

数据库应用设计开发大作业

在线教育平台管理系统

组长：沈天乐（2022300381）

成员 1：王崇宇（2022300395）

2024 年 12 月 17 日

目 录

1	选题背景	5
2	需求分析	5
2.1	数据需求分析	5
2.1.1	总体数据需求	5
2.1.2	模块数据需求分析	6
2.1.3	数据关系与流程	7
2.1.4	数据存储与安全性需求	8
2.2	功能需求分析	8
2.3	非功能需求分析	9
2.4	数据字典	10
2.4.1	数据流说明	10
2.4.2	数据存储说明	11
2.4.3	数据项说明	12
3	数据库设计	14
3.1	概念结构设计	14
3.2	逻辑结构设计	17
3.2.1	表结构设计	18
3.2.2	表之间的关系	18
3.2.3	函数依赖	19
3.3	安全性与完整性设计	20
3.3.1	数据安全性设计	20
3.3.2	数据完整性设计	20
3.3.3	约束设计	21
3.4	物理结构设计	21
3.4.1	数据库存储路径与文件	21
3.4.2	表的物理存储设计	22
3.4.3	字段数据存储设计	22
3.4.4	数据访问优化与索引设计	22
3.4.5	数据完整性约束设计	23

4	数据库实施	23
4.1	创建数据库	23
4.2	加载数据	27
4.2.1	初始化无 edu_system.db	27
4.2.2	存在 edu_system.db	27
5	数据库应用程序设计	27
5.1	系统功能模块图	27
5.2	用户管理功能	28
5.2.1	用户注册功能	28
5.2.2	用户登录功能	29
5.3	课程管理功能	29
5.3.1	课程添加功能	29
5.3.2	课程删除功能	29
5.3.3	课程修改功能	29
5.3.4	课程查看功能	29
5.4	学生管理功能	30
5.4.1	学生注册功能	30
5.4.2	报名课程功能	30
5.4.3	修改学生信息功能	30
5.4.4	删除学生功能	30
5.4.5	查看选课功能	30
5.5	教师管理功能	31
5.5.1	教师注册功能	31
5.5.2	安排课程功能	31
5.5.3	教师信息管理功能	31
5.5.4	修改删除教师功能	31
5.5.5	查看教师课程功能	32
5.6	学习进度功能	32
5.6.1	搜索学生功能	32
5.6.2	学习统计功能	32
5.6.3	保存课程成绩功能	32
5.7	作业管理功能	33

5.7.1	搜索课程功能	33
5.7.2	发布新作业功能	33
6	数据库应用程序开发	33
6.1	数据库连接	33
6.2	用户管理功能	33
6.3	课程管理功能	34
6.4	学生管理功能	34
6.5	教师管理功能	34
6.6	学习进度功能	34
6.7	作业管理功能	35
7	数据库应用系统运行环境	35

1 选题背景

随着信息技术的飞速发展，**互联网与教育**的深度融合推动了在线教育行业的蓬勃发展。在线教育平台作为传统教育的重要补充，以其高效、便捷、个性化的优势，逐渐成为现代教育体系中的关键组成部分。尤其在全球化与数字化浪潮下，远程学习、资源共享、智能管理等需求日益增加，在线教育平台的建设显得尤为重要。然而，目前许多在线教育平台在实际应用中仍存在诸多不足。例如，课程管理功能繁杂，难以高效地组织与维护课程资源；学生与教师信息管理不够系统，无法全面支撑用户的个性化需求；学习进度跟踪不够精准，导致学习效果评价存在偏差；作业管理流程不够规范，影响教学互动与反馈质量。因此，设计一个高效、智能、用户友好的在线教育平台管理系统，对于解决上述问题、提高教育管理效率具有重要的现实意义。本系统围绕**课程管理、学生管理、教师管理、学习进度跟踪和作业管理**等核心功能模块展开设计与实现，旨在为教育机构、教师与学生提供一个全面、集成化的教育管理解决方案。通过该平台，管理者能够高效地进行资源分配与管理，教师能够便捷地组织教学活动，学生能够自主规划学习进度并接受有效的教学反馈。本项目不仅有助于提升教育管理的效率与质量，同时也为教育信息化的发展提供了一定的技术与实践支持，具有较高的应用价值与研究意义。

2 需求分析

2.1 数据需求分析

针对本在线教育平台管理系统，数据需求分析主要围绕**课程管理、学生管理、教师管理、学习进度跟踪和作业管理**等模块展开。

2.1.1 总体数据需求

本系统的数据需求主要包括以下几个方面：

- **静态数据**：如课程基本信息、学生档案信息、教师档案信息等。
- **动态数据**：如学生的学习进度、作业提交与批改记录、课程报名情况等。
- **交互数据**：系统各模块之间数据的关联与流转，例如学生和课程的关联、教师与课程的管理关系等。
- **系统管理数据**：用户登录信息等。

2.1.2 模块数据需求分析

课程管理模块

- 数据需求：
 - 课程名称（唯一标识）
 - 学习时间
 - 学分
 - 平时/期中/期末时间占比
 - 上课时间
- 数据来源：管理员录入
- 存储要求：课程信息表需要长期存储，支持查询与修改。

学生管理模块

- 数据需求：
 - 学生学号（唯一标识）
 - 学生姓名
 - 入学年份
 - 报名课程列表（与课程管理模块关联）
- 数据来源：管理员录入。
- 存储要求：学生信息需要长期存储，支持增删改查。

教师管理模块

- 数据需求：
 - 教师职工号（唯一标识）
 - 教师姓名
- 数据来源：管理员录入。
- 存储要求：教师信息需要长期存储，支持查询与更新。

学习进度跟踪模块

- 数据需求：
 - 学生学号（关联学生管理模块）
 - 课程编号（关联课程管理模块）
 - 学习统计
 - 学习课程分数
- 数据来源：管理员录入和系统自动计算更新。
- 存储要求：动态存储学习记录，支持实时更新和查询。

作业管理模块

- 数据需求：
 - 所属课程名称（关联课程管理模块）
 - 作业标题
 - 作业内容
- 数据来源：管理员发布。
- 存储要求：动态存储作业记录，支持批量查询、更新和统计分析。

2.1.3 数据关系与流程

- 学生与课程的关系：一名学生可以报名多门课程，一门课程可供多名学生学习。
- 教师与课程的关系：一名教师可负责多门课程，但每门课程通常由一名教师负责。
- 作业与课程的关系：一门课程可发布多份作业，每份作业对应多个学生的提交记录。
- 学习进度与学生、课程的关系：每个学生在每门课程中都有独立的学习进度记录。

