
评分: _____



上海大学

SHANGHAI UNIVERSITY

课程论文

COURSE PAPER

课程项目提交与答辩管理系统

学院 理学院

专业 数学与应用数学

学号姓名 22122980 潘 上

学号姓名 22122991 邬 茗

课程 数据库设计与开发

打印日期 2025年7月1日

教师评分

小组论文分:

1. 工作态度与创新 (分)

★ 能按期完成规定的任务。

★ 工作量饱满，有一定创新。

2. 论文格式 (分)

符合上海大学课程论文要求。

★ 封皮规范

★ 标题、摘要、关键词规范

★ 正文规范

★ 参考文献规范

3. 论文质量 (分)

(1) 结构

★ 封皮

★ 标题、摘要、关键词

★ 正文（问题重述、问题求解、结果展示）

★ 参考文献

(2) 内容

★ 摘要部分要求指明论文所解决的问题。

★ 标题字数不要过多，关键词要求对表述论文的中心内容有实质意义。

★ 正文部分

通篇要求文字通顺；表述清晰；数学用语准确；符号统一，使用公式编辑器编辑数学公式；编号齐全；图表完备、整洁、正确，尽量美观；要回答提出的问题，结果要有应用价值。

★ 参考文献要正确引用。

4. 程序运行 (分)

提交论文程序，代码易读有注释，能运行，并得到预期结果。

小组成员分工与教师打分：默认一个成绩，可以按分工或内部约定打分。

姓名*	学号*	成绩	分工
潘上	22122980		后端实现，前端实现，项目规划，论文撰写
邬茗	22122991		项目规划，前端实现，论文撰写

课程项目提交与答辩管理系统

潘上，邬茗
(上海大学理学院数学系, 上海 200444)

摘要

本项目面向数据库课程的期末项目与答辩流程，构建一个集“学生提交”“教师评分”“管理员统筹”于一体的在线管理系统。系统采用 Bun + Elysia 构建同源 REST 服务与静态资源托管，数据库选用 SQLite 并开启外键与约束，以 users、defense_slots、projects、teacher_assignments、scores 五张核心表完成角色与流程建模；前端以原生 HTML/JS 结合 TailwindCSS 实现登录校验、答辩时段管理、项目提交、评分与审批等交互。会话使用 HttpOnly Cookie 管控并在前端做角色校验，写操作配合最小化输入校验与错误日志，关键路径通过幂等写入与关联清理维持数据一致性。实验表明，按“bun install → 初始化数据库 → 启动服务”即可完成端到端演示；默认账户可覆盖学生、教师、管理员的主要用例。项目以轻量、易复现的工程实践展示了数据库约束在教学业务闭环中的价值，并为后续扩展（集中式会话、附件上传、通知与统计审计）留出清晰演进路径。

关键词：答辩管理、教学、数据库

目录

一、引言	5
二、技术选型	5
三、数据库需求分析	6
四、数据库设计	6
五、具体实现	9
六、验证	12
七、结论	13

一、引言

在高校论文管理实践中，学生通常需要在规定时间内完成论文撰写、提交相关材料并参加现场答辩。传统以邮件或群聊收集材料的做法容易导致信息分散、材料版本不可追踪、安排沟通成本高等问题，尤其在集中答辩预约、教师分配和评分归档阶段负担更为突出。本项目聚焦“学生论文提交与答辩时间管理”的核心需求，构建统一的在线平台，将学生、教师、管理员三类角色的关键操作集中到一个 Web 系统中，以提升信息流转效率，保证数据一致性与可追踪性，并向学生与教师提供清晰、可复用的操作流程。

为适配教务与实验环境的多样性，系统在架构上坚持轻量化与易部署原则。Bun 单进程即可同时提供静态页面与 API；SQLite 使用本地文件持久化，省去额外数据库服务的配置成本。通过明确的角色权限与受控会话，系统既能支撑日常管理，也可在课堂演示与实验环境中快速上线与验证。

