# 基于 Flask 的智能教学辅助系统开发文档

## ****项目概述****

本系统旨在开发一个基于 Flask 的智能教学辅助系统，主要功能包括考勤管理、抢答互动、作业布置与批改、智能问答助手、个性化学习路径推荐和数据分析。

### ****开发成员与分工****

* **开发者 A**：考勤与抢答模块
* **开发者 B**：作业与智能问答模块
* **开发者 C**：个性化学习路径与数据分析模块

## ****系统架构设计****

* **前端**：HTML、CSS，用于实现页面展示。
* **后端**：Flask 框架，负责业务逻辑与接口服务。
* **数据库**：使用 SQLite（开发阶段），用于数据存储。
* **API**：各模块之间通过 RESTful API 进行数据交互。

## ****1.数据库设计****

| **表名** | **字段名** | **类型** | **描述** |
| --- | --- | --- | --- |
| Users | id | INTEGER | 用户唯一标识（自增主键） |
|  | identifier | TEXT | 学号（学生）或教工号（教师） |
|  | role | TEXT | 用户角色（teacher/student） |
|  | name | TEXT | 用户姓名 |
|  | email | TEXT | 邮箱 |
|  | password | TEXT | 密码（加密存储） |
| Attendance | id | INTEGER | 考勤记录唯一标识（自增主键） |
|  | user\_id | INTEGER | 学生 ID（外键关联 Users.id） |
|  | date | DATE | 考勤日期 |
|  | status | TEXT | 签到状态（present/absent） |
| Assignments | id | INTEGER | 作业唯一标识（自增主键） |
|  | teacher\_id | INTEGER | 教师 ID（外键关联 Users.id） |
|  | title | TEXT | 作业标题 |
|  | description | TEXT | 作业描述 |
|  | due\_date | DATE | 截止日期 |
| Submissions | id | INTEGER | 提交记录唯一标识（自增主键） |
|  | assignment\_id | INTEGER | 对应作业 ID（外键） |
|  | student\_id | INTEGER | 学生 ID（外键关联 Users.id） |
|  | submission\_date | DATETIME | 提交时间 |
|  | grade | REAL | 自动评分 |
| Reports | id | INTEGER | 报告唯一标识（自增主键） |
|  | teacher\_id | INTEGER | 教师 ID（外键关联 Users.id） |
|  | data | TEXT | 分析数据（JSON 格式） |
| Quiz | id | INTEGER | 抢答活动唯一标识（自增主键） |
|  | teacher\_id | INTEGER | 教师 ID（外键） |
|  | question | TEXT | 抢答题目 |
|  | start\_time | DATETIME | 抢答开始时间 |
| QuizResponses | id | INTEGER | 抢答记录唯一标识（自增主键） |
|  | quiz\_id | INTEGER | 对应抢答活动 ID（外键） |
|  | student\_id | INTEGER | 学生 ID（外键） |
|  | response | TEXT | 学生回答内容 |
|  | response\_time | DATETIME | 提交时间 |
|  |  |  |  |

### 数据库设计细节

**Users 表**

* 1. identifier 用于存储学号或教工号，确保唯一性。
  2. role 字段区分学生和教师身份。
  3. 加入 email 和 password，方便实现用户登录与管理功能。

**Attendance 表**

* 1. 每条考勤记录包含用户 ID、日期和签到状态，便于统计和查询。

**Assignments 与 Submissions 表**

* 1. 作业表中添加 teacher\_id 字段，用于标识是哪位教师布置了该作业。
  2. 提交表中记录提交时间和评分，支持后续数据分析。

**Quiz 与 QuizResponses 表**

* 1. 抢答活动单独建立表，方便管理活动的题目和时间信息。
  2. 抢答记录表记录每位学生的回答及响应时间。

## ****2.模块划分与开发细则****

### ****模块 1：考勤与抢答功能（开发者 A）****

**需求：需要有一个导入学生名单的界面（或者提供示例文件，让用户复制信息过去，具体包含的学生信息只需要包含数据库表中的信息即可），然后解析学生名单进行后续考勤。**

#### 考勤功能

* **示例代码**

from flask import Flask, request, jsonify

app = Flask(\_\_name\_\_)

attendance\_data = [] # 模拟存储考勤数据

@app.route('/attendance', methods=['GET'])

def attendance\_page():

return "考勤页面 - 教师可以发起考勤"