数据流向：

1. 学生注册登录后，查看课程信息，选择课程并报名。

2. 教师录入课程信息，发布作业并批改提交的作业。
3. 系统自动记录学生的学习进度，并展示给学生和教师。
4. 管理员管理系统中的所有信息，定期维护数据。

2.1.4 数据存储与安全性需求

- 数据存储应保证完整性、唯一性和一致性。
- 敏感数据（如密码、联系方式）需加密存储 (未实现)。
- 数据访问应根据不同用户角色（学生、教师、管理员）进行权限控制，防止越权访问 (未实现)。
- 系统应支持数据备份与恢复功能，确保数据安全性与稳定性 (未实现)。

2.2 功能需求分析

本在线教育平台管理系统主要包括以下功能模块：

1. 课程管理模块

- 管理员和教师可录入、修改、删除和查看课程信息。
- 提供课程查询功能，学生可根据课程名称、类别等条件进行检索。
- 支持课程报名功能，学生可选择并报名课程。

2. 学生管理模块

- 提供学生信息的录入、修改、删除和查询功能。
- 实现学生与所选课程的关联，提供报名课程列表查询。

3. 教师管理模块

- 提供教师信息的录入、修改、删除和查询功能。
- 支持教师管理所授课程，关联课程管理模块。。

4. 学习进度跟踪模块

- 系统自动计算学生的总成绩。

- 支持给学生各课程打分。

5. 作业管理模块

- 搜索选择课程。
- 发布新作业。
- 查看已经发布的作业。

6. 系统管理模块

- 提供系统基础配置功能，如课程类别设置、用户角色配置等。
- 记录系统操作日志，方便管理员审查和维护。
- 支持数据备份与恢复功能，保障系统数据安全。

2.3 非功能需求分析

针对本在线教育平台管理系统，非功能需求主要包括以下几个方面：

1. 性能需求

- **响应时间：**系统页面响应时间不超过 3 秒，数据查询和提交操作在 5 秒内完成。

2. 可靠性需求

- **系统稳定性：**系统年平均可用率不低于 99.9%。

3. 安全性需求

- **防止非法操作：**系统需提供防 SQL 注入等安全机制。

4. 可扩展性需求

- 系统架构应具备良好的扩展性，支持功能模块的增加和系统性能的提升。
- 数据库设计需支持大规模数据的存储和查询，确保未来用户数据增加时系统仍能高效运行。

5. 兼容性需求

- 系统需支持主流浏览器（如 Chrome、Firefox、Edge 等）的访问。

2.4 数据字典

数据字典用于描述系统中各个数据元素、数据流、数据存储及其关系。以下是本系统的数据字典详细内容。

2.4.1 数据流说明

数据流名称	说明
课程信息流	包括课程名称、课程编号、教师信息、课程分类等数据。
学生信息流	包括学生学号、姓名、入学年份等数据。
教师信息流	包括教师编号、姓名、所授课程等数据。
作业信息流	包括作业标题、内容等数据。
学习进度流	包括学生学习的课程进度、总成绩等数据。

2.4.2 数据存储说明

数据存储名称	说明
courses	存储课程信息，包括课程的 ID、名称、学时、学分、平时分、期中分和期末分等。
sqlite_sequence	存储自增 ID 序列，用于生成每个表的主键 ID。
students	存储学生信息，包括学生的 ID、姓名、学号、入学年份等。
teachers	存储教师信息，包括教师的 ID、姓名、教师编号等。
student_courses	存储学生与课程的关联信息，记录每个学生选修的课程。
teacher_courses	存储教师与课程的关联信息，记录每个教师教授的课程。
grades	存储学生的成绩信息，包括平时成绩、期中成绩和期末成绩。
assignments	存储作业信息，包括作业标题、内容、课程 ID 以及创建时间。
users	存储系统用户的信息，包括用户名、密码以及创建时间。

2.4.3 数据项说明

表 courses 的数据项说明

数据项名称	数据类型	说明	约束
id	INTEGER	课程的唯一标识符	主键
name	TEXT	课程名称	非空
learn_time	TEXT	课程的学时信息	非空
credit	REAL	课程的学分	非空
usual_score	INTEGER	课程的平时分	非空
midterm_score	INTEGER	课程的期中分	非空
final_score	INTEGER	课程的期末分	非空
times	TEXT	课程的上课时间	默认值为空字符串

表 sqlite_sequence 的数据项说明

数据项名称	数据类型	说明	约束
name	TEXT	表的名称	非空
seq	INTEGER	当前表的 ID 序列值	非空

表 students 的数据项说明

数据项名称	数据类型	说明	约束
id	INTEGER	学生的唯一标识符	主键
name	TEXT	学生姓名	非空
student_id	TEXT	学生学号	非空
enrollment_year	INTEGER	学生入学年份	非空

表 teachers 的数据项说明

数据项名称	数据类型	说明	约束
id	INTEGER	教师的唯一标识符	主键
name	TEXT	教师姓名	非空
teacher_id	TEXT	教师编号	非空

表 student_courses 的数据项说明

数据项名称	数据类型	说明	约束
student_id	INTEGER	关联学生的 ID	外键，引用 students 表中的 id
course_id	INTEGER	关联课程的 ID	外键，引用 courses 表中的 id

表 teacher_courses 的数据项说明

数据项名称	数据类型	说明	约束
teacher_id	INTEGER	关联教师的 ID	外键，引用 teachers 表中的 id
course_id	INTEGER	关联课程的 ID	外键，引用 courses 表中的 id

表 grades 的数据项说明

数据项名称	数据类型	说明	约束
student_id	INTEGER	关联学生的 ID	外键，引用 students 表中的 id
course_id	INTEGER	关联课程的 ID	外键，引用 courses 表中的 id
usual_grade	REAL	学生的平时成绩	默认值为 0
midterm_grade	REAL	学生的期中成绩	默认值为 0
final_grade	REAL	学生的期末成绩	默认值为 0

表 assignments 的数据项说明

数据项名称	数据类型	说明	约束
id	INTEGER	作业的唯一标识符	主键
course_id	INTEGER	关联课程的 ID	外键，引用 courses 表中的 id
title	TEXT	作业的标题	非空
content	TEXT	作业的内容	非空
create_time	TIMESTAMP	作业的创建时间	默认值为当前时间戳

表 users 的数据项说明

数据项名称	数据类型	说明	约束
id	INTEGER	用户的唯一标识符	主键
username	TEXT	用户名	非空
password	TEXT	用户密码	非空
created_at	TIMESTAMP	用户的创建时间	默认值为当前时间戳

