.trim();

setGeneratedPrompt(sdPrompt);

return sdPrompt;

} catch (error) {

console.error('Error generating prompt:', error);

alert('生成 SD prompt 时出现错误');

}

};

const generateImage = async () => {

if (!naturalLanguage.trim()) {

setInputError(true);

return;

}

setInputError(false);

setLoading(true);

const apiUrl = 'http://sd-eb8afe--proxy.fcv3.1279009797310410.cn-hangzhou.fc.devsapp.net/txt2img';

const sdPrompt = await generateSDPrompt();

if (!sdPrompt) {

setLoading(false);

return;

}

const positivePrompt = " ,(full-length portrait: 1.5), (8k, RAW photo, best quality, masterpiece:1.2), (realistic, photo-realistic:1.37), (male:1.3), studio light, white backgrouond, smile，fashion costume design sheet, three views, multi-angle display, clothing design, blueprint"

const finalPrompt = `${sdPrompt}${positivePrompt}`;

const payload = {

prompt: finalPrompt,

steps: 20,

width: 512,

height: 512,

sampler\_name: 'sampler\_v2',

batch\_size: 1,

n\_iter: 1,

send\_images: true,

save\_images: true,

stable\_diffusion\_model: 'chilloutmix\_NiPrunedFp32Fix',

negative\_prompt: "(NSFW:2.0),EasyNegative, paintings, sketches, (worst quality:2), (low quality:2), (normal quality:2), lowres, normal quality, ((monochrome)), ((grayscale)), skin spots, acnes, skin blemishes, age spot, ,extra fingers,fewer fingers, strange fingers, bad hand, fat ass, hole, naked, fat thigh,6 fingers, underwear, nsfw, nude,leg open, fat",

};

try {

const response = await axios.post(apiUrl, payload, {

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

});

if (response.data.ossUrl && response.data.ossUrl.length > 0) {

const imageResponse = await axios.get(response.data.ossUrl[0], { responseType: 'blob' });

const imageBlob = imageResponse.data;

const imageBlobUrl = URL.createObjectURL(imageBlob);

setImageBlobUrl(imageBlobUrl);

} else {

alert('图像生成失败');

}

} catch (error) {

console.error('Error generating image:', error);

alert('生成图像时出现错误');

} finally {

setLoading(false);

}

};

const downloadImage = () => {

const link = document.createElement('a');

link.href = imageBlobUrl;

link.download = 'generated\_image.png';

document.body.appendChild(link);

link.click();

document.body.removeChild(link);

};

return (

<div className="app-container">

<div className="input-container">

<input

type="text"

value={naturalLanguage}

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

placeholder="请输入自然语言描述"

className="input-box"

/>

{inputError && <p style={{color: 'red'}}>输入框不能为空！</p>}

<button onClick={generateImage} className="generate-button">

{loading ? '生成中...' : '生成图像'}

</button>

<button className="generate-button">

<a href="http://sd-eb8afe--sd.fcv3.1279009797310410.cn-hangzhou.fc.devsapp.net" target="\_blank"

rel="noopener noreferrer">

跳转至专业版

</a>

</button>

</div>

{loading && (

<div className="loading-message">

<h3>请稍候，正在生成图像...</h3>

</div>

)}

{generatedPrompt && (

<div className="prompt-container">

<h3>生成的 SD Prompt:</h3>

<p>{generatedPrompt}</p>

</div>

)}

{imageBlobUrl && (

<div className="image-container">

<h2>生成的图像：</h2>

<img src={imageBlobUrl} alt="Generated" className="generated-image" />

<br />

<button onClick={downloadImage} className="download-button">

下载图像

</button>

</div>

)}

</div>

);

}

export default App;

import React, { useState, useRef, useContext, createContext } from 'react';

import ReactDOM from 'react-dom';

import { BrowserRouter as Router, Route, Routes, useNavigate } from 'react-router-dom';

import './styles.css'; // 样式文件

// AuthContext

const AuthContext = createContext();

const AuthProvider = ({ children }) => {

const [user, setUser] = useState(null);

const login = (username) => {

setUser(username);

};

const logout = () => {

setUser(null);

};

return (

<AuthContext.Provider value={{ user, login, logout }}>

{children}

</AuthContext.Provider>

);

};

// Header Component

const Header = () => {

const { user, logout } = useContext(AuthContext);

const navigate = useNavigate();

return (

<header>

<h1>AI Art Generator</h1>

{user ? (

<div>

<span>{user}</span>

<button onClick={() => { logout(); navigate('/'); }}>Logout</button>

</div>

) : (

<h2>Please log in</h2>

)}

</header>

);

};

// Login Component

const Login = () => {

const { login } = useContext(AuthContext);

const [username, setUsername] = useState('');

const navigate = useNavigate();

const handleSubmit = (e) => {

e.preventDefault();

login(username);

navigate('/canvas');

};

return (

<form onSubmit={handleSubmit}>

<input

type="text"

placeholder="Username"

value={username}

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

/>

<button type="submit">Login</button>

</form>

);

};

// Canvas Component

const Canvas = () => {

const canvasRef = useRef(null);

const [drawing, setDrawing] = useState(false);

const [images, setImages] = useState([]);

const [params, setParams] = useState({ style: 'realistic', size: 'medium' });

const startDrawing = (e) => {

setDrawing(true);

const ctx = canvasRef.current.getContext('2d');

ctx.moveTo(e.nativeEvent.offsetX, e.nativeEvent.offsetY);

};

const draw = (e) => {

if (!drawing) return;

const ctx = canvasRef.current.getContext('2d');

ctx.lineTo(e.nativeEvent.offsetX, e.nativeEvent.offsetY);

ctx.stroke();

};

const stopDrawing = () => {

setDrawing(false);

};

const generateImage = async () => {

const newImage = `Image with style ${params.style} and size ${params.size}`;

setImages([...images, newImage]);

alert(newImage);

};

const handleParamChange = (e) => {

setParams({ ...params, [e.target.name]: e.target.value });

};

return (

<div>

<canvas

ref={canvasRef}

width={500}

height={500}

style={{ border: '1px solid black' }}

onMouseDown={startDrawing}

onMouseMove={draw}

onMouseUp={stopDrawing}

onMouseLeave={stopDrawing}

/>

<div>

<button onClick={generateImage}>Generate Image</button>

</div>

<div>

<label>

Style:

<select name="style" value={params.style} onChange={handleParamChange}>

<option value="realistic">Realistic</option>

<option value="cartoon">Cartoon</option>

<option value="abstract">Abstract</option>

</select>

</label>

<label>

Size:

<select name="size" value={params.size} onChange={handleParamChange}>

<option value="small">Small</option>

<option value="medium">Medium</option>

<option value="large">Large</option>

</select>

</label>

</div>

<div>

<h3>Generated Images:</h3>

<ul>

{images.map((img, index) => (

<li key={index}>{img}</li>

))}

</ul>

</div>

</div>

);

};

// Main App Component

const App = () => {

return (

<AuthProvider>

<Router>

<Header />

<Routes>

<Route path="/" element={<Login />} />

<Route path="/canvas" element={<Canvas />} />

</Routes>

</Router>

</AuthProvider>

);

};

// Render

ReactDOM.render(<App />, document.getElementById('root'));

// 样式文件 styles.css

const styles = `

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background-color: #f4f4f4;

}

header {

background: #35424a;

color: #ffffff;

padding: 10px 20px;

text-align: center;

}

h1 {

margin: 0;

}

form {

display: flex;

justify-content: center;

margin-top: 20px;

}

input {

padding: 10px;

margin-right: 10px;

}

button {

padding: 10px;

cursor: pointer;

}

canvas {

display: block;

margin: 20px auto;

}

div {

text-align: center;

}

`;

const styleSheet = document.createElement("style");

styleSheet.type = "text/css";

styleSheet.innerText = styles;

document.head.appendChild(styleSheet);

from flask import Flask, request, jsonify

from transformers import GPT2LMHeadModel, GPT2Tokenizer

import torch

import os

app = Flask(\_\_name\_\_)

class PromptGenerator:

def \_\_init\_\_(self, model\_name='gpt2'):

self.tokenizer = GPT2Tokenizer.from\_pretrained(model\_name)

self.model = GPT2LMHeadModel.from\_pretrained(model\_name)

self.model.eval()

def generate\_prompt(self, input\_text):

inputs = self.tokenizer.encode(input\_text, return\_tensors='pt')

with torch.no\_grad():

outputs = self.model.generate(

inputs,

max\_length=50,

num\_return\_sequences=1,

no\_repeat\_ngram\_size=2,

early\_stopping=True

)

prompt = self.tokenizer.decode(outputs[0], skip\_special\_tokens=True)

return prompt

generator = PromptGenerator()

history = []

@app.route('/api/generate-prompt', methods=['POST'])

def generate\_prompt():

data = request.json

if 'text' not in data or not data['text'].strip():

return jsonify({'error': 'Input text is required.'}), 400

prompt = generator.generate\_prompt(data['text'])

history.append(prompt)

return jsonify({'prompt': prompt})

@app.route('/api/history', methods=['GET'])

def get\_history():

return jsonify(history)

