数据库系统上机报告

西安电子科技大学

计算机科学与技术学院

2203015 班

团队成员：党子清，田一涵，刘恩，包乃涆

2024 年 11 月



|  |  |  |  |
| --- | --- | --- | --- |
| 学号 | 姓名 | 任务分配内容 | 项目贡献比重 |
| 22009201170 | 党子清 | 数据库创建，前后端交互 | 3.5 |
| 22009200561 | 包乃涆 | 前端设计 | 2 |
| 22009200652 | 刘恩 | 撰写报告 | 2 |
| 22009200239 | 田一涵 | 撰写报告 | 2.5 |

### 开发环境

**服务端：**

**选用的OS型号：Windows**

**选用的DBMS型号：DM8**

**客户端:**

**OS系统：安卓/Windows/IOS**

**实现：**

* **前端**
* **后端**

### 需求分析

**1.数据库设计需求**

（1）根据语义描述，设计数据库概念结构，绘制ER图。

（2）将ER图转化为关系模型，明确各关系模式的主码、外码及完整性约束。

**2.数据库创建需求**

（1）创建后台数据库。

（2）创建基本表，并建立必要的完整性约束。

**3.角色与权限需求**

（1）创建“管理员”和“普通用户”两种角色。

（2）管理员角色具有全部权限，普通用户角色具有insert、update、select权限。

（3）为普通用户创建视图，确保数据安全性。

**4.功能模块需求**

（1）管理员用户功能：

① 导入Excel表数据，包括赛题信息、评委信息、参赛团队信息、团队成员信息、指导教师信息、团队选题信息、作品成绩汇总表。

② 对以上表中的记录进行增加、修改和删除操作。

（2）普通用户功能：

① 修改团队获奖等级。

② 查询各团队最终得分，并按照赛题编号升序、团队编号升序、最终平均分降序输出。

③ 查询获奖团队详情信息。

④ 生成电子证书信息表。

⑤ 根据团队号查询该团队是否获奖。

基于项目需求，本团队决定采用达梦数据库（DM Database）官方提供的dmPython ODBC接口，以实现高级编程语言与数据库之间的交互操作，并利用游标对象进行数据检索和管理。在系统架构设计方面，我们将利用Flask框架来构建我们的Web应用系统，以确保系统的灵活性和可扩展性。

