**河北经贸大学管信学院**

**2024-2025学年第2学期**

**《商务软件开发与实训》课程设计**

项目一：用户登录校验的设计与实现

系 别 软件工程

专 业 软件工程

项目成员 孙广旭、孙晴、王梓豪、李鑫宇

学 号 202222450221、202222450212

202222450247、202222450204

指导教师 崔文阔

二〇二五年四月十九日

**河北经贸大学管信学院**

**2024-2025学年第2学期**

**《商务软件开发与实训》课程设计**

项目一：用户登录校验的设计与实现

**小组成员**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **学号** | **姓名** | **工作分工** | **个人成绩** |
| **1** | **202222450221** | **孙广旭** | 项目编写、编写数据库表 |  |
| **2** | **202222450212** | **孙晴** | 项目编写、编写数据库表 |  |
| **3** | **202222450247** | **王梓豪** | 项目编写、编写数据库表 |  |
| **4** | **202222450204** | **李鑫宇** | 项目编写、编写数据库表 |  |

二〇二五年四月十九日

摘　　要

本项目是一个基于 Flask 框架开发的用户登录认证系统，采用前后端协同设计，实现了用户的注册、登录、密码修改、自动登录（记住我）等基本功能。后端使用 Flask-WTF 进行表单处理与数据验证，Flask-Login 实现用户身份验证与会话管理，Flask-SQLAlchemy 提供数据库操作支持。前端结合 Bootstrap 框架构建响应式界面，实现良好的用户体验。同时，系统通过哈希加盐技术保障用户密码的安全性，并支持密码强度校验，提高了系统的安全性与稳定性。该系统结构清晰，功能完整，适合作为中小型 Web 项目的用户认证模块参考

关键词：Flask；用户认证；Flask-Login；表单验证；密码加密

目录

[摘　　要 1](#_Toc27950)

[1. 绪论 3](#_Toc1772)

[1.1 研究背景 3](#_Toc5613)

[1.2 研究意义 3](#_Toc8746)

[1.3 系统简介 3](#_Toc10211)

[1.4 开发准备 3](#_Toc21719)

[2. 系统关键技术 4](#_Toc6993)

[2.1 Flask 框架 4](#_Toc2260)

[2.2 Flask-WTF 表单验证 4](#_Toc17068)

[2.3 Flask-Login 身份认证 5](#_Toc32367)

[2.4 密码哈希加盐存储 5](#_Toc32241)

[2.5 Bootstrap + 前端交互设计 5](#_Toc23287)

[3. 系统分析 5](#_Toc5492)

[3.1 系统需求 5](#_Toc1217)

[3.1.1性能需求 5](#_Toc24634)

[3.1.2 安全需求 5](#_Toc161)

[3.1.3易用性需求 5](#_Toc14885)

[3.2 数据需求 6](#_Toc25571)

[3.3 系统流程分析 6](#_Toc12406)

[3.4 本章小结 7](#_Toc18244)

[4. 系统设计与实现 7](#_Toc16246)

[4.1 未登录时首页模块设计与实现 7](#_Toc19069)

[4.2 未登录时小组成员模块设计与实现 8](#_Toc4673)

[4.3 登录模块设计与实现 9](#_Toc30045)

[4.4 登录时首页模块设计与实现 11](#_Toc32261)

[4.5 登录时小组成员模块设计与实现 16](#_Toc27272)

[4.6 修改密码模块设计与实现 18](#_Toc31804)

[4.7 退出登录模块设计与实现 21](#_Toc16724)

[5. 小结 22](#_Toc17317)

[参考文献 23](#_Toc18262)

1. 绪论

1.1 研究背景

随着高校课程改革的深入开展，团队协作型项目成为培养学生综合能力的重要方式。为了更好地展示小组成员风采与学习成果，本项目设计并实现了一个基于 Flask 的“第六小组”官网系统。该系统具备用户登录、退出、修改密码等基本功能，并通过权限控制实现信息的分级访问，提升了系统的交互性与安全性。在技术实现方面，项目采用 Flask 作为后端框架，结合 Flask-WTF 表单处理、Flask-Login 用户管理、SQLAlchemy 数据库操作等扩展，构建了一个轻量但功能完善的展示型网站。通过本项目的开发，有效提升了开发者在 Web 编程、前后端协作和系统设计方面的实践能力。

