import React, { useState, useEffect, useRef } from "react";

import {

Layout,

Menu,

Button,

Modal,

Form,

Input,

Tabs,

Card,

Switch,

Upload,

List,

Avatar,

Tag,

Collapse,

message,

Alert,

} from "antd";

import {

PlusOutlined,

UploadOutlined,

FolderOutlined,

FileImageOutlined,

CaretRightOutlined,

SearchOutlined,

UserOutlined,

LockOutlined,

MailOutlined,

PhoneOutlined,

} from "@ant-design/icons";

import {

PieChart,

Pie,

Cell,

BarChart,

Bar,

XAxis,

YAxis,

Tooltip,

Legend,

ResponsiveContainer,

} from "recharts";

import exifr from "exifr";

import AMapLoader from "@amap/amap-jsapi-loader";

const { Header, Content, Sider } = Layout;

const { Panel } = Collapse;

const COLORS = ["#0088FE", "#00C49F", "#FFBB28", "#FF8042"];

// API配置 - 根据您的后端实际情况修改

const API\_BASE\_URL = "http://127.0.0.1:8000"; // 您的后端地址

const API\_ENDPOINTS = {

REGISTER: "/register",

LOGIN: "/login",

GET\_USER\_INFO: "/me",

};

function App() {

const [loggedIn, setLoggedIn] = useState(false);

const [userInfo, setUserInfo] = useState(null);

const [projects, setProjects] = useState([]);

const [currentProject, setCurrentProject] = useState(null);

const [currentVideoTask, setCurrentVideoTask] = useState(null);

const [isProjectModalVisible, setIsProjectModalVisible] = useState(false);

const [uploadList, setUploadList] = useState([]);

const [analysisActiveKey, setAnalysisActiveKey] = useState([]);

const [processedFiles, setProcessedFiles] = useState(new Set());

const [previewUrl, setPreviewUrl] = useState(null);

const [searchValue, setSearchValue] = useState("");

const [loginLoading, setLoginLoading] = useState(false);

const [registerLoading, setRegisterLoading] = useState(false);

const [loginError, setLoginError] = useState("");

const [registerError, setRegisterError] = useState("");

const [activeTab, setActiveTab] = useState("2"); // 默认选中手机号登录

const [analysisParams, setAnalysisParams] = useState({

plateNumber: true,

vehicleBrand: true,

vehicleType: true,

plateColor: false,

vehicleModel: false,

vehicleColor: false,

newEnergy: false,

});

const mapRef = useRef(null);

const markerRef = useRef(null);

// 检查本地存储的token

useEffect(() => {

const token = localStorage.getItem("auth\_token");

if (token) {

// 获取用户信息

fetchUserInfo(token);

}

}, []);

// 获取用户信息

const fetchUserInfo = async (token) => {

try {

const response = await fetch(`${API\_BASE\_URL}${API\_ENDPOINTS.GET\_USER\_INFO}?token=${token}`);

if (response.ok) {

const userData = await response.json();

setUserInfo(userData);

setLoggedIn(true);

} else {

// token无效，清除本地存储

localStorage.removeItem("auth\_token");

}

} catch (error) {

console.error("获取用户信息失败:", error);

localStorage.removeItem("auth\_token");

}

};

// 注册功能 - 适配您的后端接口

const handleRegister = async (values) => {

setRegisterLoading(true);

setRegisterError("");

// 构建表单数据以匹配后端期望的格式

const formData = new FormData();

if (values.email) formData.append("email", values.email);

if (values.phone) formData.append("phone", values.phone);

formData.append("password", values.password);

try {

const response = await fetch(`${API\_BASE\_URL}${API\_ENDPOINTS.REGISTER}`, {

method: "POST",

body: formData,

});

const data = await response.json();

if (response.ok) {

message.success("注册成功！");

setRegisterError("");

// 注册成功后切换到登录标签页

setActiveTab("2");

// 清空表单

// 这里可以通过Form的ref来重置，但为了简单，我们只是切换标签页

} else {

setRegisterError(data.detail || "注册失败，请重试");

