

# 软件工程案例实践课程管理平台

## 一 详细设计说明书

文档编号: CP-LLD-001

版本号: V1.0

状态: [√] 正式发布

密级: 内部公开

发布日期: 2025-12-23

### 文档变更历史

版本	日期	修改人	修改类型	修改描述
V0.1	2025/12/20	开发组	新建	初始化文档结构, 定义数据库物理模型
V0.2	2025/12/21	开发组	增加	补充后端 Service 层核心算法逻辑
V0.3	2025/12/22	开发组	修改	完善前端组件交互细节与 API 响应格式
V1.0	2025/12/23	项目组	发布	评审通过, 发布正式版

## 1. 引言

### 1.1 编写目的

本文档旨在对“软件工程案例实践课程管理平台”进行详细设计。在概要设计的基础上, 进一步明确系统的数据库物理结构、API 接口参数、后端业务逻辑实现及前端组件细节, 为编码实现、单元测试及系统维护提供精确的指导。

### 1.2 适用范围

本文档适用于后端开发工程师、前端开发工程师、测试工程师及数据库管理员。

## 2. 数据库详细设计

系统采用 SQLite 数据库，利用 Drizzle ORM 定义 Schema。设计核心在于利用 **Triggers** 强制执行业务规则，确保数据在物理层面的完整性。

### 2.1 命名规范

- 表名：小写复数，下划线分隔（如 `user_settings`, `class_students`）。
- 主键：统一为 `id` (INTEGER AUTOINCREMENT)。
- 外键： `target_table_singular_id`（如 `class_id`, `leader_id`）。
- 时间字段：统一存储为 ISO8601 字符串（TEXT 类型），如 `created_at`。

### 2.2 核心数据表结构

#### 2.2.1 用户与鉴权 (User Context)

表名: users

字段名	类型	约束	默认值	说明
id	INTEGER	PK	AutoInc	用户唯一标识
username	TEXT	Unique, NotNull	-	登录账号
password_hash	TEXT	NotNull	-	Argon2id 哈希值
role	TEXT	Enum	'student'	'admin'/'teacher'/'student'
must_change_password	INTEGER	Boolean	1	首次登录强制改密

表名: user\_settings

字段名	类型	约束	说明
user_id	INTEGER	PK, FK(users.id)	1对1关联
active_class_id	INTEGER	FK(classes.id)	用户当前选中的班级上下文 (Session持久化)

#### 2.2.2 团队与项目 (Team & Project Domain)

表名: teams

字段名	类型	约束	默认值	说明
id	INTEGER	PK	AutoInc	-
class_id	INTEGER	FK, NotNull	-	所属班级
leader_id	INTEGER	FK, NotNull	-	队长
status	TEXT	Enum	'recruiting'	'recruiting'/'locked'
is_locked	INTEGER	Boolean	0	核心位：立项后被触发器置为1

表名：projects

字段名	类型	约束	默认值	说明
id	INTEGER	PK	AutoInc	-
team_id	INTEGER	FK, Unique	-	1个团队仅1个项目
status	TEXT	Enum	'draft'	'draft'/'submitted'/'active'/'rejected'
source_type	TEXT	Enum	-	'case_library'/'custom'

● 2.2.3 提交物 (Submissions) ::

表名：submissions

字段名	类型	约束	说明
id	INTEGER	PK	-
assignment_id	INTEGER	FK, NotNull	关联作业
submitter_id	INTEGER	FK, NotNull	提交人
version	INTEGER	NotNull	版本号 (应用层计算 Max+1)
file_id	INTEGER	FK	关联物理文件

索引： `UNIQUE(assignment_id, team_id, submitter_id, version)` 确保版本号唯一。

2.3 数据库触发器实现 ::

触发器 1: `trg_projects_active_lock_team`

- 逻辑：当 `projects.status` 变为 `'active'` 时，强制更新关联的 `teams` 表，设置 `status='locked', is_locked=1`。
- 目的：实现“立项即锁定团队”的硬性业务规则。

触发器 2: `trg_deny_write_when_class_archived`

- 逻辑：在 `teams`, `submissions`, `grades` 等表发生 `INSERT/UPDATE/DELETE` 前，检查 `classes.status`。若为 `'archived'`，抛出 `ABORT` 异常。
- 目的：物理级封锁已结课班级的数据修改。

---

## 3. 后端模块详细设计

后端采用 **Service-Repository** 模式，逻辑与数据访问分离。

### 3.1 认证模块 (AuthService) :

- 登录流程 (`login`):
  1. 接收 `username`, `password`。
  2. 查询 `users` 表。若不存在，返回 401。
  3. 调用 `argon2.verify(hash, password)`。若失败，返回 401。
  4. 生成 JWT Token, Payload 包含 `{ id, role }`。
  5. 返回 `{ token, user }`。

## 3.2 项目管理模块 (ProjectService) ::

- 立项审核 (`reviewProject`):
  - 输入: `projectId`, `decision` ('approve'/'reject')。
  - 逻辑:
    - 权限校验: 仅 Teacher 可操作。
    - 若 `decision === 'approve'`, 更新 `projects.status = 'active'`。
    - 副作用: DB 触发器会自动锁定团队。
    - 副作用: DB 触发器会自动初始化 5 个 `project_stages` 记录。
    - 记录审计日志 `audit_logs`。