1.2 研究意义

本项目以“第六小组官网”的开发为载体，旨在通过实际的 Web 系统构建过程，全面提升开发者在前后端协同开发、用户交互设计、安全认证机制等方面的能力。在当前信息化和数字校园背景下，构建具有良好交互体验和权限控制的网站具有重要的现实意义。通过引入 Flask 框架及其相关插件，如 Flask-Login、Flask-WTF、SQLAlchemy 等，不仅锻炼了开发者对 Python Web 编程生态的掌握，同时也提高了团队协作开发、模块化设计、数据库建模等工程实践能力。

此外，系统设计中对用户登录认证、密码加密、记住登录状态等功能的实现，有助于理解现代 Web 系统中的用户身份管理与数据安全问题。该项目既是课堂学习内容的实践应用，也是面向实际需求的技术训练平台，为今后从事更复杂系统开发打下了坚实基础，具有较强的教学与实践意义。

1.3 系统简介

本系统是一个基于 Python Flask 框架开发的用户登录认证系统，主要实现了用户注册、登录、修改密码和退出登录等核心功能。系统采用工厂模式构建应用结构，结合 Flask-SQLAlchemy 管理数据库操作，使用 PyMySQL 实现对 MySQL 数据的存储和访问。为保障用户信息安全，系统对用户密码采用哈希加盐方式进行加密存储，并通过 Flask-Login 实现用户身份认证和会话管理，支持未登录用户自动跳转至登录页。前端页面基于 Flask-WTF 表单进行渲染，具备字段校验机制，确保用户输入的合法性。系统界面简洁美观，设有固定导航栏和底部页脚，用户可通过“我的”下拉菜单进行密码修改和退出操作，整体结构清晰，逻辑合理，具有良好的扩展性和教学参考价值，适用于小型网站或后台管理系统的用户认证模块。

1.4 开发准备

1. 操作系统：推荐Windows 10、11及以上。
2. 开发工具：PyCharm 2024（向下兼容）。
3. 开发语言：Python 3.12。
4. 数据库：MySQL 8.0+PyMySQL驱动。
5. Python Web框架：Flask 3.0。
6. 系统关键技术