}

} catch (error) {

console.error("注册请求错误详情:", error);

setRegisterError(`网络错误: ${error.message}，请检查后端服务是否启动`);

} finally {

setRegisterLoading(false);

}

};

// 登录功能 - 适配您的后端接口

const handleLogin = async (values) => {

setLoginLoading(true);

setLoginError("");

// 构建表单数据以匹配后端期望的格式

const formData = new FormData();

formData.append("identifier", values.identifier);

formData.append("password", values.password);

try {

const response = await fetch(`${API\_BASE\_URL}${API\_ENDPOINTS.LOGIN}`, {

method: "POST",

body: formData,

});

const data = await response.json();

if (response.ok) {

// 保存token

localStorage.setItem("auth\_token", data.access\_token);

// 获取用户信息

await fetchUserInfo(data.access\_token);

message.success("登录成功！");

} else {

setLoginError(data.detail || "登录失败，请检查用户名和密码");

}

} catch (error) {

console.error("登录请求错误:", error);

setLoginError("网络错误，请检查后端服务是否启动");

} finally {

setLoginLoading(false);

}

};

// 退出登录

const handleLogout = () => {

localStorage.removeItem("auth\_token");

setUserInfo(null);

setLoggedIn(false);

setProjects([]);

setCurrentProject(null);

setCurrentVideoTask(null);

message.success("已退出登录");

};

const showCreateProject = () => setIsProjectModalVisible(true);

const handleCreateProject = (values) => {

const newProject = { name: values.name, id: Date.now(), videoTasks: [] };

setProjects([...projects, newProject]);

setCurrentProject(newProject);

setIsProjectModalVisible(false);

};

const handleUploadChange = ({ fileList }) => {

setUploadList(fileList);

const newFiles = fileList.filter((file) => !processedFiles.has(file.uid));

if (newFiles.length > 0 && currentProject) {

const newProcessedFiles = new Set(processedFiles);

newFiles.forEach((file) => newProcessedFiles.add(file.uid));

const newVideoTasks = newFiles.map((file) => ({

id: file.uid,

name: file.name,

file: file.originFileObj || file,

location: null,

shareAllowed: false,

analysisResults: null,

createdAt: new Date(),

}));

const updatedProjects = projects.map((project) => {

if (project.id === currentProject.id) {

return {

...project,

videoTasks: [...project.videoTasks, ...newVideoTasks],

};

}

return project;

});

setProjects(updatedProjects);

setCurrentProject(updatedProjects.find((p) => p.id === currentProject.id));

setProcessedFiles(newProcessedFiles);

message.success(`成功添加 ${newFiles.length} 个文件`);

}

};

const getCurrentProjectVideoTasks = () =>

currentProject ? currentProject.videoTasks : [];

useEffect(() => {

if (!currentVideoTask?.file) return;

const url = URL.createObjectURL(currentVideoTask.file);

setPreviewUrl(url);

return () => {

URL.revokeObjectURL(url);

setPreviewUrl(null);

};

}, [currentVideoTask?.file]);

useEffect(() => {

async function readVideoGPS() {

if (currentVideoTask?.file && !currentVideoTask.location) {

try {

const gps = await exifr.gps(currentVideoTask.file);

if (gps && gps.latitude && gps.longitude) {

updateTaskLocation({ lat: gps.latitude, lng: gps.longitude });

message.success("已从视频读取地理位置信息");

}

} catch (err) {

console.warn("未能读取视频 EXIF", err);

}

}

}

readVideoGPS();

}, [currentVideoTask?.file]);

useEffect(() => {

if (!currentVideoTask || mapRef.current) return;

