

# 数据库 Lab3 Report

许元元 PB20511899

数据库 Lab3 Report

许元元 PB20511899

## 1 概述

- 1.1 系统目标
- 1.2 需求说明
- 1.3 本报告的主要内容

## 2 总体设计

- 2.1 系统模块结构
- 2.2 系统工作流程
- 2.3 数据库设计
  - E-R 图
  - 最终的物理数据库结构

## 3 详细设计

- 3.1 实现配置
- 3.2 登录及注册模块
- 3.3 论文和项目的增删模块
- 3.4 发表论文 & 承担项目 & 教授课程模块
- 3.5 查询统计模块
- 3.6 About 展示模块

## 4 实现与测试

- 4.1 主界面
- 4.2 论文和项目的增删模块
- 4.3 发表论文 & 承担项目 & 教授课程模块
- 4.4 查询统计模块
- 4.5 About 展示模块

实现中的难点问题及解决

## 5 总结与讨论

# 1 概述

## 1.1 系统目标

本系统目标是开发一个面向教师的教学科研登记系统，提供一个便利美观的教学科研登记平台，提供多种个性化登记与查询服务，方便满足各类的需求。

## 1.2 需求说明

- 主要模块需求：
  - 论文、项目、课程的增加与删除模块；
  - 发表论文、负责项目和教授课程的相关信息查询及修改；
  - 以及一个查询模块，划定时间段对指定教师信息进行查询；
- 主要功能需求：

- **登记教师论文信息**: 提供教师论文信息的增、删、改、查功能；输入时要求检查：一篇论文只能有一位通讯作者，论文的作者排名不能有重复，论文的类型和级别只能在约定的取值集合中选取。
  - **登记承担项目情况**: 提供教师承担项目信息的增、删、改、查功能；输入时要求检查：排名不能有重复，一个项目中所有教师的承担经费总额应等于项目的总经费，项目类型只能在约定的取值集合中选取。
  - **登记主讲课程情况**: 提供教师主讲课程信息的增、删、改、查功能；输入时要求检查：一门课程所有教师的主讲学时总额应等于课程的总学时，学期。
  - **查询统计**:
    - 实现按教师工号和给定年份范围汇总查询该教师的教学科研情况的功能；例如输入工号“01234”，“2023-2023”可以查询 01234 教师在 2023 年度的教学科研工作情况。
    - 实现按教师工号和给定年份范围生成教学科研工作量统计表并导出文档的功能，我选择的是输出 .md 文件：

## 1.3 本报告的主要内容

这次报告主要描述我自行实现的面向教师的教学科研登记系统的方案，包括系统目标、总体设计、详细设计和主要功能模块的描述及测试信息。本次实验我选择的是 MySQL 作为后台 DBMS，使用 Python 开发前端，所有对于数据库的操作都并未选择使用触发器和存储过程，采用的是将所有的 SQL 在前端进行拼接与实现，由 Python 层面的 API 直接调用实现。

## 2 总体设计

## 2.1 系统模块结构

```
F::.
|   db-1ab03.pdf          # 实验要求
|   run.py                # Python 源文件!
|
|
|---Database
|   Create_Table.sql      # 创建数据库基本表
|   insert_init_data.sql  # 插入初始化数据
|
|
|---output    # 导出的教师查询信息
|   10002.md      # 名称为: "教师工号.md"
|
|
|---static
|   |---css
|   |   index.css    # 个性化初始界面的 CSS 代码
|   |   style1.css   # 里运行界面格式化 CSS 代码
|   |
|   |---js
|   |   echarts.min.js      # e-charts js源码
|   |   jquery-3.6.0.min.js
|
|
|   |---plugins
|   |   |---bootstrap-3.4.1
|   |   |   |---...
```

```

|   └──bootstrap-datepicker
|       └──...
|
|   └──font-awesome-4.7.0
|       └──...
|
└──templates
    about.html          # 项目相关信息展示页面
    author_edit.html     # 编辑论文作者信息页面
    course_info_add.html # 添加授课信息页面
    course_list.html     # 课程列表页面
    course_teacher_list.html # 课程授课信息展示页面
    index.html           # 用户登录界面
    layout.html          # 内部所有页面的框架页面
    login.html           # 登陆输入页面
    manager_edit.html    # 编辑项目管理教师信息页面
    project_edit.html    # 编辑项目信息页面
    project_info_add.html # 添加项目负责信息页面
    project_list.html    # 展示项目信息页面
    project_manager_list.html # 展示项目负责信息页面
    project_reg.html     # 项目登记页面
    publication_author_list.html # 论文作者展示页面
    publication_edit.html # 论文信息编辑页面
    publication_info_add.html # 论文信息添加页面
    publication_list.html # 论文展示页面
    publication_reg.html # 论文登记页面
    query_info.html      # 信息综合查询页面
    query_list.html       # 查询信息展示页面
    register.html         # 用户注册页面
    user_list.html        # 用户信息展示页面

```

这部分我没有选择画图描述，因为实际运行逻辑就是通过点击 *html* 网页中的不同部分实现在不同 *Python* 中模块之间的跳转，每次的跳转都会使得返回一个新的展示页面，即所有的交互都在前端文件中直接体现，而前端文件之间的层次关系几乎就是我设计的结构关系。

```

F: .
|   index.html          # 用户登录界面
|
└──Inner_part
    |   layout.html        # 内部所有页面的框架页面
    |   user_list.html      # 用户信息展示页面
    |
    └──About
        |   about.html        # 项目相关信息展示页面
    |
    └──Course
        |   author_edit.html    # 编辑论文作者信息页面
        |   course_info_add.html # 添加授课信息页面
        |   course_list.html     # 课程列表页面
    |
    └──Project
        |   manager_edit.html    # 编辑项目管理教师信息页面
        |   project_edit.html    # 编辑项目信息页面
        |   project_info_add.html # 添加项目负责信息页面
        |   project_list.html     # 展示项目信息页面

```

```

|   |       project_manager_list.html      # 展示项目负责信息页面
|   |       project_reg.html               # 项目登记页面
|
|   |--Publication
|   |       author_edit.html             # 编辑论文作者信息页面
|   |       publication_author_list.html # 论文作者展示页面
|   |       publication_edit.html        # 论文信息编辑页面
|   |       publication_info_add.html   # 论文信息添加页面
|   |       publication_list.html       # 论文展示页面
|   |       publication_reg.html       # 论文登记页面
|
|   |--Query
|   |       query_info.html            # 信息综合查询页面
|   |       query_list.html           # 查询信息展示页面
|
|   |--Login
|   |       login.html                # 登陆输入页面
|
|   |--Register
|   |       register.html             # 用户注册页面

```

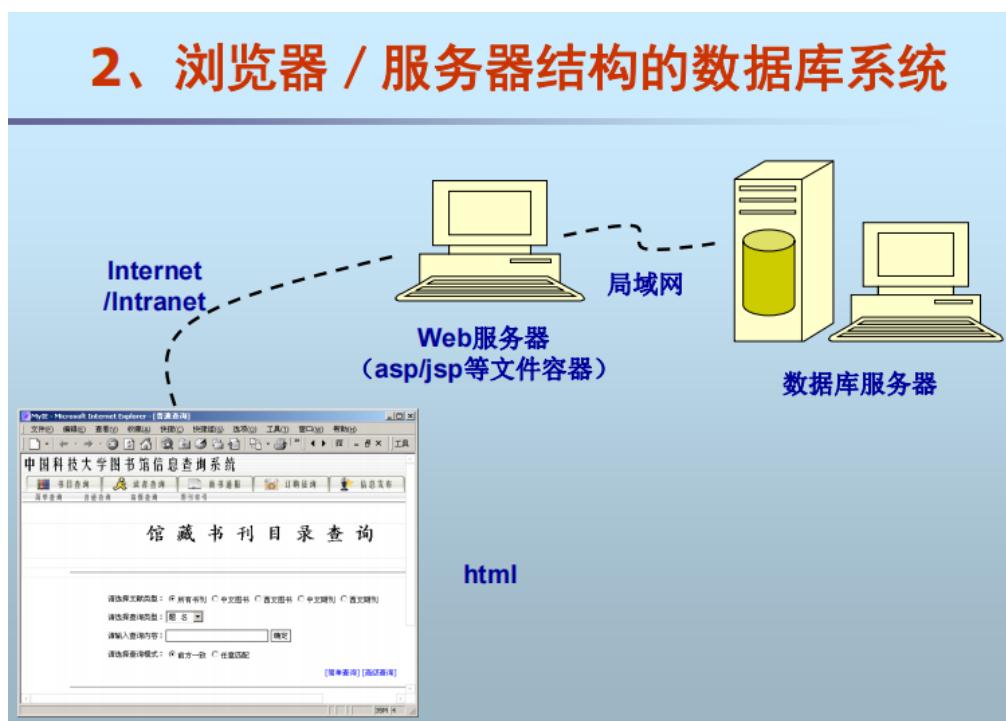
每个文件的具体功能同注释，并且在 run.py 中有一一对应的实现。

这里应为实验要求中明确说了不要求实现教师信息和课程信息的修改，于是我就省略了一部分的实现。

#### 预备数据：

1. 本项目假设教师和课程的信息已经在数据库中了，本实验不要求实现这 2 类数据的增删改查。实验需要的教师和课程记录请大家自行插入一些测试数据，测试数据中字段值的取值要满足“数据说明”中的规定。
2. 各类数据的类型可自行根据实际情况调整；
3. 测试数据自行设计。