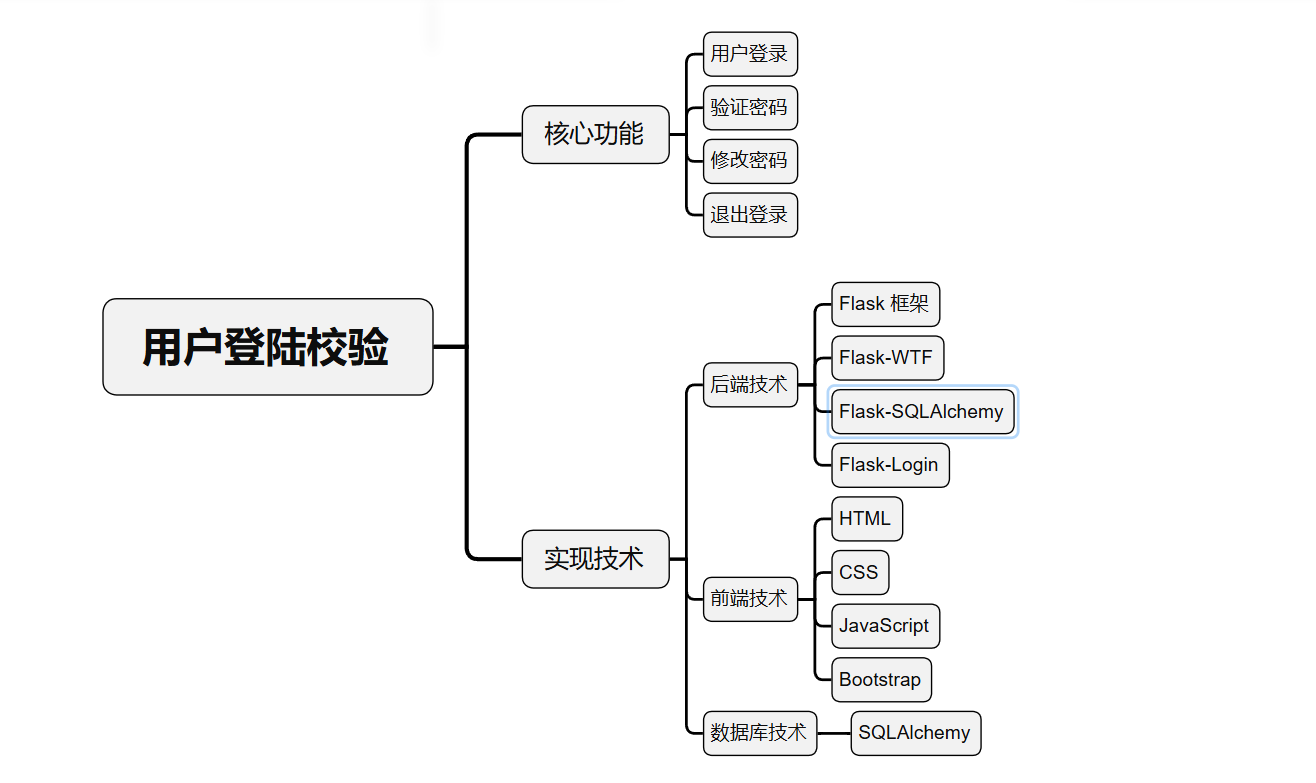


图 4.1 核心功能及实现技术总览

2.1 Flask 框架

Flask 是一个基于 Python 的轻量级 Web 框架，具备模块化、高可扩展性和灵活的路由机制。项目采用 Flask 构建整体后端逻辑，使用蓝图（Blueprint）进行模块划分，使得不同页面的逻辑独立、结构清晰。其内置的开发服务器和调试模式，便于开发和调试。在本系统中，Flask 实现了用户登录验证、页面跳转、小组成员信息展示等主要功能，为系统提供了稳定的 Web 后端支持。

2.2 Flask-WTF 表单验证

Flask-WTF 是 Flask 的表单扩展库，结合了 WTForms，使表单定义、验证更为便捷和安全。项目通过 Flask-WTF 创建了登录表单和修改密码表单，使用内置的验证器如 DataRequired、Email、EqualTo，确保用户输入的有效性。此外，还自定义了密码强度验证逻辑（包括长度限制和字符复杂度要求），提高了密码安全性，防止弱口令的使用。该技术有效提升了前后端表单交互的安全性与用户体验。

2.3 Flask-Login 身份认证

Flask-Login 是一个处理用户会话管理的扩展，负责用户登录状态的维护、权限控制和“记住我”功能的实现。在本项目中，Flask-Login 实现了用户的登录、退出、登录后访问受限资源等功能。系统通过 login\_user() 和 logout\_user() 管理用户会话，同时结合 @login\_required 装饰器保护需要登录后才能访问的页面。通过 current\_user 对象，系统可灵活判断用户是否处于登录状态，并动态更新页面显示内容。

2.4 密码哈希加盐存储

为保障用户密码的安全性，系统使用 werkzeug.security 模块提供的 generate\_password\_hash 和 check\_password\_hash 方法对密码进行哈希加盐处理。用户注册或修改密码时，密码并不会以明文形式存储在数据库中，而是转换为哈希值。用户登录时，输入的密码通过 check\_password\_hash 校验与数据库中的哈希值是否匹配。该机制有效防止数据库泄露时的密码被破解，增强系统整体的安全性。

2.5 Bootstrap + 前端交互设计

在前端页面设计中，项目采用 Bootstrap 框架构建响应式布局，实现导航栏、按钮、表单和页脚的样式统一与美观性。通过引入 CDN 的方式加载 Bootstrap5，使页面具备良好的兼容性和视觉表现。页面根据用户是否登录动态渲染不同内容，如“我的”下拉菜单或登录入口。同时，自定义样式表（style.css）进一步调整细节，如复选框、提示框及页脚吸底布局，使得整个系统界面友好、操作直观，增强了用户交互体验。

