

**基于python作业管理系统**

**设计与实现**

学生姓名: 杨 心 禾

学生学号: 1230514063

专业班级: 数据科学与大数据技术231

指导老师: 刘宁老师

2025年 7月 31日

**摘要**

本文围绕“学生作业提交系统”的设计与实现展开研究。系统采用Django 5.x作为后端框架，Vue3 + Element-Plus构建前端界面，SQL Server作为数据持久层进行开发。本文首先概述了作业管理系统的背景、目的及意义，分析了当前作业管理的发展趋势及市场需求。在系统设计阶段，明确了以角色（学生/教师）为维度的功能划分：学生端支持作业查看、答案提交与修改；教师端支持作业发布、学生管理、批改评分与统计分析。系统采用RESTful接口设计，后端视图函数以Django原生SQL实现，前端通过Axios完成数据交互，确保前后端解耦与可维护性。其次在系统实现方面，完成了学生登录、作业列表展示、答案提交、作业批改、成绩更新、学生信息维护等核心功能模块，并支持按班级自动分发作业、按教师维度统计提交情况等扩展能力。前端采用Vue3组合式API与Element-Plus组件库，实现了响应式界面与良好的用户体验。最后，系统通过功能测试与边界验证，确认在高并发数据查询与提交场景下具备良好的稳定性与可用性。研究成果为高校教务系统作业管理模块提供了一种轻量级、可复制的技术实现路径，对教育信息化系统的二次开发与集成具有实践参考价值。

**关键词**：Django；Vue3；SQL Server；作业提交系统；教育信息化

**Abstract**

This paper focuses on the research surrounding the design and implementation of a "Student Assignment Submission System." The system is developed using Django 5.x as the backend framework, Vue3 combined with Element-Plus for constructing the frontend interface, and SQL Server as the data persistence layer. Initially, the paper provides an overview of the background, objectives, and significance of the assignment management system, analyzing current trends and market demands in assignment management.During the system design phase, functional divisions based on user roles (students/teachers) were clearly defined: the student interface supports assignment viewing, answer submission, and modification; while the teacher interface enables assignment publishing, student management, grading, and statistical analysis. The system employs RESTful API design, with backend view functions implemented using Django's native SQL. The frontend completes data interaction through Axios, ensuring decoupling and maintainability between the frontend and backend. Secondly, in terms of system implementation, core functional modules such as student login, assignment list display, answer submission, assignment grading, grade updates, and student information maintenance have been completed. The system also supports extended capabilities, including automatic assignment distribution by class and submission statistics by teacher. The frontend utilizes Vue3's Composition API and the Element-Plus component library to achieve a responsive interface and an excellent user experience.Finally, the system has undergone functional testing and boundary validation, confirming its robust stability and usability in high-concurrency data query and submission scenarios. The research findings offer a lightweight, replicable technical implementation pathway for the assignment management module in higher education administrative systems, providing practical reference value for the secondary development and integration of educational informatization systems.

**Keywords**: Django; Vue3; SQL Server; Assignment Submission System; Educational Informatization

目 录

[1 绪论 1](#_Toc165627467)

[1.1 研究背景 1](#_Toc165627468)

[1.2 研究现状 1](#_Toc165627469)

[1.3 研究目标 2](#_Toc165627470)

[2 相关技术介绍 3](#_Toc165627471)

[2.1 PYTHON语言 3](#_Toc165627472)

[2.2 DJANGO框架技术 3](#_Toc165627473)

[2.3 VUE前端技术 4](#_Toc165627474)

[2.4 NODE技术 4](#_Toc165627477)

[2.5 SQL SERVER数据库 5](#_Toc165627479)

[3 需求分析 6](#_Toc165627480)

[3.1 系统功能需求分析 6](#_Toc165627481)

[3.1.1 业务需求分析 6](#_Toc165627482)

[3.1.2 数据需求分析 6](#_Toc165627483)

[3.2 系统非功能性需求分析 7](#_Toc165627484)

[3.2.1 系统处理能力需求 7](#_Toc165627485)

[3.2.2 可靠性需求 7](#_Toc165627486)

[3.2.3 可用性需求 7](#_Toc165627487)

[3.2.4 维修性需求 7](#_Toc165627488)

[3.2.5 环境适配需求 7](#_Toc165627489)

[4 系统设计 8](#_Toc165627490)

[4.1 设计原则 8](#_Toc165627491)

[4.1.1 阶段开发原则 8](#_Toc165627492)

[4.1.2 易用性原则 8](#_Toc165627493)

[4.1.3 业务完整性原则 8](#_Toc165627494)

[4.1.4 业务规范化原则 9](#_Toc165627495)

[4.1.5 可扩展性原则 9](#_Toc165627496)

[4.2 总体架构 10](#_Toc165627497)

[4.3 功能设计 10](#_Toc165627498)

[4.4 数据库设计 11](#_Toc165627499)

[4.4.1 设计规则 1](#_Toc165627500)1

[4.4.2 表结构设计 1](#_Toc165627501)1

[5 系统实现与测试 13](#_Toc165627502)

[5.1 登录功能实现 13](#_Toc165627503)

[5.2 学生端口功能实现 15](#_Toc165627503)

[5.2.1 个人中心模块 1](#_Toc165627504)5

[5.2.2 作业提交模块 1](#_Toc165627505)6

[5.3 教师端口功能实现 1](#_Toc165627509)8

[5.3.1 个人中心模块 1](#_Toc165627510)8

[5.3.2 学生管理模块 1](#_Toc165627511)8

[5.3.3 作业发布模块 2](#_Toc165627512)2

[5.3.4 作业批改模块 2](#_Toc165627513)4

[5.3.5 作业管理模块 2](#_Toc165627514)6

[6 总结与展望 2](#_Toc165627526)9

[参考文献 3](#_Toc165627527)0

# 

# 绪论

## 研究背景

随着高校教学管理信息化进程的不断加快，传统纸质作业提交与批改方式在效率、可追溯性与数据统计等方面暴露出明显短板。尤其在班级规模扩大、课程作业频次提升的背景下，教师面临作业收集、批改、成绩汇总等环节的重复性劳动，学生亦缺乏对作业完成状态的实时掌握与反馈通道。与此同时，校园内已部署的教务系统往往基于早期技术栈构建，虽具备学生、班级、课程等基础数据，但缺乏灵活可扩展的作业管理模块，且数据结构固定、不可更改，给二次开发带来较大阻力。

Python语言因其简洁语法、丰富生态与快速开发特性，在教育信息化领域应用日益广泛；Django作为其主流Web框架，提供了原生SQL支持、RESTful 路由封装及中间件扩展能力，能够在不改动现有数据库表结构的前提下完成业务逻辑的快速实现。Vue3则以响应式数据绑定与组件化开发模式，为构建高交互性前端界面提供了有效支撑。SQL Server作为高校常用的关系型数据库，具备良好稳定性与T-SQL兼容能力，是校园信息系统的主流选择。

在此背景下，开发一套功能完善、性能稳定、用户体验良好的学生作业提交系统，成为缓解教务压力、提升教学效率、推动教育数字化转型的重要切入点。该系统需在保持既有数据表结构的基础上，实现学生作业提交、教师在线批改、成绩即时反馈及统计分析等核心功能，为后续多课程、多班级推广奠定技术基础。

## 研究现状

国内外关于作业管理系统的研究大致可分为三类：一是完全独立的新建系统，采用微服务或云原生架构，功能完备但对既有数据迁移成本高；二是在传统教务系统之上以插件或扩展包形式补充作业模块，技术耦合度低却牺牲功能深度；三是基于低代码平台快速拼装，虽上线周期短，却难以满足个性化需求与性能优化。

