福录娃学生获奖智能统计系统

系统设计说明书

版本号: V1.0 **编写日期**: 2025 年 10 月 18 日**团队名称**: 月下调试人

1. 引言

1.1 编写目的

本系统设计说明书旨在为"福录娃学生获奖智能统计系统"提供详细的技术架构设计方案。系统采用 AI 驱动+人工复核的混合模式，实现从奖状图片上传、OCR 识别、智能匹配到自动计分的完整闭环处理流程。

预期读者:

- 系统开发团队成员
- 项目管理人员
- 测试工程师
- 运维工程师
- 技术评审人员

1.2 项目背景

当前高校在学生获奖证书的收集、识别和计分过程中，大量依赖人工手动录入，存在以下主要问题：

- **效率低下**：人工处理耗时耗力
- **差错率高**：手动录入容易出错
- **标准不一**：统计口径缺乏统一规范
- **追溯困难**：缺乏完整的操作记录

为解决这些痛点，本项目提出构建"AI 驱动 + 人工复核"的奖状识别与统计系统。

1.3 术语定义

术语	英文全称	说明
OCR	Optical Character Recognition	光学字符识别技术
CLIP	Contrastive Language–Image Pretraining	图像-文本对比预训练模型
LLM	Large Language Model	大语言模型
ES	Elasticsearch	分布式搜索引擎
SSO	Single Sign-On	单点登录系统
JWT	JSON Web Token	JSON 网络令牌
RBAC	Role-Based Access Control	基于角色的访问控制

2. 系统总体设计

2.1 设计目标

2.1.1 功能性目标

- 实现奖状图片的批量上传和智能识别
- 支持多种奖状格式的 OCR 文字提取
- 提供 AI 驱动的奖项分类和匹配
- 建立人工审核和自动计分机制
- 生成多维度统计报表

