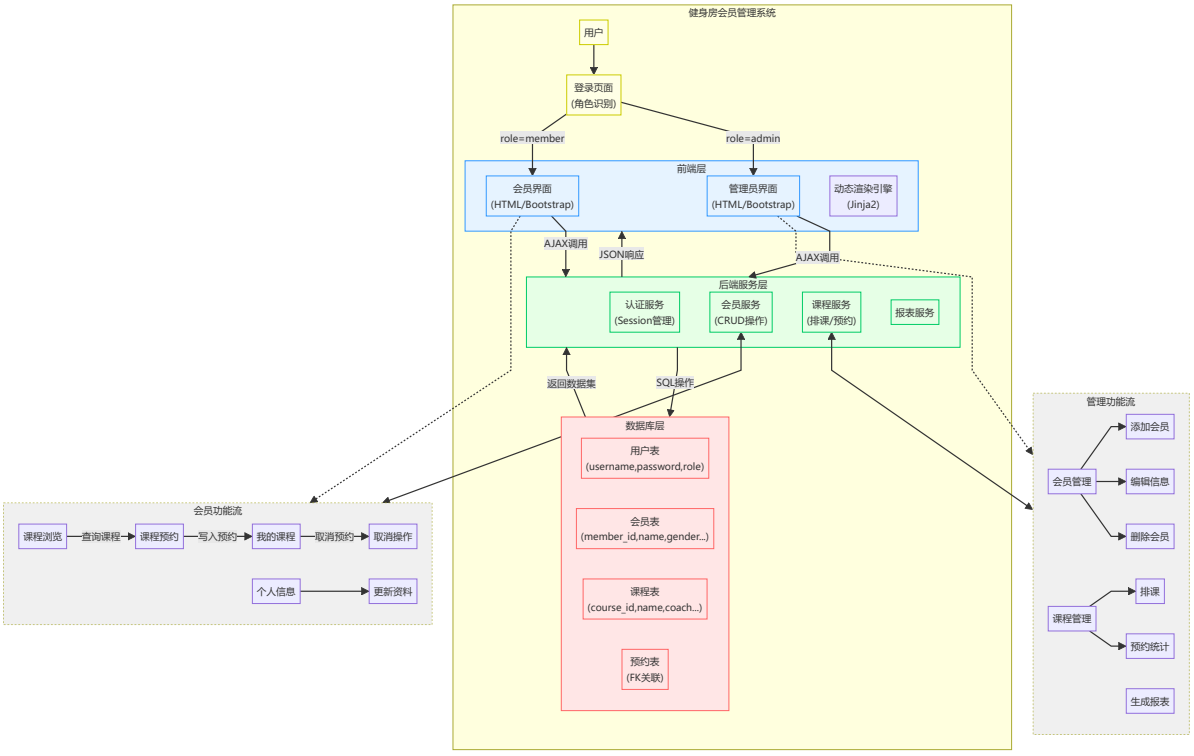
健身房会员系统架构设计文档

一、系统概述

本系统是一个基于Web的健身房会员管理系统，旨在为健身房提供高效的会员管理、课程管理和预约管理服务。系统采用分层架构设计，支持管理员和会员两种角色，提供完整的CRUD操作和业务逻辑处理能力。

二、整体架构设计

2.1 系统架构图



2.2 架构分层说明

健身房会员管理系统的整体架构设计图如上所示。大致分为三个部分：前端，后端，MySQL数据库。

- 前端：使用HTML和可能的CSS来构建和用户进行交互的页面。主要有三个界面：注册登录界面、会员界面、管理员界面。其中会员界面中有会员管理信息界面，管理员界面中有课程管理信息界面。其主要的功能有注册登录、课程信息、会员信息这三种。如架构图所示，其中后两项内部包含增删改查等多种功能，并通过URL向后端传输指令并从后端获取返回来的信息。
- 后端：从前端接受指令后会对其进行解析，并映射到相应的视图函数，通过定义与MySQL数据库相匹配的数据类型来将视图函数的指令转换成 SQL查询，从而与MySQL进行交互。
- MySQL数据库：主要包含用户表、会员表、课程表、预约表。通过查询语句和触发器与后端进行交互，从而进行表格的增删改查。

层级	组件	技术实现	主要职责
部署层	云平台	PythonAnywhere	替代本地开发环境

层级	组件	技术实现	主要职责
表示层	用户界面	HTML5, CSS3, Bootstrap	提供用户交互界面，数据展示
应用层	Web框架	Flask (Python)	请求路由，业务逻辑处理
	服务模块	认证/会员/课程服务	封装核心业务逻辑
数据层	数据库	MySQL 8.0	数据持久化存储