@app.route('/api/clear-history', methods=['POST'])

def clear\_history():

global history

history = []

return jsonify({'message': 'History cleared.'})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(port=5000)

import React, { useState, useEffect } from 'react';

import ReactDOM from 'react-dom';

import { BrowserRouter as Router, Route, Routes, useNavigate } from 'react-router-dom';

import './styles.css';

const Header = () => (

<header>

<h1>自然语言转换为提示词</h1>

</header>

);

const PromptInput = () => {

const [text, setText] = useState('');

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

const [error, setError] = useState('');

const navigate = useNavigate();

const handleSubmit = async (e) => {

e.preventDefault();

if (!text.trim()) {

setError('输入不能为空');

return;

}

setLoading(true);

setError('');

try {

const response = await fetch('http://localhost:5000/api/generate-prompt', {

method: 'POST',

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

body: JSON.stringify({ text }),

});

if (!response.ok) {

const data = await response.json();

throw new Error(data.error || '生成提示词失败');

}

const data = await response.json();

localStorage.setItem('generatedPrompt', data.prompt);

navigate('/generated');

} catch (err) {

setError(err.message);

} finally {

setLoading(false);

}

};

return (

<form onSubmit={handleSubmit}>

<textarea

rows="4"

placeholder="请输入描述..."

value={text}

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

/>

{loading ? (

<button disabled>正在生成...</button>

) : (

<button type="submit">生成提示词</button>

)}

{error && <p className="error">{error}</p>}

</form>

);

};

const GeneratedPrompt = () => {

const [prompt, setPrompt] = useState('');

const [history, setHistory] = useState([]);

useEffect(() => {

setPrompt(localStorage.getItem('generatedPrompt'));

fetchHistory();

}, []);

const fetchHistory = async () => {

const response = await fetch('http://localhost:5000/api/history');

const data = await response.json();

setHistory(data);

};

const clearHistory = async () => {

await fetch('http://localhost:5000/api/clear-history', { method: 'POST' });

setHistory([]);

};

return (

<div>

<h2>生成的提示词：</h2>

<p>{prompt || '没有生成提示词，请返回输入页面。'}</p>

<h3>历史记录：</h3>

<ul>

{history.length > 0 ? (

history.map((item, index) => <li key={index}>{item}</li>)

) : (

<li>没有历史记录</li>

)}

</ul>

<button onClick={clearHistory}>清空历史</button>

<button onClick={() => { localStorage.removeItem('generatedPrompt'); window.location.href = '/'; }}>

返回

</button>

</div>

);

};

const App = () => (

<Router>

<Header />

<Routes>

<Route path="/" element={<PromptInput />} />

<Route path="/generated" element={<GeneratedPrompt />} />

</Routes>

</Router>

);

ReactDOM.render(<App />, document.getElementById('root'));

// 内联样式

const styles = `

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

margin: 0;

padding: 20px;

}

header {

background: #35424a;

color: #ffffff;

padding: 10px;

text-align: center;

}

h1 {

margin: 0;

}

form {

display: flex;

flex-direction: column;

align-items: center;

margin-top: 20px;

}

textarea {

width: 100%;

max-width: 600px;

padding: 10px;

margin-bottom: 10px;

}

button {

padding: 10px;

cursor: pointer;

margin: 5px;

}

.error {

color: red;

}

div {

text-align: center;

margin-top: 20px;

}

`;

const styleSheet = document.createElement("style");

styleSheet.type = "text/css";

styleSheet.innerText = styles;

document.head.appendChild(styleSheet);

import os

import numpy as np

import torch

import torch.nn as nn

import torchvision.transforms as transforms

from torchvision.datasets import ImageFolder

from torch.utils.data import DataLoader, random\_split

import matplotlib.pyplot as plt

from torch.utils.tensorboard import SummaryWriter

import json

# 超参数配置

def load\_config(config\_file):

with open(config\_file, 'r') as f:

config = json.load(f)

return config

# 数据准备模块

class DataLoaderModule:

def \_\_init\_\_(self, data\_dir, batch\_size=16, val\_split=0.2):

self.data\_dir = data\_dir

self.batch\_size = batch\_size

self.val\_split = val\_split

self.transform = transforms.Compose([

transforms.Resize((256, 256)),

transforms.RandomHorizontalFlip(),

transforms.RandomRotation(10),

transforms.ToTensor(),

])

self.train\_loader, self.val\_loader = self.get\_dataloaders()

def get\_dataloaders(self):

dataset = ImageFolder(root=self.data\_dir, transform=self.transform)

val\_size = int(len(dataset) \* self.val\_split)

train\_size = len(dataset) - val\_size

train\_dataset, val\_dataset = random\_split(dataset, [train\_size, val\_size])

train\_loader = DataLoader(train\_dataset, batch\_size=self.batch\_size, shuffle=True)

val\_loader = DataLoader(val\_dataset, batch\_size=self.batch\_size, shuffle=False)

return train\_loader, val\_loader

# UNet 模型定义

class UNet(nn.Module):

def \_\_init\_\_(self, in\_channels, out\_channels):

super(UNet, self).\_\_init\_\_()

self.encoder = nn.Sequential(

nn.Conv2d(in\_channels, 64, kernel\_size=3, padding=1),

nn.ReLU(),

nn.MaxPool2d(2),

nn.Conv2d(64, 128, kernel\_size=3, padding=1),

nn.ReLU(),

nn.MaxPool2d(2)

)

self.middle = nn.Sequential(

nn.Conv2d(128, 256, kernel\_size=3, padding=1),

nn.ReLU(),

nn.Conv2d(256, 256, kernel\_size=3, padding=1),

nn.ReLU()

)

self.decoder = nn.Sequential(

nn.ConvTranspose2d(256, 128, kernel\_size=2, stride=2),

nn.ReLU(),

nn.ConvTranspose2d(128, 64, kernel\_size=2, stride=2),

nn.ReLU(),

nn.Conv2d(64, out\_channels, kernel\_size=3, padding=1),

nn.Sigmoid()

)

def forward(self, x):

enc = self.encoder(x)

mid = self.middle(enc)

dec = self.decoder(mid)

return dec

# 扩散模型定义

class DiffusionModel(nn.Module):

def \_\_init\_\_(self):

super(DiffusionModel, self).\_\_init\_\_()

self.unet = UNet(in\_channels=3, out\_channels=3)

def forward(self, x):

return self.unet(x)

# 扩散过程

def diffusion\_process(x\_0, t, noise):

return x\_0 + noise \* (t \*\* 0.5)

# 反向扩散过程

def reverse\_diffusion(model, x\_t):

return model(x\_t)

# 训练模块

class Trainer:

def \_\_init\_\_(self, model, train\_loader, val\_loader, optimizer, num\_epochs=10, log\_dir='logs'):

self.model = model

self.train\_loader = train\_loader

self.val\_loader = val\_loader

self.optimizer = optimizer

self.num\_epochs = num\_epochs

self.writer = SummaryWriter(log\_dir)

def train(self):

self.model.train()

for epoch in range(self.num\_epochs):

for images, \_ in self.train\_loader:

self.optimizer.zero\_grad()

t = torch.randint(1, 1000, (images.size(0),)) # 随机时间步

noise = torch.randn\_like(images)

x\_t = diffusion\_process(images, t, noise)

output = reverse\_diffusion(self.model, x\_t)

loss = self.custom\_loss(output, images)

loss.backward()

self.optimizer.step()

# Logging

self.writer.add\_scalar('Loss/train', loss.item(), epoch)

print(f"Epoch [{epoch+1}/{self.num\_epochs}], Loss: {loss.item():.4f}")

# 每个 epoch 结束后验证

self.validate(epoch)

self.writer.close()

def validate(self, epoch):

self.model.eval()

total\_loss = 0

with torch.no\_grad():

for images, \_ in self.val\_loader:

t = torch.randint(1, 1000, (images.size(0),))

noise = torch.randn\_like(images)

x\_t = diffusion\_process(images, t, noise)

output = reverse\_diffusion(self.model, x\_t)

loss = self.custom\_loss(output, images)

total\_loss += loss.item()

avg\_loss = total\_loss / len(self.val\_loader)

self.writer.add\_scalar('Loss/val', avg\_loss, epoch)

print(f"Validation Loss after epoch {epoch+1}: {avg\_loss:.4f}")

def custom\_loss(self, output, target):

return ((output - target) \*\* 2).mean() # 均方误差损失

# 学习率调度器

def get\_scheduler(optimizer, config):

return torch.optim.lr\_scheduler.StepLR(optimizer, step\_size=config['step\_size'], gamma=config['gamma'])

# 模型保存与加载

def save\_model(model, path):

torch.save(model.state\_dict(), path)

def load\_model(model, path):

model.load\_state\_dict(torch.load(path))

model.eval()

# 图像生成模块

def generate\_image(model, shape):

model.eval()

with torch.no\_grad():

x\_0 = torch.randn(shape) # 随机噪声

for t in reversed(range(1, 1001)):

noise = torch.randn\_like(x\_0)

x\_0 = diffusion\_process(x\_0, t, noise)

x\_0 = reverse\_diffusion(model, x\_0)

return x\_0

# 结果可视化

def visualize\_image(tensor):

image = tensor.squeeze().permute(1, 2, 0).numpy()

plt.imshow(image)

plt.axis('off')

plt.show()