二、技术选型

项目后端选择 Bun 运行时与原生 Web 框架 Elysia.js 以提供高性能的 REST API；数据库采用易于部署的 SQLite，本地文件即可持久化；前端使用纯 HTML 与原生 JavaScript，并通过 TailwindCSS CDN 快速完成界面美化；服务器同时承担静态资源与 API 服务，由 Bun 单点部署完成。这一技术栈兼顾实验环境易用性与现代 Web 特性，适合高校论文管理的教学与行政场景。

与 Node.js 相比，Bun 在启动速度和内建工具链方面具有优势，能够在演示与调试阶段提供更快的开发反馈；同时 Elysia.js 提供类型安全的路由与中间件机制，便于在小型系统中快速搭建 API，并展示现代 TypeScript 框架的写法。数据库层选用 SQLite 则体现“零运维、零部署”优势，学生与管理员可直接在本地创建数据库文件，无需额外配置数据库服务，满足现场演示与快速试运行的需求。

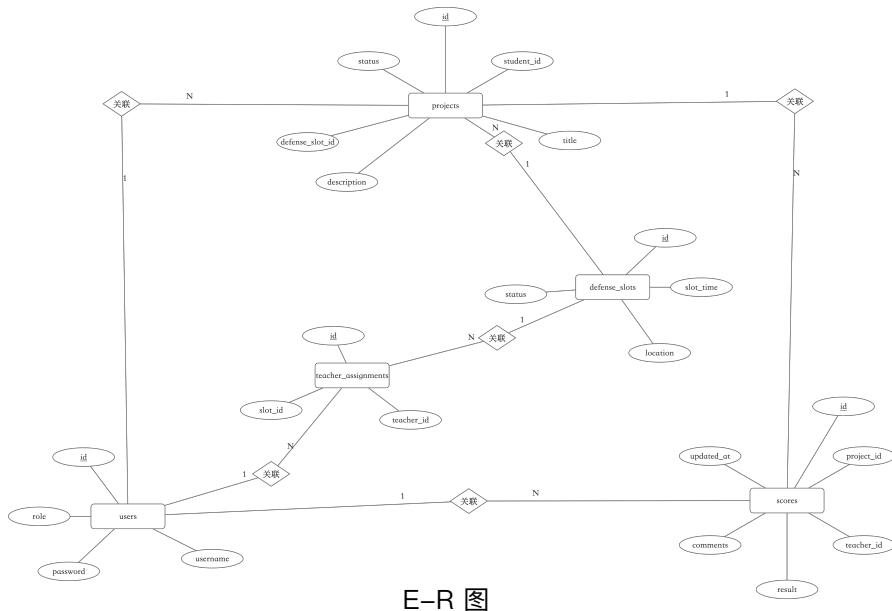
前端技术选型强调易学习与原生能力结合。TailwindCSS 通过实用类样式快速搭建界面，降低学生在 CSS 布局上的负担，同时保留 JavaScript 对 DOM 的直接操作，让学生能够聚焦于业务逻辑与接口交互。整体技术栈还便于在未来引入测试框架与持续集成工具，为课程进一步升级提供空间。

三、数据库需求分析

系统围绕三类角色展开：学生需要提交与更新论文项目，选择或查看答辩时间，并随时掌握论文状态；教师需要查看已分配的答辩时段及其对应的学生论文，并能对论文给出通过或未通过的评分与评语；管理员需要维护用户与角色、创建或更新答辩时段（含时间、地点与可用状态），并为时段分配教师，同时对学生论文进行审批与删除等日常管理。除角色操作外，系统还需提供统一登录与退出能力，确保受保护页面仅对已登录且角色匹配的用户开放。在这些业务需求下，数据库必须能够清晰地区分用户身份并维持认证凭据，确保每个学生只拥有一条论文项目记录，保证论文与答辩时段的引用关系可追踪，可在需要时进行更新或解除；对于教师与时段的多对多关系，数据库应通过中间表进行严谨建模以避免重复分配；对于评分数据，既要能明确责任教师，又要防止同一教师对同一论文重复评分。考虑到管理侧的频繁筛选与查询需求，模型还应支持按时段、教师或状态的高效过滤，同时在异常情境下保持数据一致性与可回滚性。非功能性方面，系统需在高峰期保持基本响应能力，并通过最小化的错误日志与输入校验提升可维护性；尽管暂未实现完整审计日志，模型层应保留适度扩展空间以便后续增强。