2.3 文件结构

```
/code
├── flask_app.py          # Flask主应用
├── gym.py                # 主代码，核心逻辑
├── mysite.wsgi           # WSGI入口文件
├── /static
|   ├── image.png
├── /templates                # Jinja2模板
|   ├── login.html            # 登陆界面
|   ├── logout.html           # 登出按钮
|   ├── add_member.html       # 会员添加界面
|   ├── add_courses.html      # 课程添加界面
|   ├── courses.html          # 课程管理（管理员）/课程查询预约（会员）界面
|   ├── edit_courses.html     # 课程信息修改界面
|   ├── edit_member.html      # 会员信息修改界面
|   ├── edit_profile.html     # 个人信息修改界面
|   ├── profile.html          # 个人信息界面
|   ├── members.html          # 会员管理界面
|   ├── register.html         # 注册界面
|   └── view_reservations_by_member.html      #会员课程信息界面
└── /database
    └── gym_management.sql    # 数据库文件，初始设置一个管理员
```

三、技术栈说明

3.1 部署平台：PythonAnywhere

PythonAnywhere是一个基于云端的Python开发和托管平台, 提供完整的在线编程环境与Web应用部署服务，并且提供了自带的MySQL数据库，解决了在免费开发模式下端口受限无法连接其他云数据库的问题。

生产环境配置:

```python
# Flask应用关键配置
app.secret_key = os.environ.get('SECRET_KEY')  # 从环境变量获取密钥
app.config['DEBUG'] = False  # 生产环境关闭调试模式
```

3.2 前端技术栈

- **HTML5/CSS3**：

  使用 HTML 构建页面结构，例如定义页面的布局、表格、表单、按钮等元素。

- **Bootstrap 4.5**：

  项目中使用了 Bootstrap 作为前端框架，通过 `<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">` 引入 Bootstrap 的 CSS 文件。Bootstrap 提供了大量的 CSS 样式和组件，帮助快速创建响应式和美观的页面，例如导航栏、表格、按钮等元素的样式和布局。

- **Jinja2模板引擎**：

  Jinja2 是 Flask 内置的模板引擎，它允许在 HTML 中嵌入 Python 代码，使得 HTML 页面可以动态生成。例如： 使用 `{% if session.role == 'admin' %}` 等 Jinja2 模板语法可以根据用户角色显示不同的页面元素。 使用 `{{ course.course_id }}` 等表达式将传递的数据渲染到 HTML 页面中。

- **JavaScript**：基础交互功能

3.3 后端技术栈

- **Python 3.9**：主要开发语言
- **Flask框架**：

  Flask 是一个轻量级的 Python Web 框架，它提供了路由、模板引擎、会话管理等 功能，方便开发者构建 Web 应用程序。 代码中使用 `from flask import Flask, render_template, url_for, request, redirect, session` 导入了 Flask 框架的核心模块：