**数据库交互层设计：**

本项目将采用达梦数据库的dmPython模块，该模块作为ODBC接口，为Python提供了与DM数据库的连接和操作能力。

通过配置ODBC数据源，建立与DM数据库的连接，并利用游标（Cursor）对象执行SQL语句，实现对数据库的查询、更新、插入和删除操作。

**系统架构设计：**

系统后端将基于Flask框架进行开发，Flask是一个轻量级的Web应用框架，适用于构建中小型的Web应用系统。

利用Flask的路由系统，我们将定义多个端点，以支持管理员和普通用户的各项功能需求。

通过Flask的蓝图（Blueprints）功能，我们将实现模块化的代码组织，提高系统的可维护性和可扩展性。

系统将采用RESTful API设计原则，提供清晰的接口定义，便于前端调用和数据交换。

**安全性与权限控制：**

系统将实现基于角色的访问控制（RBAC），分别为管理员和普通用户分配不同的操作权限。

通过Flask的认证机制，确保只有授权用户才能访问特定的资源和执行相关操作。

为普通用户创建受限的数据库视图，以保护敏感数据不被未授权访问。

**功能模块实现：**

系统将提供一系列功能模块，包括数据导入、记录管理、成绩查询、获奖信息检索和电子证书生成等。

各功能模块将通过Flask视图函数实现，并遵循MVC（模型-视图-控制器）设计模式，以实现业务逻辑与表现层的分离。

通过上述设计，本团队将构建一个高效、安全、可扩展的竞赛组织管理系统。

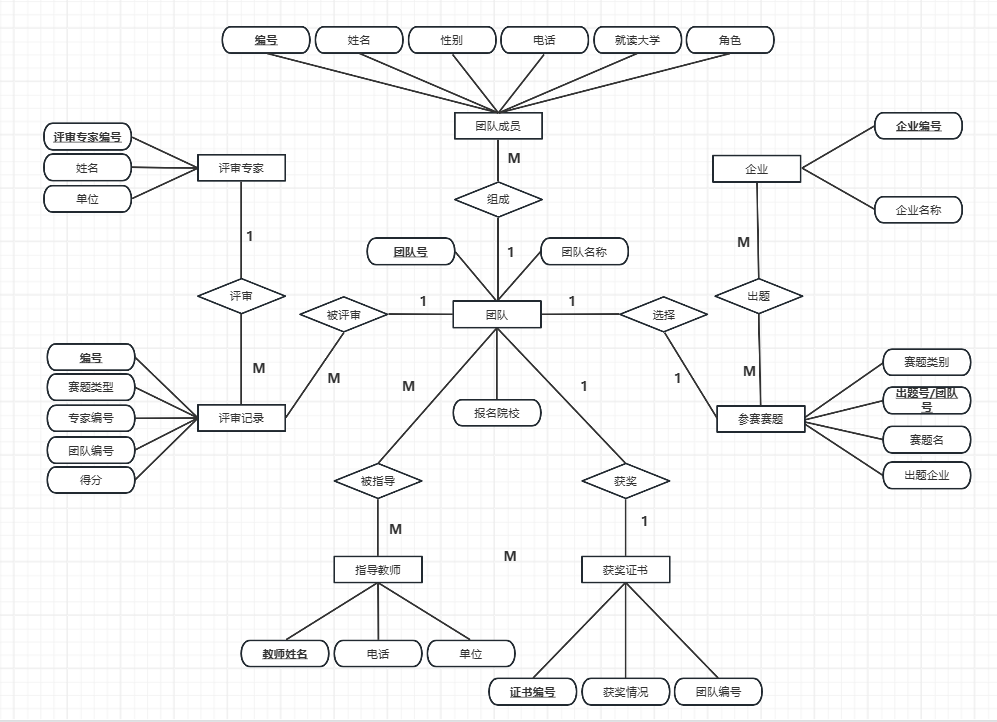
### 三、需求实现

#### 1、实体/模式设计

| **语义** | **实体/模式** |
| --- | --- |
| 团队 | Team |
| 成员 | Member |
| 指导老师 | Advisor |
| A题 | AProblem |
| B题 | BProblem |
| 专家 | Expert |
| 评审信息 | ReviewInfo |

#### 概念结构设计

#### ER图模型：



#### 逻辑结构设计

团队成员表

|  |
| --- |
| 编号(主键) |
| 姓名 |
| 性别 |
| 就读大学 |
| 角色 |
| 团队编号（外键） |

团队表

|  |
| --- |
| 团队编号 (主键) |
| 团队名称 |

企业表

|  |
| --- |
| 企业编号 (主键) |
| 企业名称 |

参赛赛题表

|  |
| --- |
| 编号(主键) |
| 赛题类型 |
| 出题与团队编号 (外键，关联团队表) |
| 赛题名称 |
| 出题企业 (外键，关联企业表) |

评审专家表

|  |
| --- |
| 评审专家编号 (主键) |
| 姓名 |
| 单位 |

评审记录表

|  |
| --- |
| 编号(主键) |
| 赛题类型 |
| 专家编号(外键，关联评审专家表) |
| 团队编号(外键，关联团队表) |
| 得分 |

指导教师表

|  |
| --- |
| 指导编号 (主键) |
| 教师姓名 |
| 电话 |
| 单位 |

8.获奖证书表

|  |
| --- |
| 证书编号(主键) |
| 获奖情况 |
| 团队编号(外键，关联团队表) |

#### 关系描述

1）团队与团队成员

关系：一对多 (一个团队有多个成员)

外键：团队成员表中的“团队编号”关联团队表。

2）参赛赛题与团队

关系：多对一 (一个赛题可以由多个团队选择)