四、数据库设计

本系统启用 SQLite 的外键支持（PRAGMA foreign_keys = ON），并以五张核心表完成论文业务的关系建模：users、defense_slots、projects、teacher_assignments 与 scores。整体设计遵循“用约束表达业务规则、以最小状态实现可读流程”的原则：用唯一键限制重复记录，用外键维护引用一致性，用少量枚举或默认值表达状态机，同时保持前端与后端的状态语义一致。



E-R 图

`users` 表承载认证与授权的最小信息集，包含自增主键 `id`、唯一的 `username`、明文示例用的 `password` 以及带有数据库级检查约束的 `role` 字段。`role` 的取值限定为 `student`、`teacher` 与 `admin`，保证从数据库层开始即具备清晰的角色边界。考虑到论文项目、评分与教师分配均以人员为核心关联对象，相关表通过外键与 `users` 建立耦合，并在教师或学生删除时依赖 `ON DELETE CASCADE` 做关联清理，从而避免产生“悬挂引用”。

`defense_slots` 用于表达可预约的答辩时段，包含主键 `id`、ISO 字符串形式的 `slot_time`、文本 `location`，以及表示可见性的 `status` 字段。该字段默认值为 `open`，配合前端使用 `open/closed` 语义进行呈现。出于表现层灵活性的考虑，数据库未对 `status` 增设枚举检查约束，管理端在保存时段时直接面向该自由文本字段写入，从而兼容不同院系对时段状态的标注差异。所有时段在查询时按 `slot_time` 排序，有利于教师与学生侧的时间轴式展示。

`projects` 作为学生论文记录表，使用 `student_id` 的唯一约束保证“一名学生仅一条论文记录”。该表包含 `title` 与可空的 `description` 字段以描述论文内容，`defense_slot_id` 可为空以表示尚未选择或尚未分配答辩时段，`status` 默认为 `pending`，对应待审批或未评分的初始态。`student_id` 外键指向 `users(id)` 并在删除学生时级联清理，`defense_slot_id` 外键指向 `defense_slots(id)` 以表达论

文与时段的从属关系。应用层的学生提交接口采用“`INSERT ... ON CONFLICT(student_id) DO UPDATE`”的幂等写入模式，既避免重复论文记录，也方便学生在截止时间前多次修改标题、摘要或时段选择。

`teacher_assignments` 用以建模教师与答辩时段的多对多关系，字段包含 `teacher_id` 与 `slot_id`，并以复合唯一约束防止同一教师在同一时段出现重复分配。两端外键分别指向 `users(id)` 与 `defense_slots(id)`，并在相对对象被删除时级联移除分配记录。管理员在保存答辩时段时，会先清空该时段的既有分配，再批量插入新的教师名单，从而保证“以提交表单一次性重建分配”的幂等性与可读性。

`scores` 表负责保存教师对论文的评定结果与评语，包含 `project_id` 与 `teacher_id` 两个外键、取值被数据库检查约束限定为 `pass/fail` 的 `result` 字段、可空的 `comments` 字段，以及带默认值的 `updated_at` 时间戳。复合唯一约束 `(project_id, teacher_id)` 保证同一教师对同一论文最多保留一条评分记录；当教师重复提交评分时，应用层以“冲突覆盖”的方式更新 `result`、`comments` 与 `updated_at`，从而显式记录最近一次评定。删除论文或教师账号将触发对评分数据的级联清理，保证评定体系不残留孤立行。