## 2.2 系统工作流程



采用的是浏览器/服务器的架构；

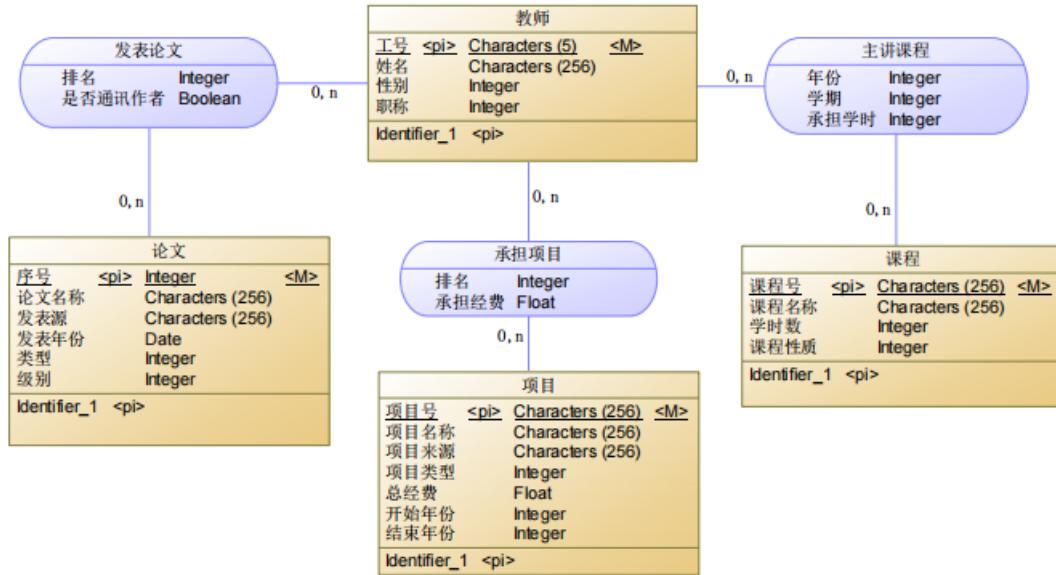
具体的工作流程感觉很简单，就不赘述了，几乎就和上面 **html** 的架构一样；

在不同的功能分区中反复跳转实现功能。

## 2.3 数据库设计

### E-R 图

教师教科研登记系统的数据需求如下：



实际的 ER 图多了一张基本表，存储用户的用户名和密码。

### 最终的物理数据库结构

在 ER 图的基础上做了一点点的修改，即：

- 在**发表论文**关系中加入了教师和论文的主码；
- 在**承担项目**关系中加入了教师和项目的主码
- 在**主讲课程**关系中加入了教师和课程的主码

## 3 详细设计

### 3.1 实现配置

#### 开发环境

MySQL Workbench 8.0 CE + VS Code

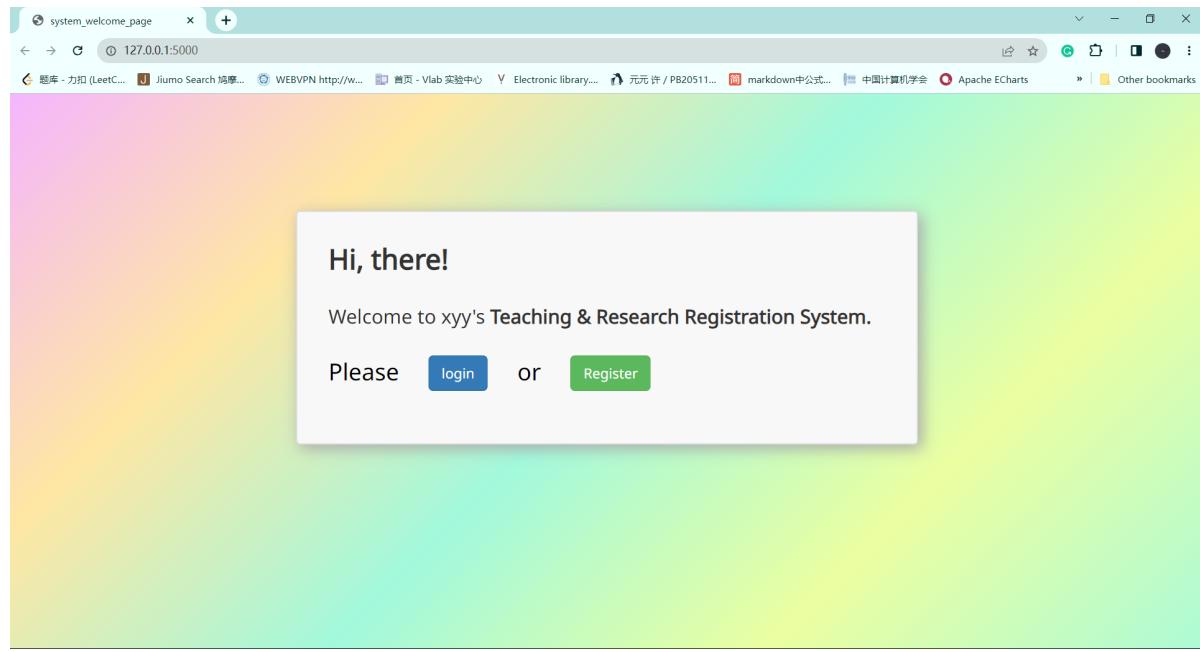
#### 数据库系统

Ver 8.0.32 for win64 on x86\_64 (MySQL Community Server - GPL)  
Copyright (c) 2000, 2023, Oracle and/or its affiliates.

## 技术栈 (Technology Stack)

```
Bootstrap CSS renderer  
Flask Framework (web)  
javascript (js)  
pymysql (python)  
HTML language (html)
```

## 3.2 登录及注册模块

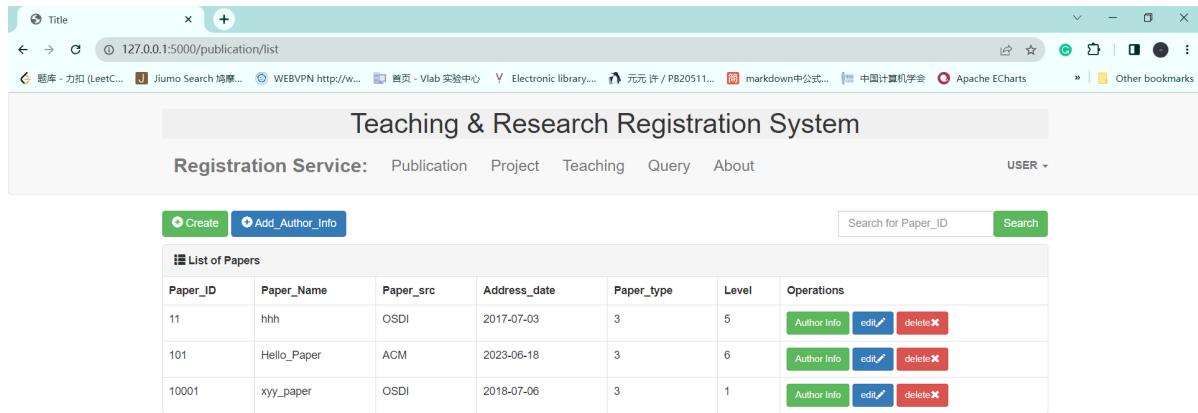


输入：登录 or 注册（鼠标点击交互）

输出 or 后续执行：

- **登录：**
  - 判断用户是否存在 & 密码是否正确
  - 都满足就直接跳转到内部页面，进行下一步的操作
- **注册：**
  - 跳转到相关页面进行账号密码的设定，**插入数据库的用户列表中**
  - 然后回到上述页面，可以进行登录

### 3.3 论文和项目的增删模块



The screenshot shows a web browser window titled "Teaching & Research Registration System". The URL is 127.0.0.1:5000/publication/list. The page has a header with "Registration Service: Publication Project Teaching Query About" and a user dropdown. Below the header is a toolbar with "Create" (green), "Add\_Author\_Info" (blue), "Search for Paper\_ID" (text input), and a "Search" button. A table titled "List of Papers" displays three rows of data:

Paper_ID	Paper_Name	Paper_src	Address_date	Paper_type	Level	Operations
11	hh	OSDI	2017-07-03	3	5	<a href="#">Author Info</a> <a href="#">edit</a> <a href="#">delete</a>
101	Hello_Paper	ACM	2023-06-18	3	6	<a href="#">Author Info</a> <a href="#">edit</a> <a href="#">delete</a>
10001	xyy_paper	OSDI	2018-07-06	3	1	<a href="#">Author Info</a> <a href="#">edit</a> <a href="#">delete</a>

**输入：**前端返回给后端的按钮信号，这里主要是左上部的 *Create* 和每个条目后的 *Delete* 和 *Edit*，还有右上的 *Search*

- **Create** 即创建新的；
- **Delete** 即删除对应的论文；
- **Edit** 即编辑论文的基本信息；
- **Search** 是按照 **Paper\_ID** 搜索论文；

**输出 or 后续执行：**

- **Create** 跳转到创建页面进行相关信息添加；
- **Delete** 即从数据库删除论文，并刷新展示的列表；
- **Edit** 即跳转到编辑页面修改论文的基本信息；
- **Search** 会按照 **Paper\_ID** 搜索并刷新展示的列表；

这里以论文为例，对项目的处理是类似的，具体的实现请见 **Part 4**。

### 3.4 发表论文 & 承担项目 & 教授课程模块

这个部分承接上一个论文的列表，点击每个条目后的 **Author\_Info** 就会展开看到每个论文的详细信息：

The screenshot shows a web browser window titled "Title" with the URL "127.0.0.1:5000/publication/list/author/11". The page header includes "Teaching & Research Registration System" and "Registration Service: Publication Project Teaching Query About". A dropdown menu "USER" is visible. The main content area is titled "List of Authors" and contains a table with two rows of data:

Paper_ID	Paper_name	Author_ID	Author_Name	Author_Rank	Is_Corresponding_Author	Operations
11	hhh	10002	吴亚颖	1	0	<a href="#">edit</a> <a href="#">delete</a>
11	hhh	10000	李诚	2	1	<a href="#">edit</a> <a href="#">delete</a>

At the top left, there are buttons for "Create\_Publication" and "Add\_Author\_Info". At the top right, there is a search bar "Search by Teacher\_ID" and a "Search" button.