外键：参赛赛题表中的“出题与团队编号”关联团队表。

3）参赛赛题与企业

关系：多对一 (一个赛题由一个企业提供)

外键：参赛赛题表中的“出题企业”关联企业表。

4）评审记录与团队、评审专家

关系：

多对一 (一个评审记录对应一个团队)

多对一 (一个评审记录对应一个评审专家)

外键：评审记录表中的“团队编号”关联团队表，“专家编号”关联评审专家表。

5）团队与指导教师

关系：多对多 (一个团队可以有多个指导教师，一个教师可以指导多个团队)

解决方式：增加中间表，包含“团队编号”和“指导编号”作为联合主键。

6）团队与获奖证书

关系：一对多 (一个团队可以获得多个证书)

外键：获奖证书表中的“团队编号”关联团队表。

### 系统实现

#### 后端

本系统采用达梦数据库的**dmPython模块**，该模块作为**ODBC**接口，为Python提供了与DM数据库的连接和操作能力。

通过配置ODBC数据源，建立与DM数据库的连接，并利用游标（Cursor）对象执行SQL语句，实现对数据库的查询、更新、插入和删除操作。

1. 管理员用户

1)根据给定的以下Excel表，将数据导入系统。给定的Excel表有“赛题信息、评委信息、参赛团队信息、团队成员信息、指导教师信息、团队选题信息、作品成绩汇总表”。

**代码**：

|  |
| --- |
| @app.route('/admin/import\_excel', methods=['POST'])  def import\_excel():  if 'file' not in request.files:  return jsonify({'error': 'No file part'}), 400    file = request.files['file']  if file.filename == '':  return jsonify({'error': 'No selected file'}), 400    if file:  df = pd.read\_excel(file)  return jsonify({'message': 'Data imported successfully'}), 200    return jsonify({'error': 'Failed to import data'}), 500 |

1. 增加、修改和删除以上表中记录。

**代码：**

|  |
| --- |
| @app.route('/admin/operation', methods=['GET', 'POST'])  def operate():  if request.method == 'POST':  operation = request.form['operation']  try:  if cursor.execute(operation):  return jsonify({'message': 'Operation completed successfully'}), 200  except Exception as e:  return jsonify({'error': 'Failed to complete operation'}), 500 |

1. 普通用户功能
2. 查看团队获奖等级。

**代码：**

|  |
| --- |
| @app.route('/user/team\_award', methods=['GET', 'POST'])  def team\_award():  team\_id = request.form["teamId"]  if not team\_id:  return jsonify({'error': 'Team ID is required'}), 400  cursor.execute(f"SELECT TeamID, Award, TeamName FROM Team WHERE TeamID = {team\_id}")  result = cursor.fetchone()    if result:  # return jsonify({'队伍号': result[0], '获奖情况': result[1]}), 200  return render\_template('team\_award\_output.html', team\_id=result[0], award=result[1], team\_name=result[2])  else:  return jsonify({'error': 'No award found for the given Team ID'}), 404 |

1. 查询各团队最终得分。输出信息有“团队编号，团队名称，团队选题，团队最终平均分“，要求按照赛题编号升序，团队编号升序，最终平均分降序输出。

**代码：**

|  |
| --- |
| @app.route('/user/team\_score', methods=['GET', 'POST'])  def team\_score():  cursor.execute("""  SELECT TeamID, TeamName, ProblemName, SCORE FROM Team  WHERE SCORE IS NOT NULL  ORDER BY score DESC;  """)  data = cursor.fetchall()  return render\_template('team\_score\_output.html', data=data) |