状态与约束方面，数据库对关键字段施加了适度的结构性限制。例如用户角色以 `CHECK` 约束限定有效集合，评分结果以 `CHECK` 约束限定为通过或未通过，学生论文以唯一的 `student_id` 避免重复提交；而答辩时段的 `status` 则以默认值与前端词汇约定来表达开放或关闭，避免在早期阶段过度约束表现层。外键策略统一开启，确保删除用户或时段时能够自动清理授课分配、评分与论文从属关系；对于某些清理路径，应用层在提交处理前仍会执行显式删除以保持意图清晰与幂等可读，这与数据库的级联机制相互补充。

索引与性能方面，主键与唯一约束隐含了必要的索引，足以支撑课堂与日常管理场景下的查询需求。典型的访问路径包括按教师查看所负责的全部时段与其下的论文、按管理员维度列出所有论文及其学生姓名、按学生加载个人论文与已选时段

等。考虑到系统以同源单体部署为主，SQLite 的单文件与内置 B-Tree 索引能够满足十至数十并发用户的读取与写入；若未来场景扩大，可在 defense_slots(slot_time)、projects(status) 等字段上增设二级索引以进一步优化过滤性能。

数据生命周期在初始化脚本中给出最小可运行样例，包括三类默认用户（student1/stupass、teacher1/teapass、admin1/admpass）、两个示例答辩时段以及一位教师对两个时段的分配。借助这些样例数据，系统在首次启动后即可完成从登录、论文提交、时段查看到评分与审批的端到端验证，便于教学与演示快速开展。随着使用深入，管理员可直接通过系统接口或迁移脚本扩展字段与约束，并在 src/db/README.migrations.md 中记录变更，以保证团队对数据库状态的共同理解与快速重放能力。

五、具体实现

系统以 Elysia 作为后端核心框架，在单个 Bun 进程中同时承担静态资源与 REST API。服务器入口位于 src/server.ts：首先注册 @elysiajs/static 插件，将 public/ 目录暴露为静态资源根路径；随后装配认证、学生、教师与管理员四类路由模块。为确保受保护页面不被未授权访问，服务器在 /student.html、/teacher.html、/admin.html 三个路径上增加了后端级访问控制，利用 utils/session.ts 中的 getSessionFromRequest 读取 Cookie 并解析会话，当不存在或会话角色不匹配时，直接以 302 重定向到登录页 index.html；若校验通过，则返回实际 HTML 文件（Bun.file 读取）。服务器额外提供 /health 健康检查端点以便教学演示与运维探活，监听端口来自 Bun.env.PORT（默认 3000），启动后在控制台打印访问地址。

数据库访问由 src/utils/dbClient.ts 统一封装。该模块在首次获取连接时解析环境变量 DATABASE_URL，否则回退到工作目录下的 data/app.db；在创建连接之前确保数据库目录存在，并显式执行 PRAGMA foreign_keys = ON 以打开外键约

束。封装暴露 `getDb()` 单例连接供各路由使用，并提供 `closeDb()` 以便测试或工具脚本在需要时关闭句柄。此设计使得应用层无需关心 SQLite 的路径解析与目录准备，部署时只需通过 `.env` 覆盖路径即可将数据文件放置到指定位置。

会话管理在 `src/utils/session.ts` 中实现。系统使用内存 Map 保存会话记录，每条记录包含会话 ID、用户 ID、用户名、角色与过期时间，默认 TTL 为四小时。`createSession()` 在登录成功后生成 UUID 形式的会话标识并写入 Map，`getSession()` 与 `getSessionFromRequest()` 负责检索与过期清理，`deleteSession()` 在登出或手动失效时移除记录。浏览器侧的会话通过 `buildSessionCookie()` 设置为 `HttpOnly`、`SameSite=Lax` 的 Cookie，前端无法读取 Cookie 明文但请求会自动携带，从而降低脚本层面的窃取风险。该方案针对单节点部署足够简单可靠，同时为未来替换为集中式存储（如 Redis）预留了统一的接口层。