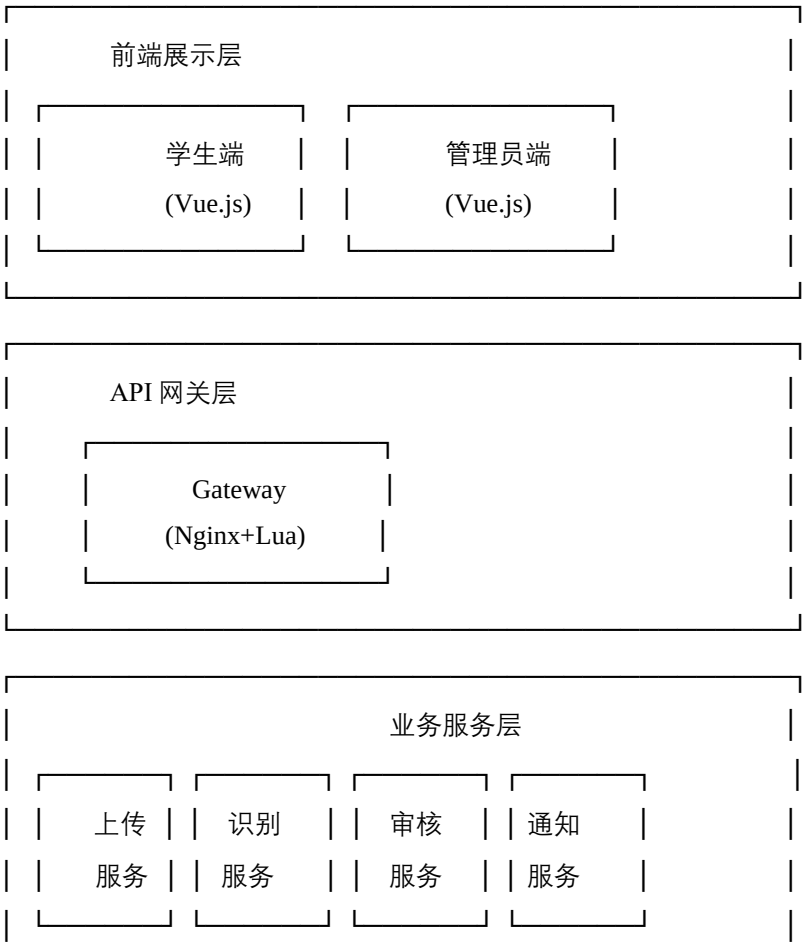
2.1.2 非功能性目标

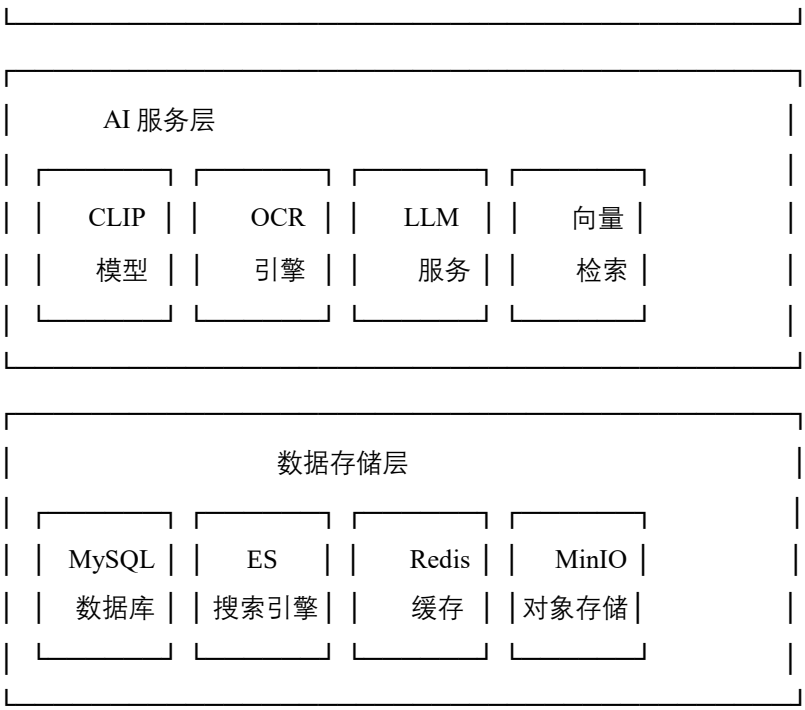
指标类别	具体目标	测量方法
性能	单次上传响应时间 < 2 秒	压力测试
并发	支持 1000+并发用户	负载测试
准确率	AI 识别准确率 ≥ 80%	样本测试
可用性	系统可用性 ≥ 99.9%	监控统计
安全	通过等保三级认证	安全测评

2.2 系统架构设计

2.2.1 总体架构

系统采用分层微服务架构，包含以下层次：





2.2.2 技术栈选择

层次	技术选型	版本	选择理由
前端	Vue.js	3.x	响应式框架，生态成熟
后端	Spring Boot	2.7.x	微服务框架，稳定可靠
数据库	MySQL	8.0	关系型数据库，性能优秀
搜索引擎	Elasticsearch	8.x	分布式搜索，支持向量检索
缓存	Redis	7.x	内存数据库，高性能
消息队列	Redis/RabbitMQ	最新	异步处理，削峰填谷
容器化	Docker + K8s	最新	弹性伸缩，便于运维

---

3. 功能模块设计

3.1 功能模块层次结构

福录娃系统
├── 用户接入层
│ ├── 学生端
│ │ ├── 登录注册
│ │ ├── 奖状上传
│ │ ├── 状态查询
│ │ └── 个人中心

```
│   └── 管理员端
│       ├── 审核管理
│       ├── 奖项库管理
│       ├── 统计分析
│       └── 系统配置
├── 核心业务层
│   ├── 上传管理服务
│   ├── 识别处理服务
│   ├── 审核管理服务
│   └── 通知消息服务
├── AI 服务层
│   ├── CLIP 预判服务
│   ├── OCR 提取服务
│   ├── LLM 分类服务
│   └── 向量检索服务
└── 基础支撑层
    ├── 文件存储服务
    ├── 消息队列服务
    ├── 缓存服务
    └── 搜索引擎服务
```

3.2 核心模块详细设计

3.2.1 上传管理服务

功能描述: 处理奖状图片的上传、验证和存储

主要接口:

