# PC管理后台 - 前端设计文档

**文档版本**: V1.0  
**日期**: 2025年2月21日  
**作者**: Grok 3 (xAI)  
**目标读者**: 前端开发团队、技术负责人

## 1. 概述

### 1.1 目标

实现“EduInsight AI”系统中PC管理后台的功能，提供管理员一个直观、高效的管理界面，用于：

* 管理模型（查看状态、调整参数、分析效果）。
* 管理共享资源（上传、导入、审核）。
* 处理批量数据（导入学生/考试数据）。
* 监控系统运行（用户活跃度、资源使用率、模型性能）。

### 1.2 技术选型

* **框架**: React（单页应用，组件化开发）。
* **UI库**: Ant Design（丰富组件，专业管理后台风格）。
* **状态管理**: Redux（复杂数据流管理）。
* **数据请求**: Axios（RESTful API 调用）。
* **构建工具**: Webpack（打包优化）。
* **部署**: 静态托管（如Netlify）。

### 1.3 设计原则

* **直观**: 一页式仪表盘，核心信息一览无余。
* **高效**: 操作流程简洁，支持批量处理。
* **可扩展**: 组件化设计，易于新增功能。

## 2. 功能设计

### 2.1 总体页面结构

text

WrapCopy

+-------------------------------------+ | 顶部导航 (Logo | 用户信息 | 退出) |   
+-------------------------------------+   
| 左侧菜单 |   
| - 模型管理 |   
| - 资源管理 |   
| - 数据管理 |   
| - 系统监控 |   
+-------------------------------------+   
| 主内容区域 (动态切换) |   
| - 仪表盘 / 列表 / 表单 / 图表 |   
+-------------------------------------+

* **顶部导航**: 固定显示品牌Logo、当前管理员信息和退出按钮。
* **左侧菜单**: 固定宽度，折叠/展开切换，导航至各功能模块。
* **主内容区域**: 动态渲染对应页面内容。

### 2.2 功能模块设计

#### 2.2.1 模型管理

* **功能**: 查看模型状态、调整参数、分析效果。
* **页面布局**:
  + **模型列表**: 表格展示所有模型（学生模型、教师模型）。
  + **模型详情**: 点击某模型，弹出抽屉查看详细信息和调整参数。
  + **效果分析**: 图表展示模型性能指标（如准确率、训练时间）。
* **组件设计**:
  + **Table**: 显示模型ID、类型、状态（训练中/已部署）、最后更新时间。
    - 操作列：查看详情、重新训练。
  + **Drawer**: 详情抽屉，含参数表单（如学习率、迭代次数）和保存按钮。
  + **Chart**: 使用ECharts绘制折线图（准确率趋势）。
* **交互流程**:
  1. 进入页面 -> 加载模型列表。
  2. 点击“查看详情” -> 打开抽屉，加载模型参数。
  3. 修改参数 -> 点击保存，调用API更新。
* **接口调用**:
  + **GET /models**: 获取模型列表。
    - 返回：{ id, type, status, updatedAt }
  + **GET /models/{id}**: 获取模型详情。
    - 返回：{ parameters, metrics }
  + **POST /models/{id}/update**: 更新模型参数。
    - 请求：{ parameters: { learningRate, epochs } }

#### 2.2.2 资源管理

* **功能**: 上传资源、导入外网资源、审核教师上传内容。
* **页面布局**:
  + **资源列表**: 表格展示资源信息。
  + **上传区域**: 拖拽上传文件或输入外网URL。
  + **审核队列**: 未审核资源列表，含通过/拒绝操作。
* **组件设计**:
  + **Table**: 显示资源ID、类型（视频/文档）、URL、标签、评分、状态（已审核/待审核）。
    - 操作列：编辑标签、删除。
  + **Upload**: Antd Upload组件，支持拖拽和批量上传。
  + **Modal**: 审核弹窗，显示资源预览（视频播放/文档查看）和操作按钮。
* **交互流程**:
  1. 上传文件 -> 选择文件/输入URL -> 调用API存储。
  2. 查看审核队列 -> 点击资源 -> 弹窗预览并审核。
  3. 编辑标签 -> 保存更新。
* **接口调用**:
  + **POST /resources**: 上传资源。
    - 请求：FormData { file, tags, type }
    - 返回：{ id, url }
  + **POST /resources/import**: 导入外网资源。
    - 请求：{ url, tags, type }
  + **GET /resources/pending**: 获取待审核资源。
    - 返回：{ id, url, uploaderId }
  + **POST /resources/{id}/review**: 审核资源。
    - 请求：{ status: "approved/rejected" }

#### 2.2.3 数据管理

* **功能**: 批量导入学生/考试数据，管理公共资源。
* **页面布局**:
  + **导入表单**: 文件上传+模板下载。
  + **数据列表**: 展示已导入数据记录。
* **组件设计**:
  + **Upload**: 支持CSV/Excel文件上传。
  + **Table**: 显示导入记录（时间、文件名称、状态）。
  + **Button**: 下载导入模板（CSV格式）。
* **交互流程**:
  1. 点击“下载模板” -> 获取标准CSV。
  2. 上传文件 -> 调用API处理 -> 显示导入结果。
  3. 查看历史记录 -> 检查导入状态。
* **接口调用**:
  + **POST /data/import**: 导入数据。
    - 请求：FormData { file }
    - 返回：{ status, errorRows }
  + **GET /data/imports**: 获取导入历史。
    - 返回：{ id, filename, status, timestamp }