1. 系统分析

3.1 系统需求

3.1.1性能需求

本系统应具备良好的响应速度和页面加载效率，确保用户在访问首页、登录、查看成员信息等操作时都能迅速反馈。在并发访问情况下，系统应保持稳定运行，避免卡顿或崩溃，确保基础功能（如登录验证、密码修改等）始终流畅执行，提升整体使用体验。

3.1.2 安全需求

系统必须保障用户数据的安全性，采用密码哈希加盐技术存储用户密码，并通过 Flask-Login 实现会话管理，防止未授权访问。表单提交需具备验证机制，防止非法输入和CSRF攻击。同时通过权限控制机制确保未登录用户无法访问敏感页面内容。

3.1.3易用性需求

系统界面设计应简洁直观，用户操作逻辑清晰。首页、登录页、小组成员页等页面结构合理，功能按钮位置明显，配合提示信息引导用户顺利完成操作。支持自动登录与手动登录切换，提升用户体验，降低操作难度，便于不同层次用户使用。

3.2 数据需求

系统数据主要存储在关系型数据库中，采用 SQLAlchemy 进行数据建模与操作。核心数据结构如下：

表1 用户信息表（user）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 是否为空 | 描述 |
| id | INT | 11 | 否 | 用户编号（主键） |
| username | VARCHAR | 125 | 否 | 用户名 |
| email | VARCHAR | 125 | 否 | 用户邮箱 |
| password\_hash | VARCHAR | 512 | 否 | 加密密码 |