AMapLoader.load({

key: "46046dbf9deeb823d973ca202a961710",

version: "2.0",

plugins: ["AMap.Marker", "AMap.ToolBar", "AMap.PlaceSearch"],

}).then((AMap) => {

mapRef.current = new AMap.Map("amap-container", {

viewMode: "2D",

zoom: 12,

center: [116.397428, 39.90923],

});

mapRef.current.on("click", (e) => {

const latlng = { lat: e.lnglat.getLat(), lng: e.lnglat.getLng() };

if (!markerRef.current) {

markerRef.current = new AMap.Marker({

position: [latlng.lng, latlng.lat],

});

mapRef.current.add(markerRef.current);

} else {

markerRef.current.setPosition([latlng.lng, latlng.lat]);

}

updateTaskLocation(latlng);

message.success("已手动标注位置");

});

});

}, [currentVideoTask]);

const updateTaskLocation = (latlng) => {

const updatedProjects = projects.map((project) => {

if (project.id === currentProject.id) {

const updatedTasks = project.videoTasks.map((task) => {

if (task.id === currentVideoTask.id) return { ...task, location: latlng };

return task;

});

return { ...project, videoTasks: updatedTasks };

}

return project;

});

setProjects(updatedProjects);

setCurrentVideoTask({ ...currentVideoTask, location: latlng });

if (markerRef.current) markerRef.current.setPosition([latlng.lng, latlng.lat]);

};

const handleSearchPlace = () => {

if (!searchValue || !mapRef.current) return;

AMapLoader.load({

key: "46046dbf9deeb823d973ca202a961710",

version: "2.0",

plugins: ["AMap.PlaceSearch"],

})

.then((AMap) => {

const placeSearch = new AMap.PlaceSearch({ pageSize: 5, pageIndex: 1, city: "全国" });

placeSearch.search(searchValue, (status, result) => {

if (status === "complete" && result.poiList.pois.length > 0) {

const poi = result.poiList.pois[0];

const latlng = { lat: poi.location.lat, lng: poi.location.lng };

updateTaskLocation(latlng);

message.success(`标记移动到: ${poi.name}`);

} else {

message.warning("未找到匹配地点");

}

});

})

.catch((err) => console.error(err));

};

// 登录和注册表单配置

const loginTabs = [

{

key: "1",

label: "微信扫码",

children: <Button type="primary" block onClick={() => message.info("微信扫码功能待实现")}>扫码登录 (演示)</Button>,

},

{

key: "2",

label: "手机号登录",

children: (

<Form

layout="vertical"

onFinish={(values) => handleLogin({ identifier: values.phone, password: values.password })}

key="phone-login"

>

{loginError && <Alert message={loginError} type="error" showIcon style={{ marginBottom: 16 }} />}

<Form.Item

label="手机号"

name="phone"

rules={[{ required: true, message: "请输入手机号" }]}

>

<Input prefix={<PhoneOutlined />} placeholder="请输入手机号" />

</Form.Item>

<Form.Item

label="密码"

name="password"

rules={[{ required: true, message: "请输入密码" }]}

>

<Input.Password prefix={<LockOutlined />} placeholder="请输入密码" />

</Form.Item>

<Button type="primary" htmlType="submit" block loading={loginLoading}>

登录

</Button>

</Form>

),

},

{

key: "3",

label: "邮箱登录",

children: (

<Form

layout="vertical"

onFinish={(values) => handleLogin({ identifier: values.email, password: values.password })}

key="email-login"

>

{loginError && <Alert message={loginError} type="error" showIcon style={{ marginBottom: 16 }} />}

<Form.Item

label="邮箱"

name="email"

rules={[

{ required: true, message: "请输入邮箱" },

{ type: 'email', message: '请输入有效的邮箱地址' }

]}

>

<Input prefix={<MailOutlined />} placeholder="请输入邮箱" />

</Form.Item>

<Form.Item

label="密码"

name="password"

rules={[{ required: true, message: "请输入密码" }]}

>

<Input.Password prefix={<LockOutlined />} placeholder="请输入密码" />

</Form.Item>

<Button type="primary" htmlType="submit" block loading={loginLoading}>

登录

</Button>

</Form>

),

},

{

key: "4",

label: "注册账号",

children: (

<Form

layout="vertical"

onFinish={handleRegister}

key="register"