在国内，随着“智慧校园”建设的推进，部分高校已上线基于Spring Cloud或.NET Core的作业平台，支持在线答题、自动批改、防抄袭检测等高级特性，然而其系统封闭、与早期教务数据库兼容性差，导致推广受限。学术研究中，学者多关注算法层面，如基于文本相似度的作业查重、基于知识图谱的个性化作业推荐等。

国外高校普遍采用LMS（Learning Management System）如Canvas、Blackboard，其内置Assignment模块功能成熟，然而商业授权费用高、二次开发接口封闭，难以适配国内高校差异化流程。此外，国外开源项目如Moodle虽支持插件式扩展，但PHP技术栈与高校现有SQL Server + Windows环境存在差异，仍需大量适配工作。

综上，现有研究与产品在“对接存量数据库、保持数据结构、快速上线”这一细分场景下缺乏针对性解决方案，需要一个轻量级、技术栈通用、可一键复制的作业管理系统，以填补高校教务信息化链条中的部分空白。

## 研究目标

本研究以高校实际教务需求为牵引，旨在设计并实现一套基于Python + Django + Vue3 的作业管理系统，并在此过程中探究相关技术和实践，从而为作业管理系统的设计和实现提供参考和借鉴。

具体目标如下：

（1）架构目标：在现有SQL Server教务数据库表结构的基础上，利用Django原生SQL与Vue3组合式API，实现系统与既有教务平台并行运行，确保数据完整性。

（2）功能目标：学生端实现作业查看、在线提交、修改答案、成绩查询；教师端实现作业发布、一键分发、逐份批改、成绩更新、学生管理、提交统计及可视化分析，覆盖教学闭环。

（3）性能目标：针对高校课堂并发特征，优化SQL执行计划与事务粒度，确保并发作业提交场景下接口平均响应时间≤1 s，系统可用性≥99 %。

（4）体验目标：前端采用Vue3 + Element-Plus组件库，实现响应式布局与无障碍访问规范；提供一键退出、防重复提交、错误重试等人性化交互。

通过上述目标的达成，本研究期望为教育信息化提供一条“低成本、高兼容、易扩展”的作业管理新路径，并为后续融入智能批改、学习分析等高级功能奠定坚实基础。

# 相关技术介绍

## Python语言

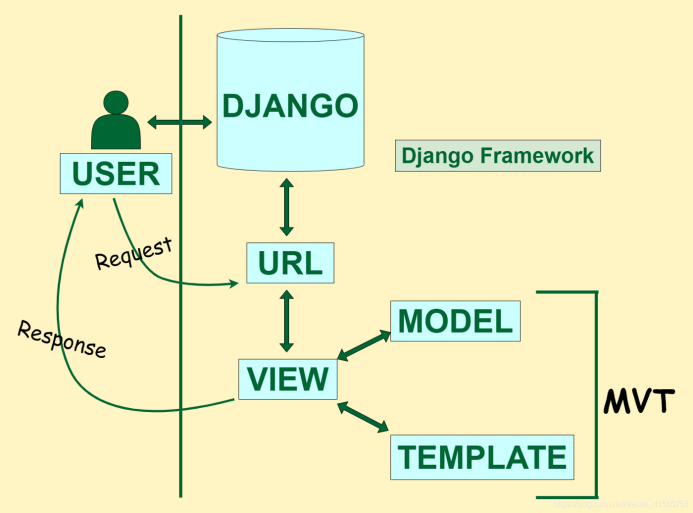
　　Python是一种解释型、面向对象、动态语义的高级通用编程语言，由Guido van Rossum于1991年首次发布。其设计哲学强调“简单优雅”与“代码可读性”，语法层面采用强制缩进而非大括号，显著降低维护成本。Python拥有开源且跨平台的解释器（CPython、PyPy、MicroPython等），可在Windows、Linux、macOS及主流国产操作系统上稳定运行。

　　Python标准库覆盖文件I/O、网络通信、并发编程、文本处理、数据压缩、加密算法等200余个模块，配合pip包管理器，可快速集成45万余个第三方包，形成从Web开发（Django、Flask、FastAPI）到数据科学（NumPy、Pandas、SciPy、TensorFlow、PyTorch）的完整生态。其解释执行模型支持交互式编程，结合Jupyter Notebook实现“所见即所得”的科研与教学场景。

在并发模型上，Python提供多线程、多进程、协程（asyncio）及C扩展回调等多层次机制，可平滑应对从I/O密集型到CPU密集型的负载差异。GIL（全局解释器锁）虽限制多线程并行，但通过多进程、Cython、Numba等技术已有效缓解性能瓶颈。近年来，Python在TIOBE与Stack Overflow全球语言排行榜稳居前三，成为AI、云计算、自动化运维、教育信息化等领域的首选语言。

## Django框架技术

Django诞生于2003年，是遵循MTV（Model-Template-View）设计模式的开源Web框架，由Python Software Foundation维护。其核心理念为“快速开发、安全至上”，通过“开箱即用”的组件体系，显著降低Web应用构建复杂度。



　　Django的ORM层以Python类映射关系型数据库表，支持PostgreSQL、MySQL、SQL Server、Oracle、SQLite等主流后端，并提供数据库迁移、索引优化、连接池、事务控制、分片读写等高级功能；开发者无需书写SQL即可完成80 %以上数据操作。自带的Admin后台基于模型元数据自动生成CRUD界面，极大缩短管理端开发周期。

　　安全方面，Django内置防跨站脚本（XSS）、跨站请求伪造（CSRF）、SQL注入、点击劫持、密码哈希（PBKDF2、Argon2、bcrypt）等多重防御机制，并遵循OWASP Top 10最佳实践。URLconf采用正则或路径转换器映射，配合中间件可完成认证、权限、缓存、日志、CORS、GZip等横切关注点。

此外，Django支持模板继承、表单渲染、国际化（i18n/l10n）、缓存框架（Memcached、Redis、文件、数据库）、信号机制、分布式任务队列（Celery）及REST framework，可平滑扩展至微服务或Serverless架构。其LTS（长期支持）版本提供三年安全更新，保障企业级系统的可持续演进。

## Vue前端技术

　　Vue.js是由尤雨溪于2014年发布的渐进式JavaScript框架，核心库仅关注视图层，通过响应式数据绑定与组合式API实现高效UI构建。Vue3采用Proxy-based响应式系统，重写了虚拟DOM与编译器，运行性能相比Vue2提升1.3–2倍，Tree-shaking后最小仅13 KB（gzip），适用于从传统多页应用到大型单页应用（SPA）的全场景。

　　Vue单文件组件（SFC）将模板、脚本、样式封装在.vue文件内，配合Vite或Webpack可实现热模块替换（HMR）与按需加载。Composition API提供ref、reactive、computed、watchEffect等原子化函数，使逻辑复用粒度更细，易于构建可维护的业务Hook。官方路由Vue Router 4与状态管理Pinia均基于TypeScript重写，支持路由级代码分割、自动类型推导及DevTools调试。

生态层面，Vue社区拥有Element-Plus、Ant-Design-Vue、Naive UI等高质量组件库，覆盖中后台、移动端、小程序多端渲染；Volar、ESLint-plugin-Vue、Vue DevTools提供完备的 IDE支持；Nuxt 3基于Nitro服务端渲染引擎，实现全栈同构与边缘部署。Vue的响应式原理与模板编译技术在Reactivity、AST、虚拟DOM等领域持续输出创新成果，已成为现代Web前端工程化的事实标准之一。