# 测试与评估

def evaluate\_model(model, dataloader):

model.eval()

total\_loss = 0

with torch.no\_grad():

for images, \_ in dataloader:

t = torch.randint(1, 1000, (images.size(0),))

noise = torch.randn\_like(images)

x\_t = diffusion\_process(images, t, noise)

output = reverse\_diffusion(model, x\_t)

loss = ((output - images) \*\* 2).mean()

total\_loss += loss.item()

avg\_loss = total\_loss / len(dataloader)

print(f"Average Loss: {avg\_loss:.4f}")

if \_\_name\_\_ == "\_\_main\_\_":

config = load\_config('config.json') # 加载配置文件

data\_loader\_module = DataLoaderModule(data\_dir=config['data\_dir'], batch\_size=config['batch\_size'], val\_split=config['val\_split'])

train\_loader, val\_loader = data\_loader\_module.train\_loader, data\_loader\_module.val\_loader

model = DiffusionModel()

optimizer = torch.optim.Adam(model.parameters(), lr=config['learning\_rate'])

scheduler = get\_scheduler(optimizer, config)

trainer = Trainer(model=model, train\_loader=train\_loader, val\_loader=val\_loader, optimizer=optimizer, num\_epochs=config['num\_epochs'])

trainer.train()

save\_model(model, 'diffusion\_model.pth')

load\_model(model, 'diffusion\_model.pth')

generated\_image = generate\_image(model, (1, 3, 256, 256))

visualize\_image(generated\_image)

evaluate\_model(model, val\_loader)

{

"name": "sd-gpt",

"version": "0.1.0",

"private": true,

"dependencies": {

"@testing-library/jest-dom": "^5.17.0",

"@testing-library/react": "^13.4.0",

"@testing-library/user-event": "^13.5.0",

"axios": "^1.7.7",

"openai": "^4.67.1",

"react": "^18.3.1",

"react-dom": "^18.3.1",

"react-scripts": "5.0.1",

"web-vitals": "^2.1.4"

},

"scripts": {

"start": "react-scripts start",

"build": "react-scripts build",

"test": "react-scripts test",

"eject": "react-scripts eject"

},

"eslintConfig": {

"extends": [

"react-app",

"react-app/jest"

]

},

"browserslist": {

"production": [

">0.2%",

"not dead",

"not op\_mini all"

],

"development": [

"last 1 chrome version",

"last 1 firefox version",

"last 1 safari version"

]

}

}

import os

import json

import numpy as np

import torch

from torch.utils.data import Dataset, DataLoader

from transformers import (

GPT2Tokenizer,

GPT2LMHeadModel,

Trainer,

TrainingArguments,

DataCollatorForLanguageModeling,

get\_linear\_schedule\_with\_warmup

)

from datasets import load\_dataset, load\_metric

from sklearn.model\_selection import train\_test\_split

import logging

import wandb

import optuna

import random

import matplotlib.pyplot as plt

wandb.init(project="gpt2-finetune")

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

logging.basicConfig(level=logging.INFO)

logger = logging.getLogger(\_\_name\_\_)

class CustomDataset(Dataset):

def \_\_init\_\_(self, texts, tokenizer, block\_size=128):

self.tokenizer = tokenizer

self.block\_size = block\_size

self.examples = []

for text in texts:

tokenized\_text = tokenizer.encode(text)

for i in range(

0,

len(tokenized\_text) - block\_size + 1,

block\_size

):

self.examples.append(tokenized\_text[i:i + block\_size])

def \_\_len\_\_(self):

return len(self.examples)

def \_\_getitem\_\_(self, idx):

return torch.tensor(self.examples[idx])

def load\_and\_prepare\_data(file\_path, tokenizer):

with open(file\_path, 'r', encoding='utf-8') as f:

text = f.read().splitlines()

text = [

line.strip()

for line in text

if line.strip()

]

augmented\_text = augment\_data(text)

train\_texts, val\_texts = train\_test\_split(

augmented\_text,

test\_size=0.1,

random\_state=42

)

train\_dataset = CustomDataset(train\_texts, tokenizer)

val\_dataset = CustomDataset(val\_texts, tokenizer)

return train\_dataset, val\_dataset

def augment\_data(texts):

synonyms = {

"happy": ["joyful", "cheerful"],

"sad": ["unhappy", "sorrowful"],

"fast": ["quick", "rapid"],

"slow": ["leisurely", "unhurried"]

}

augmented\_texts = []

for text in texts:

words = text.split()

new\_text = []

for word in words:

if word in synonyms and random.random() > 0.5:

new\_text.append(random.choice(synonyms[word]))

else:

new\_text.append(word)

augmented\_texts.append(" ".join(new\_text))

return augmented\_texts

model\_name = "gpt2"

tokenizer = GPT2Tokenizer.from\_pretrained(model\_name)

model = GPT2LMHeadModel.from\_pretrained(model\_name).to(device)

train\_file = 'path\_to\_your\_training\_file.txt'

train\_dataset, val\_dataset = load\_and\_prepare\_data(train\_file, tokenizer)

def objective(trial):

training\_args = TrainingArguments(

output\_dir='./results',

overwrite\_output\_dir=True,

num\_train\_epochs=trial.suggest\_int("num\_train\_epochs", 2, 5),

per\_device\_train\_batch\_size=trial.suggest\_int("batch\_size", 2, 8),

save\_steps=10\_000,

save\_total\_limit=2,

evaluation\_strategy="steps",

logging\_dir='./logs',

logging\_steps=500,

learning\_rate=trial.suggest\_loguniform("learning\_rate", 1e-5, 5e-4),

weight\_decay=trial.suggest\_float("weight\_decay", 0.0, 0.3),

load\_best\_model\_at\_end=True

)

trainer = Trainer(

model=model,

args=training\_args,

data\_collator=DataCollatorForLanguageModeling(

tokenizer=tokenizer,

mlm=False

),

train\_dataset=train\_dataset,

eval\_dataset=val\_dataset

)

trainer.train()

eval\_result = trainer.evaluate()

return eval\_result['eval\_loss']

study = optuna.create\_study(direction="minimize")

study.optimize(objective, n\_trials=10)

print("Best hyperparameters:", study.best\_params)

model.save\_pretrained('./gpt2-finetuned')

tokenizer.save\_pretrained('./gpt2-finetuned')

def evaluate\_model(model, tokenizer, text):

inputs = tokenizer.encode(text, return\_tensors='pt').to(device)

outputs = model.generate(inputs, max\_length=100)

return tokenizer.decode(outputs[0], skip\_special\_tokens=True)

sample\_text = "Once upon a time"

output = evaluate\_model(model, tokenizer, sample\_text)

print("Generated Text:", output)

accuracy\_metric = load\_metric("accuracy")

bleu\_metric = load\_metric("bleu")

rouge\_metric = load\_metric("rouge")

f1\_metric = load\_metric("f1")

def compute\_metrics(eval\_pred):

predictions, labels = eval\_pred

predictions = np.argmax(predictions, axis=-1)

accuracy = accuracy\_metric.compute(

predictions=predictions,

references=labels

)

bleu = bleu\_metric.compute(

predictions=predictions,

references=labels

)

rouge = rouge\_metric.compute(

predictions=predictions,

references=labels

)

f1 = f1\_metric.compute(

predictions=predictions,

references=labels

)

return {

'accuracy': accuracy['accuracy'],

'bleu': bleu['bleu'],

'rouge': rouge,

'f1': f1['f1']

}

class WandbCallback:

def on\_train\_end(self, logs=None):

wandb.finish()

trainer = Trainer(

model=model,

args=training\_args,

data\_collator=DataCollatorForLanguageModeling(

tokenizer=tokenizer,

mlm=False

),

train\_dataset=train\_dataset,

eval\_dataset=val\_dataset

)

trainer.add\_callback(WandbCallback())

trainer.train()

logger.info("Training completed and model saved.")

def save\_config(config, file\_path):

with open(file\_path, 'w') as f:

json.dump(config, f, indent=4)

config = {

"model\_name": model\_name,

"num\_train\_epochs": study.best\_params["num\_train\_epochs"],

"batch\_size": study.best\_params["batch\_size"],

"learning\_rate": study.best\_params["learning\_rate"],

"weight\_decay": study.best\_params["weight\_decay"]

}

save\_config(config, './config.json')

def load\_config(file\_path):

with open(file\_path, 'r') as f:

return json.load(f)

def plot\_metrics(training\_loss, eval\_metrics):

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)

plt.plot(training\_loss, label='Training Loss')

plt.title('Training Loss')

plt.xlabel('Steps')

plt.ylabel('Loss')

plt.legend()

plt.subplot(1, 2, 2)

plt.plot(eval\_metrics['accuracy'], label='Accuracy')

plt.plot(eval\_metrics['bleu'], label='BLEU')

plt.title('Evaluation Metrics')

plt.xlabel('Epochs')

plt.ylabel('Score')

plt.legend()

plt.tight\_layout()

plt.savefig('metrics.png')

plt.show()

def compare\_models(model\_configs):

results = {}

for config in model\_configs:

model = GPT2LMHeadModel.from\_pretrained(

config['model\_name']