1. 查询获奖团队详情信息，输出”序号、赛题类别，所获奖项、团队编号，团队名称，赛题名称，所属院校，团队成员姓名，团队成员角色，指导教师姓名，指导教师院校“。

**代码：**

|  |
| --- |
| @app.route('/user/team\_details', methods=['GET', 'POST'])  def team\_details():  team\_id = request.form['teamId']  if not team\_id:  return jsonify({'error': 'Team ID is required'}), 400  cursor.execute(f"""  SELECT  Team.TeamID AS 团队编号,  Team.TeamName AS 团队名称,  Team.SCORE as 得分,  Team.Award AS 所获奖项,  Team.Category AS 赛题类别,  Team.ProblemName AS 赛题名称,  Team.School AS 所属院校,  LISTAGG(Member.Name, ';') WITHIN GROUP (ORDER BY Member.Role) AS 团队成员姓名,  Advisor.Name AS 指导教师姓名,  Advisor.Department AS 指导教师院校  FROM  Team  JOIN  Member ON Team.TeamID = Member.TeamID  JOIN  Advisor ON Team.School = Advisor.Department  WHERE  team.TeamID = {team\_id}  GROUP BY  Team.TeamID, Team.TeamName, Team.SCORE, Team.Award, Team.Category, Team.ProblemName, Team.School, Advisor.Name, Advisor.Department  """)  data = cursor.fetchone()  team\_memeber = data[7].split(';')  team\_memeber = ','.join(list(set(team\_memeber)))  if data:  return render\_template('team\_details\_output.html', data={'队伍号': data[0], '队伍名': data[1], '得分':data[2],  '获奖情况': data[3], '赛题类别' : data[4], '赛题名':data[5],  '所属学校':data[6], '团队成员':team\_memeber, '指导老师':data[8]})  else:  return jsonify({'error': 'No award found for the given Team ID'}), 404 |

1. 生成电子证书信息表。信息有“团队编号，团队名称，团队成员姓名，团队成员院校，指导教师姓名，指导教师院校“。

**代码：**

|  |
| --- |
| @app.route('/user/generate\_certificate', methods=['GET', 'POST'])  def generate\_certificate():  team\_id = request.form['teamId']  if not team\_id:  return jsonify({'error': 'Team ID is required'}), 400 |