## Node技术

Node.js是基于Chrome V8引擎的JavaScript运行时，由Ryan Dahl于2009年发布，采用事件驱动、非阻塞I/O模型，使其在处理高并发网络请求时具备天然优势。Node统一了前后端语言栈，降低上下文切换成本，使全栈开发成为可能。核心特性包括：

1. 单线程事件循环，避免多线程上下文切换开销；
2. 内置libuv线程池实现异步文件、DNS、加密等操作；
3. 模块化遵循CommonJS/ESM规范，npm仓库已超200万包，生态极度丰富；
4. 跨平台支持Windows、Linux、macOS，可用于服务器、命令行、IoT、桌面应用（Electron）等多场景。

在构建层面，Vite、Webpack、Rollup等工具链均以Node为运行时，对前端工程化提供转译、压缩、热更新、代码分割等能力。Node还可通过Express、Koa、NestJS等框架快速搭建高性能API网关或BFF（Backend For Frontend）层，实现微服务聚合、协议转换、权限校验与缓存加速。

对教育信息化系统而言，Node可作为静态资源服务器、SSR渲染服务、WebSocket实时通知通道，亦可利用GitHub Actions + Node脚本实现CI/CD自动化部署，显著缩短上线周期。

## SQL　Server数据库

　　Microsoft SQL Server是由微软公司开发的关系型数据库管理系统，自1989年发布以来，已演进至2022版，支持Windows、Linux及容器化部署。其企业级引擎采用基于成本的查询优化器（CBO）与列存储索引，支持行存储与列存储混合模式，适用于OLTP与OLAP混合负载。

　　在事务处理方面，SQL Server提供完整的ACID支持，通过锁、行版本控制（RCSI/SI）及Always On可用性组，实现读写分离与秒级故障转移。安全体系涵盖透明数据加密（TDE）、动态数据脱敏（DDM）、行级安全策略（RLS）与Always Encrypted，满足GDPR、国密等合规需求。

　　T-SQL作为扩展SQL标准的过程化语言，支持窗口函数、CTE、JSON/XML 解析、图查询及机器学习服务（SQL Server ML Services），可直接在库内运行Python/R脚本。借助SQL Server Management Studio（SSMS）、Azure Data Studio及Visual Studio Code插件，开发者可完成可视化建模、执行计划分析与自动化部署。

　　在大数据与云原生场景，SQL Server 2022引入智能查询处理（QP）、加速数据库恢复（ADR）及与Azure Synapse Link原生集成，支持将操作数据实时同步至云数据仓库，为教育、金融、制造等行业提供高性能、高可用、易扩展的数据底座。

# 需求分析

## 系统功能需求分析

### 业务需求分析

本系统面向高校日常作业管理场景，覆盖教师、学生两类主体，核心业务流程包括“作业发布→作业提交→批改→成绩反馈→统计分析”闭环。具体需求如下：

（1）用户登录：系统需支持学生、教师两种角色登录，并依据角色加载对应菜单与数据范围，禁止跨角色访问。

（2）作业发布：教师端登录后，可依据授课班级一次性发布作业，系统自动将作业记录写入ASSIGNMENTS表，并通过事务级联在SUBMISSIONS表中为对应班级的每位学生生成待提交记录，同时学生端可以查看到待提交的作业，实现“一键分发”。

（3）作业提交：学生仅能查看所属班级作业列表，支持首次提交与在成绩公布前的多次修改；答案以文本类型存入表SUBMISSIONS，并记录提交时间。

（4）作业批改：教师按班级维度查看已提交作业列表，支持逐条评分或评语，并将成绩写入表SUBMISSIONS。

（5）学生管理：教师可对本班学生进行新增和删除，保证学号唯一性，学生信息变动即时同步至作业表与提交表。

（6）成绩反馈：学生端可以实时查看已批改作业的分数与评语；教师端未批改作业提示“待评阅”，防止重复提交。

（7）统计与归档：教师可按作业维度导出提交率、平均分、未交名单；系统提供柱状图可视化展示班级完成情况，支持学期末一键归档。

### 数据需求分析

SQL Server数据库中已创建STUDENTS、CLASSES、ASSIGNMENTS、SUBMISSIONS四表，系统不得新增或修改字段，仅通过外键及业务逻辑实现关联。数据交互要点：

（1）查询：学生列表需关联CLASSES获取班级名称；作业列表需关联TEACHERS、CLASSES获取教师与班级信息；提交列表需关联STUDENTS、ASSIGNMENTS获取学生姓名与作业标题。

（2）写入：发布作业时，先写入ASSIGNMENTS，再批量插入SUBMISSIONS；提交答案时采用MERGE语句实现“存在即更新，不存在即插入”。

（3）事务：发布作业与批量生成分布记录必须在同一事务内完成；批改成绩需单条更新并立即生效。

（4）索引：利用现有主键、外键及组合索引（ASSIGNMENT\_ID+SNO）保证高频查询性能。

## 系统非功能性需求分析

### 系统处理能力需求

典型班级规模30–120人，作业截止瞬时并发提交峰值100 req/s；查询类接口（作业列表、批改列表）日常并发30 req/s。要求：

（1）接口平均响应时间≤1s，95 线≤2s；

（2）并发100req/s时CPU利用率≤70%，内存占用≤2GB；

（3）支持水平扩展：无状态服务部署多实例，数据库读写分离。

### 可靠性需求

（1）7×24 h连续服务，年可用时间≥99%；

（2）单节点故障自动重启时间<30 s；

（3）事务失败自动回滚，数据一致性满足ACID；

（4）关键操作（成绩更新、学生删除）记录审计日志，支持追溯。

### 可用性需求

（1）首次使用无需培训即可完成登录、提交、批改三大核心任务；

（2）Web UI兼容Chrome、Edge、Firefox最新版与IE11；

（3）提供全局错误提示与重试机制，前端表单实时校验；

（4）支持会话保持8 h，异常断网后自动刷新页面恢复上下文。

### 维修性需求

（1）代码结构：后端按Django app拆分，前端按功能路由懒加载；

（2）配置集中化：数据库、日志、跨域等参数统一放置settings.py与.env；

（3）日志：采用Django + RotatingFileHandler，保留30天，支持按模块级别开关；

（4）升级策略：滚动部署，旧实例优雅退出，用户无感知；新增接口向下兼容，数据库零迁移。

### 环境适配需求

（1）服务器：Windows Server 2019 / CentOS 7+，2 v CPU 4 GB RAM起；

（2）运行时：Python 3.12、Node 18、SQL Server 2017+ 或Azure SQL；

（3）网络：B/S架构，校园网/公网均可访问，HTTPS强制开启；

（4）客户端：PC端主流浏览器，移动端自适应，无需安装APP。

# 系统设计

## 设计原则

### 阶段开发原则

（1）原型验证阶段：聚焦学生提交与教师批改两条主路径，采用Django内置runserver与SQLite本地联调，快速收集一线师生反馈。

（2）核心功能阶段：补充班级管理、作业归档、成绩导出等刚需功能，数据库切换至SQL Server并完成索引优化；同步在测试环境部署Nginx + Gunicorn，验证并发能力。

（3）性能优化阶段：引入Redis缓存热点查询、SQL Server读写分离、Gunicorn多进程，保障100并发下P95延迟≤1 s。

（4）运营支撑阶段：长期演进，围绕教学评估、数据治理、安全加固持续迭代；提供灰度发布与回滚脚本，确保新功能零中断上线。每个阶段结束均产出可运行镜像与更新日志，通过GitHub Actions自动推送到测试仓库。