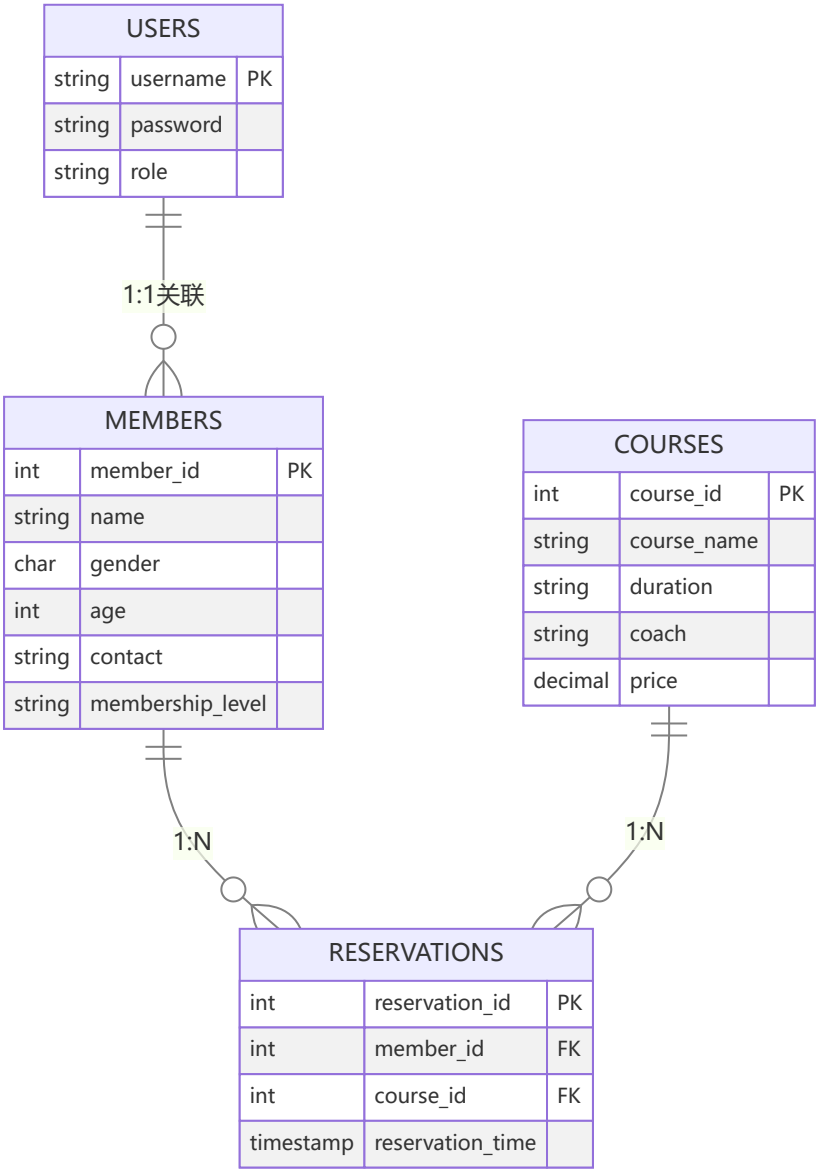
  - Flask 类用于创建 Flask 应用实例，在代码中通过 `app = Flask(__name__)` 创建了一个 Flask 应用。
  - render_template 函数用于渲染 HTML 模板，将动态数据传递给模板并生成 最终的 HTML 页面。
  - url_for 函数用于生成 URL，避免在代码中硬编码 URL，提高代码的可维护 性和灵活性。
  - request 对象用于处理客户端发送的请求，例如获取请求参数、表单数据等。
  - redirect 函数用于重定向用户到其他页面。
  - session 对象用于管理用户会话，例如存储用户登录状态、用户角色等信息。

- **mysql.connector (Python)**：MySQL数据库适配器。连接配置如下：

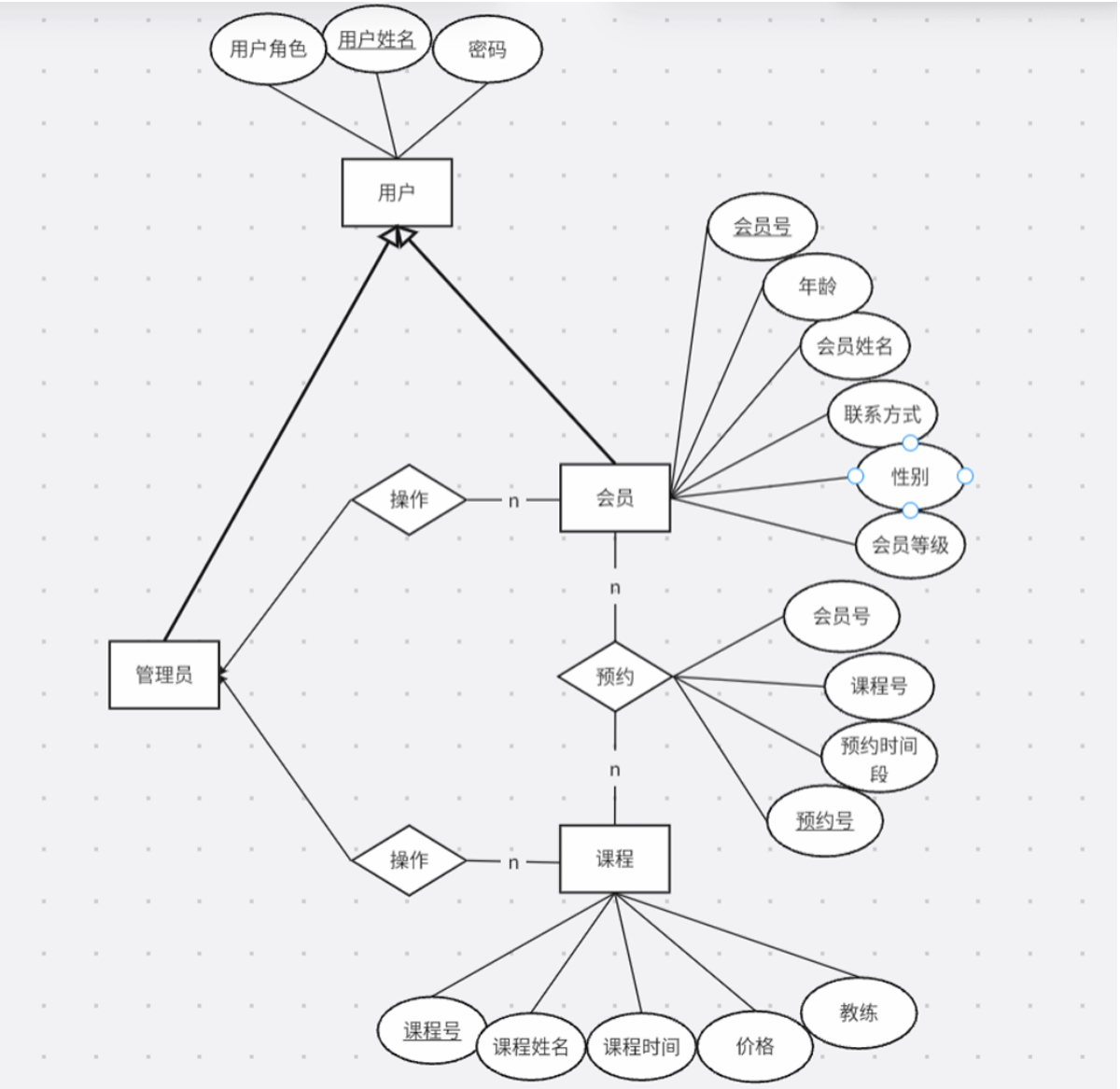
```
DB_CONFIG = {
    "user": "demonnn7",
    "password": "abc14789sysu",
    "host": "demonnn7.mysql.pythonanywhere-services.com",
    "port": "3306",
    "database": "demonnn7$gym_system",
    "raise_on_warnings": True,
    "time_zone": "+08:00" # 东八区时间支持
}
```