>

{registerError && <Alert message={registerError} type="error" showIcon style={{ marginBottom: 16 }} />}

<Form.Item

label="手机号"

name="phone"

rules={[

{

required: false,

message: "请输入手机号"

}

]}

>

<Input prefix={<PhoneOutlined />} placeholder="请输入手机号（可选）" />

</Form.Item>

<Form.Item

label="邮箱"

name="email"

rules={[

{

required: false,

message: "请输入邮箱或手机号至少一项"

},

{

type: 'email',

message: '请输入有效的邮箱地址',

// 只有当有值时才验证邮箱格式

validator: (\_, value) => {

if (!value || value.includes('@')) {

return Promise.resolve();

}

return Promise.reject(new Error('请输入有效的邮箱地址'));

}

}

]}

>

<Input prefix={<MailOutlined />} placeholder="请输入邮箱（可选）" />

</Form.Item>

<Form.Item

label="密码"

name="password"

rules={[

{ required: true, message: "请输入密码" },

{ min: 6, message: '密码至少6位' }

]}

>

<Input.Password prefix={<LockOutlined />} placeholder="请输入密码（至少6位）" />

</Form.Item>

<Form.Item

label="确认密码"

name="confirmPassword"

dependencies={['password']}

rules={[

{ required: true, message: "请确认密码" },

({ getFieldValue }) => ({

validator(\_, value) {

if (!value || getFieldValue('password') === value) {

return Promise.resolve();

}

return Promise.reject(new Error('两次输入的密码不一致'));

},

}),

]}

>

<Input.Password prefix={<LockOutlined />} placeholder="请再次输入密码" />

</Form.Item>

<Form.Item

rules={[

{

validator: (\_, value) => {

const { getFieldValue } = \_;

const email = getFieldValue('email');

const phone = getFieldValue('phone');

if (!email && !phone) {

return Promise.reject(new Error('邮箱和手机号至少填写一项'));

}

return Promise.resolve();

},

},

]}

>

{/\* 这个表单项用于整体验证邮箱和手机号至少有一项 \*/}

</Form.Item>

<Button type="primary" htmlType="submit" block loading={registerLoading}>

注册

</Button>

</Form>

),

},

];

const generateMenuItems = () => [

...projects.map((project) => ({

key: project.id,

icon: <FolderOutlined />,

label: project.name,

children: project.videoTasks.map((task) => ({

key: task.id,

icon: <FileImageOutlined />,

label: task.name,

onClick: () => setCurrentVideoTask(task),

})),

onTitleClick: () => { setCurrentProject(project); setCurrentVideoTask(null); },

})),

{ key: "new-project", icon: <PlusOutlined />, label: "新建项目", onClick: showCreateProject },

];

if (!loggedIn) {

return (

<div style={{ display: "flex", justifyContent: "center", alignItems: "center", minHeight: "100vh", width: "100vw" }}>

<Card title="车辆信息智能识别与数据分析平台" style={{ width: 450 }}>

<Tabs

activeKey={activeTab}

onChange={setActiveTab}

items={loginTabs}

/>

<div style={{ marginTop: 16, textAlign: 'center', color: '#999', fontSize: '12px' }}>

提示: 请确保后端服务已启动在 {API\_BASE\_URL}

<br />

测试账号: 使用您注册的邮箱/手机号和密码

</div>

</Card>

</div>

);