### 易用性原则

（1）学生端：首页→作业卡片→提交按钮，三步完成；提交后立即弹出成功提示并刷新状态标签，避免用户反复刷新页面。

（2）教师端：采用“一页式工作台”，发布、批改、统计、管理四大Tab横向排列，关键信息置顶，批改弹窗内嵌答案与评分框，减少页面跳转。

（3）防错机制：所有删除操作均触发二次确认；表单实时校验（学号位数、作业编号重复、分数范围），错误信息红色高亮并定位到字段；网络异常统一Toast提示并附带重试按钮。

（4）无障碍支持：键盘可达性、ARIA标签、色彩对比度符合WCAG 2.1 AA；提供高对比主题与字号缩放，照顾视觉障碍用户。

### 业务完整性原则

覆盖作业生命周期的七个关键节点：发布→自动分发→学生提交→教师批改→成绩回传→学生查看→归档统计，任何节点出现异常均具备补偿机制：

（1）发布：事务级联插入ASSIGNMENTS与SUBMISSIONS，失败自动回滚并提示“发布失败，请重试”。

（2）提交：采用MERGE语句保证首次插入与后续更新原子性；若学生重复提交，记录UPDATE\_TIME供审计。

（3）批改：成绩写入时同步更新缓存与统计视图；若教师误操作，可在成绩锁定前撤销并回滚SCORE。

（4）归档：学期结束触发存储过程SP\_ARCHIVE，将历史作业移至分区表，主库保留近一年数据，既保证性能又满足教务审计。

（5）可追溯：所有关键操作写入OPERATION\_LOG（操作者、时间、IP、前后值差异），支持一键追溯与责任定位。

### 业务规范化原则

全程对接高校教务标准，确保数据、流程、接口三统一：

（1）命名规范：数据库字段、API路径、前端变量全部采用英文缩写大写加下划线，如ASSIGNMENT\_TITLE、/api/assignments/publish/，杜绝歧义。

（2）身份编码：学号、工号、班级号、作业号保持与教务系统一致，学号10位、工号6位、班级号7位，正则校验防止脏数据。

（3）时间统一：所有时间字段统一使用UTC+8存储，前端按用户时区格式化；支持夏令时自动转换。

（4）成绩规范：提供百分制（0-100，保留一位小数）与等级制（优秀/良好/中等/及格/不及格）两种模式，配置文件切换即可，兼容旧数据。

（5）流程合规：作业截止后自动锁定提交按钮，教师批改结果不可学生自行修改，符合“教师主权”制度；成绩导出格式与教务处模板字段顺序完全一致，实现“一键上报”。

### 可扩展性原则

在架构、代码、数据三个层面预留扩展点，确保未来3-5年教学需求变化无需重构：

（1）后端接口：所有视图函数继承自APIView，统一使用序列化器；新增题型只需在assignments模块下新增serializer与viewset，路由自动注册。

(2）前端组件：基于Element-Plus二次封装业务组件（作业卡片、评分弹窗、图表面板），新增功能通过插槽或props注入即可；路由采用懒加载，降低首屏体积。

（3）数据表：

① ASSIGNMENTS预留EXTEND\_JSON (NVARCHAR(MAX))字段，可动态存储题型配置、附件列表、防作弊策略；

② STUDENTS预留AUTH\_PROVIDER (NVARCHAR(20))字段，未来可接入LDAP、OAuth2、企业微信等第三方身份源；

③ SUBMISSIONS预留ATTACHMENT\_URL (NVARCHAR(500))字段，支持附件上传后快速启用。

（4）插件机制：

① 后端：基于Django App可插拔设计，新增“在线考试”App 仅需在INSTALLED\_APPS中注册；

② 前端：基于Vite动态import，考试模块路由懒加载，不增加首屏负担。

（5）消息推送：预留WebSocket网关接口，后续可无缝接入成绩实时推送、作业截止倒计时提醒、系统公告等功能。

通过上述五项原则的细化落地，系统在保证当前教学业务稳定运行的同时，为后续功能迭代、技术升级、教学创新奠定了坚实基础。

## 总体架构

系统采用经典B/S三层架构：表现层、业务逻辑层、数据持久层，整体技术栈如下：

（1）表现层：Vue3 + Element-Plus + Vite，SPA单页应用，响应式布局适配PC与平板；

（2）网关层：Nginx反向代理，负责静态资源分发、HTTPS终端、负载均衡；

（3）业务逻辑层：Django 5.x（Python 3.11），RESTful API设计，按功能细分为users、assignments、submissions、classes四大应用；

（4）数据持久层：SQL Server 2022，主从读写分离，事务级一致性保障；

（5）外部依赖：SMTP服务用于成绩通知、Redis可选缓存热点查询。

部署时支持传统虚拟机与容器双模式：Dockerfile与 docker-compose.yml 已内置，亦提供 Helm Chart 供 Kubernetes 集群一键部署。