于是我们进入了一个新的页面，可以对作者的信息进行增加，修改或者查找。

输入：

- **Add\_Author\_Info**: 添加新的作者信息；
- **edit**: 编辑已有的作者信息；
- **delete**: 删除已有的作者信息；

输出 or 后续执行：

- **Add\_Author\_Info**: 跳转到新的页面，添加作者信息（并进行约束检测）；
- **edit**: 跳转到新的页面，编辑已有的作者信息（并进行约束检测）；
- **delete**: 跳转到新的页面，删除已有的作者信息（并进行约束检测）；

这里以发表论文为例，对负责项目，教授课程的处理是类似的，具体的实现请见 Part 4。

## 3.5 查询统计模块

The screenshot shows a web browser window titled "Title" with the URL "127.0.0.1:5000/query". The page header includes "Teaching & Research Registration System" and "Registration Service: Publication Project Teaching Query About". A dropdown menu "USER" is visible. The main content area is titled "Query Information" and contains form fields for "Teacher\_ID", "Start\_Year", and "End\_Year", each with a corresponding input field. A "submit" button is at the bottom.

输入：

- **Teacher\_ID**: 教师工号;
- **Start\_Year**: 查询起始年份;
- **End\_Year**: 查询终止年份

### 程序执行流程:

- 返回结果, 跳转到展示页面, 如下:

**Registration Service:** Publication Project Teaching Query About USER -

Teacher Basic Information			
Teacher_ID	Teacher_Name	Sex	Title
10002	吴亚颖	女	博士后

Course Teaching Information				
Course_ID	Course_Name	Course_Hour	Year	Semester
SBZKD01	大数据算法	60	2002	夏
SBZKD00	数据库应用与管理	20	2023	春

Publication Information					
Paper_Name	Paper_src	Address_Date	Level	Pub_Rank	Is_Cor_Author
hh	OSDI	2017-07-03	中文CCF-B	排名第1	否
Hello_Paper	ACM	2023-06-18	无级别	排名第3	是
xyy_paper	OSDI	2018-07-06	CCF-A	排名第2	否

Project Incharge Information						
Project_name	Project_src	Project_type	start_year	end_year	Total_funding	Incharge_funding
XYY_Database	ADSL_Lab	其他项目类型	2023	2025	50000.0	20000.0

- 并且会输出一个 `.md` 文档到: `./output/<教师工号>.md`, 打开如下:

**教师教学科研工作统计 (1950-2030)**

### 教师基本信息

教师编号	教师姓名	教师性别	教师职称
10002	吴亚颖	女	博士后

### 教师授课信息

课程编号	课程名称	主讲学时	开课年份	开课学期
SBZKD01	大数据算法	60	2002	夏
SBZKD00	数据库应用与管理	20	2023	春

### 教师发表论文信息

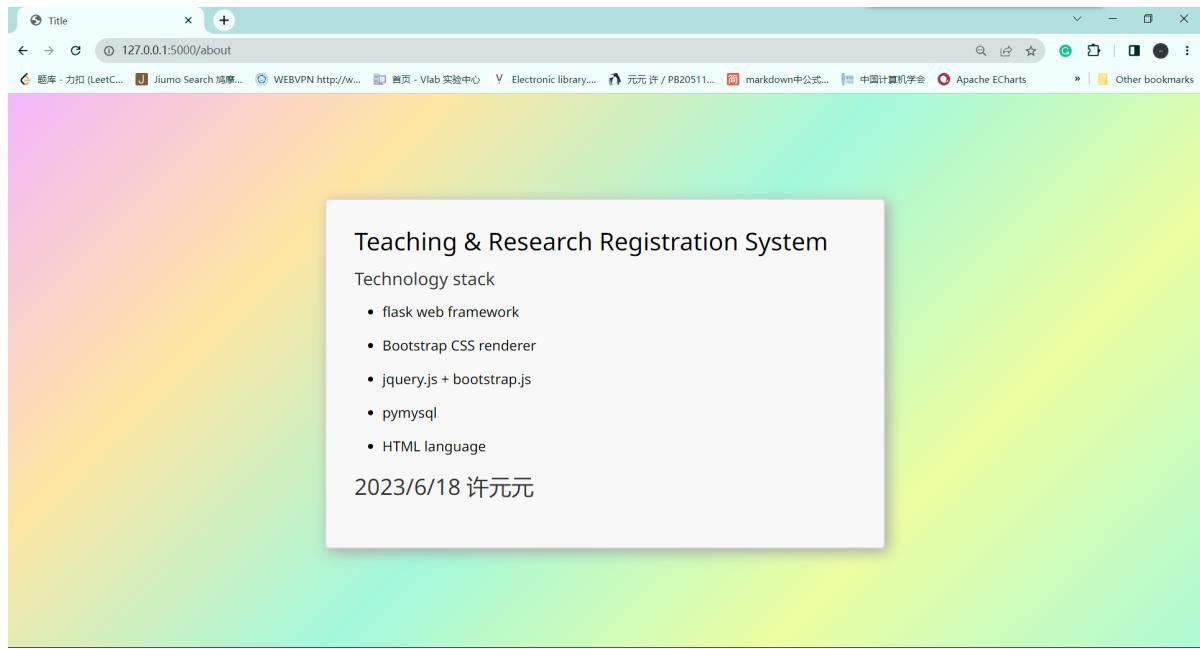
论文名称	论文来源	发表日期	论文级别	作者排名	是否为通讯作者
hh	OSDI	2017-07-03	中文CCF-B	排名第1	否
Hello_Paper	ACM	2023-06-18	无级别	排名第3	是
xyy_paper	OSDI	2018-07-06	CCF-A	排名第2	否

### 教师承担项目信息

项目名称	项目来源	项目级别	开始时间	结束时间	总经费	承担经费
XYY_Database	ADSL_Lab	其他项目类型	2023	2025	50000.0	20000.0

## 3.6 About 展示模块

点进去就是个展示模块, 没什么输出就。



## 4 实现与测试

这个部分我感觉没必要分成两部分说了，就直接边介绍实现，边展示对各种输入的测试输出结果！

因为这种依托前端的测试并没有什么样例的说法吧，我就手动进行一些输入，贴上输入输出的结果。

我的项目文件是完整的，如果想实际测试，只需要：

- 运行两个 **mysql** 文件创建数据库；
- 替换修改 **python** 源文件中对数据库连接时的用户名和密码；
- 运行即可：

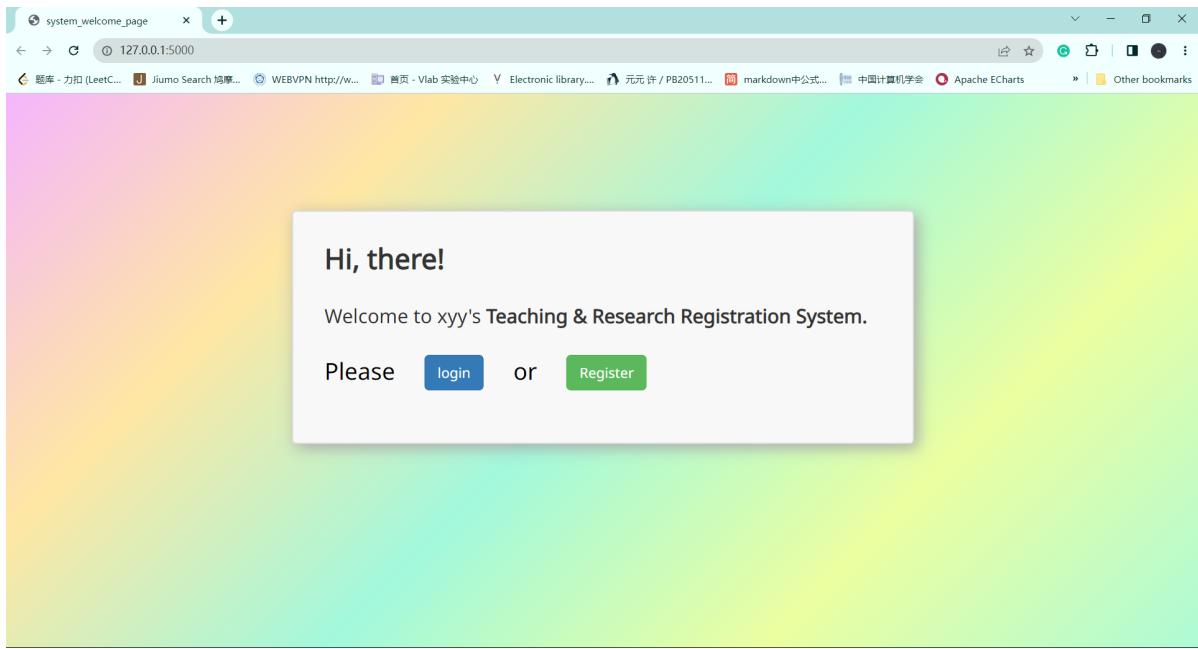
```
python run.py
```

因为功能过于繁复，包括报错信息，不可能每个都展示，可以实际跑一下，我个人是把实验要求中所有提到了的输入条件约束，以及运行中的条件判断全部都加入了我的判断中，并且都在前端设计了会有弹窗提醒。