## 3.3 作业提交模块 (SubmissionService) ::

- 创建提交 (`createSubmission`):
  - 输入: `assignmentId`, `file` (Multipart), `submitterId`。
  - 逻辑:
    - 文件处理: 计算 SHA256, 存储文件到磁盘 (UUID命名), 写入 `files` 表。
    - 版本控制:

```
1 | const maxVer = await db.select({ v:  
  |   max(submissions.version) })  
2 |                               .from(submissions)  
3 |                               .where(...)  
4 | const newVer = (maxVer[0]?.v || 0) + 1;
```

- 事务写入: 插入 `submissions` 记录。

## 3.4 权限控制 (ABAC) ::

使用 CASL 风格定义细粒度权限:

- **Rule:** `can('update', 'Project', (p, user) => p.team.leaderId === user.id && p.status === 'draft')`
  - 检查点：所有 Service 修改操作前，必须调用 `guard.assert(action, resource)`。
- 

## 4. API 接口规范

RESTful 风格，前缀 `/api/v1`。

### 4.1 通用响应格式

```
1 {
2   "data": { ... },           // 成功时返回的数据
3   "error": null              // 失败时为 null
4 }
```

错误响应：

```
1 {
2   "error": {
3     "code": "CLASS_ARCHIVED_READONLY",
4     "message": "Class is archived.",
5     "details": {}
6   }
7 }
```

## 4.2 核心接口定义 ::

### ● 4.2.1 提交立项 ::

- **Method:** POST
- **URL:** /teams/:teamId/projects
- **Body:**

```
1 {  
2   "name": "Library Mgmt System",  
3   "sourceType": "custom",  
4   "techStack": "Vue + Java"  
5 }
```

json

### ● 4.2.2 教师审核立项 ::

- **Method:** POST
- **URL:** /projects/:id/reviews
- **Body:**

```
1 {  
2   "decision": "approve",  
3   "feedback": "同意立项，请按计划推进。"  
4 }
```

json

### ● 4.2.3 下载文件 ::

- **Method:** GET
  - **URL:** /files/:id/download
  - **Response:** Binary Stream (Blob)。
  - **Header:** Content-Disposition: attachment; filename="report.pdf"
- 

## 5. 前端详细设计

前端基于 Vue 3 + Element Plus, 组件化开发。

### 5.1 全局状态管理 (stores) ::

- **userStore:**
  - **state:** { token, userInfo, roles }
  - **actions:** login(), logout() (清除 Token 及 LocalStorage)。
- **contextStore:**
  - **state:** { activeClassId }
  - 持久化: 每次变更 activeClassId, 调用后端 API 更新 user\_settings 表, 确保跨设备同步。

### 5.2 核心组件实现 ::

#### ● 5.2.1 JsonDiff.vue (版本/审计对比) ::

- 功能: 展示两个 JSON 对象的差异。
- 实现:
  1. Props: oldData, newData.
  2. 利用 diff 算法库计算差异。
  3. 渲染: 使用 <pre> 标签, 新增行加绿色背景类 .bg-green-100, 删除行加红色背景类 .bg-red-100。

## ● 5.2.2 ProjectWorkspace.vue (项目工作台) ::

- 逻辑：
  - 状态机：根据 `project.status` 切换视图。
    - `draft`：显示 `<el-form>` 编辑表单。
    - `active`：显示 `<el-steps>` 阶段导航条。
  - 文件上传：
    - 使用 `<el-upload>`，配置 `http-request` 自定义上传方法。
    - 上传成功后，触发 `refreshSubmissions()` 重新拉取列表。

## ● 5.2.3 文件下载工具 (download.ts) ::

- 实现：
  1. 发起 Axios 请求，`responseType: 'blob'`。
  2. 检查 Blob 类型。若为 `application/json`，说明发生错误（如班级归档禁止下载）。
    - 使用 `FileReader` 读取 Blob 内容，解析 JSON 错误信息，弹出 `ErrorMessage.error`。
  3. 若为文件流，创建 `<a>` 标签，设置 `href = URL.createObjectURL(blob)`，模拟点击下载。

# 6. 安全与审计设计

## 6.1 密码安全 ::

使用 **Argon2id** 算法存储密码，配置参数：

- Memory: 64 MB
- Iterations: 3
- Parallelism: 1
- 避免使用 MD5 或 SHA1 等不安全算法。

## 6.2 审计日志不可篡改

- 机制：`audit_logs` 表仅允许 `INSERT`。
- 防护：数据库层设置 `BEFORE UPDATE` 和 `BEFORE DELETE` 触发器，直接 `RAISE(ABORT)`，防止即使是管理员账号通过 SQL 篡改日志。

## 6.3 归档保护

当 `classes.status = 'archived'` 时，不仅前端禁用按钮，后端 Service 层第一步校验班级状态，数据库层 Trigger 作为最后一道防线拦截写入，实现三级防护。