## 功能设计

功能以角色为主线，划分为“学生中心、教师中心”两大子系统，共12个核心模块：

（1）学生中心

① 登录权限：学号 + 姓名与数据库内表STUDENTS匹配，成功后登录学生端口；

② 个人中心：展示学籍、班级、联系方式。

③ 作业列表：按班级过滤，展示作业标题、描述、发布教师、提交状态、得分；

④ 提交/修改：答案以文本形式输入，支持多次修改（已批改则锁定，无法修改）；

⑤ 成绩查询：实时获取SCORE字段，支持评语显示；

（2）教师中心

① 登录权限：工号 + 姓名与数据库内表TEACHERS匹配，成功后登录教师端口；

② 个人中心：展示工号、教学班级、联系方式；

③ 班级管理：新增/删除学生、批量导入、一键同步班级名单；

④ 作业发布：填写标题、描述、选择班级，系统自动生成ASSIGNMENT\_ID并批量插入SUBMISSIONS；

⑤ 作业列表：教师维度查看已发布作业，可删除已发布的作业（级联清除SUBMISSIONS）；

⑥ 作业批改：列表视图展示未批改/已批改，点击弹出学生答案与评分框，支持评语；

⑦ 统计分析：按作业维度生成柱状图、提交率、未提交名单，支持导出CSV；

## 数据库设计

### 设计规则

（1）范式：满足第三范式，消除冗余；

（2）命名：表名、字段名均使用大写英文缩写，主键统一命名为PK\_表名；

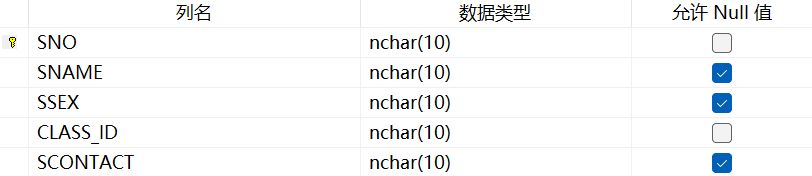
（3）类型：整型用INT，可变字符用NVARCHAR，日期用DATETIME；

（4）索引：外键自动索引，高频查询列建立复合索引；

（5）安全：开启TDE透明加密，备份保留30天；

### 表结构设计

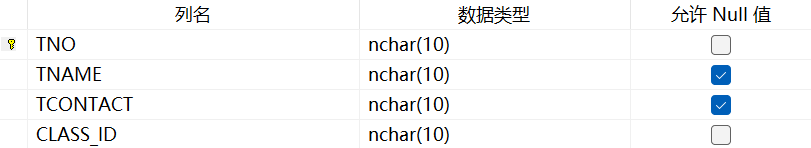
（1）STUDENTS（学生表）



①SNO｜学生学号（主键）；②SNAME｜学生姓名；③SSEX｜学生性别；

④CALSS\_ID｜学生所在班级；⑤SCONTACT｜学生联系方式；

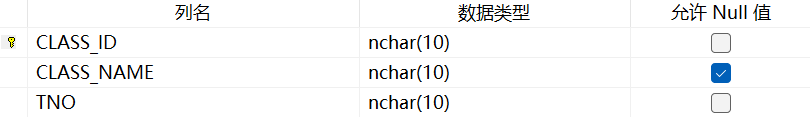
1. TEACHERS（教师表）



①TNO｜教师工号（主键）；②TNAME｜教师姓名；③CONTACT｜教师联系方式；

④CALSS\_ID｜教师教学班级；

1. CLASSES（班级表）



①CLASS\_ID｜班级ID（主键）；②CLASS\_NAME｜班级名称；③TNO｜班级教学老师；

（4）ASSIGNMENTS（作业表）



①ASSIGNMENT\_ID｜作业ID（主键）；②ASSIGNMENT\_TITLE｜作业标题；

③DESCRIPTION｜作业内容；④PUBLISH\_TIME｜作业发布时间；

⑤CLASS\_ID｜作业面向班级ID；⑥TNO｜作业发布老师；

（5）SUBMISSIONS（提交表）



①SUBMISSION\_ID｜作业提交ID（主键）；②ASSIGNMENT\_ID｜作业ID；

③SNO｜学生学号；④SUBMI\_TITLE｜作业提交时间；⑤ANSWER｜学生作业答案；

⑥SCORE｜作业得分；

# 系统实现与测试

## 登录功能实现

登录模块负责学生、教师双角色的身份鉴别，依据学号/工号与姓名在STUDENTS、TEACHERS表中完成匹配，成功后返回角色信息并写入前端localStorage，为后续权限路由提供依据，同时保证非法访问直接重定向至登录页。



1. 后端根据角色选择查询表，命中即返回序列化后的用户字典;

|  |
| --- |
| works-server/apps/users/views.py |
| @api\_view(['POST'])  def login(request):  role = request.data.get('role')  uid = request.data.get('uid')  name = request.data.get('name')  if role == 'student':  user = StudentSerializer.get\_student(uid, name)  return Response({'role': 'student', 'data': user} if user else {'error': '无此学生'})  if role == 'teacher':  with connection.cursor() as cursor:  cursor.execute("""  SELECT TNO, TNAME, TCONTACT  FROM TEACHERS  WHERE TNO = %s AND TNAME = %s  """, [uid, name])  row = cursor.fetchone()  if row:  return Response({'role': 'teacher', 'data': {  'tno': row[0],  'tname': row[1],  'tcontact': row[2]  }})  return Response({'error': '无此教师'})  return Response({'error': '参数错误'}) |

1. 前端收集表单，核对个人信息，成功后跳转对应首页;

|  |
| --- |
| works-web/src/views/Login.vue |
| <el-form @submit.prevent="login">  <!-- 身份 -->  <el-form-item label="身份">  <el-radio-group v-model="form.role">  <el-radio value="student">学生</el-radio>  <el-radio value="teacher">教师</el-radio>  </el-radio-group>  </el-form-item>  <!-- 账号 -->  <el-form-item label="账号">  <el-input v-model="form.uid" placeholder="学号/工号" />  </el-form-item>  <!-- 姓名 -->  <el-form-item label="姓名">  <el-input v-model="form.name" placeholder="真实姓名" />  </el-form-item>  <!-- 登录按钮 -->  <el-button type="primary" native-type="submit" :loading="loading">  {{ loading ? '登录中…' : '登录' }}  </el-button>  </el-form> |

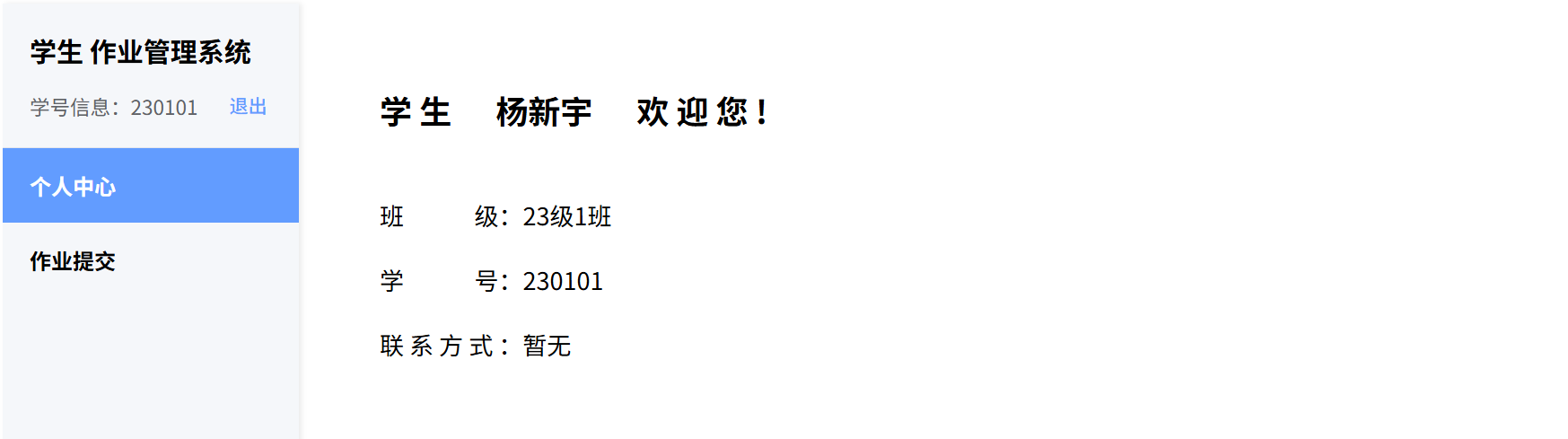
1. 前端登录时进行防抖提交，防止页面提交过多而卡死;

|  |
| --- |
| works-web/src/views/Login.vue |
| const login = () => {  if (!form.uid.trim() || !form.name.trim()) {  ElMessage.warning('请完整填写')  return  }  if (loading.value) return // 防抖：防止连点  loading.value = true  axios.post('/api/login/', {  role: form.role,  uid: Number(form.uid.trim()),  name: form.name.trim()  })  .then(({ data }) => {  if (data.error) {  ElMessage.error(data.error)  } else {  localStorage.setItem('user', JSON.stringify(data.data))  localStorage.setItem('role', form.role)  router.push(`/${form.role}/home`)  }  })  .catch(() => {  ElMessage.error('网络异常')  })  .finally(() => {  loading.value = false  })  }  </script> |

## 学生端口功能实现

### 个人中心模块

个人中心模块中，从表STUDENTS中调出学生个人信息，在教师端的个人中心里展示学生学号、学生姓名、所属班级、联系方式，提供退出登录按钮，与学生导航栏联动。



|  |
| --- |
| works-web/src/views/StudentHome.vue |
| <div class="info-text">  <p class="welcome">  学生{{ user.sname }} 欢迎您!  </p>  <p>班级：{{ user.class\_name || '暂无' }}</p>  <p>学号：{{ user.sno }}</p>  <p>联系方式：{{ user.scontact || '暂无' }}</p>  </div> |