).to(device)

training\_args = TrainingArguments(

output\_dir='./results',

overwrite\_output\_dir=True,

num\_train\_epochs=config['num\_train\_epochs'],

per\_device\_train\_batch\_size=config['batch\_size'],

evaluation\_strategy="epoch",

logging\_dir='./logs',

logging\_steps=500,

save\_total\_limit=2

)

trainer = Trainer(

model=model,

args=training\_args,

data\_collator=DataCollatorForLanguageModeling(

tokenizer=tokenizer,

mlm=False

),

train\_dataset=train\_dataset,

eval\_dataset=val\_dataset

)

trainer.train()

eval\_result = trainer.evaluate()

results[config['model\_name']] = eval\_result['eval\_loss']

return results

model\_configs = [

{

"model\_name": "gpt2",

"num\_train\_epochs": 3,

"batch\_size": 4

},

{

"model\_name": "gpt2-medium",

"num\_train\_epochs": 3,

"batch\_size": 4

}

]

comparison\_results = compare\_models(model\_configs)

print("Model Comparison Results:", comparison\_results)

def generate\_adversarial\_samples(model, tokenizer, text):

inputs = tokenizer.encode(text, return\_tensors='pt').to(device)

outputs = model.generate(

inputs,

max\_length=100,

do\_sample=True,

top\_k=50,

top\_p=0.95,

num\_return\_sequences=5

)

return [

tokenizer.decode(output, skip\_special\_tokens=True)

for output in outputs

]

adversarial\_samples = generate\_adversarial\_samples(

model,

tokenizer,

"Once upon a time"

)

function disableLogs() {

{

if (disabledDepth === 0) {

/\* eslint-disable react-internal/no-production-logging \*/

prevLog = console.log;

prevInfo = console.info;

prevWarn = console.warn;

prevError = console.error;

prevGroup = console.group;

prevGroupCollapsed = console.groupCollapsed;

prevGroupEnd = console.groupEnd; // https://github.com/facebook/react/issues/19099

var props = {

configurable: true,

enumerable: true,

value: disabledLog,

writable: true

}; // $FlowFixMe Flow thinks console is immutable.

Object.defineProperties(console, {

info: props,

log: props,

warn: props,

error: props,

group: props,

groupCollapsed: props,

groupEnd: props

});

/\* eslint-enable react-internal/no-production-logging \*/

}

disabledDepth++;

}

}

function reenableLogs() {

{

disabledDepth--;

if (disabledDepth === 0) {

/\* eslint-disable react-internal/no-production-logging \*/

var props = {

configurable: true,

enumerable: true,

writable: true

}; // $FlowFixMe Flow thinks console is immutable.

Object.defineProperties(console, {

log: assign({}, props, {

value: prevLog

}),

info: assign({}, props, {

value: prevInfo

}),

warn: assign({}, props, {

value: prevWarn

}),

error: assign({}, props, {

value: prevError

}),

group: assign({}, props, {

value: prevGroup

}),

groupCollapsed: assign({}, props, {

value: prevGroupCollapsed

}),

groupEnd: assign({}, props, {

value: prevGroupEnd

})

});

/\* eslint-enable react-internal/no-production-logging \*/

}

if (disabledDepth < 0) {

error('disabledDepth fell below zero. ' + 'This is a bug in React. Please file an issue.');

}

}

}

var ReactCurrentDispatcher$1 = ReactSharedInternals.ReactCurrentDispatcher;

var prefix;

function describeBuiltInComponentFrame(name, source, ownerFn) {

{

if (prefix === undefined) {

// Extract the VM specific prefix used by each line.

try {

throw Error();

} catch (x) {

var match = x.stack.trim().match(/\n( \*(at )?)/);

prefix = match && match[1] || '';

}

} // We use the prefix to ensure our stacks line up with native stack frames.

return '\n' + prefix + name;

}

}

var reentry = false;

var componentFrameCache;

{

var PossiblyWeakMap = typeof WeakMap === 'function' ? WeakMap : Map;

componentFrameCache = new PossiblyWeakMap();

}

function describeNativeComponentFrame(fn, construct) {

// If something asked for a stack inside a fake render, it should get ignored.

if ( !fn || reentry) {

return '';

}

{

var frame = componentFrameCache.get(fn);

if (frame !== undefined) {

return frame;

}

}

var control;

reentry = true;

var previousPrepareStackTrace = Error.prepareStackTrace; // $FlowFixMe It does accept undefined.

Error.prepareStackTrace = undefined;

var previousDispatcher;

{

previousDispatcher = ReactCurrentDispatcher$1.current; // Set the dispatcher in DEV because this might be call in the render function

// for warnings.

ReactCurrentDispatcher$1.current = null;

disableLogs();

}

try {

// This should throw.

if (construct) {

// Something should be setting the props in the constructor.

var Fake = function () {

throw Error();

}; // $FlowFixMe

Object.defineProperty(Fake.prototype, 'props', {

set: function () {

// We use a throwing setter instead of frozen or non-writable props

// because that won't throw in a non-strict mode function.

throw Error();

}

});

if (typeof Reflect === 'object' && Reflect.construct) {

// We construct a different control for this case to include any extra

// frames added by the construct call.

try {

Reflect.construct(Fake, []);

} catch (x) {

control = x;

}

Reflect.construct(fn, [], Fake);

} else {

try {

Fake.call();

} catch (x) {

control = x;

}

fn.call(Fake.prototype);

}

} else {

try {

throw Error();

} catch (x) {

control = x;

}

fn();

}

} catch (sample) {

// This is inlined manually because closure doesn't do it for us.

if (sample && control && typeof sample.stack === 'string') {

// This extracts the first frame from the sample that isn't also in the control.

// Skipping one frame that we assume is the frame that calls the two.

var sampleLines = sample.stack.split('\n');

var controlLines = control.stack.split('\n');

var s = sampleLines.length - 1;

var c = controlLines.length - 1;

while (s >= 1 && c >= 0 && sampleLines[s] !== controlLines[c]) {

// We expect at least one stack frame to be shared.

// Typically this will be the root most one. However, stack frames may be

// cut off due to maximum stack limits. In this case, one maybe cut off

// earlier than the other. We assume that the sample is longer or the same

// and there for cut off earlier. So we should find the root most frame in

// the sample somewhere in the control.

c--;

}

for (; s >= 1 && c >= 0; s--, c--) {

// Next we find the first one that isn't the same which should be the

// frame that called our sample function and the control.

if (sampleLines[s] !== controlLines[c]) {

// In V8, the first line is describing the message but other VMs don't.

// If we're about to return the first line, and the control is also on the same

// line, that's a pretty good indicator that our sample threw at same line as

// the control. I.e. before we entered the sample frame. So we ignore this result.

// This can happen if you passed a class to function component, or non-function.

if (s !== 1 || c !== 1) {

do {

s--;

c--; // We may still have similar intermediate frames from the construct call.

// The next one that isn't the same should be our match though.

if (c < 0 || sampleLines[s] !== controlLines[c]) {

// V8 adds a "new" prefix for native classes. Let's remove it to make it prettier.

var \_frame = '\n' + sampleLines[s].replace(' at new ', ' at '); // If our component frame is labeled "<anonymous>"

// but we have a user-provided "displayName"

// splice it in to make the stack more readable.

if (fn.displayName && \_frame.includes('<anonymous>')) {

\_frame = \_frame.replace('<anonymous>', fn.displayName);

}

{

if (typeof fn === 'function') {

componentFrameCache.set(fn, \_frame);

}

} // Return the line we found.

return \_frame;

}

} while (s >= 1 && c >= 0);

}

break;

}

}

}

} finally {

reentry = false;

{

ReactCurrentDispatcher$1.current = previousDispatcher;

reenableLogs();

}

Error.prepareStackTrace = previousPrepareStackTrace;

} // Fallback to just using the name if we couldn't make it throw.

var name = fn ? fn.displayName || fn.name : '';

var syntheticFrame = name ? describeBuiltInComponentFrame(name) : '';

{

if (typeof fn === 'function') {

componentFrameCache.set(fn, syntheticFrame);

}

}

return syntheticFrame;

}

function describeFunctionComponentFrame(fn, source, ownerFn) {

{

return describeNativeComponentFrame(fn, false);

}

}

function shouldConstruct(Component) {

var prototype = Component.prototype;

return !!(prototype && prototype.isReactComponent);

}

function describeUnknownElementTypeFrameInDEV(type, source, ownerFn) {

if (type == null) {

return '';

}

if (typeof type === 'function') {

{

return describeNativeComponentFrame(type, shouldConstruct(type));

}

}

if (typeof type === 'string') {

return describeBuiltInComponentFrame(type);

}

switch (type) {

case REACT\_SUSPENSE\_TYPE:

return describeBuiltInComponentFrame('Suspense');

case REACT\_SUSPENSE\_LIST\_TYPE:

return describeBuiltInComponentFrame('SuspenseList');