- `POST /api/v1/upload` - 文件上传
- `GET /api/v1/upload/status` - 上传状态查询
- `DELETE /api/v1/upload/{fileId}` - 文件删除

技术实现:

- 支持批量上传，单次最多 20 张图片
- 文件格式限制：JPG、PNG、PDF
- 单文件大小限制：10MB
- 图片预处理：压缩、格式转换
- 存储到 MinIO 对象存储

3.2.2 识别处理服务

功能描述: AI 驱动的奖状识别和分类处理

处理流程:

1. **CLIP 预判**: 验证图片是否为有效证书
2. **OCR 提取**: 识别图片中的文字内容
3. **向量转换**: 将文本转换为向量表示
4. **混合检索**: ES 中搜索相似奖项
5. **LLM 分类**: 智能匹配最佳奖项

6. **查重检测**: 避免重复提交

7. **结果入库**: 保存识别结果

性能指标:

- 识别准确率 ≥ 80%
- 处理时间 ≤ 1 分钟
- 支持并发处理 ≥ 50 个任务

3.2.3 审核管理服务

功能描述: 人工审核和结果管理

审核类型:

- **自动通过**: 高置信度匹配直接通过
- **人工审核**: 低置信度或无法匹配的项目
- **批量审核**: 管理员批量处理相似项目

审核操作:

- 通过审核并计分
- 驳回并注明原因
- 关联到正确奖项
- 新增奖项类型

---

4. 接口设计

4.1 接口设计原则

- **RESTful 风格**: 遵循 REST 设计规范
- **统一格式**: 所有接口返回统一的数据格式
- **版本控制**: API 版本号管理，支持向后兼容
- **安全认证**: JWT 令牌机制
- **限流控制**: 防止接口滥用

4.2 数据格式规范

4.2.1 统一响应格式

json

```
{
  "code": 200,
  "message": "success",
  "data": { /* 响应数据 */ },
  "timestamp": 1640995200000,
  "requestId": "uuid"
}
```

4.2.2 错误响应格式

json

```
{
  "code": 400,
  "message": "参数错误",
  "error": {
    "field": "file",
    "message": "文件格式不支持"
  },
  "timestamp": 1640995200000,
  "requestId": "uuid"
}
```

4.3 核心接口定义

4.3.1 文件上传接口

接口地址: `POST /api/v1/upload`

请求参数:

参数名	类型	必填	说明
files	File[]	是	奖状图片文件
studentId	String	是	学生 ID
notes	String	否	备注信息

响应示例:

Json

```
{
  "code": 201,
  "message": "上传成功",
  "data": {
    "taskIds": ["task_20241018_001", "task_20241018_002"],
    "uploadTime": "2024-10-18T10:00:00Z"
  }
}
```

4.3.2 任务查询接口

接口地址: `GET /api/v1/task/{taskId}`

响应示例:

Json

```
{
  "code": 200,
```

```
  "message": "success",
  "data": {
    "taskId": "task_20241018_001",
    "status": "AI_REVIEW",
    "ocrText": "张三 优秀学生奖学金 ...",
    "candidates": [
      {
        "awardId": "award_001",
        "name": "三好学生",
        "score": 5,
        "similarity": 0.92
      }
    ],
    "suggestedAwardId": "award_001",
    "suggestedScore": 5,
    "createTime": "2024-10-18T10:00:00Z"
  }
}
```

4.3.3 审核操作接口

接口地址: `PUT /api/v1/audit/{taskId}`

请求参数:

参数名	类型	必填	说明
action	String	是	操作类型: APPROVE/REJECT/ASSOCIATE
awardId	String	否	关联奖项 ID
reason	String	否	驳回原因