#### 2.2.4 系统监控

* **功能**: 监控用户活跃度、资源使用率、模型性能。
* **页面布局**:
  + **仪表盘**: 图表+关键指标卡片。
* **组件设计**:
  + **Card**: 显示关键指标（如活跃用户数、资源下载量）。
  + **Chart**: 折线图（用户活跃趋势）、柱状图（资源使用分布）。
* **交互流程**:
  1. 进入页面 -> 加载监控数据。
  2. 切换时间范围（如7天/30天） -> 刷新图表。
* **接口调用**:
  + **GET /metrics**: 获取监控数据。
    - 返回：{ users: { active, trend }, resources: { downloads, top }, models: { accuracy } }

## 3. 页面设计

### 3.1 模型管理页面

text

WrapCopy

+-------------------------------------+ | [筛选: 类型/状态] [刷新] |   
+-------------------------------------+   
| Table: ID | 类型 | 状态 | 更新时间 | 操作 |   
+-------------------------------------+   
| Drawer (右侧抽屉): |   
| - 参数表单 |   
| - 效果折线图 |   
+-------------------------------------+

### 3.2 资源管理页面

text

WrapCopy

+-------------------------------------+ | [Tab: 已审核 | 待审核] |   
+-------------------------------------+   
| [上传区域: 拖拽/URL输入] |   
+-------------------------------------+   
| Table: ID | 类型 | URL | 标签 | 状态 | 操作 |   
+-------------------------------------+   
| Modal: 资源预览 | 通过/拒绝按钮 |   
+-------------------------------------+

### 3.3 数据管理页面

text

WrapCopy

+-------------------------------------+ | [上传文件] [下载模板] |   
+-------------------------------------+   
| Table: ID | 文件名 | 状态 | 时间 |   
+-------------------------------------+

### 3.4 系统监控页面

text

WrapCopy

+-------------------------------------+ | Card: 活跃用户 | 资源下载 | 模型准确率 |   
+-------------------------------------+   
| Chart: 用户趋势 | 资源分布 |   
+-------------------------------------+

## 4. 技术实现细节

### 4.1 项目结构

text

WrapCopy

src/ ├── components/ # 公共组件   
│ ├── ModelTable.js   
│ ├── ResourceUpload.js   
│ └── MetricCard.js   
├── pages/ # 页面模块   
│ ├── ModelManage/   
│ ├── ResourceManage/   
│ ├── DataManage/   
│ └── SystemMonitor/   
├── redux/ # 状态管理   
│ ├── actions.js   
│ └── reducers.js   
├── api/ # 接口调用   
│ └── index.js   
├── App.js # 主入口   
└── index.js # 启动文件

### 4.2 示例代码（模型管理）

jsx

WrapCopy

// src/pages/ModelManage/index.js import { Table, Button } from 'antd'*;* import { useEffect, useState } from 'react'*;* import { getModels, updateModel } from '../../api'*;* import ModelDrawer from './ModelDrawer'*;* const ModelManage = () => { const [models, setModels] = useState([])*;* const [selectedModel, setSelectedModel] = useState(null)*;* useEffect(() => { fetchModels()*;* }, [])*;* const fetchModels = async () => { const res = await getModels()*;* setModels(res.data)*;* }*;* const columns = [ { title: 'ID', dataIndex: 'id' }, { title: '类型', dataIndex: 'type' }, { title: '状态', dataIndex: 'status' }, { title: '更新时间', dataIndex: 'updatedAt' }, { title: '操作', render: (\_, record) => ( <Button onClick={() => setSelectedModel(record)}>查看详情</Button> ),   
}, ]*;* return (   
<div>   
<Table dataSource={models} columns={columns} />   
{selectedModel && (   
<ModelDrawer   
model={selectedModel} onClose={() => setSelectedModel(null)} onSave={async (params) => { await updateModel(selectedModel.id, params)*;* fetchModels()*;* }}   
/>   
)}   
</div> )*;* }*;* export default ModelManage*;*

### 4.3 API封装

js

WrapCopy

*// src/api/index.js* import axios from 'axios'; const api = axios.create({ baseURL: '/api' }); export const getModels = () => api.get('/models'); export const updateModel = (id, params) => api.post(`/models/${id}/update`, params); export const uploadResource = (formData) => api.post('/resources', formData); export const getMetrics = () => api.get('/metrics');

## 5. 非功能性需求

### 5.1 性能

* **加载优化**: 懒加载非核心页面，异步加载图表数据。
* **响应**: 表格分页，每页20条，避免大数据阻塞。

### 5.2 可维护性

* **组件复用**: 抽取通用Table和Upload组件。
* **文档化**: 注释关键逻辑，维护API映射表。

### 5.3 兼容性

* **浏览器**: Chrome、Firefox、Edge（最新版本）。

## 6. 风险与应对

* **接口延迟**: 添加加载动画，使用本地缓存兜底。
* **数据复杂**: 支持筛选和搜索，简化操作。

### 设计亮点

1. **统一风格**: Ant Design保证专业感。
2. **交互流畅**: 抽屉和弹窗提升操作效率。
3. **可视化**: 图表直观展示监控和效果数据。