3 数据库设计

3.1 概念结构设计

在本系统中，概念结构设计主要依赖于实体关系模型（Entity-Relationship Model, ER 模型）来表示不同实体及其相互之间的关系。通过 ER 模型，我们能够清晰地描述出系统中的各类实体及它们之间的联系，确保系统的数据能够有效地支持业务需求。

本系统中的核心实体包括学生、教师、课程、作业、成绩等，实体之间的关系如学生与课程之间的多对多关系（学生选课），教师与课程之间的多对多关系（教师授课），以及学生和作业之间的关系（学生提交作业）等。以下是系统概念结构设计的主要内容：

- 实体：

- 学生（Student）：记录学生的基本信息，如学号、姓名等。
- 教师（Teacher）：记录教师的基本信息，如工号、姓名、所授课程等。

- 课程 (Course): 包含课程的名称、学分、学时、课程成绩组成等。
- 作业 (Assignment): 记录与课程相关的作业信息，如作业标题、内容等。
- 成绩 (Grade): 存储学生的作业、平时成绩、期中成绩和期末成绩。

• 关系:

- 选课 (Student-Course): 学生与课程之间的多对多关系。
- 授课 (Teacher-Course): 教师与课程之间的多对多关系。
- 等等等等

概念结构设计的关键任务是确保在设计过程中，所有的数据项都能够得到合理的关联，且能够高效地支持系统功能的实现。为此，我们将采用 ER 图对实体、关系及属性进行可视化表示，帮助开发人员和使用者更好地理解系统的数据结构。