3.4 数据库设计

数据库架构图：



ER图如下：



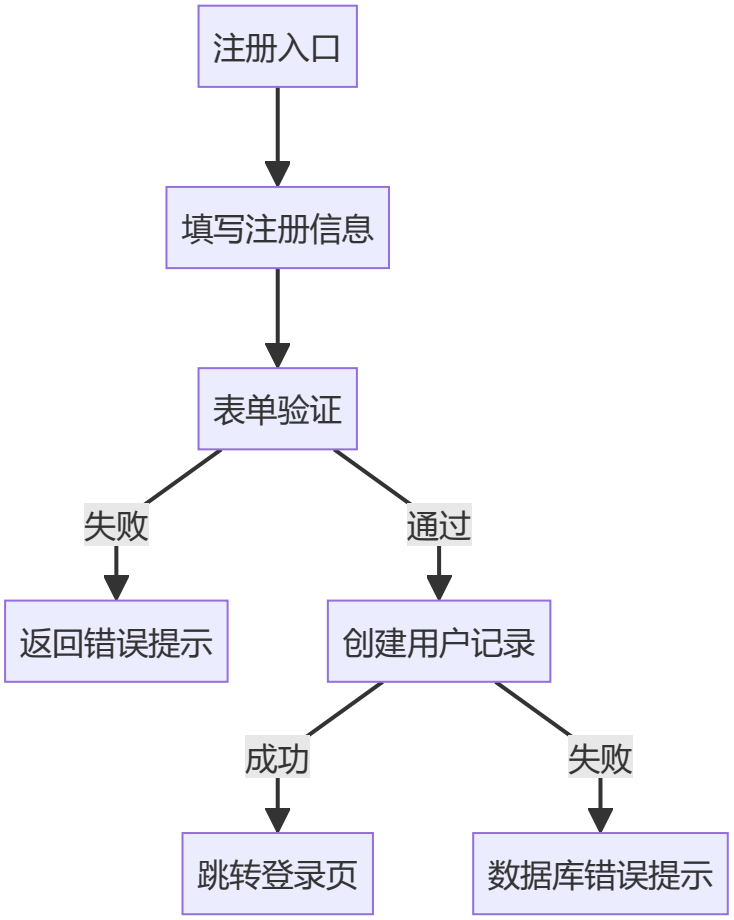
对应的关系模型如下(其中黑体代表主键)：

- 用户（User）（**用户姓名**，密码，用户角色）
- 会员（Member）（**会员号**，年龄，会员姓名，联系方式，性别，会员等级）
- 课程（Course）（**课程号**，课程姓名，课程时间，价格，教练）
- 预约（Reservation）（**预约号**，会员号，课程号，预约时间段）