### 4.1 主界面

```
@app.route('/', methods=['GET', 'POST'])
def front():
    return render_template('index.html')
```

展示结果如下：



通过单机两个不同的按钮可以实现不同的操作：

实际运行时，是接收到了 `index.html` 的跳转信号，从而进入了不同 `html` 文件

```
@app.route('/register', methods=['GET', "POST"])
def register():
    if request.method == "GET":
        return render_template('register.html')

    username = request.form.get("User_Name")
    password = request.form.get("User_Password")
    print(username, password)
    conn = pymysql.connect(host="127.0.0.1", port=3306, user='root',
                           passwd='123456', db='tmsys', charset='utf8')
    cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)

    sql = "select * from User where User_Name = %s"
    sql1 = "insert into User(User_Name, User_Password) values(%s, %s)"

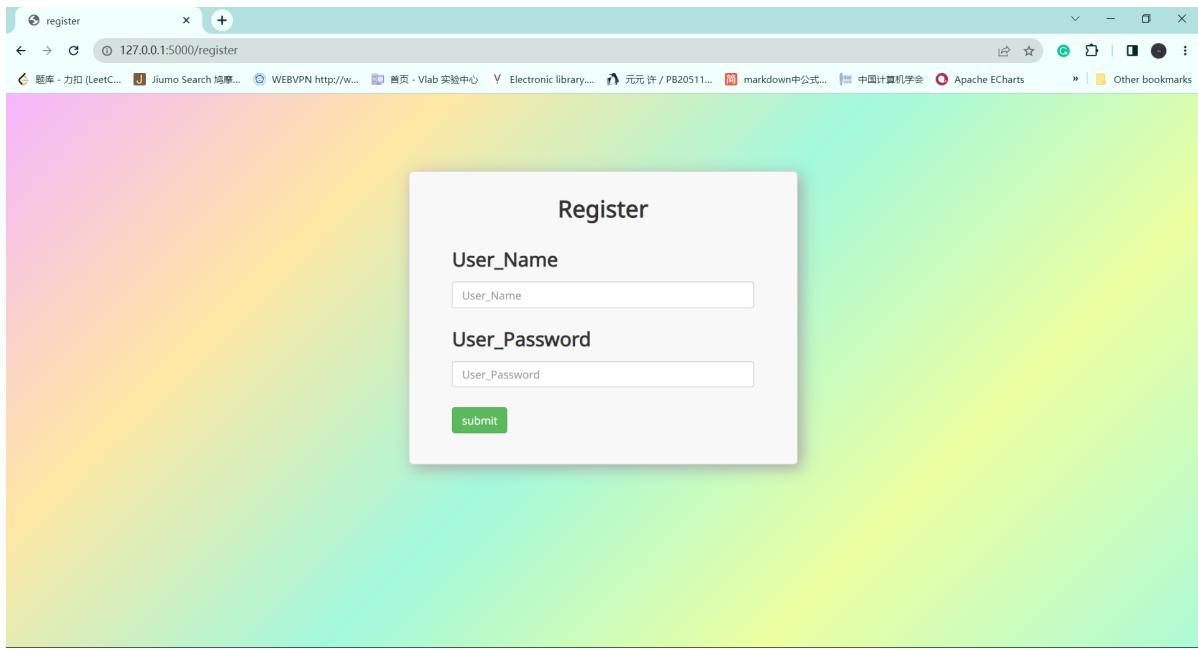
    cursor.execute(sql, (username))
    data = cursor.fetchall()
    if data:
        return render_template('register.html', error="The username already
exists.")

    cursor.execute(sql1, (username, password))
    conn.commit()

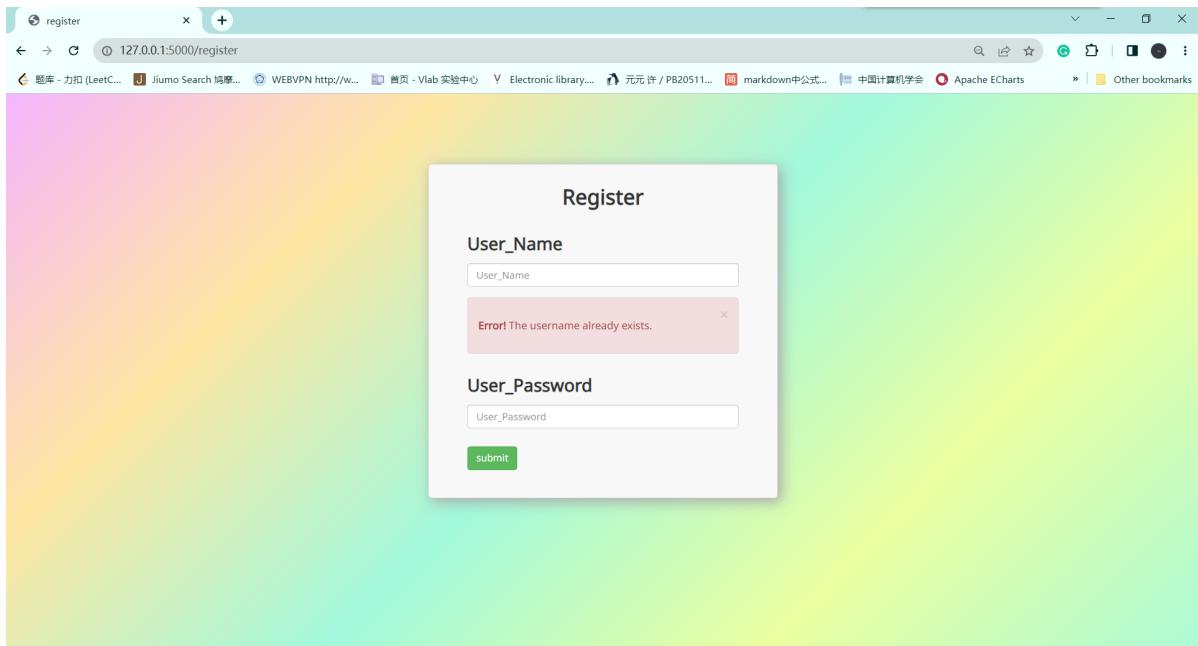
    cursor.close()
    conn.close()

    return redirect('/')
```