下图为本系统的 ER 图，展示了各实体及其关系：

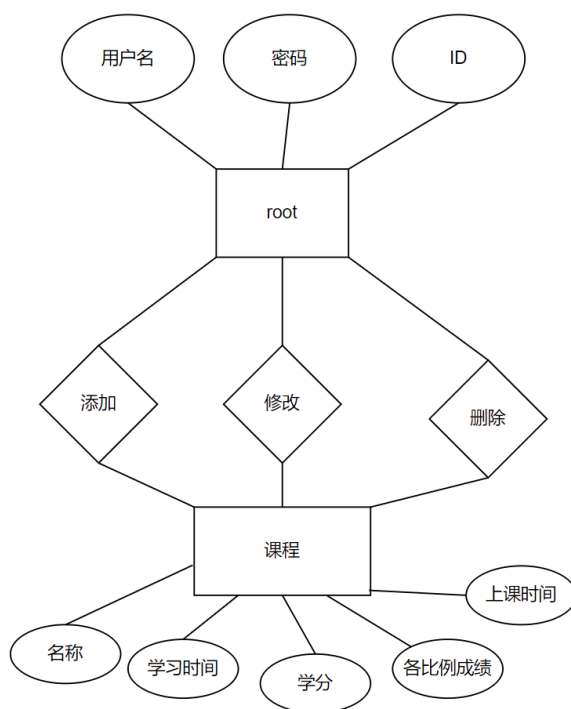


图 1: 用户-课程

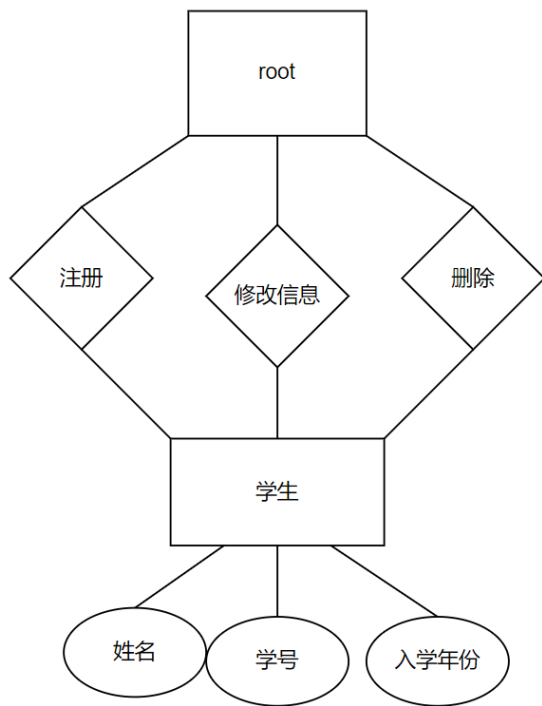


图 2: 用户-学生

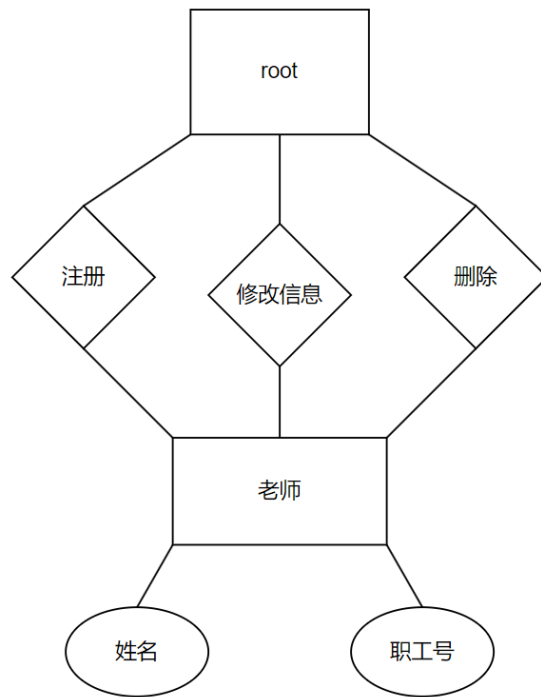


图 3: 用户-老师

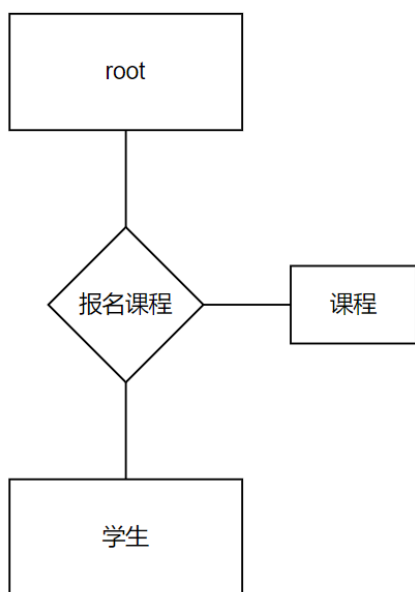


图 4: 学生报名课程

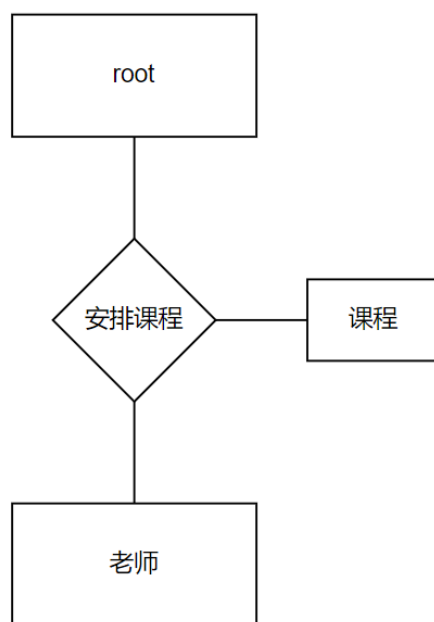


图 5: 老师安排课程

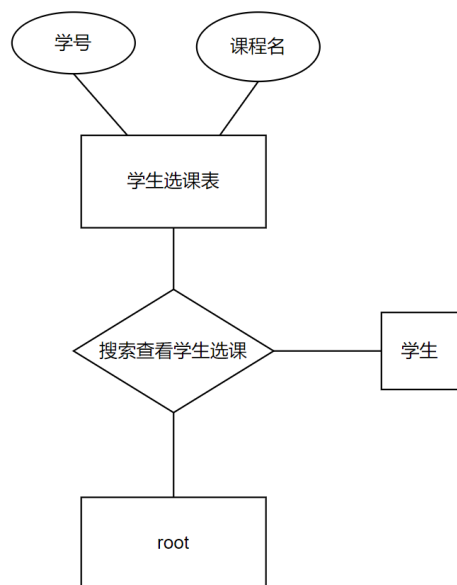


图 6: 学生查看选课

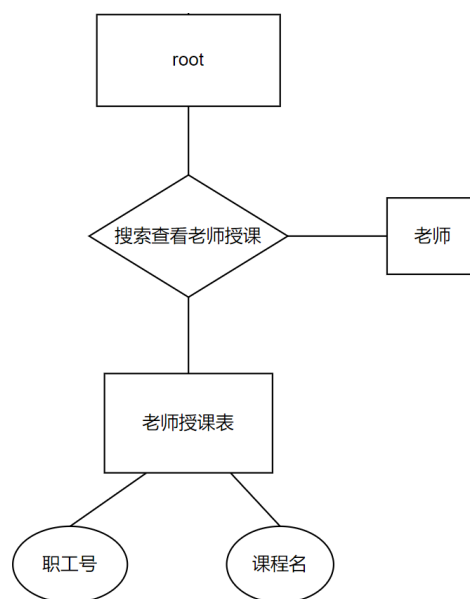


图 7: 老师查看授课

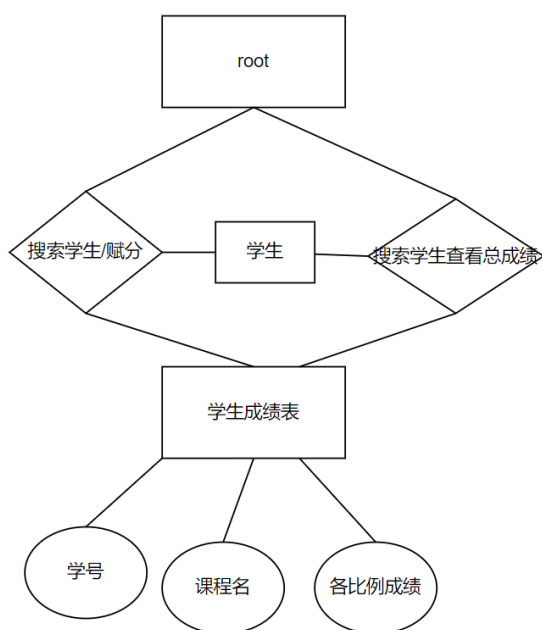


图 8: 查看/给成绩

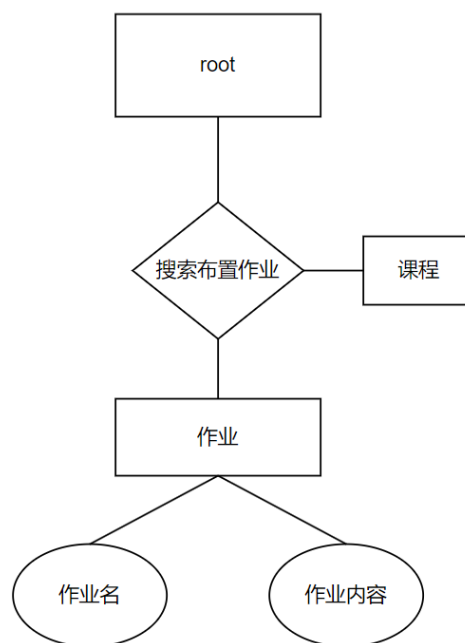


图 9: 布置作业

3.2 逻辑结构设计

在本系统中，逻辑结构设计的主要任务是根据业务需求和数据量，确定合适的表结构，并通过合理的字段设计确保数据的一致性、完整性和可靠性。同时，我们还需要考虑查询的效率、存储的空间以及数据维护的便利性。以下是本系统的逻辑结构设计的关键内容：

3.2.1 表结构设计

- **学生表 (students):** 存储学生的基本信息, 如学号、姓名、入学年份等。字段设计包括: 学号 (student_id) 作为主键, 姓名 (name)、入学年份 (enrollment_year) 等。
- **教师表 (teachers):** 存储教师的基本信息, 如工号、姓名等。字段设计包括: 工号 (teacher_id) 作为主键, 姓名 (name) 等。
- **课程表 (courses):** 存储课程的基本信息, 如课程名称、学时、学分等。字段设计包括: 课程 ID (id) 作为主键, 课程名称 (name)、学分 (credit)、学时 (learn_time) 等。
- **成绩表 (grades):** 存储学生的成绩信息, 如平时成绩、期中成绩、期末成绩等。字段设计包括: 学生 ID (student_id)、课程 ID (course_id) 作为复合主键, 平时成绩 (usual_grade)、期中成绩 (midterm_grade)、期末成绩 (final_grade) 等。
- **作业表 (assignments):** 存储作业的相关信息, 如作业标题、内容、创建时间等。字段设计包括: 作业 ID (id) 作为主键, 课程 ID (course_id) 作为外键, 作业标题 (title)、作业内容 (content)、创建时间 (create_time) 等。