四、核心功能模块设计

4.1 会员子系统

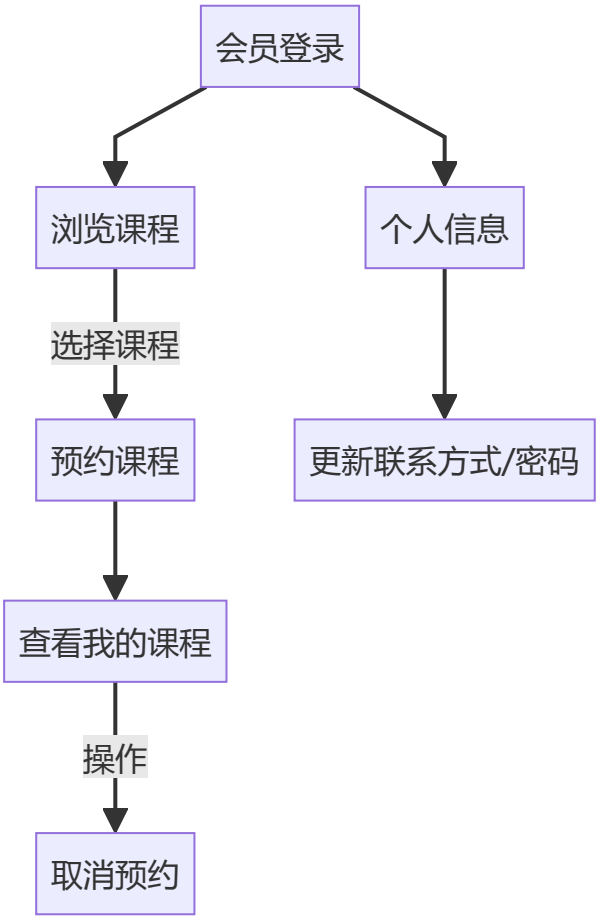
4.1.1 注册功能



关键验证规则：

1. 用户名长度≤8字符
2. 电话号码必须为11位数字
3. 年龄≥12岁
4. 密码与确认密码一致

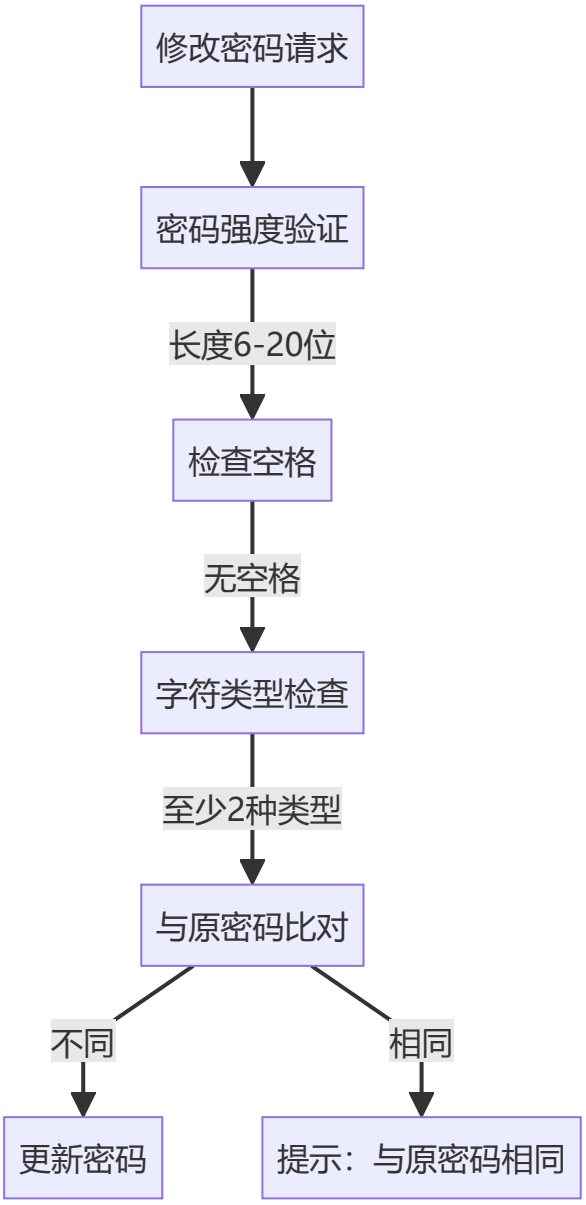
4.1.2 会员登录



会员登陆后主要有两个界面：一是主界面，可以浏览课程并对其进行选择或取消预约，同时能查看“我的课程”；二是可以查看个人信息，并可以对个人信息进行修改。

4.1.3 密码更新验证

```
cursor = conn.cursor()
        try:
            # 检查密码是否与原密码相同
            if new_password:
                # 获取原密码
                cursor.execute('''
                    SELECT password FROM Users
                    WHERE username = (SELECT name FROM Members WHERE member_id =
%s)
                ''', (member_id,))
                old_password = cursor.fetchone()[0]

                if new_password == old_password:
                    flash("该密码与原密码一致，请重新填写", 'error')
                    return redirect(url_for('edit_profile'))
```