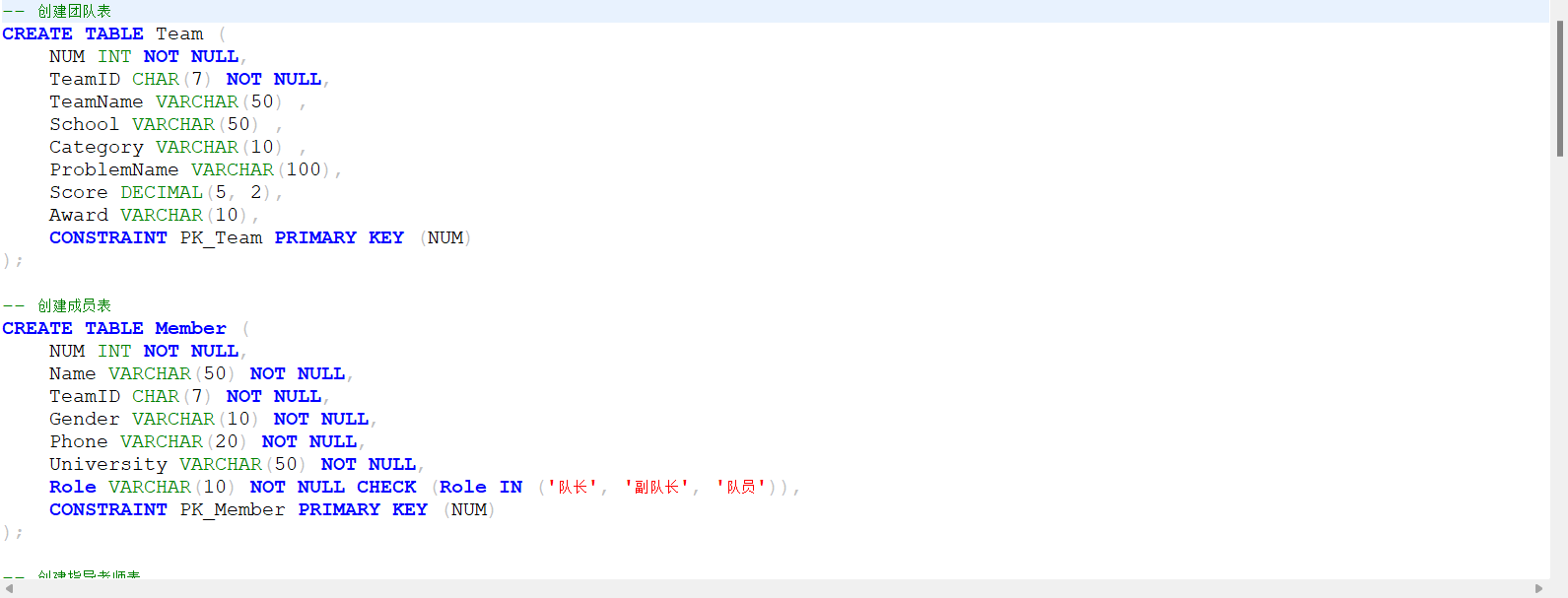
1. 根据给定的团队号，查询该团队是否获奖。

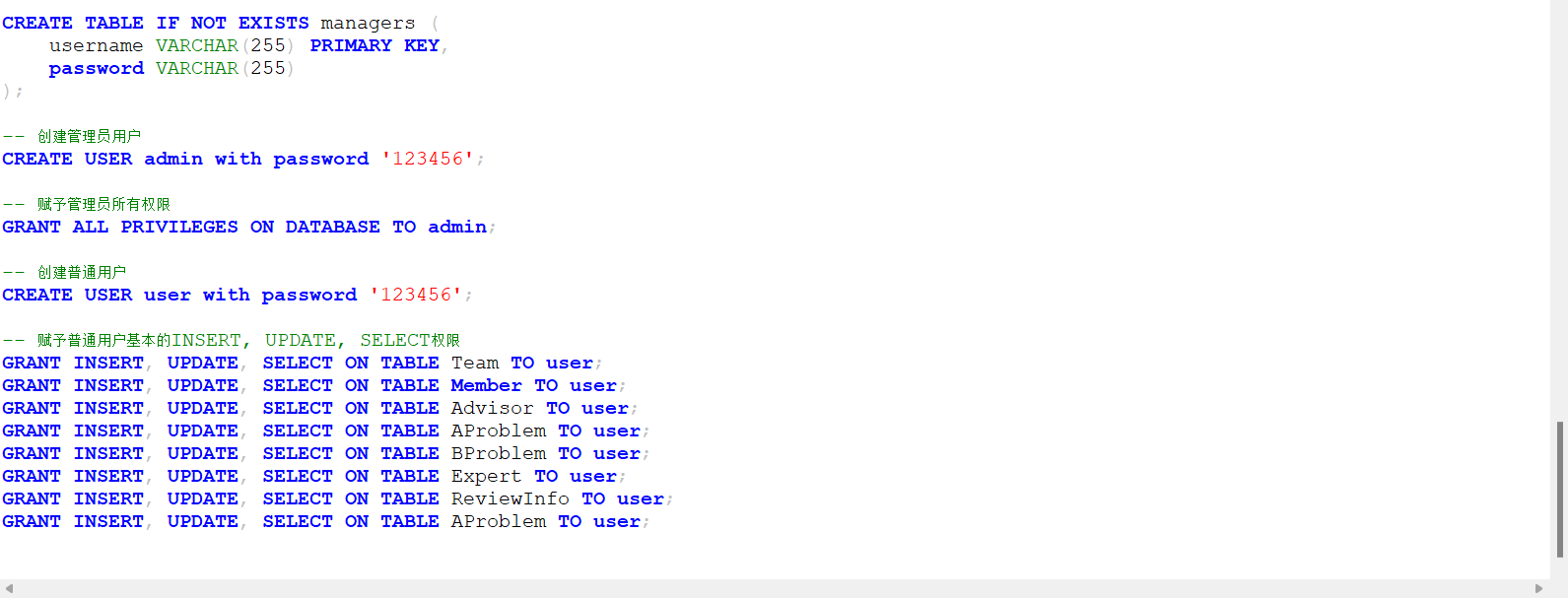
**代码**：

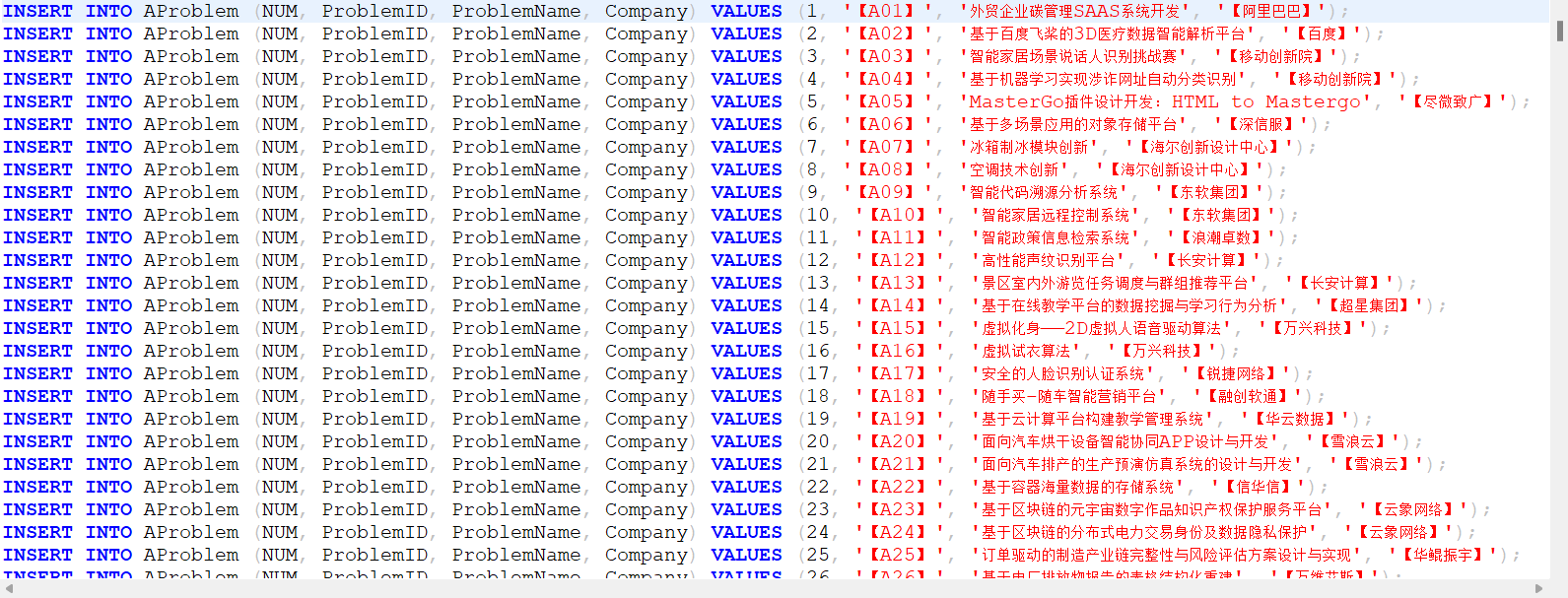
|  |
| --- |
| cursor.execute(f"""  SELECT  Team.TeamID AS 团队编号,  Team.TeamName AS 团队名称,  Team.SCORE as 得分,  Team.Award AS 所获奖项,  Team.Category AS 赛题类别,  Team.ProblemName AS 赛题名称,  Team.School AS 所属院校,  LISTAGG(Member.Name, ';') WITHIN GROUP (ORDER BY Member.Role) AS 团队成员姓名,  Advisor.Name AS 指导教师姓名,  Advisor.Department AS 指导教师院校  FROM  Team  JOIN  Member ON Team.TeamID = Member.TeamID  JOIN  Advisor ON Team.School = Advisor.Department  WHERE  team.TeamID = {team\_id}  GROUP BY  Team.TeamID, Team.TeamName, Team.SCORE, Team.Award, Team.Category, Team.ProblemName, Team.School, Advisor.Name, Advisor.Department  """)  data = cursor.fetchone()  team\_memeber = data[7].split(';')  team\_memeber = ','.join(list(set(team\_memeber)))  if data:  return render\_template('generate\_output.html', data={'队伍号': data[0], '队伍名': data[1], '得分':data[2],  '获奖情况': data[3], '赛题类别' : data[4], '赛题名':data[5],  '所属学校':data[6], '团队成员':team\_memeber, '指导老师':data[8]})  else:  return jsonify({'error': 'No award found for the given Team ID'}), 404 |