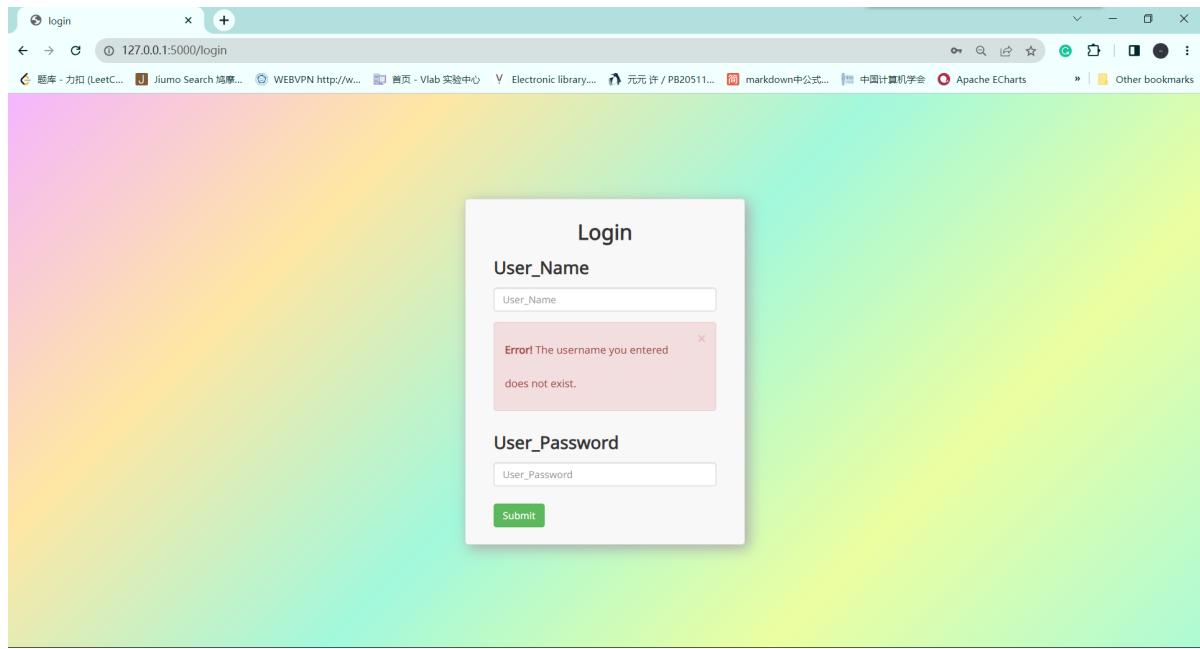
Register 模块示意图如下



添加了一些基础报错信息，比如注册用户名已存在，或者密码为空等，运行示例如下：



**Login** 模块示意图如下



也添加了一些基础的报错信息，比如账户不存在等，就如上面的报错信息。

```
@app.route('/login', methods=["GET", "POST"])
def login():
    if request.method == "GET":
        return render_template('login.html')

    username = request.form.get("User_Name")
    password = request.form.get("User_Password")

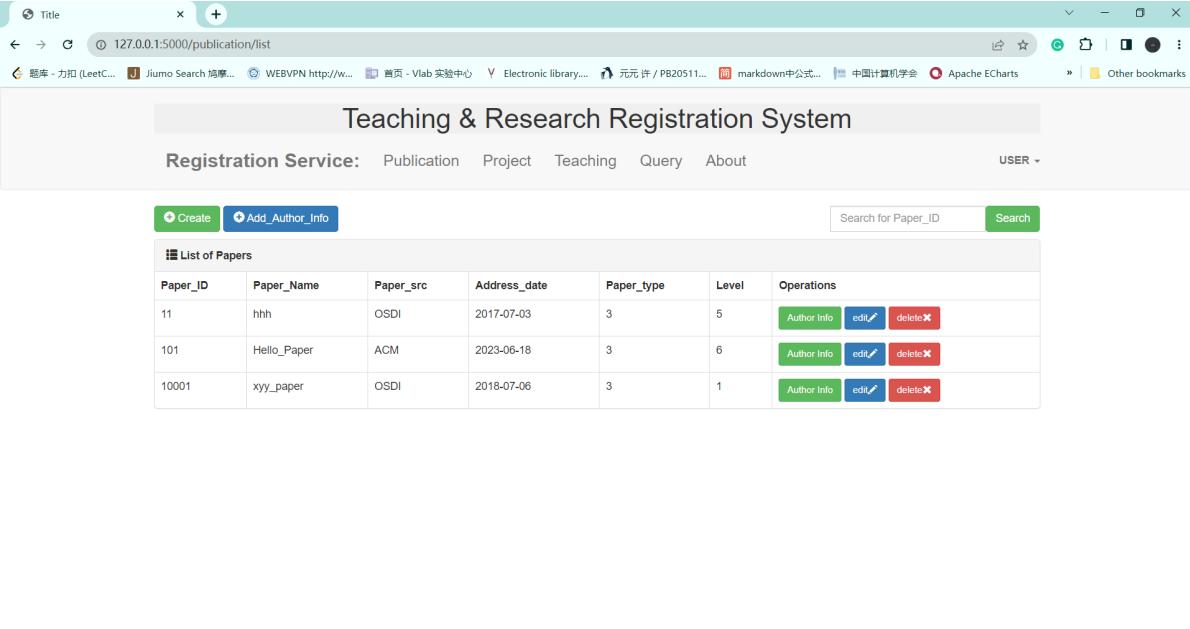
    conn = pymysql.connect(host="127.0.0.1", port=3306, user='root',
                           passwd='123456', db='tmsys', charset='utf8')
    cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)

    sql = "select * from User where User_Name = %s"

    cursor.execute(sql, (username))
    data = cursor.fetchall()
    cursor.close()
    conn.close()
    if not data:
        error = "The username you entered does not exist."
        return render_template('login.html', user_not_exist=error)

    if data[0]['User_Password'] == password:
        return redirect('/layout')
    else:
        error = "wrong password."
        return render_template('login.html', wrong_pswd=error)
```

## 4.2 论文和项目的增删模块



The screenshot shows a web application titled "Teaching & Research Registration System". The main menu includes "Registration Service" with options for Publication, Project, Teaching, Query, and About. A user dropdown is also present. Below the menu, there are two buttons: "Create" and "Add\_Author\_Info". A search bar with placeholder "Search for Paper\_ID" and a "Search" button are located on the right. The main content area displays a table titled "List of Papers" with columns: Paper\_ID, Paper\_Name, Paper\_src, Address\_date, Paper\_type, Level, and Operations. The table contains three rows of data:

Paper_ID	Paper_Name	Paper_src	Address_date	Paper_type	Level	Operations
11	hh	OSDI	2017-07-03	3	5	<button>Author Info</button> <button>edit</button> <button>delete</button>
101	Hello_Paper	ACM	2023-06-18	3	6	<button>Author Info</button> <button>edit</button> <button>delete</button>
10001	xyy_paper	OSDI	2018-07-06	3	1	<button>Author Info</button> <button>edit</button> <button>delete</button>

进行若干的操作：

- **Create** 跳转到创建页面进行相关信息添加；
  - **python 代码：**

```
@app.route("/publication/registration", methods=["GET", "POST"])
def publication_reg():
    if request.method == "GET":
        return render_template('publication_reg.html')

    Paper_ID = request.form.get("Paper_ID")
    Paper_name = request.form.get("Paper_name")
    Paper_src = request.form.get("Paper_src")
    address_date = request.form.get("address_date")
    Paper_type = request.form.get("Paper_type")
    level = request.form.get("level")
    Paper_name = Paper_name.replace("'", '')
    Paper_name = Paper_name.replace('"', '')

    Teacher_ID = request.form.get("Teacher_ID")
    Pub_Rank = request.form.get("Pub_Rank")
    coauth = request.form.get("coauth")

    if not Paper_ID:
        return render_template('publication_reg.html', error="Paper_ID cannot be null.")
    if not Paper_name:
        return render_template('publication_reg.html', error1="Paper_name cannot be null.")

    print(int(Paper_type))
    if int(Paper_type) < 1 or int(Paper_type) > 4:
        return render_template('publication_reg.html', error2="Please input legal Paper_type.")
```

```

        if int(level) < 1 or int(level) > 6:
            return render_template('publication_reg.html', error3="Please
input legal Paper_level.")

        if not Teacher_ID or not Pub_Rank or not coauth:
            return render_template('publication_reg.html', error="Please add
full information of at least one author.")

        if not (coauth == "0" or coauth == "1"):
            return render_template('publication_reg.html', error4="Please
input legal Is_Corresponding_Author.")

        conn = pymysql.connect(host="127.0.0.1", port=3306, user='root',
passwd='123456', db='tmsys', charset='utf8')
        cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)

        sql = "select * from Paper where Paper_ID = %s"
        cursor.execute(sql, (Paper_ID))
        data = cursor.fetchall()
        if data:
            return render_template('publication_reg.html', error="The
Paper_ID you entered already exist.")

        sql = "insert into Paper(Paper_ID, Paper_name, Paper_src,
address_date, Paper_type, level) values(%s, %s, %s, %s, %s, %s)"
        cursor.execute(sql, (Paper_ID, Paper_name, Paper_src, address_date,
Paper_type, level))
        conn.commit()

        sql = "insert into Publish(Teacher_ID, Paper_ID, Pub_Rank, coauth)
values(%s, %s, %s, %s)"
        cursor.execute(sql, (Teacher_ID, Paper_ID, Pub_Rank, coauth))
        conn.commit()

        cursor.close()
        conn.close()

    return redirect('/publication/registration')

```

- 呈现效果：

The screenshot shows a web browser window with the following details:

- Title Bar:** Shows the URL `127.0.0.1:5000/publication/registration`.
- Header:** The page title is "Teaching & Research Registration System". Below it, a navigation bar includes links for "Publication", "Project", "Teaching", "Query", and "About".
- Content Area:** A form titled "Publication Registration" with the following fields and their values:
  - Paper\_ID: 1001
  - Paper\_name: "Hello\_Paper"
  - Paper\_src: SOSP
  - Address\_date: 2017/7/3
  - Paper\_type (1 - 4): 3
  - Paper\_level (1 - 6): 1
  - Teacher\_ID: 10000
  - Author\_Rank: 1
  - Is\_Corresponding\_Author (0 or 1): 1
- Buttons:** A green "submit" button is located at the bottom right of the form.

- 插入结果：

Paper_ID	Paper_Name	Paper_src	Address_date	Paper_type	Level	Operations
11	hh	OSDI	2017-07-03	3	5	<button>Author Info</button> <button>edit</button> <button>delete</button>
101	Hello_Paper	ACM	2023-06-18	3	6	<button>Author Info</button> <button>edit</button> <button>delete</button>
1001	Hello_Paper	SOSP	2017-07-03	3	1	<button>Author Info</button> <button>edit</button> <button>delete</button>
10001	xyy_paper	OSDI	2018-07-06	3	1	<button>Author Info</button> <button>edit</button> <button>delete</button>

可见，我们允许同名，但不允许同 **Paper\_ID** 论文的存在，并且添加了操作处理论文名称可能带的单双引号。

- 同样根据对数据的约束添加了一些**报错信息**，比如效果：

The screenshot shows a form for publication registration. The 'Paper\_ID' field contains an error message: "Error! The Paper\_ID you entered already exist." The form includes fields for Paper\_name, Paper\_src, Address\_date, Paper\_type (1-4), Paper\_level (1-6), Teacher\_ID, Author\_Rank, and Is\_Corresponding\_Author.