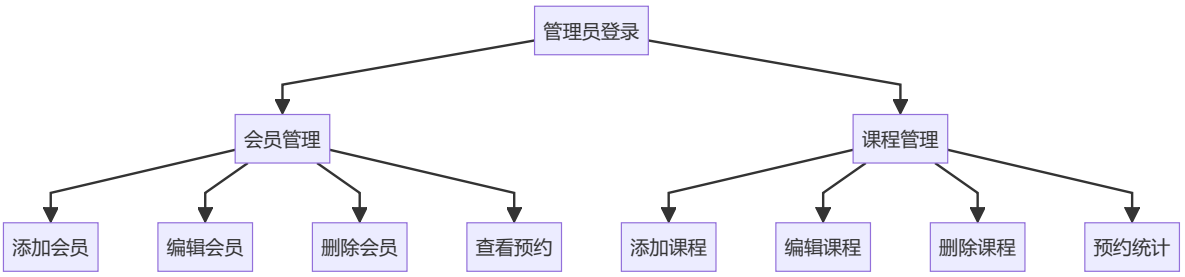


4.1.4 会员等级管理

- 会员表包含 `membership_level` 字段（整数类型）
- 管理员可设置会员等级（黄金会员、铂金会员等）
- 会员可查看自身等级（个人主页）
- 等级字段用于未来扩展（如差异化服务）

4.2 管理员子系统

4.2.1 管理员登录

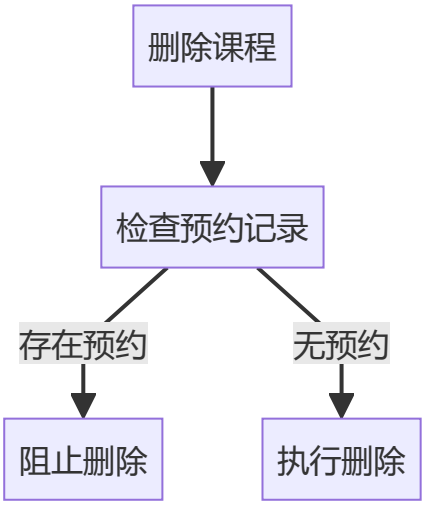


管理员登陆后主要是主界面中有两个子页面：一是会员管理，可以对会员信息进行增删改查；二是课程管理，可以对课程信息进行增删改查。

4.2.2 课程删除保护

系统实现课程删除前检查机制，若存在预约则无法删除，该课程无预约才能删除。

```
    try:
        # 首先检查课程是否被预约
        cursor.execute('SELECT 1 FROM Reservations WHERE course_id = %s LIMIT 1',
(course_id,))
        if cursor.fetchone():
            flash("该课程已被预约，无法删除", 'error')
            return redirect(url_for('view_courses'))

        # 如果没有预约，则删除课程
        cursor.execute('DELETE FROM Courses WHERE course_id = %s', (course_id,))
        conn.commit()
        flash("课程删除成功", 'success')
    except mysql.connector.Error as e:
        print(f"An error occurred while deleting a course: {e}")
        conn.rollback()
        flash("删除课程失败，请重试", 'error')
```



4.2.3 会员与课程搜索

管理员可按姓名搜索会员，可按课程名字搜索课程，并支持模糊查询（如输入"张"可查所有张姓会员）。若没有搜索内容，则显示所有用户和课程。

```
if search_query:
```