}

// 演示统计数据

const demoPieData = [

{ name: "SUV", value: 40 },

{ name: "Sedan", value: 30 },

{ name: "Truck", value: 20 },

{ name: "Other", value: 10 },

];

const demoBarData = [

{ name: "Brand A", count: 12 },

{ name: "Brand B", count: 20 },

{ name: "Brand C", count: 8 },

{ name: "Brand D", count: 16 },

];

const demoPlateColorData = [

{ name: "蓝牌", value: 50 },

{ name: "黄牌", value: 20 },

{ name: "白牌", value: 20 },

{ name: "黑牌", value: 10 },

];

const demoNewEnergyData = [

{ name: "新能源", value: 25 },

{ name: "非新能源", value: 75 },

];

const demoVehicleModelData = [

{ name: "Model A", count: 15 },

{ name: "Model B", count: 22 },

{ name: "Model C", count: 10 },

];

return (

<Layout style={{ minHeight: "100vh", width: "100vw" }}>

<Header style={{ color: "white", fontSize: 20, padding: "0 24px", position: "sticky", top: 0, zIndex: 1, display: 'flex', justifyContent: 'space-between', alignItems: 'center' }}>

<span>车辆信息智能识别与数据分析平台</span>

<div style={{ display: 'flex', alignItems: 'center' }}>

<span style={{ marginRight: 16 }}>

<UserOutlined /> {userInfo?.email || userInfo?.phone || '用户'}

</span>

<Button type="link" onClick={handleLogout} style={{ color: 'white' }}>

退出登录

</Button>

</div>

</Header>

<Layout style={{ flexDirection: "row", flex: 1 }}>

<Sider width={280} style={{ background: "#001529", overflow: "auto", height: "calc(100vh - 64px)" }}>

<Menu

mode="inline"

selectedKeys={[currentProject?.id?.toString(), currentVideoTask?.id?.toString()]}

defaultOpenKeys={currentProject ? [currentProject.id.toString()] : []}

style={{ height: "100%", borderRight: 0 }}

items={generateMenuItems()}

/>

</Sider>

<Content style={{ padding: 24, background: "#fff", overflow: "auto", flex: 1, height: "calc(100vh - 64px)", minWidth: 0 }}>

{!currentProject && (

<div style={{ display: "flex", justifyContent: "center", alignItems: "center", height: "100%" }}>

<div style={{ textAlign: "center" }}>

<p>请选择或创建一个项目</p>

<Button type="primary" onClick={showCreateProject}>创建新项目</Button>

</div>

</div>

)}

{currentProject && !currentVideoTask && (

<div style={{ width: "100%" }}>

<Card title={`项目: ${currentProject.name}`} style={{ marginBottom: 24 }}>

<Upload

fileList={uploadList}

onChange={handleUploadChange}

beforeUpload={() => false}

multiple

showUploadList={false}

style={{ marginBottom: 24 }}

>

<Button icon={<UploadOutlined />}>上传视频</Button>

</Upload>

<Card title="视频任务列表" style={{ marginTop: 16 }}>

{getCurrentProjectVideoTasks().length > 0 ? (

<List

itemLayout="horizontal"

dataSource={getCurrentProjectVideoTasks()}

renderItem={(task) => (

<List.Item

actions={[

<Button type="link" onClick={() => setCurrentVideoTask(task)}>查看详情</Button>,

<Button type="link" danger onClick={() => {

const updatedProjects = projects.map(project => {

if (project.id === currentProject.id) {

return { ...project, videoTasks: project.videoTasks.filter(t => t.id !== task.id) };

}

return project;

});

setProjects(updatedProjects);

setCurrentProject(updatedProjects.find(p => p.id === currentProject.id));

message.success(`已删除任务: ${task.name}`);

}}>删除</Button>

]}

>

<List.Item.Meta avatar={<Avatar icon={<FileImageOutlined />} />} title={task.name} description={`上传于: ${task.createdAt.toLocaleString()}`} />

<div>{task.analysisResults ? <Tag color="green">已分析</Tag> : <Tag color="blue">待分析</Tag>}</div>

</List.Item>

)}

/>

) : (

<div style={{ textAlign: "center", padding: 20 }}>

<p>暂无视频任务</p>

<p>请上传视频/影像文件</p>

</div>

)}

</Card>

</Card>

</div>

)}

{currentProject && currentVideoTask && (

<div style={{ width: "100%" }}>

<div style={{ marginBottom: 16 }}>

<Button type="link" onClick={() => setCurrentVideoTask(null)} style={{ padding: 0, marginBottom: 8 }}>&larr; 返回项目</Button>