**数据库代码部分：**

1.创建表：



1. 创建用户
2. 插入数据：



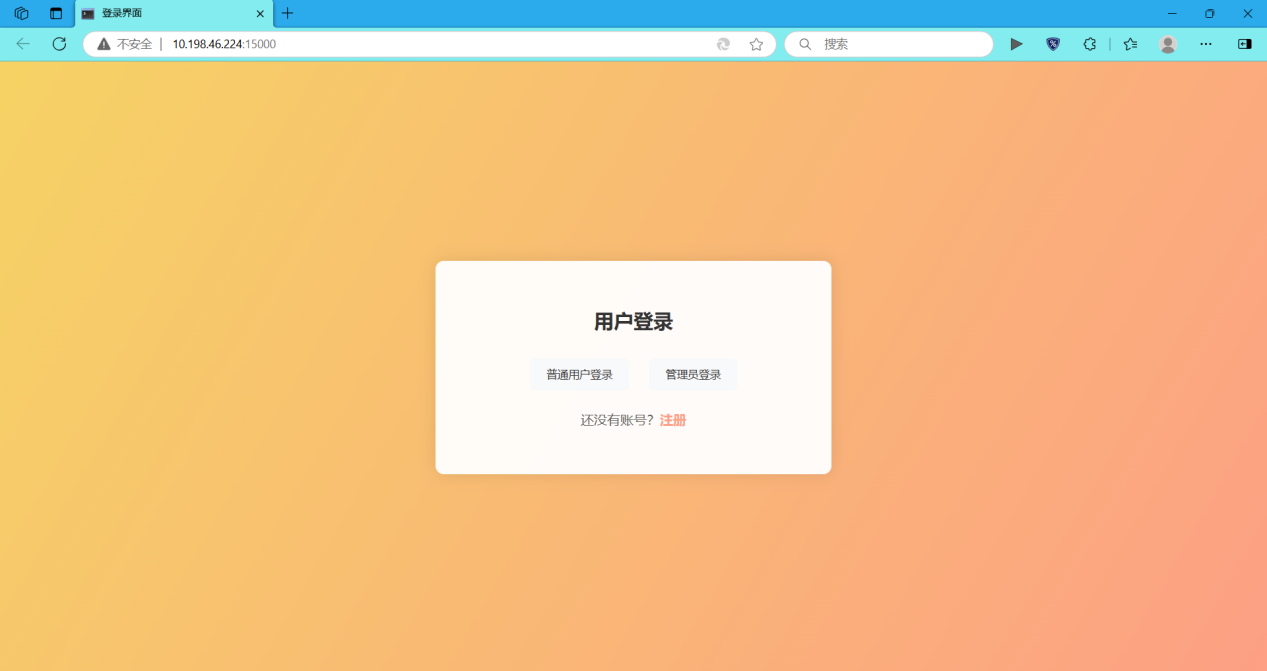
### 界面展示

#### 前端实现

采用html语言结合css，实现前端的设计。

#### 前端展示

登陆与注册界面：







管理员界面：



普通用户界面



管理员直接操作界面：



查看团队获奖等级：



查询队伍详情：

结果展示：



生成电子证书：