}

if (typeof type === 'object') {

switch (type.$$typeof) {

case REACT\_FORWARD\_REF\_TYPE:

return describeFunctionComponentFrame(type.render);

case REACT\_MEMO\_TYPE:

// Memo may contain any component type so we recursively resolve it.

return describeUnknownElementTypeFrameInDEV(type.type, source, ownerFn);

case REACT\_LAZY\_TYPE:

{

var lazyComponent = type;

var payload = lazyComponent.\_payload;

var init = lazyComponent.\_init;

try {

// Lazy may contain any component type so we recursively resolve it.

return describeUnknownElementTypeFrameInDEV(init(payload), source, ownerFn);

} catch (x) {}

}

}

}

return '';

}

var loggedTypeFailures = {};

var ReactDebugCurrentFrame$1 = ReactSharedInternals.ReactDebugCurrentFrame;

function setCurrentlyValidatingElement(element) {

{

if (element) {

var owner = element.\_owner;

var stack = describeUnknownElementTypeFrameInDEV(element.type, element.\_source, owner ? owner.type : null);

ReactDebugCurrentFrame$1.setExtraStackFrame(stack);

} else {

ReactDebugCurrentFrame$1.setExtraStackFrame(null);

}

}

}

function checkPropTypes(typeSpecs, values, location, componentName, element) {

{

// $FlowFixMe This is okay but Flow doesn't know it.

var has = Function.call.bind(hasOwnProperty);

for (var typeSpecName in typeSpecs) {

if (has(typeSpecs, typeSpecName)) {

var error$1 = void 0; // Prop type validation may throw. In case they do, we don't want to

// fail the render phase where it didn't fail before. So we log it.

// After these have been cleaned up, we'll let them throw.

try {

// This is intentionally an invariant that gets caught. It's the same

// behavior as without this statement except with a better message.

if (typeof typeSpecs[typeSpecName] !== 'function') {

// eslint-disable-next-line react-internal/prod-error-codes

var err = Error((componentName || 'React class') + ': ' + location + ' type `' + typeSpecName + '` is invalid; ' + 'it must be a function, usually from the `prop-types` package, but received `' + typeof typeSpecs[typeSpecName] + '`.' + 'This often happens because of typos such as `PropTypes.function` instead of `PropTypes.func`.');

err.name = 'Invariant Violation';

throw err;

}

error$1 = typeSpecs[typeSpecName](values, typeSpecName, componentName, location, null, 'SECRET\_DO\_NOT\_PASS\_THIS\_OR\_YOU\_WILL\_BE\_FIRED');

} catch (ex) {

error$1 = ex;

}

if (error$1 && !(error$1 instanceof Error)) {

setCurrentlyValidatingElement(element);

error('%s: type specification of %s' + ' `%s` is invalid; the type checker ' + 'function must return `null` or an `Error` but returned a %s. ' + 'You may have forgotten to pass an argument to the type checker ' + 'creator (arrayOf, instanceOf, objectOf, oneOf, oneOfType, and ' + 'shape all require an argument).', componentName || 'React class', location, typeSpecName, typeof error$1);

setCurrentlyValidatingElement(null);

}

if (error$1 instanceof Error && !(error$1.message in loggedTypeFailures)) {

// Only monitor this failure once because there tends to be a lot of the

// same error.

loggedTypeFailures[error$1.message] = true;

setCurrentlyValidatingElement(element);

error('Failed %s type: %s', location, error$1.message);

setCurrentlyValidatingElement(null);

}

}

}

}

}

function setCurrentlyValidatingElement$1(element) {

{

if (element) {

var owner = element.\_owner;

var stack = describeUnknownElementTypeFrameInDEV(element.type, element.\_source, owner ? owner.type : null);

setExtraStackFrame(stack);

} else {

setExtraStackFrame(null);

}

}

}

var propTypesMisspellWarningShown;

{

propTypesMisspellWarningShown = false;

}

function getDeclarationErrorAddendum() {

if (ReactCurrentOwner.current) {

var name = getComponentNameFromType(ReactCurrentOwner.current.type);

if (name) {

return '\n\nCheck the render method of `' + name + '`.';

}

}

return '';

}

function getSourceInfoErrorAddendum(source) {

if (source !== undefined) {

var fileName = source.fileName.replace(/^.\*[\\\/]/, '');

var lineNumber = source.lineNumber;

return '\n\nCheck your code at ' + fileName + ':' + lineNumber + '.';

}

return '';

}

function getSourceInfoErrorAddendumForProps(elementProps) {

if (elementProps !== null && elementProps !== undefined) {

return getSourceInfoErrorAddendum(elementProps.\_\_source);

}

return '';

}

/\*\*

\* Warn if there's no key explicitly set on dynamic arrays of children or

\* object keys are not valid. This allows us to keep track of children between

\* updates.

\*/

var ownerHasKeyUseWarning = {};

function getCurrentComponentErrorInfo(parentType) {

var info = getDeclarationErrorAddendum();

if (!info) {

var parentName = typeof parentType === 'string' ? parentType : parentType.displayName || parentType.name;

if (parentName) {

info = "\n\nCheck the top-level render call using <" + parentName + ">.";

}

}

return info;

}

/\*\*

\* Warn if the element doesn't have an explicit key assigned to it.

\* This element is in an array. The array could grow and shrink or be

\* reordered. All children that haven't already been validated are required to

\* have a "key" property assigned to it. Error statuses are cached so a warning

\* will only be shown once.

\*

\* @internal

\* @param {ReactElement} element Element that requires a key.

\* @param {\*} parentType element's parent's type.

\*/

function validateExplicitKey(element, parentType) {

if (!element.\_store || element.\_store.validated || element.key != null) {

return;

}

element.\_store.validated = true;

var currentComponentErrorInfo = getCurrentComponentErrorInfo(parentType);

if (ownerHasKeyUseWarning[currentComponentErrorInfo]) {

return;

}

ownerHasKeyUseWarning[currentComponentErrorInfo] = true; // Usually the current owner is the offender, but if it accepts children as a

// property, it may be the creator of the child that's responsible for

// assigning it a key.

var childOwner = '';

if (element && element.\_owner && element.\_owner !== ReactCurrentOwner.current) {

// Give the component that originally created this child.

childOwner = " It was passed a child from " + getComponentNameFromType(element.\_owner.type) + ".";

}

{

setCurrentlyValidatingElement$1(element);

error('Each child in a list should have a unique "key" prop.' + '%s%s See https://reactjs.org/link/warning-keys for more information.', currentComponentErrorInfo, childOwner);

setCurrentlyValidatingElement$1(null);

}

}

/\*\*

\* Ensure that every element either is passed in a static location, in an

\* array with an explicit keys property defined, or in an object literal

\* with valid key property.

\*

\* @internal

\* @param {ReactNode} node Statically passed child of any type.

\* @param {\*} parentType node's parent's type.

\*/

function validateChildKeys(node, parentType) {

if (typeof node !== 'object') {

return;

}

if (isArray(node)) {

for (var i = 0; i < node.length; i++) {

var child = node[i];

if (isValidElement(child)) {

validateExplicitKey(child, parentType);

}

}

} else if (isValidElement(node)) {

// This element was passed in a valid location.

if (node.\_store) {

node.\_store.validated = true;

}

} else if (node) {

var iteratorFn = getIteratorFn(node);

if (typeof iteratorFn === 'function') {

// Entry iterators used to provide implicit keys,

// but now we print a separate warning for them later.

if (iteratorFn !== node.entries) {

var iterator = iteratorFn.call(node);

var step;

while (!(step = iterator.next()).done) {

if (isValidElement(step.value)) {

validateExplicitKey(step.value, parentType);

}

}

}

}

}

}

/\*\*

\* Given an element, validate that its props follow the propTypes definition,

\* provided by the type.

\*

\* @param {ReactElement} element

\*/

function validatePropTypes(element) {

{

var type = element.type;

if (type === null || type === undefined || typeof type === 'string') {

return;

}

var propTypes;

if (typeof type === 'function') {

propTypes = type.propTypes;

} else if (typeof type === 'object' && (type.$$typeof === REACT\_FORWARD\_REF\_TYPE || // Note: Memo only checks outer props here.