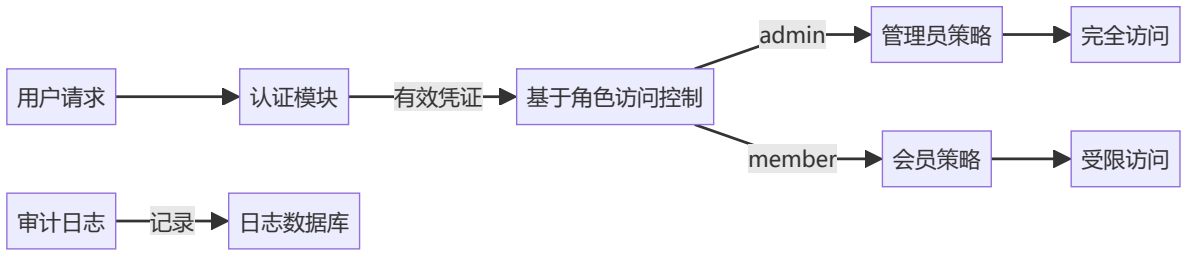
```
            # 搜索用户
            cursor.execute('''
                SELECT *
                FROM Members
                WHERE name LIKE %s
            ''', (f'%{search_query}%',))
        else:
            # 如果没有搜索内容，显示全部用户
            cursor.execute('SELECT * FROM Members')
        members = cursor.fetchall()

if search_query:
            # 只搜索课程基本信息，不检查预约状态
            cursor.execute('''
                SELECT c.course_id, c.course_name, c.duration, c.coach, c.price
                FROM Courses c
                WHERE c.course_name LIKE %s
            ''', (f'%{search_query}%',))
        else:
            # 如果没有搜索内容，显示全部课程
            cursor.execute('''
                SELECT c.course_id, c.course_name, c.duration, c.coach, c.price
                FROM Courses c
            ''')
```

五、安全设计

5.1 安全机制

安全层面	实现措施
认证安全	明文存储
会话安全	Flask Session加密
数据安全	MySQL角色权限分离
操作安全	关键操作二次确认
输入安全	参数化SQL查询防止注入

5.2 访问控制模型



六、关键问题与解决

6.1 角色分离与界面动态渲染

问题：同一HTML模板需要根据用户角色呈现不同界面，这里需要在该HTML文件中运用if-else语句来实现，此时我们就需要在主python文件中记录用户登入时的角色，该角色会在后面用于判断界面选择。

解决方案：

```
cursor.execute('SELECT role FROM Users WHERE username = %s AND password = %s',
(username, password))
    user = cursor.fetchone()
    if user:
        session['role'] = user[0]  # 将用户角色存储在会话中
```

6.2 个人选课界面链接

问题：如果以会员身份登入，无论在哪一个页面下，上方的导航栏都有一个要链接的个人选课界面，但由于我们的users中包括一个管理员账号，不能和会员账号共同编号，所以我们的数据库设计时，在users没有member_id。在实现会员界面时出现错误：

```
werkzeug.routing.exceptions.BuildError: Could not build url for endpoint
'view_reservations_by_member'. Did you forget to specify values ['member_id']?
```

解决方案：需要在登入时通过用户名查找到对应的member_id并存储，这样就可以通过该member_id来查找到该会员对应的选课记录，以此来保证个人选课界面的正确跳转。

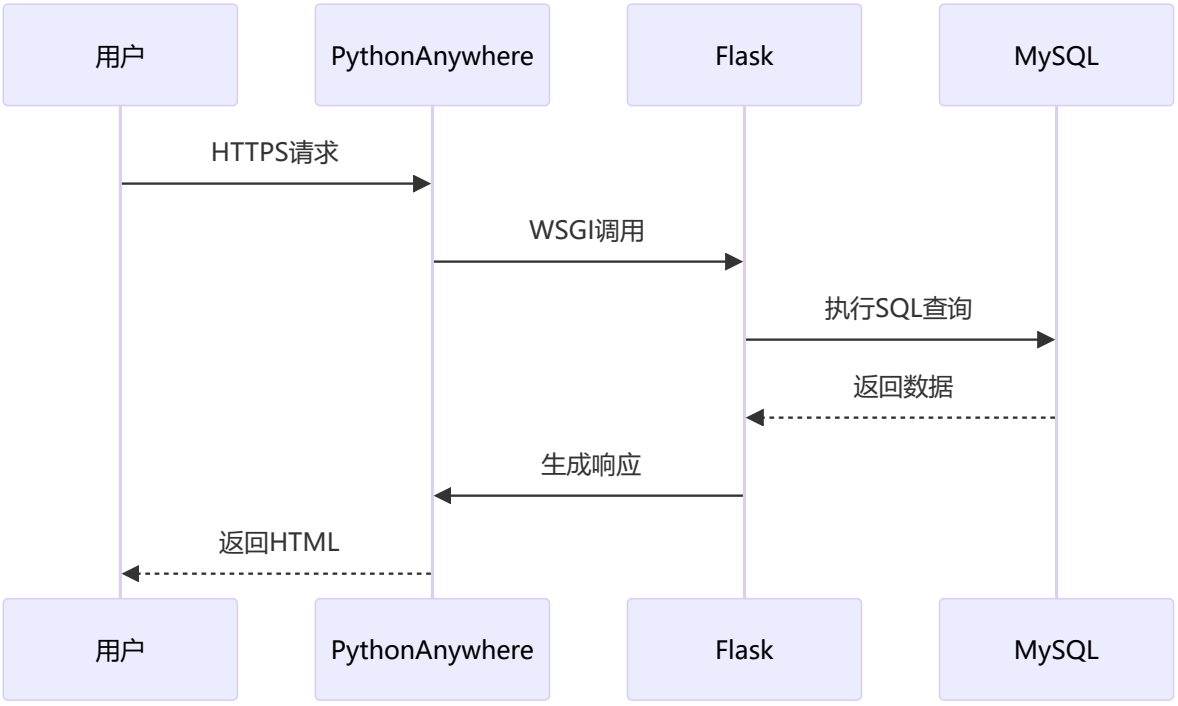
```
# 从 Members 表中根据用户名查找 member_id
     cursor.execute('SELECT member_id FROM Members WHERE name = %s',(username,))
            member_info = cursor.fetchone()
            if member_info:
                session['member_id'] = member_info[0]  # 将 member_id 存储在会话中
                return redirect(url_for('view_courses'))
            else:
                return "Member not found.", 404
```