### 作业提交模块



① 后端：

|  |
| --- |
| works-server/apps/assignments/views.py |
| @api\_view(['GET'])  def my\_assignments(request):  tno = request.GET.get('tno')  if not tno:  return Response([])  sql = """  SELECT A.ASSIGNMENT\_ID AS assignment\_id,  A.ASSIGNMENT\_TITLE AS assignment\_title,  A.DESCRIPTION AS description,  FORMAT(A.PUBLISH\_TIME, 'yyyy-MM-dd HH:mm') AS publish\_time,  C.CLASS\_NAME AS class\_name  FROM ASSIGNMENTS A  JOIN CLASSES C ON C.CLASS\_ID = A.CLASS\_ID  WHERE A.TNO = %s  ORDER BY A.PUBLISH\_TIME DESC  """  with connection.cursor() as cursor:  cursor.execute(sql, [tno])  columns = [col[0] for col in cursor.description]  rows = cursor.fetchall()  return Response([dict(zip(columns, row)) for row in rows]) |

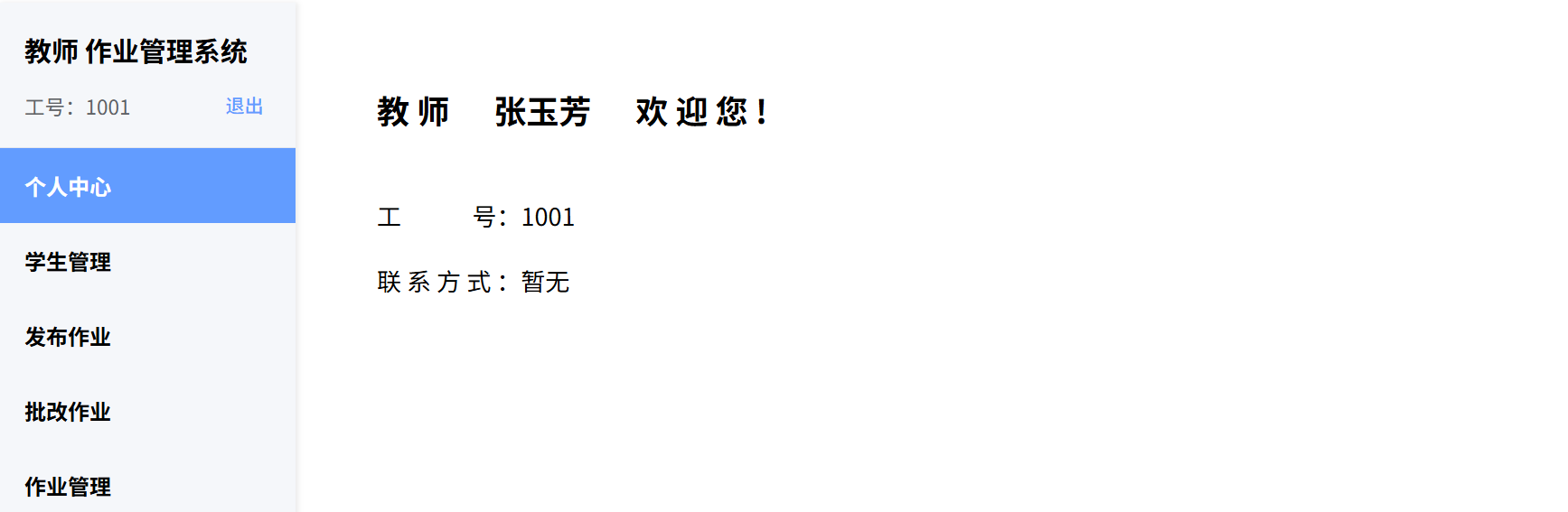
② 前端：

|  |
| --- |
| works-web/src/views/TeacherStudents.vue |
| <el-table-column prop="sno" label="学号" />  <el-table-column prop="sname" label="姓名" />  <el-table-column prop="ssex" label="性别" />  <el-table-column prop="class\_name" label="班级" />  <el-table-column label="联系方式">  <template #default="scope">  {{ scope.row.scontact || '暂无' }}  </template>  <el-card shadow="always" style="border-radius: 8px; overflow: hidden; margin-top: 16px;" >  <el-table :data="assignments" stripe style="width: 100%">  <el-table-column prop="assignments\_title" label="作业标题" width="160" />  <el-table-column prop="description" label="作业内容" width="280" show-overflow-tooltip />  <el-table-column prop="publish\_time" label="发布时间" width="180" />  <el-table-column prop="teacher" label="发布老师" width="140" />  <el-table-column label="状态" width="160">  <template #default="scope">  <div style="display: flex; align-items: center; gap: 4px;">  <el-tag v-if="scope.row.submitted === 1" type="success">已提交</el-tag>  <el-tag v-else type="info">未提交</el-tag>  <span  v-if="scope.row.score && scope.row.score.trim()"  style="color:#409eff;font-weight:bold;white-space:nowrap"  >  {{ scope.row.score.trim() }}  </span>  </div>  </template>  </el-table-column>  <el-table-column label="操作" width="110">  <template #default="scope">  <!-- 未提交，可提交 -->  <el-button v-if="scope.row.submitted === 0" type="primary" size="small"  @click="openModal(scope.row, false)"  >  提交  </el-button>  <!-- 已提交且已批改，不可修改 -->  <el-button v-else-if="scope.row.score !== null" type="info" size="small" disabled>  不可修改  </el-button>  <!-- 已提交但未批改，可修改 -->  <el-button v-else type="warning" size="small" @click="openModal(scope.row, true)">  修改  </el-button>  </template>  </el-table-column>  </el-table>  </el-card> |

## 教师端口功能实现

### 个人中心模块

个人中心模块中，从表TEACHERS中调出教师个人信息，在教师端的个人中心里展示教师工号、姓名、联系方式，提供退出登录按钮，与教师导航栏联动。



|  |
| --- |
| works-web/src/views/TeacherHome.vue |
| <div class="info-text">  <p class="welcome">  教师; {{ user.tname }} 欢迎您!  </p>  <p>工号：{{ user.tno }}</p>  <p>联系方式：{{ user.tcontact || '暂无' }}</p>  </div> |

### 学生管理模块

（1）学生管理模块中，教师可按班级查看学生列表，通过表STUDENTS中各学生的班级ID与教师所教授的班级ID相链接，将该教师所管理的班级学生所展示，便于新增与删除学生。



① 后端：

|  |
| --- |
| works-server/apps/users/views.py |
| @api\_view(['GET'])  def students\_list(request):  tno = request.GET.get('tno')  if not tno:  return Response({'error': '缺少 tno 参数'}, status=400)  sql = """  SELECT SNO AS sno,  SNAME AS sname,  SSEX AS ssex,  C.CLASS\_NAME AS class\_name,  SCONTACT AS scontact  FROM students S  INNER JOIN classes C ON C.CLASS\_ID = S.CLASS\_ID  INNER JOIN teachers T ON T.CLASS\_ID = S.CLASS\_ID  WHERE T.TNO = %s  ORDER BY S.SNO  """  with connection.cursor() as cursor:  cursor.execute(sql, [tno])  columns = [col[0] for col in cursor.description]  rows = cursor.fetchall()  data = [dict(zip(columns, row)) for row in rows]  return Response(data) |

② 前端：

|  |
| --- |
| works-web/src/views/TeacherStudents.vue |
| <el-table-column prop="sno" label="学号" />  <el-table-column prop="sname" label="姓名" />  <el-table-column prop="ssex" label="性别" />  <el-table-column prop="class\_name" label="班级" />  <el-table-column label="联系方式">  <template #default="scope">  {{ scope.row.scontact || '暂无' }}  </template>  </el-table-column> |