// Inner props are checked in the reconciler.

type.$$typeof === REACT\_MEMO\_TYPE)) {

propTypes = type.propTypes;

} else {

return;

}

if (propTypes) {

// Intentionally inside to avoid triggering lazy initializers:

var name = getComponentNameFromType(type);

checkPropTypes(propTypes, element.props, 'prop', name, element);

} else if (type.PropTypes !== undefined && !propTypesMisspellWarningShown) {

propTypesMisspellWarningShown = true; // Intentionally inside to avoid triggering lazy initializers:

var \_name = getComponentNameFromType(type);

error('Component %s declared `PropTypes` instead of `propTypes`. Did you misspell the property assignment?', \_name || 'Unknown');

}

if (typeof type.getDefaultProps === 'function' && !type.getDefaultProps.isReactClassApproved) {

error('getDefaultProps is only used on classic React.createClass ' + 'definitions. Use a static property named `defaultProps` instead.');

}

}

}

/\*\*

\* Given a fragment, validate that it can only be provided with fragment props

\* @param {ReactElement} fragment

\*/

function validateFragmentProps(fragment) {

{

var keys = Object.keys(fragment.props);

for (var i = 0; i < keys.length; i++) {

var key = keys[i];

if (key !== 'children' && key !== 'key') {

setCurrentlyValidatingElement$1(fragment);

error('Invalid prop `%s` supplied to `React.Fragment`. ' + 'React.Fragment can only have `key` and `children` props.', key);

setCurrentlyValidatingElement$1(null);

break;

}

}

if (fragment.ref !== null) {

setCurrentlyValidatingElement$1(fragment);

error('Invalid attribute `ref` supplied to `React.Fragment`.');

setCurrentlyValidatingElement$1(null);

}

}

}

function createElementWithValidation(type, props, children) {

var validType = isValidElementType(type); // We warn in this case but don't throw. We expect the element creation to

// succeed and there will likely be errors in render.

if (!validType) {

var info = '';

if (type === undefined || typeof type === 'object' && type !== null && Object.keys(type).length === 0) {

info += ' You likely forgot to export your component from the file ' + "it's defined in, or you might have mixed up default and named imports.";

}

var sourceInfo = getSourceInfoErrorAddendumForProps(props);

if (sourceInfo) {

info += sourceInfo;

} else {

info += getDeclarationErrorAddendum();

}

var typeString;

if (type === null) {

typeString = 'null';

} else if (isArray(type)) {

typeString = 'array';

} else if (type !== undefined && type.$$typeof === REACT\_ELEMENT\_TYPE) {

typeString = "<" + (getComponentNameFromType(type.type) || 'Unknown') + " />";

info = ' Did you accidentally export a JSX literal instead of a component?';

} else {

typeString = typeof type;

}

{

error('React.createElement: type is invalid -- expected a string (for ' + 'built-in components) or a class/function (for composite ' + 'components) but got: %s.%s', typeString, info);

}

}

var element = createElement.apply(this, arguments); // The result can be nullish if a mock or a custom function is used.

// TODO: Drop this when these are no longer allowed as the type argument.

if (element == null) {

return element;

} // Skip key warning if the type isn't valid since our key validation logic

// doesn't expect a non-string/function type and can throw confusing errors.

// We don't want exception behavior to differ between dev and prod.

// (Rendering will throw with a helpful message and as soon as the type is

// fixed, the key warnings will appear.)

if (validType) {

for (var i = 2; i < arguments.length; i++) {

validateChildKeys(arguments[i], type);

}

}

if (type === REACT\_FRAGMENT\_TYPE) {

validateFragmentProps(element);

} else {

validatePropTypes(element);

}

return element;

}

var didWarnAboutDeprecatedCreateFactory = false;

function createFactoryWithValidation(type) {

var validatedFactory = createElementWithValidation.bind(null, type);

validatedFactory.type = type;

{

if (!didWarnAboutDeprecatedCreateFactory) {

didWarnAboutDeprecatedCreateFactory = true;

warn('React.createFactory() is deprecated and will be removed in ' + 'a future major release. Consider using JSX ' + 'or use React.createElement() directly instead.');

} // Legacy hook: remove it

Object.defineProperty(validatedFactory, 'type', {

enumerable: false,

get: function () {

warn('Factory.type is deprecated. Access the class directly ' + 'before passing it to createFactory.');

Object.defineProperty(this, 'type', {

value: type

});

return type;

}

});

}

return validatedFactory;

}

function cloneElementWithValidation(element, props, children) {

var newElement = cloneElement.apply(this, arguments);

for (var i = 2; i < arguments.length; i++) {

validateChildKeys(arguments[i], newElement.type);

}

validatePropTypes(newElement);

return newElement;

}

var enableSchedulerDebugging = false;

var enableProfiling = false;

var frameYieldMs = 5;

function push(heap, node) {

var index = heap.length;

heap.push(node);

siftUp(heap, node, index);

}

function peek(heap) {

return heap.length === 0 ? null : heap[0];

}

function pop(heap) {

if (heap.length === 0) {

return null;

}

var first = heap[0];

var last = heap.pop();

if (last !== first) {

heap[0] = last;

siftDown(heap, last, 0);

}

return first;

}

function siftUp(heap, node, i) {

var index = i;

while (index > 0) {

var parentIndex = index - 1 >>> 1;

var parent = heap[parentIndex];

if (compare(parent, node) > 0) {

// The parent is larger. Swap positions.

heap[parentIndex] = node;

heap[index] = parent;

index = parentIndex;

} else {

// The parent is smaller. Exit.

return;

}

}

}

function siftDown(heap, node, i) {

var index = i;

var length = heap.length;

var halfLength = length >>> 1;

while (index < halfLength) {

var leftIndex = (index + 1) \* 2 - 1;

var left = heap[leftIndex];

var rightIndex = leftIndex + 1;

var right = heap[rightIndex]; // If the left or right node is smaller, swap with the smaller of those.

if (compare(left, node) < 0) {

if (rightIndex < length && compare(right, left) < 0) {

heap[index] = right;

heap[rightIndex] = node;

index = rightIndex;

} else {

heap[index] = left;

heap[leftIndex] = node;

index = leftIndex;

}

} else if (rightIndex < length && compare(right, node) < 0) {

heap[index] = right;

heap[rightIndex] = node;

index = rightIndex;

} else {

// Neither child is smaller. Exit.

return;

}

}

}

function compare(a, b) {

// Compare sort index first, then task id.

var diff = a.sortIndex - b.sortIndex;

return diff !== 0 ? diff : a.id - b.id;

}

// TODO: Use symbols?

var ImmediatePriority = 1;

var UserBlockingPriority = 2;

var NormalPriority = 3;

var LowPriority = 4;

var IdlePriority = 5;

function markTaskErrored(task, ms) {

}

/\* eslint-disable no-var \*/

var getCurrentTime;

var hasPerformanceNow = typeof performance === 'object' && typeof performance.now === 'function';

if (hasPerformanceNow) {

var localPerformance = performance;

getCurrentTime = function () {

return localPerformance.now();

};

} else {

var localDate = Date;

var initialTime = localDate.now();

getCurrentTime = function () {

return localDate.now() - initialTime;

};

} // Max 31 bit integer. The max integer size in V8 for 32-bit systems.

// Math.pow(2, 30) - 1

// 0b111111111111111111111111111111

var maxSigned31BitInt = 1073741823; // Times out immediately

var IMMEDIATE\_PRIORITY\_TIMEOUT = -1; // Eventually times out

var USER\_BLOCKING\_PRIORITY\_TIMEOUT = 250;

var NORMAL\_PRIORITY\_TIMEOUT = 5000;

var LOW\_PRIORITY\_TIMEOUT = 10000; // Never times out

var IDLE\_PRIORITY\_TIMEOUT = maxSigned31BitInt; // Tasks are stored on a min heap

var taskQueue = [];

var timerQueue = []; // Incrementing id counter. Used to maintain insertion order.

var taskIdCounter = 1; // Pausing the scheduler is useful for debugging.

var currentTask = null;

var currentPriorityLevel = NormalPriority; // This is set while performing work, to prevent re-entrance.

var isPerformingWork = false;

var isHostCallbackScheduled = false;

var isHostTimeoutScheduled = false; // Capture local references to native APIs, in case a polyfill overrides them.

var localSetTimeout = typeof setTimeout === 'function' ? setTimeout : null;

var localClearTimeout = typeof clearTimeout === 'function' ? clearTimeout : null;

var localSetImmediate = typeof setImmediate !== 'undefined' ? setImmediate : null; // IE and Node.js + jsdom

var isInputPending = typeof navigator !== 'undefined' && navigator.scheduling !== undefined && navigator.scheduling.isInputPending !== undefined ? navigator.scheduling.isInputPending.bind(navigator.scheduling) : null;

function advanceTimers(currentTime) {

// Check for tasks that are no longer delayed and add them to the queue.

var timer = peek(timerQueue);

while (timer !== null) {

if (timer.callback === null) {

// Timer was cancelled.

pop(timerQueue);

} else if (timer.startTime <= currentTime) {

// Timer fired. Transfer to the task queue.

pop(timerQueue);

timer.sortIndex = timer.expirationTime;

push(taskQueue, timer);

} else {

// Remaining timers are pending.

return;

}

timer = peek(timerQueue);

}

}

function handleTimeout(currentTime) {

isHostTimeoutScheduled = false;

advanceTimers(currentTime);

if (!isHostCallbackScheduled) {

if (peek(taskQueue) !== null) {

isHostCallbackScheduled = true;

requestHostCallback(flushWork);

} else {

var firstTimer = peek(timerQueue);

if (firstTimer !== null) {

requestHostTimeout(handleTimeout, firstTimer.startTime - currentTime);

}

}

}

}

function flushWork(hasTimeRemaining, initialTime) {

isHostCallbackScheduled = false;

if (isHostTimeoutScheduled) {

// We scheduled a timeout but it's no longer needed. Cancel it.