---

5. 数据架构设计

5.1 数据存储架构

系统采用多存储引擎的混合架构，针对不同数据类型选择最适合的存储方案：

数据存储架构

```
├─── 关系型数据 (MySQL)
│    ├─── 用户表 (users)
│    ├─── 奖项表 (awards)
│    ├─── 任务表 (tasks)
│    ├─── 审核记录表 (audit_logs)
│    └─── 统计表 (statistics)
├─── 搜索引擎 (Elasticsearch)
```

```
│    ├──── OCR 文本索引 (ocr_text_index)
│    ├──── 向量索引 (vector_index)
│    └──── 奖项索引 (award_index)
├──── 缓存数据 (Redis)
│    ├──── 会话缓存 (session:{userId})
│    ├──── 任务队列 (task:queue)
│    └──── 热点数据 (hot:data:*)
└──── 文件存储 (MinIO)
     ├──── 原始图片 (original/)
     ├──── 压缩图片 (compressed/)
     └──── 缩略图 (thumbnail/)
```

5.2 数据库设计

5.2.1 用户表 (users)

sql

```
CREATE TABLE users (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    student_id VARCHAR(20) UNIQUE NOT NULL,
    name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE,
    password_hash VARCHAR(255),
    role ENUM('STUDENT', 'ADMIN') DEFAULT 'STUDENT',
    college VARCHAR(100),
    major VARCHAR(100),
    class_name VARCHAR(50),
    total_score INT DEFAULT 0,
    status ENUM('ACTIVE', 'INACTIVE') DEFAULT 'ACTIVE',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    INDEX idx_student_id (student_id),
    INDEX idx_email (email),
    INDEX idx_role (role)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

5.2.2 奖项表 (awards)

sql

```
CREATE TABLE awards (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(200) NOT NULL,
    category VARCHAR(100) NOT NULL,
    level ENUM('NATIONAL', 'PROVINCIAL', 'UNIVERSITY', 'COLLEGE') NOT NULL,
    score INT NOT NULL,
```

```
    aliases JSON,
    keywords JSON,
    description TEXT,
    is_active BOOLEAN DEFAULT TRUE,
    created_by BIGINT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    INDEX idx_category (category),
    INDEX idx_level (level),
    INDEX idx_is_active (is_active),
    FULLTEXT idx_name_desc (name, description)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

5.2.3 任务表 (tasks)

sql

```
CREATE TABLE tasks (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    task_id VARCHAR(50) UNIQUE NOT NULL,
    student_id BIGINT NOT NULL,
    file_name VARCHAR(255) NOT NULL,
    file_url VARCHAR(500) NOT NULL,
    status ENUM('PENDING', 'PROCESSING', 'AI_REVIEW', 'MANUAL_REVIEW',
'APPROVED', 'REJECTED', 'FAILED') DEFAULT 'PENDING',
    ocr_text TEXT,
    ocr_confidence FLOAT,
    suggested_award_id BIGINT,
    suggested_score INT,
    final_award_id BIGINT,
    final_score INT,
    review_status ENUM('PENDING', 'APPROVED', 'REJECTED') DEFAULT
'PENDING',
    reviewer_id BIGINT,
    review_time TIMESTAMP,
    review_comment TEXT,
    error_message TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    INDEX idx_task_id (task_id),
    INDEX idx_student_id (student_id),
    INDEX idx_status (status),
```

```
    INDEX idx_created_at (created_at),
    FOREIGN KEY (student_id) REFERENCES users(id),
    FOREIGN KEY (suggested_award_id) REFERENCES awards(id),
    FOREIGN KEY (final_award_id) REFERENCES awards(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

5.3 Elasticsearch 索引设计

5.3.1 OCR 文本索引

json

```
{
  "mappings": {
    "properties": {
      "taskId": {"type": "keyword"},
      "studentId": {"type": "long"},
      "ocrText": {
        "type": "text",
        "analyzer": "ik_max_word",
        "search_analyzer": "ik_smart"
      },
      "ocrVector": {
        "type": "dense_vector",
        "dims": 768
      },
      "confidence": {"type": "float"},
      "createTime": {"type": "date"}
    }
  },
  "settings": {
    "number_of_shards": 3,
    "number_of_replicas": 1
  }
}
```

---

6. 安全设计

6.1 安全架构

系统采用多层次的安全防护策略，确保数据和服务的安全性：

安全架构层次

├── 网络安全层