被错信息都会显示在出错行的下方，起到提醒作用。

- **Delete** 即从数据库删除论文，并刷新展示的列表；

- **python** 代码：

```

@app.route("/publication/delete/<string:PID>")
def publication_delete(PID):
    conn = pymysql.connect(host="127.0.0.1", port=3306, user='root',
                           passwd='123456', db='tmsys', charset='utf8')
    cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)

    sql = "delete from Publish where Paper_ID = %s"
    cursor.execute(sql, (PID))
    conn.commit()

    sql = "delete from Paper where Paper_ID = " + "''" + PID + "''"
    # sql_find = "select * from Publish where Paper_ID = %s"

    # cursor.execute(sql_find, (PID))

```

```

# data = cursor.fetchall()

# if data:
#     error = "The Paper " + str(PID) + " has a publish record and
# is not allowed to be deleted."
#     data_list = publication(Flag=True)
#     return render_template('publication_list.html',
# data_list=data_list, error=error)

cursor.execute(sql)
conn.commit()

cursor.close()
conn.close()

return redirect('/publication/list')

```

- **呈现效果:**

比如在上述的结果中，删除掉刚插入的内容：

Paper_ID	Paper_Name	Paper_src	Address_date	Paper_type	Level	Operations
11	hh	OSDI	2017-07-03	3	5	<a href="#">Author Info</a> <a href="#">edit</a> <a href="#">delete</a>
101	Hello_Paper	ACM	2023-06-18	3	6	<a href="#">Author Info</a> <a href="#">edit</a> <a href="#">delete</a>
10001	xyy_paper	OSDI	2018-07-06	3	1	<a href="#">Author Info</a> <a href="#">edit</a> <a href="#">delete</a>

- **Edit** 即跳转到编辑页面修改论文的基本信息；

- **python 代码:**

```

@app.route("/publication/edit/<string:PID>", methods=["GET", "POST"])
def publication_edit(PID):
    if request.method == "GET":
        return render_template('publication_edit.html')

    Paper_ID = PID
    Paper_name = request.form.get("Paper_name")
    Paper_src = request.form.get("Paper_src")
    address_date = request.form.get("address_date")
    Paper_type = request.form.get("Paper_type")
    level = request.form.get("level")

    conn = pymysql.connect(host="127.0.0.1", port=3306, user='root',
    passwd='123456', db='tmsys', charset='utf8')
    cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)

    sql = "select * from Paper where Paper_ID = %s"

```

```

        cursor.execute(sql, (Paper_ID))
        data = cursor.fetchall()

        if not Paper_name:
            Paper_name = data[0]['Paper_name']

        if not Paper_src:
            Paper_src = data[0]['Paper_src']

        if not address_date:
            address_date = data[0]['address_date']

        if not Paper_type:
            Paper_type = data[0]['Paper_type']

        if not level:
            level = data[0]['level']

        sql = "update Paper set Paper_name = %s, Paper_src = %s,
address_date = %s, Paper_type = %s, level = %s where Paper_ID = %s"

        cursor.execute(sql, (Paper_name, Paper_src, address_date,
Paper_type, level, Paper_ID))
        conn.commit()

        cursor.close()
        conn.close()

    return redirect('/publication/list')

```

- 呈现效果：

The screenshot shows a web browser window with the URL `127.0.0.1:5000/publication/edit/11`. The page is titled "Teaching & Research Registration System" and has a sub-header "Registration Service: Publication". Below this is a form titled "Update Publication Information" with five input fields: "New\_Paper\_name", "New\_Paper\_src", "New\_Address\_date", "New\_Paper\_type", and "New\_Paper\_level". At the bottom of the form is a green "submit" button.

---

输入新的信息即可进行论文信息的编辑，并保留所有的作者信息。

- Search 会按照 Paper\_ID 搜索并刷新展示的列表；

- Python 代码：

```

@app.route("/publication/list", methods=["GET", "POST"])
def publication(flag=False):
    conn = pymysql.connect(host="127.0.0.1", port=3306, user='root',
passwd='123456', db='tmsys', charset='utf8')

```

```

        cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)
        if request.method == "GET": # 主动 Search 调用的分支
            sql = "select * from Paper"
            cursor.execute(sql)
            data_list = cursor.fetchall()

            cursor.close()
            conn.close()
            if Flag == True:
                return data_list
            return render_template('publication_list.html',
data_list=data_list)

        Paper_ID = request.form.get("Paper_ID")
        print(Paper_ID)
        sql = "select * from Paper where Paper_ID = %s"
        print(sql)

        cursor.execute(sql, (Paper_ID))
        data_list = cursor.fetchall()
        print(data_list)

        cursor.close()
        conn.close()

    return render_template('publication_list.html', data_list=data_list)

```

- 呈现效果：

比如 **Search(11)** 的结果：

Paper_ID	Paper_Name	Paper_src	Address_date	Paper_type	Level	Operations
11	hth	OSDI	2017-07-03	3	5	<a href="#">Author Info</a> <a href="#">edit</a> <a href="#">delete</a>

---

这里还是主要以**论文**为例，对**项目的**处理是类似的。

有一点不同是**添加操作**：

- 论文添加时至少有一名作者，这个我们通过创建时加上一个人的信息实现；

- 但是项目要求总经费等于每个人承担的部分之和，也就要求所有的人承担信息需要随着项目一同加到我们的信息中
  - Python 代码：**  
不多赘述了，就是添加完项目信息直接跳转到添加承担项目信息的部分。
  - 实现效果：**  
与论文类似。

## 4.3 发表论文 & 承担项目 & 教授课程模块

这个部分承接上一个论文的列表，点击每个条目后的 **Author\_Info** 就会展开看到每个论文的详细信息：

Paper_ID	Paper_name	Author_ID	Author_Name	Author_Rank	Is_Corresponding_Author	Operations
11	hhh	10002	吴亚颖	1	0	<a href="#">edit</a> <a href="#">delete</a>
11	hhh	10000	李诚	2	1	<a href="#">edit</a> <a href="#">delete</a>

于是我们进入了一个新的页面，可以对作者的信息进行增加，修改或者查找。

**具体的实现与效果：**

- Add\_Author\_Info：** 跳转到新的页面，添加作者信息（并进行约束检测）；
  - Python 代码：**

```

@app.route("/publication/author", methods=["GET", "POST"])
def add_publication_info():
    if request.method == "GET":
        return render_template('publication_info_add.html')

    Teacher_ID = request.form.get("Teacher_ID")
    Paper_ID = request.form.get("Paper_ID")
    Pub_Rank = request.form.get("Pub_Rank")
    coauth = request.form.get("coauth")

    conn = pymysql.connect(host="127.0.0.1", port=3306, user='root',
                           passwd='123456', db='tmsys', charset='utf8')
    cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)

    sql_1 = "select * from Teacher where Teacher_ID = %s"
    cursor.execute(sql_1, (Teacher_ID))
    data_1 = cursor.fetchall()

    sql_2 = "select * from Paper where Paper_ID = %s"
    cursor.execute(sql_2, (Paper_ID))
    data_2 = cursor.fetchall()

    sql_3 = "select * from Publish where Paper_ID = %s and Pub_Rank = %s"
    cursor.execute(sql_3, (Paper_ID, Pub_Rank))
    data_3 = cursor.fetchall()

    sql_4 = "select * from Publish where Paper_ID = %s and coauth = 1"
    cursor.execute(sql_4, (Paper_ID))
    data_4 = cursor.fetchall()

    sql_5 = "select * from Publish where Paper_ID = %s and Teacher_ID = %s"
    cursor.execute(sql_5, (Paper_ID, Teacher_ID))
    data_5 = cursor.fetchall()

    if not Teacher_ID:
        return render_template('publication_info_add.html', error="The Teacher_ID can not be null.")
    elif not Paper_ID:
        return render_template('publication_info_add.html', error="The Paper_ID can not be null.")
    elif not Pub_Rank:
        return render_template('publication_info_add.html', error="The Publication_Rank can not be null.")
    elif not coauth:
        return render_template('publication_info_add.html', error="The Is_Corresponding_Author can not be null.")

    elif not data_1:
        return render_template('publication_info_add.html', error="The Teacher_ID you entered must exist.")
    elif not data_2:
        return render_template('publication_info_add.html', error="The Paper_ID you entered must exist.")

```

```

    elif data_5:
        return render_template('publication_info_add.html',
error="Duplicate Author information exists.")

    elif data_3:
        return render_template('publication_info_add.html',
error="Duplicate rank information exists.")

    elif data_4 and coauth == "1":
        return render_template('publication_info_add.html',
error="Corresponding Author already exists.")
    else:
        sql = "insert into Publish(Teacher_ID , Paper_ID, Pub_Rank,
coauth) values(%s, %s, %s, %s)"
        cursor.execute(sql, (Teacher_ID, Paper_ID, Pub_Rank, coauth))
        conn.commit()

    cursor.close()
    conn.close()

    return redirect('/publication/author')

```

- 实现效果：

The screenshot shows a web application interface for publication registration. At the top, there's a navigation bar with links like 'Publication', 'Project', 'Teaching', 'Query', and 'About'. On the right, there's a 'USER' dropdown. The main content area is titled 'Publication Registration'. It features four input fields: 'Teacher\_ID', 'Paper\_ID', 'Publication\_Rank', and 'Is\_Corresponding\_Author', each with its respective label above it. Below these fields is a green 'submit' button.

- 
- 一些报错信息其实在代码中就能看到，也就是在对应的条目下面弹出相应的红框，就不再贴图了，看一下 **html** 的实现好了：

```

<div id="error"></div>
<script>
    const error_html = "<div class=\"alert alert-danger alert-dismissible\" role=\"alert\">\n" +
        "                <button type=\"button\" class=\"close\" data-dismiss=\"alert\" aria-label=\"Close\"><span>\n" +
        "                    aria-hidden=\"true\">&times;</span>\n" +
        "                </button>\n" +
        "                <strong>Error!</strong> {{error}}\n" +
        "            </div>";
    const error = '{{error}}';
    if (error)
        document.getElementById('error').innerHTML = error_html;
</script>

```

这就是将报错信息转换为前端的 **html** 代码，效果和先前的内容都差不多。

- **edit**: 跳转到新的页面，编辑已有的作者信息（并进行约束检测）；
- **Python** 代码：

```

@app.route("/author/edit/<string:PID>", methods=["GET", "POST"])
def author_info_edit(PID):
    if request.method == "GET":
        return render_template('author_edit.html')

    Paper_ID = PID
    Teacher_ID = request.form.get("Teacher_ID")
    Pub_Rank = request.form.get("Pub_Rank")
    coauth = request.form.get("coauth")

    conn = pymysql.connect(host="127.0.0.1", port=3306, user='root',
                           passwd='123456', db='tmsys', charset='utf8')
    cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)

    sql = "select * from Publish where Paper_ID = %s"
    cursor.execute(sql, (Paper_ID))
    data = cursor.fetchall()

    if not Teacher_ID:
        Teacher_ID = data[0]['Teacher_ID']

    if not Pub_Rank:
        Pub_Rank = data[0]['Pub_Rank']

    if not coauth:
        coauth = data[0]['coauth']

    sql = "update Publish set Paper_ID = %s, Teacher_ID = %s, Pub_Rank = %s, coauth = %s"

    cursor.execute(sql, (Paper_ID, Teacher_ID, Pub_Rank, coauth))
    conn.commit()

    cursor.close()
    conn.close()

```

```
return redirect('/publication/list')
```

- 实现效果：

The screenshot shows a web application interface for updating publication information. The title bar says 'Teaching & Research Registration System'. Below it, a navigation menu includes 'Publication', 'Project', 'Teaching', 'Query', and 'About'. On the left, there's a sidebar with 'Update Publication Information' and several input fields labeled 'New\_Paper\_name', 'New\_Paper\_src', 'New\_Address\_date', 'New\_Paper\_type', and 'New\_Paper\_level'. At the bottom is a green 'submit' button.