认证模块位于 `src/api/auth.ts`。`POST /api/login` 接收用户名与密码，直接在 `users` 表内进行匹配校验，命中后调用会话模块创建会话并在响应头写入 Cookie，同时将 `{ id, username, role }` 返回给前端用于本地跳转；若认证失败则返回 `401`。`POST /api/logout` 解析请求 Cookie 并删除会话，同时下发一个 `Max-Age=0` 的 Cookie 以指导浏览器清除本地会话。该模块对请求体采用 Elysia 的 `t.Object` 进行最小化模式校验，异常路径在服务端以 `console.error` 记录并返回 `500`。

学生模块位于 `src/api/student.ts`。提交论文接口 `POST /api/student/submit` 要求 `studentId`、`title` 并可选携带 `description` 与 `defenseSlotId`，在数据库层使用 `INSERT ... ON CONFLICT(student_id) DO UPDATE` 的 UPSERT 语义实现幂等写入，避免重复记录并在更新时自动重置论文状态为 `pending`；若学生尚未选择答辩时段，可将 `defenseSlotId` 置空。查询个人论文接口 `GET /api/student/project/:studentId` 返回该学生的论文详情；查询答辩时段接口 `GET /api/defense/slots` 直接按时间排序列出全部时段并暴露 `id`、`slotTime`、

location 与 status 字段，供学生侧选择。所有接口均包裹 try/catch，并在失败时设置 HTTP 状态码并返回统一错误消息。

教师模块位于 src/api/teacher.ts。GET /api/teacher/slots/:id 以教师 ID 为参数，先通过 teacher_assignments 与 defense_slots 关联获取该教师负责的全部时段，再以 LEFT JOIN 将每个时段下的论文项目合并到结果集；路由层随后在内存中按 slotId 进行分组与聚合，最终返回“时段 + 项目列表”的结构，便于前端以表格渲染。评分接口 POST /api/teacher/score 首先根据 studentId 查找学生的论文并确保其已选择时段 (defense_slot_id IS NOT NULL)，随后向 scores 表以 UPSERT 方式写入或覆盖评分与评语，同时更新论文状态为 approved 或 rejected，从而显式闭合“评分—状态变更”的流程。若学生未找到有效论文或尚未选择时段，接口返回 404，并提示前端引导学生完善信息。

管理员模块位于 src/api/admin.ts，覆盖用户、时段与论文三条主线。用户侧提供列表、保存与删除接口：保存接口以 id 作为幂等键，若主键存在则更新，否则插入；删除接口按 ID 清理用户，依赖外键实现对论文与评分的级联清理。时段侧的列表接口在 SQL 中利用 GROUP_CONCAT 汇总教师 ID 并计算教师/论文统计数量；保存接口支持更新或插入时段记录，并在每次保存后“先清空、再批量插入”的方式重建教师分配，确保分配结果与前端多选框保持一致；删除接口在移除时段前，会先删除该时段的全部分配并将引用该时段的论文 defense_slot_id 置空，再删除时段本身，从而避免产生悬挂引用。论文侧的列表接口联结 users 返回学生姓名，审批接口直接更新 projects.status，删除接口先清理评分再删除论文，以明确表达清理意图并与外键级联形成互补。

前端脚本集中在 public/js/main.js。工具函数 formToJson() 用于序列化表单数据，request() 统一封装 fetch 的 JSON 头与 credentials: 'include'，确保浏览器在跨路径请求时携带会话 Cookie；getCurrentUser() 与 requireRole() 基于 localStorage.currentUser 做本地身份校验并在失败时重定向登录页；bindLogout() 将“退出”按钮与后端登出接口绑定；renderTable() 以统一 TailwindCSS 类渲染所有列表页。登录页初始化逻辑 initLogin() 支持密码显隐切