6.3 PythonAnywhere部署适配

问题：本地开发环境与云平台差异。

我们刚开始是在本地进行开发，用的是PostgreSQL以及连接云数据库https://supabase.com/dashboard/project/gmevpselpkclylnjgity，Supabase提供了三种连接方式，分别是直接连接（仅允许IPv6地址，端口为5432）、事务池（兼容IPv4地址，端口为6543）、会话池（仅允许IPv4地址，端口为5432）。

后续我们转成PythonAnywhere替代本地开发环境，但由于是在平台的免费开发模式下进行，只允许出站连接到特定端口：80和443，而PythonAnywhere的服务器是IPv4网络，与Supabase云数据库的三种连接方式均不适配，因此我们的在线版本将数据库转成PythonAnywhere平台自带的免费MySQL数据库。本地版本仍可以使用Supabase云数据库。



6.4 时区处理

问题：数据库默认UTC时间与本地时间不一致。时间比中国时间早八小时。

解决方案：

```
# 预约时使用东八区时间
local_time = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
cursor.execute('''
    INSERT INTO Reservations (..., %s)
''', (local_time))
```

七、性能优化策略

- **连接管理**：按需创建/关闭数据库连接

```
def get_db_connection():
    try:
        conn = mysql.connector.connect(**DB_CONFIG)
        return conn
    except mysql.connector.Error as e:
        print(f"数据库连接失败: {e}")
        return None
```

- 查询优化：使用LIMIT检查预约状态

```
# 首先检查课程是否被预约
        cursor.execute('SELECT 1 FROM Reservations WHERE course_id = %s LIMIT
1', (course_id,))
```

- 缓存机制：会话(session)存储member_id减少查询

八、错误处理设计

- **分层处理**：
  - 前端：Bootstrap警报提示
  - 后端：try/except捕获数据库异常
  - 日志：print记录操作异常
- **关键保护**

```
except mysql.connector.Error as e:
        print(f"An error occurred while adding a member: {e}")
        conn.rollback()                          # 事务回滚
        flash("添加会员失败，请重试", 'error')  # 用户反馈
        return redirect(url_for('add_member'))
```

九、总结与展望

9.1 系统总结

本健身房会员管理系统实现了以下核心能力：

- 多角色支持（管理员/会员）
- 完整的会员生命周期管理
- 课程安排与预约管理
- 实时数据统计与展示
- 响应式前端界面

系统采用模块化设计，各层职责清晰，便于维护和扩展。通过Flask框架实现轻量级高效后端，结合MySQL保证数据可靠性。系统现已在PythonAnywhere上稳定运行，可通过分配的域名访问：demonnn7.pythonanywhere.com。

9.2 未来扩展方向

1. 移动端支持：开发React Native跨平台移动应用
2. 智能推荐：基于会员偏好和会员等级推荐课程
3. 物联网集成：接入健身设备数据采集
4. 财务模块：增加会员费管理和支付接口
5. 大数据分析：会员行为分析和业务预测

本架构设计平衡了系统功能性与技术可行性，为健身房的数字化转型提供了可靠的技术支持，具有良好的可扩展性和可维护性。