3.2.2 表之间的关系

- **学生与课程的关系:** 学生可以选修多门课程, 课程也可以被多个学生选修。通过建立“学生-课程”表 (student_courses) 来实现该多对多关系。该表包含学生 ID (student_id) 和课程 ID (course_id) 两个外键, 作为联合主键。
- **教师与课程的关系:** 教师可以教授多门课程, 课程也可以由多位教师教授。通过建立“教师-课程”表 (teacher_courses) 来实现该多对多关系。该表包含教师 ID (teacher_id) 和课程 ID (course_id) 两个外键, 作为联合主键。
- **学生与成绩的关系:** 每个学生在每门课程上有一个成绩记录。通过“成绩”表 (grades) 来存储学生的成绩信息, 字段包括学生 ID (student_id) 和课程 ID (course_id) 作为联合主键, 记录学生的平时成绩、期中成绩和期末成绩。

3.2.3 函数依赖

- **courses** 表的函数依赖:

$$\text{id} \rightarrow \{\text{name}, \text{learn_time}, \text{credit}, \text{usual_score}, \text{midterm_score}, \text{final_score}, \text{times}\}$$

- **sqlite_sequence** 表的函数依赖:

$$\text{name} \rightarrow \text{seq}$$

- **students** 表的函数依赖:

$$\text{id} \rightarrow \{\text{name}, \text{student_id}, \text{enrollment_year}\}$$

- **teachers** 表的函数依赖:

$$\text{id} \rightarrow \{\text{name}, \text{teacher_id}\}$$

- **student_courses** 表的函数依赖:

$$\text{student_id}, \text{course_id} \rightarrow \text{student_id}, \text{course_id}$$

- **teacher_courses** 表的函数依赖:

$$\text{teacher_id}, \text{course_id} \rightarrow \text{teacher_id}, \text{course_id}$$

- **grades** 表的函数依赖:

$$\text{student_id}, \text{course_id} \rightarrow \{\text{usual_grade}, \text{midterm_grade}, \text{final_grade}\}$$

- **assignments** 表的函数依赖:

$$\text{id} \rightarrow \{\text{course_id}, \text{title}, \text{content}, \text{create_time}\}$$

- **users** 表的函数依赖:

$id \rightarrow \{username, password, created_at\}$

3.3 安全性与完整性设计

在在线教育平台管理系统中，数据的安全性和完整性至关重要，尤其是在存储用户信息、学生成绩和教师信息等敏感数据时。为了确保系统中数据的保密性、一致性和可靠性，需要对数据进行适当的安全性和完整性设计。

3.3.1 数据安全性设计

数据安全性主要保证数据在存储和传输过程中不被未授权的用户或恶意程序访问、篡改或泄露。在本系统中，主要采取以下措施来保障数据的安全性：

- **数据加密**：对存储在数据库中的敏感信息（如用户密码、学生个人信息等）进行加密处理。密码采用哈希加密算法（如 SHA-256 或 bcrypt）存储，确保即使数据库遭到泄露，用户密码也无法被直接获取。
- 但由于时间关系，本系统并没有实现加密。

3.3.2 数据完整性设计

数据完整性设计确保数据库中的数据在创建、修改和删除过程中始终保持一致性和准确性。以下是本系统的主要完整性设计：

- **实体完整性**：每个表中的主键必须唯一且不可为空。系统中每个学生、教师和课程都有唯一的标识符（如学生 ID、教师 ID、课程 ID 等），确保实体数据的唯一性。例如，学生表（students）中的学生 ID、教师表（teachers）中的教师 ID 等字段作为主键，都不能为 NULL，并且具有唯一性。
- **参照完整性**：系统中的外键约束确保了不同表之间的关系一致性。例如，成绩表（grades）中的学生 ID 和课程 ID 必须与学生表（students）和课程表（courses）中的相应记录一致，避免孤立数据的出现。通过使用外键约束，确保删除、更新等操作不会导致参照完整性破坏。
- **字段完整性**：为确保字段的有效性，每个字段都有明确的约束条件。例如，课程表（courses）中的学分（credit）字段设为 REAL 类型，且要求大于 0；学生表

(students) 中的学号字段 (student_id) 要求唯一；作业表 (assignments) 中的创建时间 (create_time) 字段自动生成，确保数据的时效性。

- **值域完整性：**对字段的取值范围进行限定，避免非法值的插入。例如，成绩表 (grades) 中的平时成绩、期中成绩和期末成绩字段可以设置为 0 到 100 的范围，确保成绩数据的有效性和合理性。

3.3.3 约束设计

为了进一步保证数据的一致性和安全性，系统还将使用约束 (Constraints) 来自动化执行一些数据完整性检查

- **非空约束：**确保关键字段（如学号、课程 ID 等）不能为空，以保证数据的完整性。
- **唯一性约束：**确保每个学生 ID、教师 ID、课程 ID 等字段的值唯一，避免数据重复。
- **外键约束：**确保学生与课程之间、教师与课程之间的关系一致，避免孤立的记录出现。

3.4 物理结构设计

对于本在线教育平台管理系统，数据库采用 SQLite 进行存储，以下是该系统的物理结构设计内容。

3.4.1 数据库存储路径与文件

本系统使用 SQLite 作为数据库管理系统，数据库文件存储在本地磁盘中，所有数据表、索引、触发器和存储的文件都会被保存在一个 SQLite 数据库文件中。在设计时，需要确保数据库文件的位置和存取路径合理，且具有一定的容错机制。

- **数据库文件路径：**数据库文件使用相对路径进行存储，以确保系统的可移植性。数据库文件命名为 'edu_system.db'，存储在 '/database' 目录下。
- **数据库文件大小：**SQLite 数据库支持大文件，默认情况下没有大小限制，但系统应根据实际使用需求设置合理的文件大小限制，以避免文件过大影响性能。
- **数据备份：**定期备份 '.db' 文件，以防止数据丢失。可以设置自动备份机制，确保数据的安全性。

3.4.2 表的物理存储设计

SQLite 会将所有数据表、索引和其他数据库对象存储在一个数据库文件中，表的数据存储与文件系统的操作相关。以下是数据库表的物理存储设计：

- **数据表存储：**每个数据表（如学生表、课程表、教师表等）在 SQLite 数据库文件中存储为不同的页面，每个页面的默认大小为 4096 字节。SQLite 会根据实际存储的数据量分配多个页面，以保证数据存储的高效性。
- **索引存储：**对于需要频繁查询的字段（如学生 ID、教师 ID、课程 ID 等），我们将创建索引来加速查询操作。SQLite 会将索引结构与数据表分开存储，并使用 B 树结构来存储索引，确保快速定位数据。
- **表与索引的分布：**为了提高查询效率，常用查询的表（如成绩表、课程表、学生表等）会考虑加上适当的索引。表的存储会根据表的大小动态调整页的数量，确保数据读取和写入的高效性。