换，登录成功后按角色映射跳转到学生、教师或管理员页面。学生页 `initStudent()` 在加载时并行拉取可用时段与个人论文，提交表单时构造包含 `studentId` 的载荷并调用写入接口；教师页 `initTeacher()` 渲染负责的答辩时段与其下项目，并在评分后刷新时段数据；管理员页 `initAdmin()` 则围绕“用户、时段、论文”三块进行渲染与操作，包括多选教师分配、保存与删除动作、以及与后端结果的即时回显。所有结果与错误信息通过 `withTimestamp()` 添加时间戳，便于操作追踪与课堂演示。

在请求生命周期上，认证成功后服务器以 `Set-Cookie` 返回会话，前端后续请求通过 `credentials: 'include'` 自动携带 `Cookie`；前端页面在进入时基于本地存储做快速角色判断，而真正的访问控制由服务器在受保护页面路由处强制执行。关键写路径遵循幂等与可恢复原则：学生提交采用 `ON CONFLICT(student_id)` 避免重复；教师评分采用 `ON CONFLICT(project_id, teacher_id)` 覆盖最近结果并更新时间戳；管理员保存时段以“重建分配”的方式确保前后端状态一致；管理员删除时段与论文时分别进行显式的关联清理，既使意图可读，也降低了对读者理解外键级联细节的依赖。

错误处理方面，路由在数据库操作失败时以最小可辨识的消息体与 `500` 响应码进行返回，并在服务端 `console.error` 标注来源（如 `admin.slot.save`、`teacher.score` 等），便于快速定位。由于演示环境使用明文示例密码，正式部署时应切换为哈希存储与更严格的输入校验策略；若需要提升稳定性，可在请求入口增加速率限制与更细粒度的参数验证，并在前端对字段长度与格式进行预校验，从而进一步增强系统的健壮性。

六、验证

系统提供清晰的运行流程：执行 `bun install` 安装依赖、`bun run src/db/initDB.ts` 初始化数据库、`bun run dev` 启动服务，即可使用默认账号进行登录并验证不同角色功能。同时，服务器暴露的 `/health` 接口可用于快速检测服务是否正常响应。通过登录后操作学生、教师与管理员页面，可验证项目提交流程、答辩安

排展示与审批/评分闭环的正确性，前端的即时反馈机制也为操作结果提供了可视化确认。

进一步的验证措施包括：对数据库约束进行异常操作测试，例如尝试为同一学生重复创建项目、为已评分项目再次提交评分，以确认唯一约束与业务逻辑的正确性；同时，可通过模拟并发请求检查答辩时段分配在高频操作下的事务一致性。若将系统部署到教学服务器，可结合日志分析与简单的压力测试，评估 Bun 服务器在 30-50 并发用户下的响应情况，为课程运维提供数据支撑。

七、结论

本项目以 SQLite 为核心数据存储，通过 Bun + Elysia 架构实现了角色分明、流程完整的课程项目管理平台，展示了数据库设计与 Web 应用协同的教学范例。系统从需求分析到数据模型、API 实现与前端交互，形成完整的教学闭环，帮助学生理解从概念模型到代码落地的全过程。当前版本在单节点部署、轻量化环境中已能满足课程演示与小规模使用需求。

未来可进一步扩展多会话持久化、文件附件上传、评分多维度量化以及消息通知等功能，并可引入单元测试与权限审计机制，以提升系统的可靠性与可维护性。针对大班教学，可考虑加入批量导入导出功能与数据备份策略；对于学术诚信需求，可扩展论文相似度检测接口。最终目标是将该系统打造成课程教学的持续实践平台，支持跨学期复用与多院系协作。