输入新的论文信息，即可替换原先的论文内容，并且会保留论文的 **ID** 和先前所有的**已有发表**关系。

- **delete**: 跳转到新的页面，删除已有的作者信息（并进行约束检测）；

- **Python** 代码：

```
@app.route("/author/delete<string:content>")
def author_delete(content):
    conn = pymysql.connect(host="127.0.0.1", port=3306, user='root',
                           passwd='123456', db='tmsys', charset='utf8')
    cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)

    str1 = content.replace("''", "").replace(" ", "")
    str2 = str1.replace("(", "").replace(")", "")
    listi = str2.split(",")
    sql = "delete from Publish where Paper_ID = " + "''" + listi[0] + "''"
    + " and Teacher_ID = " + "''" + listi[1] + "''"
    print(sql)

    # 不能没有作者！
    # sql_find = "select * from Publish where Paper_ID = " + "''" +
    listi[0] + "'"
    sql_find = "select Publish.Paper_ID, Paper.Paper_name,
Teacher.Teacher_ID, Teacher.Teacher_name, Pub_Rank, coauth \
from Teacher, Publish, Paper \
where Publish.Paper_ID = %s \
and Publish.Paper_ID = Paper.Paper_ID \
and Teacher.Teacher_ID = Publish.Teacher_ID \
order by Pub_Rank"
    cursor.execute(sql_find, (listi[0]))
    data_list = cursor.fetchall()
    print(data_list)
    print(len(data_list))
    if len(data_list) == 1:
        return render_template('publication_author_list.html',
                               data_list=data_list, error="The Paper must have at least one author.")
```

```

print("seriously?")
cursor.execute(sql)
conn.commit()

cursor.close()
conn.close()

return redirect('/publication/list')

```

- 实现效果：

**删除前：**

Paper_ID	Paper_name	Author_ID	Author_Name	Author_Rank	Is_Corresponding_Author	Operations
10001	xyy_paper	10001	刘琪	1	1	<a href="#">edit</a> <a href="#">delete</a>
10001	xyy_paper	10002	吴亚颖	2	0	<a href="#">edit</a> <a href="#">delete</a>

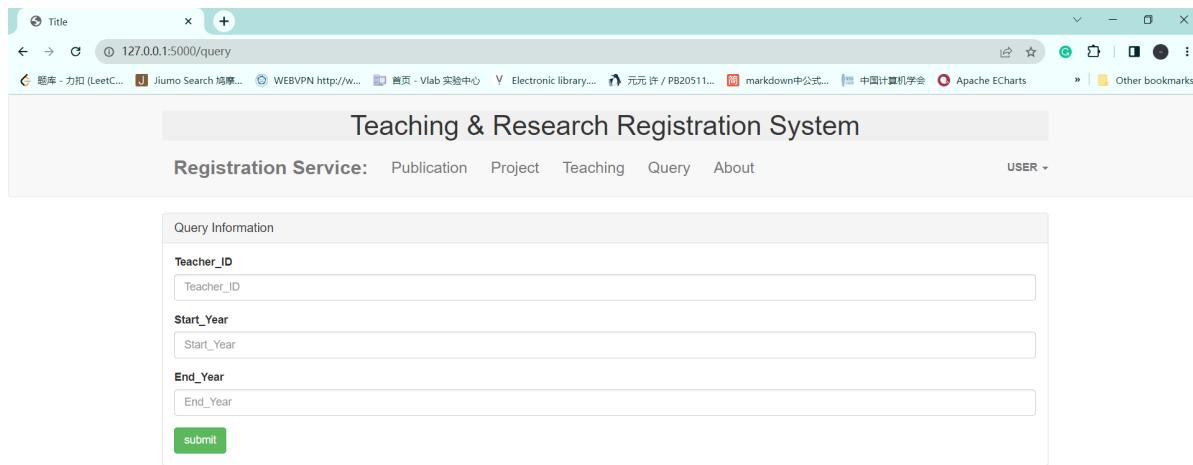
**删除后：**

Paper_ID	Paper_name	Author_ID	Author_Name	Author_Rank	Is_Corresponding_Author	Operations
10001	xyy_paper	10002	吴亚颖	2	0	<a href="#">edit</a> <a href="#">delete</a>

还有很多的报错信息，基本不符合要求的非法输入或者空我都留了相关的报错信息，报错的格式与前几个模块所展示的都是类似的，因此不再重复地说明和展示了。

这里还是主要以**发表论文**为例，对**负责项目**，**教授课程**的处理是类似的，不一一赘述了。

## 4.4 查询统计模块



- 需要输入：

- Teacher\_ID：教师工号；
- Start\_Year：查询起始年份；
- End\_Year：查询终止年份

- Python 代码

这里涉及到了需要将相应的内容从数值转化为对应的内容，所以代码显得有一些冗长

转换依据：

1. 性别为整数，1-男，2-女
2. 教师职称为整数：1-博士后，2-助教，3-讲师，4-副教授，5-特任教授，6-教授，7-助理研究员，8-特任副研究员，9-副研究员，10-特任研究员，11-研究员。
3. 论文类型为整数：1-full paper, 2-short paper, 3-poster paper, 4-demo paper。
4. 论文级别为整数：1-CCF-A, 2-CCF-B, 3-CCF-C, 4-中文CCF-A, 5-中文CCF-B, 6-无级别。
5. 项目类型为整数：1-国家级项目，2-省部级项目，3-市厅级项目，4-企业合作项目，5-其它类型项目。
6. 发表论文和承担项目中的排名：1-表示排名第一，以此类推。论文排名第一即为第一作者，承担项目排名第一即为项目负责人。
7. 主讲课程中的学期取值为：1-春季学期，2-夏季学期，3-秋季学期。
8. 课程性质为整数：1-本科生课程，2-研究生课程

代码如下：

```
@app.route("/query", methods=["GET", "POST"])
def query():
    if request.method == "GET":
        return render_template('query_info.html')

    conn = pymysql.connect(host="127.0.0.1", port=3306, user='root',
                           passwd='123456', db='tmsys', charset='utf8')
```

```
cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)

Teacher_ID = request.form.get("Teacher_ID")
Start_Year = request.form.get("Start_Year")
End_Year = request.form.get("End_Year")

sql_1 = "select * from Teacher where Teacher_ID = %s"
cursor.execute(sql_1, (Teacher_ID))
data_1 = cursor.fetchall()

if not Teacher_ID:
    return render_template('query_info.html', error = "The Teacher_ID can not be NULL!")

if not data_1:
    return render_template('query_info.html', error="The Teacher_ID you entered must exist.")

if not Start_Year:
    return render_template('query_info.html', error = "The Start_Year can not be NULL!")

if not End_Year:
    return render_template('query_info.html', error = "The End_Year can not be NULL!")

if int(Start_Year) > int(End_Year):
    return render_template('query_info.html', error1="Start Year can not be larger than end year!")

f = open("./output/" + Teacher_ID + ".md", "w")
f.write("# 教师教学科研工作统计 (" + Start_Year + "-" + End_Year + ")\n\n")

sql = "select Teacher_ID, Teacher_name, sex, title\
from Teacher\
where Teacher_ID = %s"

cursor.execute(sql, (Teacher_ID))
data_list = cursor.fetchall()
temp = data_list[0]['sex']
if temp == 1:
    data_list[0]['sex'] = '男'
else:
    data_list[0]['sex'] = '女'

temp = data_list[0]['title']
if temp == 1:
    data_list[0]['title'] = '博士后'
elif temp == 2:
    data_list[0]['title'] = '助教'
elif temp == 3:
    data_list[0]['title'] = '讲师'
elif temp == 4:
    data_list[0]['title'] = '副教授'
elif temp == 5:
    data_list[0]['title'] = '特任教授'
```

```

        elif temp == 6:
            data_list[0]['title'] = '教授'
        elif temp == 7:
            data_list[0]['title'] = '助理研究员'
        elif temp == 8:
            data_list[0]['title'] = '特任副研究员'
        elif temp == 9:
            data_list[0]['title'] = '副研究员'
        elif temp == 10:
            data_list[0]['title'] = '特任研究员'
        elif temp == 11:
            data_list[0]['title'] = '研究员'
    # print(data_list)

    teacher_info = "## 教师基本信息\n" +
                    "| 教师编号 | 教师姓名 | 教师性别 | 教师职称 |\n" +
                    "| :-----: | :-----: | :-----: | :-----: |\n" +
                    "| " + data_list[0]['Teacher_ID'] + " | " + data_list[0]
['Teacher_name'] + " | " +
                    + data_list[0]['sex'] + " | " + data_list[0]['title'] + "
|\n";
f.write(teacher_info)

sql = "select Teach.Course_ID, Course.Course_name, teach_hour, year,
semester \
       from Teach, Course \
      where Teach.Teacher_ID = %s \
        and Teach.Course_ID = Course.Course_ID \
      order by year, semester"
cursor.execute(sql, (Teacher_ID))
data_list1 = cursor.fetchall()
# print(data_list1)
teach_info = "## 教师授课信息\n" +
                    "| 课程编号 | 课程名称 | 主讲学时 | 开课年份 | 开课学期 |\n" +
                    "| :-----: | :-----: | :-----: | :-----: | :-----: |\n";
f.write(teach_info);

for index, item in enumerate(data_list1):
    temp = data_list1[index]['semester']
    if temp == 1:
        data_list1[index]['semester'] = '春'
    elif temp == 2:
        data_list1[index]['semester'] = '夏'
    elif temp == 3:
        data_list1[index]['semester'] = '秋'
    info = "| " + data_list1[index]['Course_ID'] + " | " +
data_list1[index]['Course_name'] + " | " + str(data_list1[index]
['teach_hour']) + " | " +
                    + str(data_list1[index]['year']) + " | "
+data_list1[index]['semester'] + " | \n"
    f.write(info)

sql = "select Paper_name, Paper_src, address_date, level, Pub_Rank,
coauth \
       from Paper, Publish \
      where Publish.Teacher_ID = %s \