3.4.3 字段数据存储设计

字段的存储设计主要基于字段的数据类型。SQLite 会根据每个字段的数据类型选择合适的存储格式。以下是各字段类型的存储方式：

- **INTEGER：**存储整数类型的数据，通常采用 4 字节或 8 字节进行存储，具体取决于整数的大小。
- **TEXT：**存储字符串类型的数据，SQLite 会根据字符串的实际长度分配相应的存储空间。对于较短的文本，SQLite 可能会使用动态长度存储，确保存储空间的高效使用。
- **REAL：**存储浮动小数点类型的数据，SQLite 使用 8 字节存储每个 REAL 类型的数据。
- **TIMESTAMP：**存储时间戳类型的数据，通常以 64 位整数存储，表示自 1970 年 1 月 1 日以来的秒数或毫秒数。

3.4.4 数据访问优化与索引设计

为提高系统的查询性能，特别是对于频繁查询的操作，需要设计高效的索引。以下是一些索引设计和优化的策略：

- **主键索引**: 对于每个主键字段（如 'student_id'、'course_id'、'teacher_id' 等），自动生成主键索引，以确保数据的唯一性和查询的高效性。
- **外键索引**: 对于涉及外键关系的字段（如 'student_id'、'course_id'、'teacher_id' 等），可以创建外键索引，以加速联接查询。
- **联合索引**: 对于经常组合查询的字段（如成绩表中的 'student_id' 和 'course_id'），可以创建联合索引，以优化多字段查询的效率。
- **查询优化**: 对于频繁执行的查询操作，可以使用 SQLite 的查询优化功能，如查询缓存、统计信息、EXPLAIN 命令等，确保查询执行计划的优化。

3.4.5 数据完整性约束设计

为了保证数据的完整性和一致性，SQLite 支持各种约束和触发器设计。以下是主要的设计措施：

- **NOT NULL 约束**: 确保关键字段（如 'student_id'、'course_id' 等）不能为空，以避免无效数据的插入。
- **唯一性约束**: 对于如学号、教师 ID 等需要唯一的字段，使用唯一性约束来防止数据重复。
- **外键约束**: 保证表与表之间的参照关系一致性，如学生选修课程时，必须确保学生和课程在各自的表中存在相应的记录。
- **触发器**: 使用触发器实现数据的自动维护，如当学生成绩表中的成绩更新时，自动更新学生的综合成绩。

4 数据库实施

4.1 创建数据库

/py/init_database.py

```
import sqlite3
from config import DATABASE_PATH
import os
```

```

def init_database():
    # 确保数据库目录存在
    os.makedirs(os.path.dirname(DATABASE_PATH), exist_ok=
        True)

    conn = sqlite3.connect(DATABASE_PATH)
    cursor = conn.cursor()

    # 创建课程表
    cursor.execute('''
CREATE TABLE IF NOT EXISTS courses (
id INTEGER PRIMARY KEY AUTOINCREMENT,
name TEXT NOT NULL UNIQUE,
learn_time TEXT NOT NULL,
credit REAL NOT NULL,
usual_score INTEGER NOT NULL,
midterm_score INTEGER NOT NULL,
final_score INTEGER NOT NULL,
times TEXT NOT NULL DEFAULT ''
)
''')

    # 创建学生表
    cursor.execute('''
CREATE TABLE IF NOT EXISTS students (
id INTEGER PRIMARY KEY AUTOINCREMENT,
name TEXT NOT NULL,
student_id TEXT NOT NULL UNIQUE,
enrollment_year INTEGER NOT NULL
)
''')

    # 创建教师表
    cursor.execute('''

```



```

CREATE TABLE IF NOT EXISTS teachers (
id INTEGER PRIMARY KEY AUTOINCREMENT,
name TEXT NOT NULL,
teacher_id TEXT NOT NULL UNIQUE
)
'''

# 创建学生课程关系表
cursor.execute('''
CREATE TABLE IF NOT EXISTS student_courses (
student_id INTEGER,
course_id INTEGER,
FOREIGN KEY (student_id) REFERENCES students (id) ON
DELETE CASCADE,
FOREIGN KEY (course_id) REFERENCES courses (id) ON
DELETE CASCADE,
PRIMARY KEY (student_id, course_id)
)
'''

# 创建教师课程关系表
cursor.execute('''
CREATE TABLE IF NOT EXISTS teacher_courses (
teacher_id INTEGER,
course_id INTEGER,
FOREIGN KEY (teacher_id) REFERENCES teachers (id) ON
DELETE CASCADE,
FOREIGN KEY (course_id) REFERENCES courses (id) ON
DELETE CASCADE,
PRIMARY KEY (teacher_id, course_id)
)
'''

# 创建成绩表

```

```

cursor.execute('''
CREATE TABLE IF NOT EXISTS grades (
student_id INTEGER,
course_id INTEGER,
usual_grade REAL DEFAULT 0,
midterm_grade REAL DEFAULT 0,
final_grade REAL DEFAULT 0,
FOREIGN KEY (student_id) REFERENCES students (id) ON
    DELETE CASCADE,
FOREIGN KEY (course_id) REFERENCES courses (id) ON
    DELETE CASCADE,
PRIMARY KEY (student_id, course_id)
)
''')

# 创建作业表
cursor.execute('''
CREATE TABLE IF NOT EXISTS assignments (
id INTEGER PRIMARY KEY AUTOINCREMENT,
course_id INTEGER NOT NULL,
title TEXT NOT NULL,
content TEXT NOT NULL,
create_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY (course_id) REFERENCES courses (id) ON
    DELETE CASCADE
)
''')

conn.commit()
conn.close()
print("数据库初始化完成")

if __name__ == '__main__':
    init_database()

```

4.2 加载数据

在 templates 各文件夹下各 html 文件里面有类似如下函数来加载数据：

```
function loadStudents() {  
  window.getStudents()  
    .then(response => {  
    if (response && response.success) {  
      window.studentsList = response.data; // 保存学生列表  
      filterStudents(''); // 初始显示所有学生  
    }  
  })  
    .catch(error => {  
      console.error('加载学生列表失败:', error);  
      alert('加载学生列表失败: ' + error.message);  
    });  
}
```

通过 window.getStudents() 获取学生数据，并在成功获取后将数据存储在 window.studentsList 中，然后调用 filterStudents 函数来显示学生。若发生错误，它会捕获错误并显示相关信息。