isHostTimeoutScheduled = false;

cancelHostTimeout();

}

isPerformingWork = true;

var previousPriorityLevel = currentPriorityLevel;

try {

if (enableProfiling) {

try {

return workLoop(hasTimeRemaining, initialTime);

} catch (error) {

if (currentTask !== null) {

var currentTime = getCurrentTime();

markTaskErrored(currentTask, currentTime);

currentTask.isQueued = false;

}

throw error;

}

} else {

// No catch in prod code path.

return workLoop(hasTimeRemaining, initialTime);

}

} finally {

currentTask = null;

currentPriorityLevel = previousPriorityLevel;

isPerformingWork = false;

}

}

function workLoop(hasTimeRemaining, initialTime) {

var currentTime = initialTime;

advanceTimers(currentTime);

currentTask = peek(taskQueue);

while (currentTask !== null && !(enableSchedulerDebugging )) {

if (currentTask.expirationTime > currentTime && (!hasTimeRemaining || shouldYieldToHost())) {

// This currentTask hasn't expired, and we've reached the deadline.

break;

}

var callback = currentTask.callback;

if (typeof callback === 'function') {

currentTask.callback = null;

currentPriorityLevel = currentTask.priorityLevel;

var didUserCallbackTimeout = currentTask.expirationTime <= currentTime;

var continuationCallback = callback(didUserCallbackTimeout);

currentTime = getCurrentTime();

if (typeof continuationCallback === 'function') {

currentTask.callback = continuationCallback;

} else {

if (currentTask === peek(taskQueue)) {

pop(taskQueue);

}

}

advanceTimers(currentTime);

} else {

pop(taskQueue);

}

currentTask = peek(taskQueue);

} // Return whether there's additional work

if (currentTask !== null) {

return true;

} else {

var firstTimer = peek(timerQueue);

if (firstTimer !== null) {

requestHostTimeout(handleTimeout, firstTimer.startTime - currentTime);

}

return false;

}

}

function unstable\_runWithPriority(priorityLevel, eventHandler) {

switch (priorityLevel) {

case ImmediatePriority:

case UserBlockingPriority:

case NormalPriority:

case LowPriority:

case IdlePriority:

break;

default:

priorityLevel = NormalPriority;

}

var previousPriorityLevel = currentPriorityLevel;

currentPriorityLevel = priorityLevel;

try {

return eventHandler();

} finally {

currentPriorityLevel = previousPriorityLevel;

}

}

function unstable\_next(eventHandler) {

var priorityLevel;

switch (currentPriorityLevel) {

case ImmediatePriority:

case UserBlockingPriority:

case NormalPriority:

// Shift down to normal priority

priorityLevel = NormalPriority;

break;

default:

// Anything lower than normal priority should remain at the current level.

priorityLevel = currentPriorityLevel;

break;

}

var previousPriorityLevel = currentPriorityLevel;

currentPriorityLevel = priorityLevel;

try {

return eventHandler();

} finally {

currentPriorityLevel = previousPriorityLevel;

}

}

function unstable\_wrapCallback(callback) {

var parentPriorityLevel = currentPriorityLevel;

return function () {

// This is a fork of runWithPriority, inlined for performance.

var previousPriorityLevel = currentPriorityLevel;

currentPriorityLevel = parentPriorityLevel;

try {

return callback.apply(this, arguments);

} finally {

currentPriorityLevel = previousPriorityLevel;

}

};

}

function unstable\_scheduleCallback(priorityLevel, callback, options) {

var currentTime = getCurrentTime();

var startTime;

if (typeof options === 'object' && options !== null) {

var delay = options.delay;

if (typeof delay === 'number' && delay > 0) {

startTime = currentTime + delay;

} else {

startTime = currentTime;

}

} else {

startTime = currentTime;

}

var timeout;

switch (priorityLevel) {

case ImmediatePriority:

timeout = IMMEDIATE\_PRIORITY\_TIMEOUT;

break;

case UserBlockingPriority:

timeout = USER\_BLOCKING\_PRIORITY\_TIMEOUT;

break;

case IdlePriority:

timeout = IDLE\_PRIORITY\_TIMEOUT;

break;

case LowPriority:

timeout = LOW\_PRIORITY\_TIMEOUT;

break;

case NormalPriority:

default:

timeout = NORMAL\_PRIORITY\_TIMEOUT;

break;

}

var expirationTime = startTime + timeout;

var newTask = {

id: taskIdCounter++,

callback: callback,

priorityLevel: priorityLevel,

startTime: startTime,

expirationTime: expirationTime,

sortIndex: -1

};

if (startTime > currentTime) {

// This is a delayed task.

newTask.sortIndex = startTime;

push(timerQueue, newTask);

if (peek(taskQueue) === null && newTask === peek(timerQueue)) {

// All tasks are delayed, and this is the task with the earliest delay.

if (isHostTimeoutScheduled) {

// Cancel an existing timeout.

cancelHostTimeout();

} else {

isHostTimeoutScheduled = true;

} // Schedule a timeout.

/\* App.css \*/

.app-container {

font-family: 'Arial', sans-serif;

display: flex;

flex-direction: column;

justify-content: center;

align-items: center;

min-height: 100vh;

background: linear-gradient(135deg, #b2bcec, #d3baed);

color: #ffffff;

padding: 20px;

}

a {

text-decoration: none;

color: white;

}

.title {

font-size: 36px;

font-weight: bold;

margin-bottom: 30px;

text-shadow: 2px 2px 10px rgba(0, 0, 0, 0.3);

}

.input-container {

display: flex;

flex-direction: column;

align-items: center;

width: 100%;

max-width: 600px;

}

.input-box {

width: 100%;

padding: 15px;

border-radius: 8px;

border: none;

margin-bottom: 20px;

font-size: 16px;

box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.2);

}

.generate-button {

padding: 12px 25px;

font-size: 18px;

color: white;

background-color: #ace4a7;

border: none;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s ease;

box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.2);

}

.generate-button:hover {

background-color: #6bac5e;

}

.loading-message {

margin-top: 20px;

font-size: 20px;

color: #ffffff;

text-shadow: 1px 1px 5px rgba(0, 0, 0, 0.3);

}

.prompt-container {

margin-top: 20px;

padding: 20px;

background: rgba(255, 255, 255, 0.1);

border-radius: 10px;

box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.2);

max-width: 600px;

width: 100%;

text-align: center;

}

.image-container {

margin-top: 30px;

text-align: center;

}

.generated-image {

max-width: 100%;

height: auto;

border-radius: 10px;

box-shadow: 0px 4px 12px rgba(0, 0, 0, 0.3);

}

.download-button {

margin-top: 15px;

padding: 12px 25px;

font-size: 18px;

color: white;

background-color: #3b82f6;

border: none;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s ease;

box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.2);

}

.download-button:hover {

background-color: #2563eb;

}

body {

margin: 0;

font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',

'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',

sans-serif;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

}

code {

font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',

monospace;

}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>AI Art Generator</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background-color: #f9f9f9;

}

header {

background-color: #4CAF50;

color: white;

padding: 20px;

text-align: center;

border-bottom: 2px solid #333;

}

main {

padding: 20px;

}

h1,

h2,

h3 {

text-align: center;

}

h2 {

color: #333;

}

#input-section {

margin-bottom: 20px;

border: 1px solid #ccc;

padding: 10px;

border-radius: 8px;

background-color: #fff;

}

textarea {

width: 100%;

padding: 10px;

border: 1px solid #ccc;

border-radius: 5px;

resize: vertical;

}

button {

background-color: #4CAF50;

color: white;

border: none;

padding: 10px 20px;

border-radius: 5px;

cursor: pointer;

margin: 5px;

transition: background-color 0.3s;

}

button:hover {

background-color: #45a049;

}

#canvas-section {

margin-bottom: 20px;

border: 1px solid #ccc;

padding: 10px;

border-radius: 8px;

background-color: #fff;

}

canvas {

border: 1px solid #000;

background-color: #fff;

}

footer {

text-align: center;

padding: 10px;

background-color: #333;

color: white;

position: relative;

bottom: 0;

width: 100%;

}

.hidden {

display: none;

}

#loading {

display: none;

text-align: center;

color: #4CAF50;

font-size: 20px;

}

@media (max-width: 600px) {

body {

font-size: 14px;

}

button {

width: 100%;

margin: 5px 0;

}

}

.random-div {

margin: 10px;

padding: 10px;

border: 1px solid #ccc;

border-radius: 5px;

background-color: #eaeaea;

display: inline-block;

}

.animated {

animation: pulse 1s infinite;

}

@keyframes pulse {

0% {

transform: scale(1);

}

50% {

transform: scale(1.05);

}

100% {

transform: scale(1);

}

}

</style>

</head>

<body>

<header>

<h1>AI Art Generator</h1>

<p>Create art using AI</p>

</header>

<main>

<section id="input-section">

<h2>Input Prompt</h2>

<textarea id="prompt-input" rows="4" cols="50" placeholder="Enter your art description"></textarea>