数据库在初始化时通过 createdb.py 文件建立，同时系统通过表单获取用户数据，并与数据库进行比对和更新。

3.3 系统流程分析

本系统的流程主要为以下流程图所示：

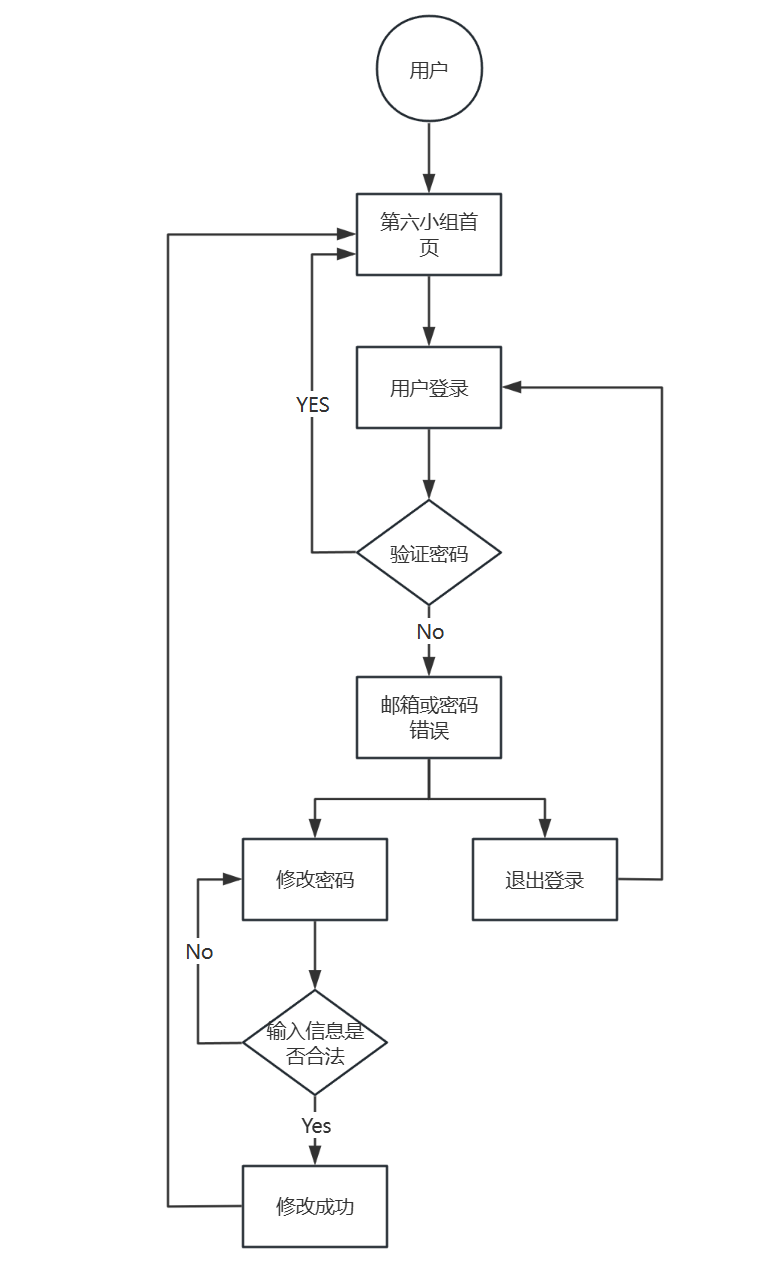


图 3.1 业务流程图

3.4 本章小结

本章围绕系统的功能需求、数据结构及流程设计进行了详细分析。系统以用户登录认证为核心，结合 Flask 系列组件，实现了身份识别、密码校验、信息权限控制等功能，为后续系统实现打下了坚实基础。通过合理的数据结构和流程设计，系统具备良好的可扩展性和实用性。

1. 系统设计与实现

4.1 未登录时首页模块设计与实现

在未登录状态下，首页模块主要展示欢迎信息与提示信息，界面简洁明了。用户访问系统首页时，通过 Flask 的路由控制判断当前用户登录状态（current\_user.is\_authenticated）。若未登录，页面中央显示系统欢迎语句与提示文字，如“登录后才可显示详情哦~”，以引导用户先进行身份验证。前端页面使用 Jinja2 模板语言进行条件渲染，并结合 Bootstrap 实现居中排版和样式美化。该模块在功能上不涉及用户隐私与数据库操作，确保安全性，同时在设计上兼顾用户体验，为用户提供清晰的操作路径。该实现体现了前后端联动的基本设计思想，也是用户认证系统中重要的访问控制入口之一。



图4.1 用户未登录时首页页面

关键代码：

{% else %}

<!-- 未登录内容 -->

<h2 style="text-align:center;">欢迎来到第六小组首页！</h2>

<p style="text-align:center; color: gray;">登录后才可显示详情哦~</p>

{% endif %}

4.2 未登录时小组成员模块设计与实现

在未登录状态下，小组成员模块采用信息隐藏的方式进行设计，实现了访问权限的基本控制。当用户未登录系统时，该模块仅展示标题“第六小组成员信息”，并以趣味性提示语替代真实数据内容，营造神秘感并引导用户登录。该功能通过 Flask 模板语法中的 if current\_user.is\_authenticated 进行权限判断，仅在用户登录后才加载具体的成员信息模板。前端使用居中排版和表情符号增强提示效果，同时保持页面风格一致性。此模块设计不仅保护了数据隐私，也提升了系统的交互性和用户体验。



图4.2 用户未登录时小组成员页面

关键代码：

{% else %}

<!-- 未登录状态显示提示信息 -->

<div class="alert alert-info text-center mt-4" role="alert">

🤫 神秘的小组成员信息，你想看嘛~ 请先登录解锁哦~

</div>

{% endif %}

4.3 登录模块设计与实现

本系统的登录模块基于 Flask 框架开发，结合 Flask-WTF 实现前端表单渲染与后端数据校验。用户登录表单包括邮箱、密码两个基本字段及一个“记住我”复选框，通过 LoginForm 表单类定义。系统使用 Flask-Login 管理用户会话，login\_user(user, remember=...) 实现持久登录功能，并结合 current\_user.is\_authenticated 控制页面显示逻辑。后端登录逻辑在 /login 路由中完成：校验表单合法性后查询数据库匹配用户，并通过加盐哈希验证密码正确性。若用户勾选“记住我”，则浏览器会持久保存登录状态，反之下次需要重新登录。此外，为避免未勾选时仍被记住的情况，系统在用户未选择“记住我”时强制调用 logout\_user() 清除持久会话，有效保证登录逻辑的安全性与用户控制权。