5 数据库应用程序设计

5.1 系统功能模块图

- 用户管理模块：负责用户的注册、登录等功能。
- 课程管理模块：用于管理课程的创建、修改、删除以及课程信息的查询。
- 学生管理模块：用于管理学生的信息，支持学生信息的增、删、改、查。
- 教师管理模块：用于管理教师的信息，支持教师信息的增、删、改、查。
- 学习进度模块：用于跟踪学生的学习进度。

- 作业管理模块：支持布置作业

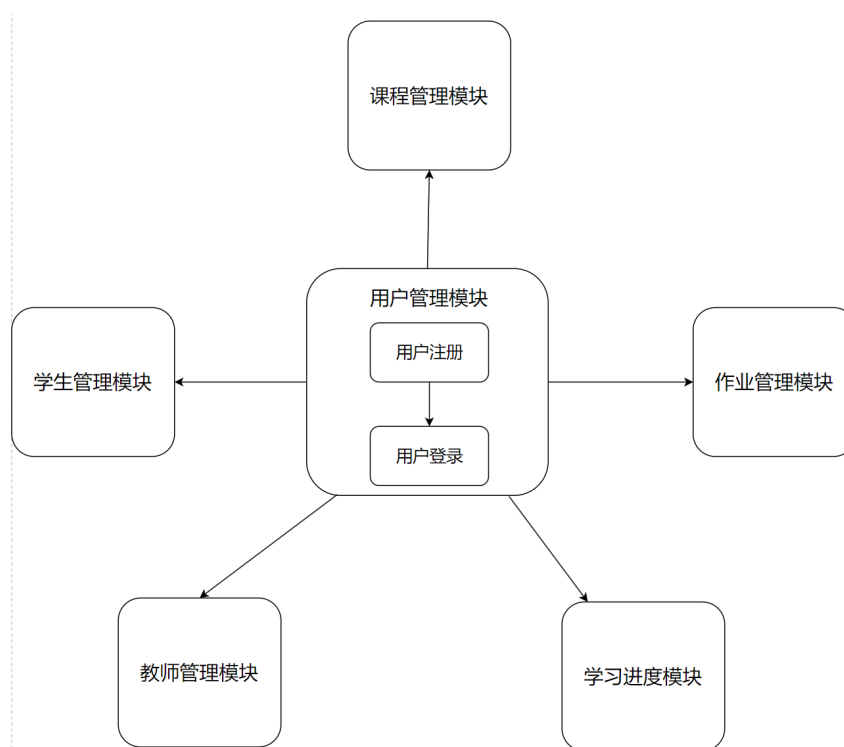


图 10: 系统模块图

5.2 用户管理功能

用户管理功能是系统的重要组成部分，主要包括用户的注册、登录和身份验证等功能。通过该功能，用户可以注册账号、登录系统以及进行基本的身份验证。下面将详细介绍该模块的具体功能和实现过程。

5.2.1 用户注册功能

用户注册功能允许新用户创建账户。用户需要提供一个唯一的用户名和密码，系统会验证用户名是否已被注册，如果未被注册，则会将新用户的信息存入数据库。在实际应用中，密码应进行加密存储，避免直接存储明文密码。注册成功后，系统会为用户分配一个唯一的用户标识符，并返回注册成功的消息。

5.2.2 用户登录功能

用户登录功能允许已注册的用户通过提供用户名和密码进行登录。系统首先验证输入的用户名是否存在，如果存在，则将存储的密码与输入的密码进行比对。如果密码正

确，系统会为用户创建会话，并允许用户访问受保护的资源。在实际开发中，登录时应使用哈希后的密码进行比对，并确保会话的安全性。

5.3 课程管理功能

课程管理功能是系统中一个核心的管理模块，主要用于课程的增、删、改、查操作。管理员或教师可以通过该功能进行课程信息的管理，包括课程名称、学习时间、学分、成绩比重以及上课时间等。以下是该模块的具体功能和操作流程。

5.3.1 课程添加功能

课程添加功能允许管理员或教师新增课程信息。用户可以填写课程的相关信息，包括课程名称、学习时间、学分、成绩构成（平时成绩、期中成绩、期末成绩的比例）以及课程的上课时间段。系统提供了选择时间段的功能，允许用户选择具体的上课时间段。填写完成后，用户点击“添加”按钮，课程信息将被保存到数据库中。

5.3.2 课程删除功能

课程删除功能允许管理员删除已存在的课程。当用户点击“删除”按钮时，系统将提示是否确认删除该课程。如果用户确认，课程信息将从数据库中移除。此功能有助于确保课程信息的及时更新，避免过时的课程信息仍然出现在系统中。

5.3.3 课程修改功能

课程修改功能允许管理员或教师对已有课程信息进行修改。如果课程的任何信息（如课程名称、学分、成绩比例、上课时间等）需要更新，用户可以在编辑框中修改相关内容，修改后点击“保存”按钮，系统将更新数据库中的课程信息。

5.3.4 课程查看功能

课程查看功能使管理员或教师能够查看所有已添加的课程及其详细信息。系统将列出所有课程的基本信息，包括课程名称、学分、上课时间等，并可根据需要查看具体的成绩比重等详细内容。用户可以选择查看某一课程的具体信息进行修改或删除。

5.4 学生管理功能

学生管理功能是系统中的一个关键模块，旨在提供对学生信息和选课情况的全面管理。该模块包括学生的注册、选课、修改信息、删除学生等功能。管理员或教师通过该功能可以方便地进行学生信息的管理和课程的报名、查看等操作。以下是该模块的具体功能和操作流程。

5.4.1 学生注册功能

学生注册功能允许新生或已有学生进行系统注册。在注册过程中，学生需要提供基本信息，如用户名、密码等。管理员可以通过该功能将学生的个人信息（如学号、姓名、年级等）录入系统，学生在注册后即可使用系统进行选课和查询等操作。

5.4.2 报名课程功能

报名课程功能允许学生根据自身需求选择课程。学生可以查看所有可用的课程，并根据课程的时间安排、学分、课程内容等信息选择合适的课程。通过选课系统，学生将选定的课程加入到个人课程表中，并自动记录每门课程的选课信息。选课过程通常包括选择课程、确认报名等步骤，确保学生顺利报名。

5.4.3 修改学生信息功能

修改学生信息功能允许学生或管理员更新学生的个人信息。学生可以修改自己的基本信息（如联系方式、地址等），管理员可以帮助学生修改学号、年级等更为关键的信息。通过该功能，系统能够保持学生信息的最新状态，确保信息准确无误。

5.4.4 删除学生功能

删除学生功能允许管理员移除系统中不再需要的学生信息。学生的删除操作会涉及到从数据库中彻底移除学生的个人资料和相关的选课记录。为了避免误删除，系统通常会要求管理员进行二次确认。在执行删除操作时，系统应确保所有相关的选课信息也被删除，以免造成数据不一致。