```
│   ├── HTTPS/TLS 加密传输
│   ├── WAF 防火墙防护
│   └── DDoS 攻击防护
├── 身份认证层
│   ├── SSO 统一认证
│   ├── JWT 令牌机制
│   └── 多因素认证
├── 权限控制层
│   ├── RBAC 权限模型
│   ├── 细粒度访问控制
│   └── 动态权限验证
├── 数据安全层
│   ├── 数据加密存储
│   ├── 敏感信息脱敏
│   └── 备份恢复机制
└── 操作审计层
    ├── 操作日志记录
    ├── 异常行为监控
    └── 安全事件响应
```

6.2 身份认证与授权

6.2.1 认证流程

1. **用户登录**: 对接学校 SSO 系统
2. **令牌颁发**: 验证成功后颁发 JWT 令牌
3. **请求验证**: 每个 API 请求携带 JWT 令牌
4. **权限验证**: 基于 RBAC 模型验证权限
5. **会话管理**: Redis 存储会话信息

6.2.2 JWT 令牌结构

json

```
{
  "header": {
    "alg": "HS256",
    "typ": "JWT"
  },
  "payload": {
    "userId": 12345,
    "studentId": "2024001",
```

```
    "role": "STUDENT",
    "exp": 1640995200,
    "iat": 1640991600
  }
}
```

6.3 数据安全

6.3.1 数据加密

- **传输加密**: 所有数据传输使用 HTTPS/TLS 1.3
- **存储加密**: 敏感数据使用 AES-256 加密
- **密钥管理**: 使用专门的密钥管理系统

6.3.2 敏感信息保护

- **个人信息**: 姓名、学号等个人信息脱敏显示
- **图片文件**: 私有访问，临时签名 URL
- **操作日志**: 记录所有敏感操作

---

7. 性能设计

7.1 性能指标

指标项	目标值	测试方法
并发用户数	≥ 1000	JMeter 压力测试
上传响应时间	≤ 2 秒	API 性能测试
AI 处理时间	≤ 1 分钟	端到端测试
页面加载时间	≤ 3 秒	前端性能测试
数据库查询时间	≤ 100ms	SQL 性能测试
系统可用性	≥ 99.9%	监控统计

7.2 性能优化策略

7.2.1 前端优化

- **资源压缩**: JS/CSS 文件压缩和合并
- **图片优化**: 自动压缩和格式转换
- **CDN 加速**: 静态资源 CDN 分发
- **懒加载**: 图片和组件懒加载
- **缓存策略**: 浏览器缓存优化

7.2.2 后端优化

- **连接池**: 数据库连接池管理
- **缓存策略**: 多级缓存架构
- **异步处理**: 消息队列异步化
- **代码优化**: 算法和数据结构优化
- **JVM 调优**: 垃圾回收和内存优化

7.2.3 数据库优化

- **索引优化**: 合理创建索引
- **查询优化**: SQL 语句优化
- **读写分离**: 主从复制架构
- **分库分表**: 水平拆分策略
- **连接池**: 数据库连接池

7.3 缓存架构

缓存层次结构

```
├─── L1 缓存 (本地缓存)
│    ├─── 用户会话信息
│    ├─── 系统配置信息
│    └─── 热点数据缓存
├─── L2 缓存 (Redis 集群)
│    ├─── 分布式会话
│    ├─── 业务数据缓存
│    └─── 计算结果缓存
└─── L3 缓存 (CDN)
     ├─── 静态资源
     ├─── 图片文件
     └─── 前端资源
```

---

8. 扩展性设计

8.1 水平扩展能力

8.1.1 微服务扩展

- **服务无状态**: 所有业务服务设计为无状态
- **容器化部署**: Docker 容器化，支持快速扩展
- **负载均衡**: Nginx 负载均衡，支持多实例
- **服务发现**: 服务注册与发现机制

8.1.2 数据库扩展

- **读写分离**: 主从复制，读写分离

- **分库分表**: 按业务模块和数据量分片
- **分布式事务**: 保证数据一致性
- **数据迁移**: 在线数据迁移方案

8.2 垂直功能扩展

8.2.1 插件化架构

- **AI 模型插件**: 支持多种 OCR 和分类模型
- **存储插件**: 支持多种文件存储方式
- **通知插件**: 支持多种消息通知渠道
- **认证插件**: 支持多种认证方式

8.2.2 配置化扩展

- **流程配置**: 审核流程可配置
- **规则引擎**: 匹配规则可配置
- **评分标准**: 计分规则可配置
- **权限配置**: 权限粒度可配置

---

9. 监控与运维

9.1 监控体系