图4.3 登录模块页面

关键代码：

{% extends 'base.html' %}

{% block content %}

<div class="login-container">

<h2>用户登录</h2>

<form method="POST" class="login-form">

{{ form.hidden\_tag() }}

<div>

{{ form.email.label }}

{{ form.email(size=32, autocomplete="email") }}

{% for error in form.email.errors %}

<span class="error-message">{{ error }}</span>

{% endfor %}

</div>

<div>

{{ form.password.label }}

{{ form.password(size=32, autocomplete="current-password") }}

{% for error in form.password.errors %}

<span class="error-message">{{ error }}</span>

{% endfor %}

</div>

<!-- ✅ 记住我复选框 -->

<div class="form-check" style="margin: 10px 0;">

<input class="form-check-input" id="remember" name="remember" type="checkbox" value="y">

<label class="form-check-label" for="remember">记住我</label>

</div>

<div>

{{ form.submit(class\_='login-button') }}

</div>

</form>

</div>

{% endblock %}

config.py

# 记住我 - cookie - 的有效期

REMEMBER\_COOKIE\_DURATION = timedelta(days=7)

routes.py

@main.route('/login', methods=['GET', 'POST'])

def login():

form = LoginForm()

if form.validate\_on\_submit():

user = User.query.filter\_by(email=form.email.data).first()

if user and user.check\_password(form.password.data):

# ✅ 如果这次没勾选“记住我”，主动清除之前的持久登录状态

if not form.remember.data:

logout\_user()

login\_user(user, remember=form.remember.data)

return redirect(url\_for('main.home'))

else:

flash('邮箱或密码错误')

return render\_template('login.html', form=form)

4.4 登录时首页模块设计与实现

在用户成功登录后，系统会根据 current\_user.is\_authenticated 判断用户是否处于已认证状态，从而在首页展示完整内容。首页采用了 Jinja2 模板引擎，根据登录状态动态切换页面内容。当用户已登录时，页面中央显示欢迎信息，并加载一个基于 Bootstrap 实现的轮播图组件，展示多张小组相关图片。轮播图通过 HTML 和 Bootstrap 的 carousel 类实现，支持自动轮播、左右切换等功能，并通过 CSS 控制尺寸与页面布局，确保界面美观。同时，为提升用户体验，在轮播图与底部页脚之间通过自定义样式设置了额外间距，避免页面拥挤。整体设计逻辑清晰，功能展示直观，增强了用户的沉浸感与系统的可视化效果。



图4.4 用户未登录时首页页面

关键代码：

{% extends 'base.html' %}

{% block content %}

{% if current\_user.is\_authenticated %}

<!-- 登录后的内容 -->

<h2 style="text-align:center;">欢迎来到第六小组首页！</h2>

<p style="text-align:center;">没错！૮・ﻌ・ა 这就是展示的内容！</p>

<!-- 登录后显示轮播图 -->

<div id="carouselExample" class="carousel slide mx-auto mt-4 spacing-bottom" data-bs-interval="2000" data-bs-ride="carousel" data-bs-pause="hover" style="width: 700px;height:600px;">

<div class="carousel-inner">

<div class="carousel-item active">

<img src="{{ url\_for('static', filename='img/lt1.jpg') }}" class="d-block w-100" alt="图1">

</div>

<div class="carousel-item">

<img src="{{ url\_for('static', filename='img/lt2.jpg') }}" class="d-block w-100" alt="图2">

</div>

<div class="carousel-item">

<img src="{{ url\_for('static', filename='img/lt3.jpg') }}" class="d-block w-100" alt="图3">

</div>

<div class="carousel-item">

<img src="{{ url\_for('static', filename='img/lt4.jpg') }}" class="d-block w-100" alt="图4">

</div>

</div>

<!-- 控制按钮 -->

<button class="carousel-control-prev" type="button" data-bs-target="#carouselExample" data-bs-slide="prev">

<span class="carousel-control-prev-icon" aria-hidden="true"></span>

</button>

<button class="carousel-control-next" type="button" data-bs-target="#carouselExample" data-bs-slide="next">