5.4.5 查看选课功能

查看选课功能允许学生查看自己已报名的所有课程以及相关的课程信息，如课程名称、上课时间等。学生可以通过此功能查看自己当前的学习安排，了解课程的学习进度

和学分情况。此功能帮助学生有效地管理自己的学习计划，并根据需要进行调整或补充选课。

5.5 教师管理功能

教师管理功能是系统中的一个重要模块，旨在提供对教师信息及其所授课程的全面管理。该模块包括教师的注册、课程安排、教师信息的管理、教师的修改与删除等功能。管理员可以通过该模块管理教师信息，安排教师教授的课程，并查看教师的授课情况。以下是该模块的具体功能和操作流程。

5.5.1 教师注册功能

教师注册功能允许新教师或已有教师进行系统注册。在注册过程中，教师需要提供基本信息，如姓名、教师编号等。管理员可以通过该功能将教师的个人信息录入系统，教师在注册后即可使用系统进行课程安排、查看自己教授的课程等操作。

5.5.2 安排课程功能

安排课程功能允许管理员将课程分配给教师。在课程安排过程中，管理员可以根据教师的专长、教学能力及时间安排等因素，将具体课程指派给合适的教师。通过该功能，教师可以查看自己所教授的课程，掌握教学任务，进行必要的准备工作。

5.5.3 教师信息管理功能

教师信息管理功能允许管理员查看、修改教师的基本信息。管理员可以在系统中查看教师的姓名、教师编号、联系方式等信息，也可以根据需要更新教师的个人资料。此功能保证了教师信息的准确性，确保每位教师的联系方式和其他关键信息在系统中保持最新状态。

5.5.4 修改删除教师功能

修改教师功能允许管理员对教师的信息进行修改，包括更新联系方式、修改教学安排等。删除教师功能则允许管理员移除不再在系统中需要的教师信息。当删除教师信息时，系统会自动处理与该教师相关的课程安排，确保数据的一致性和完整性。删除操作通常会涉及到从数据库中删除教师的所有相关数据，如授课记录、课程安排等。

5.5.5 查看教师课程功能

查看教师课程功能允许管理员和教师查看某位教师所教授的所有课程信息。管理员可以通过此功能查看每位教师的授课情况，了解教师的工作安排，便于合理调配教学资源。教师本人则可以查看自己所授课程的具体信息，如课程名称、上课时间等，帮助教师更好地管理自己的教学任务。

5.6 学习进度功能

学习进度功能是系统中的核心模块之一，主要用于跟踪和记录学生在各个课程中的学习进度，包括学习情况的统计分析、学生成绩的管理、以及根据学习进度为学生提供反馈。该模块帮助教师和学生了解课程的学习进度，并能及时发现学生在学习过程中可能遇到的问题。以下是该模块的具体功能和操作流程。

5.6.1 搜索学生功能

搜索学生功能允许管理员通过学生的学号、姓名等信息快速查找特定学生。在使用该功能时，系统会根据输入的搜索条件，检索数据库中的学生信息，并显示与条件匹配的学生。此功能对于教师和管理员来说，能够提高查询效率，方便地找到目标学生，查看其学习进度、成绩等详细信息。

5.6.2 学习统计功能

学习统计功能通过对学生在各门课程中的学习情况进行统计和分析，按照西工大现行的绩点计算方法展示该学生的所有课程的平均分和绩点。

5.6.3 保存课程成绩功能

保存课程成绩功能录入并保存学生在每门课程中的成绩。管理员可以根据学生的平时成绩、期中成绩、期末成绩等评定学生的总成绩，并将其保存在系统中。系统支持对成绩的修改和更新，确保学生的成绩信息始终准确无误。在成绩录入过程中，系统通常会进行必要的验证，确保成绩的合理性（例如成绩不能超出 100 分），并可以自动计算学生的最终成绩。

5.7 作业管理功能

作业管理功能是系统中的重要组成部分，主要用于帮助教师和管理员管理和发布作业，跟踪学生的作业完成情况。该功能通过提供作业的发布等操作。以下是该模块的具体功能和操作流程。

5.7.1 搜索课程功能

搜索课程功能允许教师或管理员根据课程名称、课程编号等信息快速查找特定的课程。当教师需要发布作业时，首先通过该功能选择要发布作业的课程。系统会显示所有相关的课程信息，教师可以在课程列表中找到需要的课程，确保作业发布到正确的课程。该功能提高了操作效率，避免了误操作和重复发布。

5.7.2 发布新作业功能

发布新作业功能允许教师将作业布置给学生。教师可以在该功能中设置作业的详细信息，如作业名称、作业内容等。通过该功能，教师可以为每个课程发布一个或多个作业。

6 数据库应用程序开发

6.1 数据库连接

打开自动连接数据库。

6.2 用户管理功能



登录

用户名:

密码:

登录 注册新账号

图 11

6.3 课程管理功能

添加课程

删除课程

修改课程

查看课程

添加课程

课程名称:

学习期限:

大一

学分:

平时成绩占比:

期中成绩占比:

期末成绩占比:

上课时间:

选择星期

选择时间

删除

选择星期

选择时间

删除

选择的课程

保存

图 12

6.4 学生管理功能

注册学生

报名信息

修改学生信息

删除学生

查看选课

注册学生

学生姓名:

学号:

入学年份:

选择入学年份

注册

图 13

6.5 教师管理功能

注册教师

安排课程

教师信息

修改删除教师

查看教师课程

注册教师

教师姓名:

教师职工号:

注册

图 14

6.6 学习进度功能

学习进度跟踪

搜索学生:

输入学号或姓名搜索

选择学生

学习统计

总学分: 0

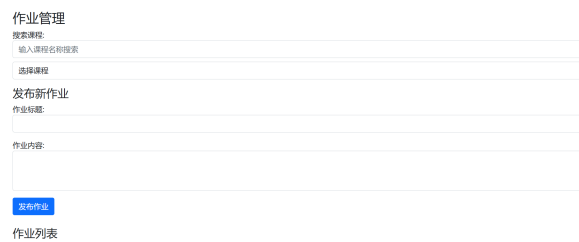
平均分: 0

平均分: 0

保存并生成报表

图 15

6.7 作业管理功能



作业管理

搜索课程:
输入课程名称搜索

选择课程

发布新作业

作业标题:

作业内容:

发布作业

作业列表

图 16

7 数据库应用系统运行环境

- python
- SQLite
- 浏览器

详见 README.md(使用步骤、作业分工等)