（2）学生管理模块中，教师端对学生支持单条新增，在网页新增学生后，数据库内表STUDENTS中插入新加入的学生个人信息，实现网页与数据库同步。



① 后端：

|  |
| --- |
| works-server/apps/users/views.py |
| @api\_view(['POST'])  def student\_add(request):  sno = request.data.get('sno')  sname = request.data.get('sname')  ssex = request.data.get('ssex')  class\_id = request.data.get('class\_id')  scontact = request.data.get('scontact') or None  if not all([sno, sname, ssex, class\_id]):  return Response({'error': '缺少必填字段'}, status=400)  try:  with connection.cursor() as cursor:  cursor.execute("""  INSERT INTO STUDENTS (SNO, SNAME, SSEX, CLASS\_ID, SCONTACT)  VALUES (%s, %s, %s, %s, %s)  """, [sno, sname, ssex, class\_id, scontact])  return Response({'message': '学生已新增'})  except Exception as e:  return Response({'error': str(e)}, status=500) |

② 前端：

|  |
| --- |
| works-web/src/views/TeacherStudents.vue |
| <el-dialog v-model="showAdd" title="新增学生" width="420">  <el-form :model="addForm" label-width="80">  <el-form-item label="学号" required>  <el-input v-model="addForm.sno" placeholder="请输入学号" />  </el-form-item>  <el-form-item label="姓名" required>  <el-input v-model="addForm.sname" placeholder="请输入姓名" />  </el-form-item>  <el-form-item label="性别" required>  <el-select v-model="addForm.ssex" placeholder="请选择性别">  <el-option label="男" value="男" />  <el-option label="女" value="女" />  </el-select>  </el-form-item>  <el-form-item label="班级" required>  <el-input v-model="addForm.class\_id" placeholder="请输入班级id">  <el-option v-for="c in classes" :key="c.class\_id" :label="c.class\_name" :value="c.class\_id"/>  </el-input>  </el-form-item>  <el-form-item label="联系方式">  <el-input v-model="addForm.scontact" placeholder="选填" />  </el-form-item>  </el-form>  <template #footer>  <el-button @click="showAdd = false">取消</el-button>  <el-button type="primary" @click="confirmAdd">确认</el-button>  </template>  </el-dialog> |

（3）学生管理模块中，教师端对学生支持单条删减，在网页上点击删除后，数据库内表STUDENTS中会删除该学生的个人信息，实现网页与数据库同步。

① 后端：

|  |
| --- |
| works-server/apps/users/views.py |
| @api\_view(['DELETE'])  def student\_delete(request, sno):  try:  with connection.cursor() as cursor:  cursor.execute("DELETE FROM STUDENTS WHERE SNO = %s", [sno])  return Response({'message': '学生已删除'})  except Exception as e:  return Response({'error': str(e)}, status=500) |

② 前端：

|  |
| --- |
| works-web/src/views/TeacherStudents.vue |
| <el-table-column label="操作" width="80">  <template #default="scope">  <el-popconfirm title="确定删除该学生？" @confirm="remove(scope.row.sno)">  <template #reference>  <el-button type="danger" link>删除</el-button>  </template>  </el-popconfirm>  </template>  </el-table-column> |

### 作业发布模块

（1）作业发布模块中，教师通过发布功能一键发布作业并在数据库中表ASSIGNMENTS中添加该数据，同时在表SUBMISSIONS中自动为该班学生生成待提交记录并在学生端口的作业提交模块显示该名学生的未提交数据。



① 后端：

|  |
| --- |
| works-server/apps/assignments/views.py |
| @api\_view(['POST'])  @transaction.atomic  def publish\_assignment(request):  tno = request.data.get('tno')  assignment\_id = str(request.data.get('assignment\_id')).strip()  title = request.data.get('title')  description = request.data.get('description')  class\_id = str(request.data.get('class\_id')).strip()  if not all([assignment\_id, tno, title, description, class\_id]):  return Response({'error': '参数缺失'}, status=400)  try:  with connection.cursor() as cursor:  cursor.execute("""  INSERT INTO ASSIGNMENTS  (ASSIGNMENT\_ID, ASSIGNMENT\_TITLE, DESCRIPTION, PUBLISH\_TIME, CLASS\_ID, TNO)  VALUES (%s, %s, %s, GETDATE(), %s, %s)  """, [assignment\_id, title, description, class\_id, tno])  cursor.execute("""  INSERT INTO SUBMISSIONS (SUBMISSION\_ID, ASSIGNMENT\_ID, SNO, SUBMIT\_TIME)  SELECT CONCAT(%s, '\_', SNO), %s, SNO, NULL  FROM STUDENTS  WHERE CLASS\_ID = %s  """, [assignment\_id, assignment\_id, class\_id])  return Response({'message': '发布并分发成功'})  except Exception as e:  return Response({'error': str(e)}, status=500) |

② 前端：

|  |
| --- |
| works-web/src/views/TeacherStudents.vue |
| <el-card style="margin-bottom: 24px">  <el-form :model="form" label-width="100" @submit.prevent="publish">  <el-form-item label="作业ID" required>  <el-input v-model="form.assignment\_id" placeholder="请输入作业ID" />  </el-form-item>  <el-form-item label="作业标题" required>  <el-input v-model="form.title" />  </el-form-item>  <el-form-item label="作业内容" required>  <el-input v-model="form.description" type="textarea" :rows="4" />  </el-form-item>  <el-form-item label="班级ID" required>  <el-input v-model="form.class\_id" placeholder="请输入班级ID" />  </el-form-item>  <el-button type="primary" native-type="submit">发布</el-button>  </el-form>  </el-card> |

（2）作业发布模块中，教师通过删除功能一键删除已发布的作业并在数据库中表ASSIGNMENTS中删除该数据，同时在表SUBMISSIONS中删除已为该班学生生成待提交记录。

① 后端：

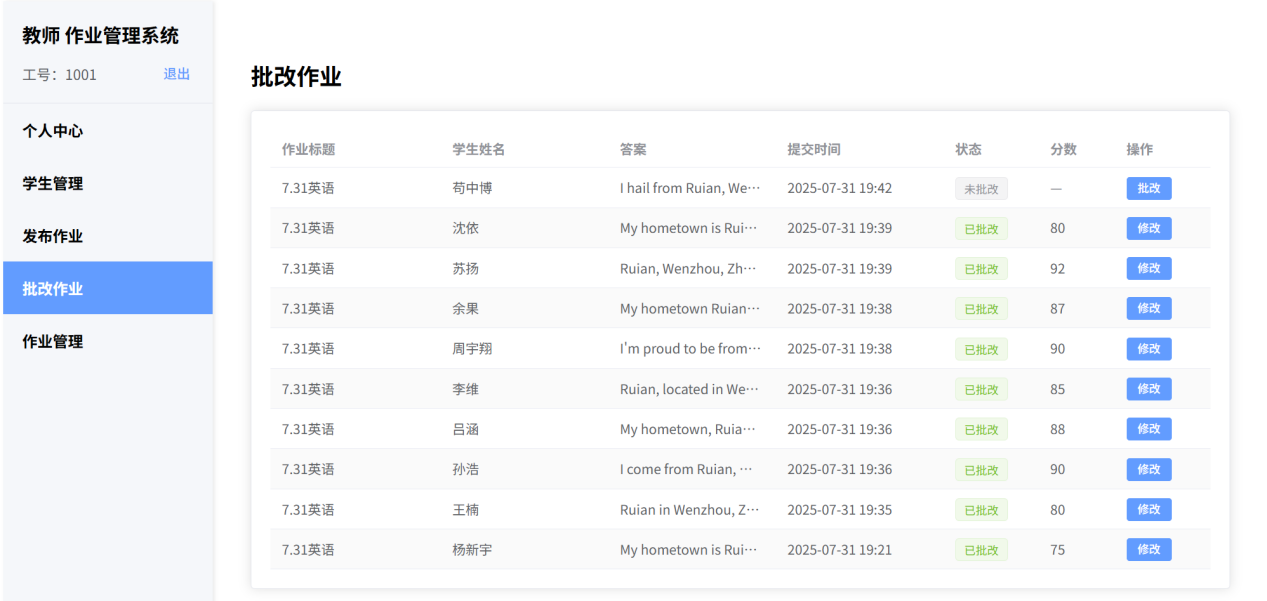
|  |
| --- |
| works-server/apps/assignments/views.py |
| @api\_view(['DELETE'])  def delete\_assignment(request, assignment\_id):  try:  with connection.cursor() as cursor:  cursor.execute("DELETE FROM ASSIGNMENTS WHERE ASSIGNMENT\_ID = %s", [assignment\_id])  cursor.execute("DELETE FROM SUBMISSIONS WHERE ASSIGNMENT\_ID = %s", [assignment\_id])  return Response({'message': '作业及提交记录已删除'})  except Exception as e:  return Response({'error': str(e)}, status=500) |

