# 服务端 - 详细设计文档

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

## 1. 概述

### 1.1 目标

服务端支持“EduInsight AI”系统的核心功能，包括：

* 处理PC管理后台和移动端的请求（数据上传、查询、推送）。
* 管理数据存储（用户、作业、资源等）。
* 协调模型端调用，整合分析结果。
* 实现共享资源管理和审核流程。
* 执行定时/触发式推送任务。

### 1.2 技术选型

* **框架**: Flask（轻量、易扩展，与Python生态兼容）。
* **数据库**:
  + **MySQL**: 结构化数据存储（如用户信息、资源元数据）。
  + **Redis**: 缓存热点数据和推送任务。
* **存储**: AWS S3（视频、文档等大文件）。
* **异步任务**: Celery + RabbitMQ（推送、批量处理）。
* **认证**: JWT（多角色权限管理）。
* **日志**: Logging（记录请求和错误）。

### 1.3 设计原则

* **模块化**: 分离API、数据、推送和资源服务。
* **高性能**: 异步任务和缓存优化响应速度。
* **安全性**: 数据加密、权限控制。

## 2. 功能设计

### 2.1 总体架构

服务端分为四个子模块：

* **API服务**: 处理前端和移动端请求，提供RESTful接口。
* **数据服务**: 管理数据库和文件存储操作。
* **推送服务**: 执行定时和触发式推送任务。
* **资源服务**: 处理资源上传、导入、审核和推荐。

### 2.2 子模块功能

#### 2.2.1 API服务

* **功能**:
  + 接收并验证请求（PC后台和移动端）。
  + 调用数据服务存储/查询数据。
  + 转发分析请求至模型端，整合返回结果。
* **核心接口**:
  + **用户相关**: 登录、角色管理。
  + **学生相关**: 作业提交、进度查询。
  + **教师相关**: 仪表盘数据、资源上传。
  + **家长相关**: 反馈提交、报告查询。
  + **资源管理**: 上传、导入、审核。
  + **模型管理**: 查看、更新参数。

#### 2.2.2 数据服务

* **功能**:
  + MySQL操作：用户、作业、资源元数据的CRUD。
  + Redis缓存：热点数据（如推荐资源）和会话状态。
  + S3存储：上传文件并生成访问URL。
* **数据表**:
  + users: id, role, name, class\_id, password\_hash
  + homework: id, student\_id, content, file\_url, submit\_time
  + resources: id, type, url, tags, rating, status, uploader\_id
  + feedback: id, parent\_id, content, rating, timestamp

#### 2.2.3 推送服务

* **功能**:
  + 定时推送：每日任务、每周建议。
  + 触发推送：成绩异常、新反馈提醒。
* **实现**:
  + Celery定时任务调度。
  + Redis存储推送队列。

#### 2.2.4 资源服务

* **功能**:
  + 上传资源：保存文件至S3并记录元数据。
  + 导入资源：解析外网URL并存储。
  + 审核流程：管理待审核资源，更新状态。
  + 推荐逻辑：调用模型端匹配资源。

## 3. 接口设计

### 3.1 API接口列表

#### 3.1.1 用户相关

* **POST /auth/login**
  + **描述**: 用户登录，返回JWT令牌。
  + **请求**: { "username": "admin", "password": "123" }
  + **响应**: { "token": "jwt\_token", "role": "admin" }

#### 3.1.2 学生端接口

* **POST /homework**
  + **描述**: 提交作业（含图片和OCR文本）。
  + **请求**: FormData { "image": file, "text": "ocr\_result" }
  + **响应**: { "homework\_id": 1, "status": "submitted" }
* **GET /student/progress**
  + **描述**: 获取学习进度和任务。
  + **响应**: { "trend": [{"date": "2025-02-20", "score": 85}], "tasks": [{"id": 1, "name": "数学作业"}] }
* **GET /resources/recommended**
  + **描述**: 获取推荐资源。
  + **响应**: [{ "id": 1, "url": "s3\_url", "tags": ["math"], "rating": 4.8 }]
* **POST /assistant/ask**
  + **描述**: 提交问题给智能助手。
  + **请求**: { "question": "这题怎么解？" }
  + **响应**: { "answer": "步骤如下..." }
* **GET /student/points**
  + **描述**: 获取积分和排名。
  + **响应**: { "points": 100, "rank": 5 }

#### 3.1.3 教师端接口

* **GET /dashboard**
  + **描述**: 获取仪表盘数据。
  + **响应**: { "students": {"avg\_score": 85, "layers": {"leading": 10}}, "suggestions": ["加练习"], "feedback": ["好评"] }
* **GET /analysis/teacher**
  + **描述**: 获取教学分析和推荐。
  + **响应**: { "coverage": {"math": 60}, "recommendations": [{"id": 1, "url": "s3\_url"}] }
* **POST /upload/resource**
  + **描述**: 上传资源。
  + **请求**: FormData { "file": file, "tags": ["math"] }
  + **响应**: { "resource\_id": 1, "url": "s3\_url" }