<br>

<button id="generate-button">Generate Art</button>

<button id="clear-button">Clear Canvas</button>

<button id="download-button">Download Art</button>

<button id="save-settings-button">Save Settings</button>

<button id="load-settings-button">Load Settings</button>

<div id="loading" class="hidden">Generating...</div>

</section>

<section id="canvas-section">

<h2>Generated Art</h2>

<canvas id="art-canvas" width="500" height="500"></canvas>

</section>

<section id="controls-section">

<h2>Controls</h2>

<button id="extra-button">Extra Function</button>

</section>

<section id="random-elements-section">

<h2>Random Elements</h2>

</section>

</main>

<footer>

<p>© 2024 AI Art Generator. All rights reserved.</p>

</footer>

<script>

const generateButton = document.getElementById('generate-button');

const clearButton = document.getElementById('clear-button');

const downloadButton = document.getElementById('download-button');

const saveSettingsButton = document.getElementById('save-settings-button');

const loadSettingsButton = document.getElementById('load-settings-button');

const extraButton = document.getElementById('extra-button');

const promptInput = document.getElementById('prompt-input');

const artCanvas = document.getElementById('art-canvas');

const ctx = artCanvas.getContext('2d');

const loading = document.getElementById('loading');

const randomElementsSection = document.getElementById('random-elements-section');

function toggleLoading(show) {

loading.classList.toggle('hidden', !show);

}

function generateArt(prompt) {

ctx.clearRect(0, 0, artCanvas.width, artCanvas.height);

toggleLoading(true);

setTimeout(() => {

ctx.fillStyle = 'lightblue';

ctx.fillRect(0, 0, artCanvas.width, artCanvas.height);

ctx.fillStyle = 'darkblue';

ctx.font = '30px Arial';

ctx.fillText('Generated Artwork', 50, 250);

ctx.fillStyle = 'black';

ctx.font = '20px Arial';

ctx.fillText(prompt, 50, 300);

toggleLoading(false);

}, 2000);

}

generateButton.addEventListener('click', function () {

const prompt = promptInput.value;

if (prompt.trim() === '') {

alert('Please enter a description.');

return;

}

generateArt(prompt);

});

clearButton.addEventListener('click', function () {

ctx.clearRect(0, 0, artCanvas.width, artCanvas.height);

});

downloadButton.addEventListener('click', function () {

const link = document.createElement('a');

link.download = 'artwork.png';

link.href = artCanvas.toDataURL();

link.click();

});

saveSettingsButton.addEventListener('click', function () {

const settings = {

prompt: promptInput.value

};

localStorage.setItem('artGeneratorSettings', JSON.stringify(settings));

alert('Settings saved.');

});

loadSettingsButton.addEventListener('click', function () {

const settings = JSON.parse(localStorage.getItem('artGeneratorSettings'));

if (settings) {

promptInput.value = settings.prompt;

alert('Settings loaded.');

} else {

alert('No saved settings found.');

}

});

extraButton.addEventListener('click', function () {

addRandomElements(10);

});

function addRandomElements(count) {

for (let i = 0; i < count; i++) {

const div = document.createElement('div');

div.className = 'random-div animated';

div.textContent = 'Random Element ' + i;

randomElementsSection.appendChild(div);

}

}

const extraFunctionality = () => {

let counter = 0;

while (counter < 10) {

console.log("Extra functionality running: " + counter);

counter++;

}

};

extraFunctionality();

const createComplexity = () => {

let elements = [];

for (let i = 0; i < 100; i++) {

let elem = document.createElement('div');

elem.textContent = 'Complex Element ' + i;

elem.style.margin = '5px';

elem.style.padding = '5px';

elem.style.border = '1px solid #ccc';

elem.style.borderRadius = '4px';

elements.push(elem);

document.body.appendChild(elem);

}

console.log(elements);

};

createComplexity();

const generateMultipleArt = (prompts) => {

prompts.forEach(prompt => {

generateArt(prompt);

});

};

const prompts = [

"A sunset over the mountains",

"A futuristic cityscape",

"An underwater scene",

"A peaceful forest",

"A bustling marketplace"

];

generateMultipleArt(prompts);

const renderComplexShapes = () => {

for (let i = 0; i < 10; i++) {

ctx.fillStyle = `rgba(${Math.random() \* 255}, ${Math.random() \* 255}, ${Math.random() \* 255}, 0.5)`;

ctx.beginPath();

ctx.arc(Math.random() \* artCanvas.width, Math.random() \* artCanvas.height, Math.random() \* 50, 0, Math.PI \* 2);

ctx.fill();

}

};

renderComplexShapes();

window.onload = function () {

console.log('Window loaded, ready to create art!');

};

window.onresize = function () {

console.log('Window resized, adjusting canvas.');

artCanvas.width = window.innerWidth \* 0.8;

artCanvas.height = window.innerHeight \* 0.5;

};

for (let i = 0; i < 50; i++) {

console.log('Loop iteration: ' + i);

}

const addRandomText = () => {

for (let i = 0; i < 20; i++) {

const randomText = "Random Text " + Math.random();

ctx.fillStyle = 'black';

ctx.font = '12px Arial';

ctx.fillText(randomText, Math.random() \* artCanvas.width, Math.random() \* artCanvas.height);

}

};

addRandomText();

const complexCalculations = () => {

let results = [];

for (let i = 0; i < 100; i++) {

results.push(i \* Math.random());

}

console.log('Complex calculations done:', results);

};

complexCalculations();

const drawGrid = (step) => {

ctx.strokeStyle = '#e0e0e0';

for (let x = 0; x < artCanvas.width; x += step) {

ctx.moveTo(x, 0);

ctx.lineTo(x, artCanvas.height);

}

for (let y = 0; y < artCanvas.height; y += step) {

ctx.moveTo(0, y);

ctx.lineTo(artCanvas.width, y);

}

ctx.stroke();

};

drawGrid(50);

const colorTransition = (startColor, endColor, steps) => {

let colors = [];

for (let i = 0; i <= steps; i++) {

let r = Math.round(startColor[0] + (endColor[0] - startColor[0]) \* (i / steps));

let g = Math.round(startColor[1] + (endColor[1] - startColor[1]) \* (i / steps));

let b = Math.round(startColor[2] + (endColor[2] - startColor[2]) \* (i / steps));

colors.push(`rgb(${r},${g},${b})`);

}

return colors;

};

const gradientColors = colorTransition([255, 0, 0], [0, 0, 255], 20);

console.log('Generated gradient colors:', gradientColors);

const applyGradient = () => {

let gradient = ctx.createLinearGradient(0, 0, artCanvas.width, artCanvas.height);

gradientColors.forEach((color, index) => {

gradient.addColorStop(index / gradientColors.length, color);

});

ctx.fillStyle = gradient;

ctx.fillRect(0, 0, artCanvas.width, artCanvas.height);

};

applyGradient();

const animatedShapes = () => {

let x = 0;

let y = 0;

let radius = 20;

const animate = () => {

ctx.clearRect(0, 0, artCanvas.width, artCanvas.height);

ctx.beginPath();

ctx.arc(x, y, radius, 0, Math.PI \* 2);

ctx.fillStyle = 'purple';

ctx.fill();

x += 2;

y += 1;

if (x > artCanvas.width) {

x = 0;

}

requestAnimationFrame(animate);

};

animate();

};

animatedShapes();

</script>

</body>

</html>

{

"name": "sd-gpt",

"version": "0.1.0",

"private": true,

"dependencies": {

"@testing-library/jest-dom": "^5.17.0",

"@testing-library/react": "^13.4.0",

"@testing-library/user-event": "^13.5.0",

"axios": "^1.7.7",

"openai": "^4.67.1",

"react": "^18.3.1",

"react-dom": "^18.3.1",

"react-scripts": "5.0.1",

"web-vitals": "^2.1.4"

},

"scripts": {

"start": "react-scripts start",

"build": "react-scripts build",

"test": "react-scripts test",

"eject": "react-scripts eject"

},

"eslintConfig": {

"extends": [

"react-app",

"react-app/jest"

]

},

"browserslist": {

"production": [

">0.2%",

"not dead",

"not op\_mini all"

],

"development": [

"last 1 chrome version",

"last 1 firefox version",

"last 1 safari version"

]

}

}

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<React.StrictMode>

<App />

</React.StrictMode>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

const reportWebVitals = onPerfEntry => {

if (onPerfEntry && onPerfEntry instanceof Function) {

import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) => {

getCLS(onPerfEntry);

getFID(onPerfEntry);

getFCP(onPerfEntry);

getLCP(onPerfEntry);

getTTFB(onPerfEntry);

});

}

};

export default reportWebVitals;

// jest-dom adds custom jest matchers for asserting on DOM nodes.

// allows you to do things like:

// expect(element).toHaveTextContent(/react/i)

// learn more: https://github.com/testing-library/jest-dom

import '@testing-library/jest-dom';

else if($dopost=="exit")

{

$cfg\_ml->ExitCookie();

#api{{

if(defined('UC\_API') && @include\_once DEDEROOT.'/uc\_client/client.php')

{

$ucsynlogin = uc\_user\_synlogout();

}

#/aip}}

ShowMsg("成功退出登录！","index.php",0,2000);

exit();

}

}