<h2>{currentProject.name} / {currentVideoTask.name}</h2>

</div>

<Card title="视频预览" style={{ marginBottom: 24, width: "100%" }}>

{previewUrl ? <video controls style={{ width: "100%", maxHeight: 400, background: "#000" }} src={previewUrl} /> : <p>暂无视频预览</p>}

</Card>

<Card title="地理信息" style={{ marginBottom: 24, width: "100%", position: "relative" }}>

<div style={{ display: "flex", gap: 16, marginBottom: 8 }}>

<Input

addonBefore="纬度"

value={currentVideoTask.location?.lat || ""}

onChange={(e) => updateTaskLocation({ lat: parseFloat(e.target.value) || 0, lng: currentVideoTask.location?.lng || 0 })}

/>

<Input

addonBefore="经度"

value={currentVideoTask.location?.lng || ""}

onChange={(e) => updateTaskLocation({ lat: currentVideoTask.location?.lat || 0, lng: parseFloat(e.target.value) || 0 })}

/>

<Input

placeholder="搜索地点"

value={searchValue}

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

suffix={<SearchOutlined onClick={handleSearchPlace} style={{ cursor: "pointer" }} />}

/>

<Button

type="primary"

icon={<SearchOutlined />}

onClick={() => {

if (markerRef.current && mapRef.current) {

mapRef.current.setCenter(markerRef.current.getPosition());

}

}}

/>

</div>

<div id="amap-container" style={{ height: 300, border: "1px solid #ddd", borderRadius: 8 }} />

</Card>

<Card title="识别参数" style={{ marginTop: 16 }}>

<div style={{ display: "flex", flexWrap: "wrap", gap: 16 }}>

{[

{ key: "plateNumber", label: "车牌号码" },

{ key: "vehicleBrand", label: "车辆品牌" },

{ key: "vehicleType", label: "车辆类型" },

{ key: "plateColor", label: "车牌颜色" },

{ key: "vehicleModel", label: "车辆型号" },

{ key: "vehicleColor", label: "车辆颜色" },

{ key: "newEnergy", label: "新能源车" }

].map(item => (

<Switch

key={item.key}

checkedChildren={item.label}

unCheckedChildren={item.label}

checked={analysisParams[item.key]}

onChange={(checked) => setAnalysisParams({ ...analysisParams, [item.key]: checked })}

/>

))}

</div>

</Card>

<Button type="primary" style={{ marginTop: 16 }} onClick={() => {

const updatedProjects = projects.map(project => {

if (project.id === currentProject.id) {

const updatedTasks = project.videoTasks.map(task => {

if (task.id === currentVideoTask.id) {

return {

...task,

analysisResults: {

pieData: [...demoPieData],

barData: [...demoBarData],

plateColorData: [...demoPlateColorData],

newEnergyData: [...demoNewEnergyData],

vehicleModelData: [...demoVehicleModelData],

params: { ...analysisParams }

}

};

}

return task;

});

return { ...project, videoTasks: updatedTasks };

}

return project;

});

setProjects(updatedProjects);

setCurrentVideoTask({

...currentVideoTask,

analysisResults: {

pieData: [...demoPieData],

barData: [...demoBarData],

plateColorData: [...demoPlateColorData],

newEnergyData: [...demoNewEnergyData],

vehicleModelData: [...demoVehicleModelData],

params: { ...analysisParams }

}

});

setAnalysisActiveKey(['1']);

}}>开始分析</Button>

{currentVideoTask.analysisResults && (

<Card title="演示统计图表" style={{ marginTop: 24 }}>

<Collapse activeKey={analysisActiveKey} onChange={(key) => setAnalysisActiveKey(key)} expandIcon={({ isActive }) => <CaretRightOutlined rotate={isActive ? 90 : 0} />}>