```
监控架构
├─── 基础设施监控
│      ├─── 服务器性能 （CPU、内存、磁盘）
│      ├─── 网络状态 （带宽、延迟、丢包）
│      └─── 中间件状态 （数据库、缓存、消息队列）
├─── 应用性能监控
│      ├─── 接口响应时间
│      ├─── 错误率和异常统计
│      ├─── 业务指标监控
│      └─── 用户体验监控
├─── 日志监控
│      ├─── 应用日志
│      ├─── 访问日志
│      ├─── 错误日志
│      └─── 审计日志
└─── 告警机制
       ├─── 阈值告警
       ├─── 异常告警
       ├─── 故障告警
       └─── 恢复通知
```

9.2 运维自动化

9.2.1 CI/CD 流程

- **代码管理**: Git 版本控制
- **自动构建**: Jenkins 持续集成
- **自动测试**: 单元测试、集成测试
- **自动部署**: 蓝绿部署、滚动部署
- **回滚机制**: 快速回滚策略

9.2.2 运维工具

- **配置管理**: Ansible 自动化配置
- **容器编排**: Kubernetes 集群管理
- **监控告警**: Prometheus + Grafana
- **日志分析**: ELK 日志分析平台
- **链路追踪**: SkyWalking 分布式追踪

---

10. 风险评估与应对

10.1 技术风险

风险描述	影响程度	应对措施
AI 模型准确率不达标	高	模型优化、数据增强、人工兜底
系统性能瓶颈	中	架构优化、缓存策略、水平扩展
第三方服务故障	中	服务降级、备用方案、容错机制
数据安全泄露	高	加密存储、访问控制、安全审计

10.2 业务风险

风险描述	影响程度	应对措施
用户使用率低	中	用户培训、体验优化、推广活动
审核标准争议	中	标准细化、流程透明、沟通机制
数据质量不佳	中	上传引导、质量检查、预处理优化

---

11. 实施计划

11.1 开发阶段划分

阶段	工期	主要工作内容	交付物
第一阶段	4 周	基础架构搭建、核心功能开发	MVP 版本
第二阶段	3 周	AI 模型集成、优化调试	Alpha 版本
第三阶段	3 周	功能完善、性能优化	Beta 版本
第四阶段	2 周	测试验收、部署上线	正式版本

11.2 资源需求

资源类型	数量	配置要求	用途
开发服务器	3 台	8 核 16G 内存	开发环境
测试服务器	2 台	16 核 32G 内存	测试环境
生产服务器	5 台	32 核 64G 内存	生产环境
GPU 服务器	2 台	V100/A100	AI 模型推理
存储空间	10TB	SSD	文件存储

---

12. 总结

本系统设计说明书详细阐述了福录娃学生获奖智能统计系统的整体技术架构和实现方案。通过采用现代化的微服务架构、AI 技术和人工审核相结合的混合模式，系统能够有效解决传统奖状统计工作中的效率低、差错率高、标准不一等问题。

12.1 设计亮点

1. **AI 驱动的智能识别**: 集成多种 AI 模型，实现高准确率的自动识别
2. **灵活的审核机制**: AI 自动处理+人工审核的混合模式
3. **高性能架构**: 微服务+缓存+消息队列的高性能设计
4. **多层次安全保障**: 从网络到数据的全方位安全防护
5. **良好的扩展性**: 支持水平扩展和功能插件化

12.2 预期效果

- **效率提升**: 处理效率提升 80%以上
- **准确率提高**: 识别准确率达到 80%以上
- **标准化管理**: 统一评分标准和审核流程
- **数据可追溯**: 完整的操作记录和审计日志
- **用户体验优化**: 简化操作流程，提升使用便利性

本设计为系统的后续开发和实施提供了清晰的技术指导，确保项目能够按时、高质量地交付使用。