@app.route('/attendance/submit', methods=['POST'])

def submit\_attendance():

data = request.json

attendance\_data.append(data)

return jsonify({"message": "签到成功", "data": attendance\_data})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

* **任务细化**
  1. 创建教师发起考勤的页面。
  2. 实现学生签到数据提交功能。
  3. 将考勤记录存储到数据库。

#### 抢答功能

* **示例代码**

quiz\_data = [] # 存储抢答活动数据

@app.route('/quiz/start', methods=['POST'])

def start\_quiz():

quiz = request.json

quiz\_data.append(quiz)

return jsonify({"message": "抢答活动已开始", "quiz": quiz})

@app.route('/quiz/response', methods=['POST'])

def quiz\_response():

response = request.json

quiz\_data.append(response)

return jsonify({"message": "答案已提交", "response": response})

* **任务细化**
  1. 教师发起抢答活动，记录题目和时间。
  2. 学生提交答案，并记录抢答顺序。
  3. 设计用于显示抢答排名的前端页面。

### ****模块 2：作业与智能问答功能（开发者 B）****

#### 作业功能

* **示例代码**

assignments = []

@app.route('/assignments', methods=['GET'])

def get\_assignments():

return jsonify(assignments)

@app.route('/assignments/create', methods=['POST'])

def create\_assignment():

assignment = request.json

assignments.append(assignment)

return jsonify({"message": "作业已创建", "assignment": assignment})

@app.route('/assignments/submit', methods=['POST'])

def submit\_assignment():

submission = request.json

return jsonify({"message": "作业已提交", "submission": submission})

* **任务细化**
  1. 教师可以创建作业，输入标题和内容。
  2. 学生通过表单提交作业，支持文件上传。
  3. 实现系统自动批改作业的逻辑。

#### 智能问答功能

* **示例代码**

@app.route('/chat', methods=['GET'])

def chat\_page():

return "智能问答助手页面"

@app.route('/chat/ask', methods=['POST'])

def ask\_question():

question = request.json.get('question')

# 模拟调用大模型 API

answer = f"您问的是：{question}，答案是：示例答案。"

return jsonify({"question": question, "answer": answer})

* **任务细化**
  1. 创建学生提问页面。
  2. 模拟调用大模型生成答案并返回。

### ****模块 3：学习路径与数据分析功能（开发者 C）****

#### 个性化学习路径

* **示例代码**

@app.route('/learning-path', methods=['GET'])

def get\_learning\_path():

return jsonify({"path": ["学习目标 1", "学习目标 2"]})

@app.route('/learning-path/update', methods=['POST'])

def update\_learning\_path():

new\_path = request.json.get('path')

return jsonify({"message": "学习路径已更新", "new\_path": new\_path})

* **任务细化**
  1. 根据学生的学习数据生成学习路径。
  2. 实现路径更新的功能。

#### 数据分析功能

* **示例代码**

@app.route('/reports', methods=['GET'])

def get\_reports():

return jsonify({"reports": "分析报告示例"})

@app.route('/reports/generate', methods=['POST'])

def generate\_reports():

data = request.json

return jsonify({"message": "报告生成成功", "data": data})

* **任务细化**
  1. 收集考勤、抢答、作业数据。（要和其他两位同学商量这些数据的格式）
  2. 用 Python 库（如 Matplotlib 或 Pandas）生成数据可视化报告。

## ****3.开发流程与时间表****

1. **第一周**：
   * 数据库设计。
   * 确定模块接口规范。
   * 学习 Flask 基础语法和示例代码。
2. **第二至第三周**：
   * 独立开发各模块，参考示例代码。
   * 使用模拟数据进行测试。
3. **第四周**：
   * 集成模块，进行系统测试。
   * 开发首页，整合所有功能入口。
4. **第五周**：
   * 修复 Bug，优化系统性能。
   * 完成最终上线。

## ****注意事项****

* 确保接口返回格式一致（如 JSON）。
* 使用 Postman 测试接口。
* 模块开发完成后，及时更新接口文档。