#### 3.1.4 家长端接口

* **POST /feedback**
  + **描述**: 提交反馈。
  + **请求**: { "content": "孩子进步了", "rating": 4 }
  + **响应**: { "feedback\_id": 1 }
* **GET /student/report**
  + **描述**: 获取学习报告。
  + **响应**: { "trend": [{"date": "2025-02-20", "score": 85}], "comparison": {"class\_avg": 80} }
* **GET /parent/guidance**
  + **描述**: 获取指导建议和资源。
  + **响应**: { "weaknesses": ["fractions"], "resources": [{"id": 1, "url": "s3\_url"}] }

#### 3.1.5 资源管理接口

* **POST /resources**
  + **描述**: 上传系统资源。
  + **请求**: FormData { "file": file, "tags": ["math"], "type": "video" }
  + **响应**: { "id": 1, "url": "s3\_url" }
* **POST /resources/import**
  + **描述**: 导入外网资源。
  + **请求**: { "url": "http://example.com/video", "tags": ["math"], "type": "video" }
  + **响应**: { "id": 2 }
* **GET /resources/pending**
  + **描述**: 获取待审核资源。
  + **响应**: [{ "id": 3, "url": "s3\_url", "uploader\_id": 5 }]
* **POST /resources/{id}/review**
  + **描述**: 审核资源。
  + **请求**: { "status": "approved" }
  + **响应**: { "id": 3, "status": "approved" }

#### 3.1.6 模型管理接口

* **GET /models**
  + **描述**: 获取模型列表。
  + **响应**: [{ "id": 1, "type": "student", "status": "trained", "updatedAt": "2025-02-20" }]
* **GET /models/{id}**
  + **描述**: 获取模型详情。
  + **响应**: { "parameters": {"learningRate": 0.01}, "metrics": {"accuracy": 0.95} }
* **POST /models/{id}/update**
  + **描述**: 更新模型参数。
  + **请求**: { "parameters": {"learningRate": 0.02} }
  + **响应**: { "status": "updated" }

#### 3.1.7 监控接口

* **GET /metrics**
  + **描述**: 获取系统监控数据。
  + **响应**: { "users": {"active": 500}, "resources": {"downloads": 1000}, "models": {"accuracy": 0.95} }

## 4. 技术实现细节

### 4.1 项目结构

text

WrapCopy

app/ ├── api/ # API路由   
│ ├── auth.py   
│ ├── student.py   
│ ├── teacher.py   
│ ├── parent.py   
│ ├── resources.py   
│ └── models.py   
├── services/ # 服务层   
│ ├── data.py   
│ ├── push.py   
│ └── resource.py   
├── models/ # 数据模型   
│ ├── user.py   
│ └── resource.py   
├── config.py # 配置   
├── tasks.py # Celery任务   
└── main.py # 主入口

### 4.2 示例代码（学生作业提交）

python

WrapCopy

*# app/api/student.py* from flask import Blueprint, request, jsonify from services.data import save\_homework from services.resource import upload\_to\_s3 import requests student\_bp = Blueprint('student', \_\_name\_\_) @student\_bp.route('/homework', methods=['POST']) def submit\_homework(): file = request.files.get('image') text = request.form.get('text') *# 上传文件到S3* file\_url = upload\_to\_s3(file, 'homework') *# 保存到数据库* homework\_id = save\_homework(student\_id=request.user\_id, content=text, file\_url=file\_url)   
*# 调用模型端分析* model\_response = requests.get(f'http://model-service/inference/student?homework\_id={homework\_id}') analysis = model\_response.json()   
return jsonify({'homework\_id': homework\_id, 'status': 'submitted', 'analysis': analysis}) *# app/services/data.py* from models.homework import Homework from config import db def save\_homework(student\_id, content, file\_url): homework = Homework(student\_id=student\_id, content=content, file\_url=file\_url)   
db.session.add(homework)   
db.session.commit() return homework.id

### 4.3 推送服务实现

python

WrapCopy

*# app/tasks.py* from celery import Celery from config import celery\_app from services.push import send\_push\_notification @celery\_app.task def daily\_student\_push(student\_id): tasks = get\_pending\_tasks(student\_id) *# 假设获取待完成任务* send\_push\_notification(student\_id, f"今天有{len(tasks)}个任务待完成") *# 定时任务配置* celery\_app.conf.beat\_schedule = { 'daily-push': { 'task': 'app.tasks.daily\_student\_push', 'schedule': crontab(hour=8, minute=0), *# 每天8点* },   
}

## 5. 非功能性需求

### 5.1 性能

* **并发**: Gunicorn + Nginx支持5000用户在线。
* **响应**: Redis缓存热点数据，API<5秒。
* **异步**: Celery处理推送和批量任务。

### 5.2 安全

* **认证**: JWT验证角色权限。
* **加密**: S3文件加密存储，MySQL密码哈希。
* **日志**: 记录所有请求和错误，便于排查。

### 5.3 可扩展性

* **微服务**: 可拆分为独立服务（资源、推送等）。
* **配置**: 通过config.py支持多环境切换。

## 6. 风险与应对

* **模型延迟**: 设置超时和默认响应。
* **文件上传失败**: 提供重试机制并记录日志。
* **推送丢失**: Redis持久化队列，确保不丢消息。

### 设计亮点

1. **模块化**: 子模块清晰分离，易于维护。
2. **高效**: 异步任务和缓存优化性能。
3. **灵活**: 支持模型端和外部LLM的无缝集成。