<Panel header="分析结果与数据共享设置" key="1">

<Form.Item label="授权共享">

<Switch

checked={currentVideoTask.shareAllowed || false}

onChange={(checked) => {

const updatedProjects = projects.map(project => {

if (project.id === currentProject.id) {

const updatedTasks = project.videoTasks.map(task => {

if (task.id === currentVideoTask.id) {

return { ...task, shareAllowed: checked };

}

return task;

});

return { ...project, videoTasks: updatedTasks };

}

return project;

});

setProjects(updatedProjects);

setCurrentVideoTask({ ...currentVideoTask, shareAllowed: checked });

}}

/>

</Form.Item>

<h4>用户选择的分析参数</h4>

<table style={{ width: "100%", borderCollapse: "collapse", marginBottom: 24 }}>

<thead>

<tr>

<th style={{ border: "1px solid #ddd", padding: 8 }}>参数项</th>

<th style={{ border: "1px solid #ddd", padding: 8 }}>状态</th>

</tr>

</thead>

<tbody>

{Object.entries(currentVideoTask.analysisResults.params || {}).map(([key, value]) => (

<tr key={key}>

<td style={{ border: "1px solid #ddd", padding: 8 }}>{key}</td>

<td style={{ border: "1px solid #ddd", padding: 8 }}>{value ? "开启" : "关闭"}</td>

</tr>

))}

</tbody>

</table>

{/\* 图表和对应表格 \*/}

{[

{ title: "车辆类型分布", data: currentVideoTask.analysisResults.pieData, type: "pie" },

{ title: "车辆品牌统计", data: currentVideoTask.analysisResults.barData, type: "bar" },

{ title: "车牌颜色分布", data: currentVideoTask.analysisResults.plateColorData, type: "pie" },

{ title: "新能源车比例", data: currentVideoTask.analysisResults.newEnergyData, type: "pie" },

{ title: "车辆型号统计", data: currentVideoTask.analysisResults.vehicleModelData, type: "bar" },

].map((chart, idx) => (

<div key={idx} style={{ marginBottom: 32 }}>

<h4>{chart.title}</h4>

<ResponsiveContainer width="100%" height={300}>

{chart.type === "pie" ? (

<PieChart>

<Pie data={chart.data} dataKey="value" nameKey="name" cx="50%" cy="50%" outerRadius={100} label>

{chart.data.map((entry, index) => (

<Cell key={index} fill={COLORS[index % COLORS.length]} />

))}

</Pie>

<Tooltip />

<Legend />

</PieChart>

) : (

<BarChart data={chart.data}>

<XAxis dataKey="name" />

<YAxis />

<Tooltip />

<Legend />

<Bar dataKey="count" fill="#82ca9d" />

</BarChart>

)}

</ResponsiveContainer>

{/\* 对应表格 \*/}

<table style={{ width: "100%", borderCollapse: "collapse", marginTop: 8 }}>

<thead>

<tr>

<th style={{ border: "1px solid #ddd", padding: 8 }}>名称</th>

<th style={{ border: "1px solid #ddd", padding: 8 }}>{chart.type === "pie" ? "数量/比例" : "数量"}</th>

</tr>

</thead>

<tbody>

{chart.data.map((item, i) => (

<tr key={i}>

<td style={{ border: "1px solid #ddd", padding: 8 }}>{item.name}</td>

<td style={{ border: "1px solid #ddd", padding: 8 }}>{item.value ?? item.count}</td>

</tr>

))}

</tbody>

</table>

</div>

))}

</Panel>

</Collapse>

</Card>

)}

</div>

)}

<Modal title="创建新项目" open={isProjectModalVisible} footer={null} onCancel={() => setIsProjectModalVisible(false)}>

<Form layout="vertical" onFinish={handleCreateProject}>

<Form.Item label="项目名称" name="name" rules={[{ required: true, message: "请输入项目名称" }]}>

<Input />

</Form.Item>

<Form.Item>

<Button type="primary" htmlType="submit">创建</Button>

</Form.Item>

</Form>

</Modal>

</Content>

</Layout>

</Layout>

);

}

export default App;