```

```

        and Publish.Paper_ID = Paper.Paper_ID"
cursor.execute(sql, (Teacher_ID))
data_list2 = cursor.fetchall()
# print(data_list2)
publish_Info = "## 教师发表论文信息 \n" + \
    "| 论文名称 | 论文来源 | 发表日期 | 论文级别 | 作者排名 | 是否为通讯作者 |\n" +
    "| :-----: | :-----: | :-----: | :-----: | :-----: | :-----: |\n" +
    "----: |\n"
f.write(publish_Info)

for index, item in enumerate(data_list2):
    temp = data_list2[index]['level']
    if temp == 1:
        data_list2[index]['level'] = 'CCF-A'
    elif temp == 2:
        data_list2[index]['level'] = 'CCF-B'
    elif temp == 3:
        data_list2[index]['level'] = 'CCF-C'
    elif temp == 4:
        data_list2[index]['level'] = '中文CCF-A'
    elif temp == 5:
        data_list2[index]['level'] = '中文CCF-B'
    elif temp == 6:
        data_list2[index]['level'] = '无级别'

    temp = data_list2[index]['Pub_Rank']
    data_list2[index]['Pub_Rank'] = '排名第' + str(temp)

    temp = data_list2[index]['coauth']
    if temp == 1:
        data_list2[index]['coauth'] = '是'
    else:
        data_list2[index]['coauth'] = '否';

    # Paper_name, Paper_src, address_date, level, Pub_Rank, coauth
    info = "| " + data_list2[index]['Paper_name'] + " | " +
data_list2[index]['Paper_src'] + " | " + str(data_list2[index]
['address_date']) + " | " +
            + str(data_list2[index]['level']) + " | " +
data_list2[index]['Pub_Rank'] + " | " + data_list2[index]['coauth'] + " | \n"
    f.write(info)

sql = "select Project_name, Project_src, Project_type, start_year,
end_year, Project.funding, Incharge.funding as Ifunding \
        from Project, Incharge \
        where Incharge.Teacher_ID = %s \
            and Incharge.Project_ID = Project.Project_ID"
cursor.execute(sql, (Teacher_ID))
data_list3 = cursor.fetchall()
# print(data_list3)
Incharge_Info = "## 教师承担项目信息 \n" + \
    "| 项目名称 | 项目来源 | 项目级别 | 开始时间 | 结束时间 | 总经费 |\n" +
    "| :-----: | :-----: | :-----: | :-----: | :-----: | :-----: |\n" +
    "----: | :-----: | \n"

```

```

f.write(Incharge_Info)

for index, item in enumerate(data_list3):
    temp = data_list3[index]['Project_type']
    if temp == 1:
        data_list3[index]['Project_type'] = '国家级项目'
    elif temp == 2:
        data_list3[index]['Project_type'] = '省部级项目'
    elif temp == 3:
        data_list3[index]['Project_type'] = '市厅级项目'
    elif temp == 4:
        data_list3[index]['Project_type'] = '企业合作项目'
    elif temp == 5:
        data_list3[index]['Project_type'] = '其他项目类型'

# Project_name, Project_src, Project_type, start_year, end_year,
Project.funding, Incharge.funding as Ifunding \


info = "| " + data_list3[index]['Project_name'] + " | " +
data_list3[index]['Project_src'] + " | " + data_list3[index]['Project_type'] +
" | " +
str(data_list3[index]['start_year']) + " | " +
str(data_list3[index]['end_year']) + " | " + str(data_list3[index]
['funding']) + " | " +
str(data_list3[index]['Ifunding']) + " |\n"
f.write(info)

cursor.close()
conn.close()

f.close()

return render_template('query_list.html', data_list=data_list,
data_list1=data_list1, data_list2=data_list2, data_list3=data_list3)

```

- 展示结果：

跳转到展示页面，如下：

The screenshot shows a web application interface with the following data:

Teacher Basic Information				
Teacher_ID	Teacher_Name	Sex	Title	
10002	吴亚颖	女	博士后	

Course Teaching Information				
Course_ID	Course_Name	Course_Hour	Year	Semester
SBZKD01	大数据算法	60	2002	夏
SBZKD00	数据库应用与管理	20	2023	春

Publication Information					
Paper_Name	Paper_src	Address_Date	Level	Pub_Rank	Is_Cor_Author
hh	OSDI	2017-07-03	中文CCF-B	排名第1	否
Hello_Paper	ACM	2023-06-18	无级别	排名第3	是
xyy_paper	OSDI	2018-07-06	CCF-A	排名第2	否

Project Incharge Information						
Project_name	Project_src	Project_type	start_year	end_year	Total_funding	Incharge_funding
XYY_Database	ADSL_Lab	其他项目类型	2023	2025	50000.0	20000.0

并且会输出一个 `.md` 文档到: `./output/<教师工号>.md`, 打开如下:

The screenshot shows a table titled "教师教学科研工作统计 (1950-2030)" in a Typora editor window. The table has four columns: 教师编号 (Teacher ID), 教师姓名 (Teacher Name), 教师性别 (Teacher Gender), and 教师职称 (Teacher Title). One row is visible with data: 10002, 吴亚颖, 女, 博士后.

**教师基本信息**

教师编号	教师姓名	教师性别	教师职称
10002	吴亚颖	女	博士后

**教师授课信息**

课程编号	课程名称	主讲学时	开课年份	开课学期
SBZKD01	大数据算法	60	2002	夏
SBZKD00	数据库应用与管理	20	2023	春

**教师发表论文信息**

论文名称	论文来源	发表日期	论文级别	作者排名	是否为通讯作者
hh	OSDI	2017-07-03	中文CCF-B	排名第1	否
Hello_Paper	ACM	2023-06-18	无级别	排名第3	是
xyy_paper	OSDI	2018-07-06	CCF-A	排名第2	否

**教师承担项目信息**

项目名称	项目来源	项目级别	开始时间	结束时间	总经费	承担经费
XYY_Database	ADSL_Lab	其他项目类型	2023	2025	50000.0	20000.0

## 4.5 About 展示模块

点进去就是个展示模块，没什么输出就。

The screenshot shows a web browser window with a title bar "Title" and address bar "127.0.0.1:5000/about". The page content is a white box with a shadow, containing the text "Teaching & Research Registration System" and "Technology stack". The technology stack list includes: flask web framework, Bootstrap CSS renderer, jquery.js + bootstrap.js, pymysql, and HTML language. At the bottom of the box is the date "2023/6/18 许元元". The background of the browser window has a colorful gradient.

## 实现中的难点问题及解决

感觉没有什么特别困难的点，问就是可能对前端开发的了解有点少，所以导致工作的推进很缓慢，上手难度非常的大；

尽管之前已经做过前端开发的相关工作，还是很容易在各种细节上出问题，并且工作量是真的大，写那么十几个 `html` 虽然写好第一个之后都是各种复制粘贴，但是调整起来真的是非常费劲。

数据库这边的操作还算容易，并且随着 `python API` 的开发升级，感觉开发和协同还是对用户非常友好的，因此这方面很顺利。

# 5 总结与讨论

---

我将这次实验视作一个软件工程类型的大作业，总结下来收获大概有几点吧

- **在实现系统前，要对全局的思路有一个完整的统筹规划，设计的步骤是至关重要的**

也就是说做的过程中，不能仅仅是充分考虑用户需求，满足了就完了；我们必须有自己对全局的设计，合理划分模块和功能，以确保系统的可行性和有效性，不然很可能在实现的过程中遇到奇奇怪怪的思路上的问题；

- **合理设计满足用户多样化的需求**

在实现查询功能时，需要考虑到查询条件的灵活性和数据的可视化展示等方面，对自己开发中问题的定位，和未来提高用户体验和系统的实用性都是非常重要的（本来我还想引用 **e-charts** 的图表 **JS** 源码插入到 **html** 显示做数据可视化的，可惜最后时间和精力真的不足，而且我也没有特别想好这个项目有什么出彩的展示设计，于是就放弃了）；

- **选择合适的前端开发技术**

当然是技术越完善，越高级的前端，可以提高用户体验和系统的易用性，而且必须应该考虑到系统的可扩展性和维护性等方面，比如我做的 **html** 开发，最直观的感受就是“所见即所实现”，给用户提供的每一分友好都是用开发者的头发和时间换来的，如果追求开发的高效性真的不如快速搓一个 **QT** 前端，要注重用户反馈和需求，避免过度复杂或过于简单化的设计。