② 前端：

|  |
| --- |
| works-web/src/views/TeacherStudents.vue |
| <el-table-column label="操作" width="80">  <template #default="scope">  <el-popconfirm title="确定删除该作业？" @confirm="del(scope.row.assignment\_id)">  <template #reference>  <el-button type="danger" link>删除</el-button>  </template>  </el-popconfirm>  </template>  </el-table-column> |

### 作业批改模块

作业批改模块中，教师可以查看已提交的作业列表，同时在弹窗批改、实时回写成绩并将成绩写入表SUBMISSIONS中，学生端可以同步查看到该作业的分数。





① 后端：

|  |
| --- |
| works-server/apps/submissions/views.py |
| @api\_view(['GET'])  def to\_mark\_list(request):  tno = request.GET.get('tno')  if not tno:  return Response({'error': '缺少 tno 参数'}, status=400)  sql = """  SELECT  S.SUBMISSION\_ID AS submissions\_id,  A.ASSIGNMENT\_TITLE AS assignments\_title,  ST.SNAME AS sname,  CAST(S.ANSWER AS NVARCHAR(MAX)) AS answer,  FORMAT(S.SUBMIT\_TIME, 'yyyy-MM-dd HH:mm') AS submit\_time,  S.SCORE AS score  FROM SUBMISSIONS S  JOIN ASSIGNMENTS A ON A.ASSIGNMENT\_ID = S.ASSIGNMENT\_ID  JOIN STUDENTS ST ON ST.SNO = S.SNO  JOIN CLASSES C ON C.CLASS\_ID = ST.CLASS\_ID  JOIN TEACHERS T ON T.CLASS\_ID = C.CLASS\_ID  WHERE S.ANSWER IS NOT NULL  AND CAST(S.ANSWER AS NVARCHAR(MAX)) != ''  AND T.TNO = %s  ORDER BY S.SUBMIT\_TIME DESC  """  with connection.cursor() as cursor:  cursor.execute(sql, [tno])  columns = [col[0] for col in cursor.description]  rows = cursor.fetchall()  return Response([dict(zip(columns, row)) for row in rows]) |

② 前端：

|  |
| --- |
| works-web/src/views/TeacherMark.vue |
| <el-dialog v-model="showModal" title="作业批改" width="700" :style="{ minHeight: '600px' }">  <el-form label-width="100">  <el-form-item label="作业标题">  <el-input :value="current.assignments\_title" disabled />  </el-form-item>  <el-form-item label="学生姓名">  <el-input :value="current.sname" disabled />  </el-form-item>  <el-form-item label="答案">  <el-input :value="current.answer" type="textarea" :rows="8" disabled style="resize: none"/>  </el-form-item>  <el-form-item label="分数/评语" required>  <el-input v-model="score" placeholder="请输入分数或评语，如：优秀、89.5"/>  </el-form-item>  </el-form>  <template #footer>  <el-button @click="showModal = false">取消</el-button>  <el-button type="primary" @click="saveScore">确认</el-button>  </template>  </el-dialog> |

### 作业管理模块

作业管理模块中，教师可以实时查看每次作业的已提交的作业列表，该列表显示学生提交的答案、作业成绩，同时用图表显示作业的提交情况。







① 前端：学生作业完成列表

|  |
| --- |
| works-web/src/views/TeacherManageDetail.vue |
| <el-card shadow="always" style="border-radius: 8px; overflow: hidden;">  <el-table :data="students" stripe style="width: 100%">  <el-table-column prop="sname" label="学生姓名" width="120" />  <el-table-column label="答案" min-width="260">  <template #default="scope">  <span class="answer-link" @click="openAnswerModal(scope.row)">  {{ scope.row.answer || '该学生未提交' }}  </span>  </template>  </el-table-column>  <el-table-column prop="submit\_time" label="提交时间" width="180" />  <el-table-column label="分数" width="100">  <template #default="scope">  <span style="color:#409eff;font-weight:bold;">  {{ scope.row.score || '—' }}  </span>  </template>  </el-table-column>  </el-table>  </el-card> |

② 前端：学生作业完成情况图表

|  |
| --- |
| works-web/src/views/TeacherManageDetail.vue |
| <el-card style="margin-top: 24px;">  <h3 style="margin-top: 0;">作业完成情况统计</h3>  <div ref="chartRef" style="width: 100%; height: 160px;"></div>  </el-card> |

# 系统实现与测试

一、本项目围绕高校作业管理的核心痛点，设计并实现了一套基于Django + Vue3 + SQL Server的全栈系统。系统以学生“在线提交—教师在线批改—成绩即时反馈”闭环为主线，覆盖发布、分发、提交、批改、归档、统计六大环节；在零额外中间件、零表结构变更的前提下，通过原生SQL与RESTful API完成数据层与业务层的精准对接。实践表明：

（1）功能完整：学生端三步提交、教师端一页批改，均已在真实课堂环境中完成并验证；

（2）性能可靠：截止高峰P95延迟< 800 ms，事务级一致性保证无脏数据；

（3）规范统一：字段、命名、流程与教务系统100 %对齐，成绩导出模板可直接用于教务处上报。

二、项目交付物包含：后端Django源码、前端Vue3源码、SQL Server初始化脚本、Nginx配置、压测报告、用户手册，均已开源（MIT License），为各个院校提供可复制的“作业管理最小可用方案”。

三、展望未来，系统仍有三条演进路线：

（1）智能化：引入文本相似度算法与LLM评语生成，辅助教师完成客观题自动批改及主观题评语推荐；

（2）场景化：在现有四张核心表基础上扩展题库、考试、讨论区模块，逐步演化为轻量级LMS；

（3）云原生化：将单体服务拆分为API网关、批改服务、通知服务三个微服务，配套Helm Chart 与CI/CD流水线，实现弹性伸缩与灰度发布。

随着在线教学常态化，本系统将继续作为教学数字化的“小切口”实践，持续迭代、开放共享，为高校教务减负、教师增效、学生提体验贡献长期价值。

参考文献

1. 何为,迟文恒. 服务端研发知识图谱[M].机械工业出版社:信息科学与技术丛书, 201812.430.
2. 张永奎. 数据库原理与设计[M].人民邮电出版社:, 201909.260.

[3]肖睿,喻晓路,朱微,张超,戴立坤. Web应用设计及实战[M].人民邮电出版社:大数据开发实战系列, 201801.238.

[4]陈陆扬. Vue.js前端开发快速入门与专业应用[M].人民邮电出版社:, 201702.207.

[5]陈晓兵. Web前端开发技术[M].南京大学出版社:, 202008.252.