<span class="carousel-control-next-icon" aria-hidden="true"></span>

</button>

</div>

{% else %}

<!-- 未登录内容 -->

<h2 style="text-align:center;">欢迎来到第六小组首页！</h2>

<p style="text-align:center; color: gray;">登录后才可显示详情哦~</p>

{% endif %}

{% endblock %}

base.html

<!DOCTYPE html>

<html lang="zh-CN">

<head>

<meta charset="UTF-8">

<title>第六小组网站</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">

<!-- Bootstrap CSS -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet" crossorigin="anonymous">

</head>

<body>

<header>

<nav class="my-navbar">

<div class="logo">

<a href="{{ url\_for('main.home') }}" class="logo-link">第六小组</a>

</div>

<div class="nav-links">

<a href="{{ url\_for('main.home') }}">首页</a>

<a class="nav-link" href="{{ url\_for('main.members') }}">小组成员</a>

<a href="#">个人简介</a>

<a href="#">信息下载</a>

</div>

<div class="user-menu">

{% if current\_user.is\_authenticated %}

<div class="dropdown">

<button class="dropbtn" onclick="toggleDropdown()">我的</button>

<div class="dropdown-content" id="dropdownMenu">

<a href="{{ url\_for('main.change\_password') }}">修改密码</a>

<a href="{{ url\_for('main.logout') }}">退出登录</a>

</div>

</div>

{% else %}

<a href="{{ url\_for('main.login') }}">登录</a>

{% endif %}

</div>

</nav>

</header>

<main>

{% block content %}{% endblock %}

</main>

<footer>

<div class="footer-content">

<div class="qr-code">

<img src="{{ url\_for('static', filename='picture1.png') }}" alt="二维码" width="100">

</div>

<div class="site-info">

<p>这是第六小组的官网，展示我们的信息与成果。</p>

<ul class="footer-links">

<li><a href="#">家长监护</a></li>

<li><a href="#">网络报警</a></li>

<li><a href="#">免责声明</a></li>

<li><a href="#">出版许可</a></li>

</ul>

</div>

</div>

<p class="copyright">© 第六小组 所有</p>

</footer>

<!-- Bootstrap Modal 用于 flash 消息弹窗 -->

<div class="modal fade" id="alertModal" tabindex="-1" aria-labelledby="alertModalLabel" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content">

<!-- 下面这个是提示框颜色-->

<div class="modal-header">

<h5 class="modal-title" id="alertModalLabel">提示</h5>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="关闭"></button>

</div>

<div class="modal-body" id="alertMessage">

<!-- flash 消息将被插入这里 -->

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-bs-dismiss="modal">关闭</button>

</div>

</div>

</div>

</div>

<!-- Bootstrap JS -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js"></script>

<!-- 处理 flash 消息显示 Modal 弹窗 -->

<script>

window.onload = function() {

var messages = {{ get\_flashed\_messages()|tojson }};

if (messages.length > 0) {

document.getElementById('alertMessage').innerText = messages[0];

var alertModal = new bootstrap.Modal(document.getElementById('alertModal'));

alertModal.show();

}

<!-- 错误提示信息 -->

var flashMessage = document.getElementById('flash-message');

if (flashMessage) {

flashMessage.classList.add('show'); // 显示提示消息

// 2.5秒后自动消失

setTimeout(function() {

flashMessage.classList.remove('show');

}, 2500);

}

};

function toggleDropdown() {

var dropdown = document.querySelector('.dropdown');

dropdown.classList.toggle("show");

}

</script>

</body>

</html>

4.5 登录时小组成员模块设计与实现

在用户登录成功后，系统允许访问“小组成员”模块。该模块通过后端路由控制，仅在用户认证通过后才允许访问对应页面。页面内容为固定的小组成员信息展示，采用结构清晰的 HTML 页面呈现，内容包括成员姓名、学号、班级等。前端页面结合了 Bootstrap 样式进行布局优化，确保信息排列整齐、排版美观。使用 Jinja2 模板判断用户认证状态，未登录用户访问该模块时将显示提示语并隐藏具体内容。该设计实现了权限控制与信息保护，同时提升了模块的交互性与趣味性，增强了系统的用户体验与安全性。

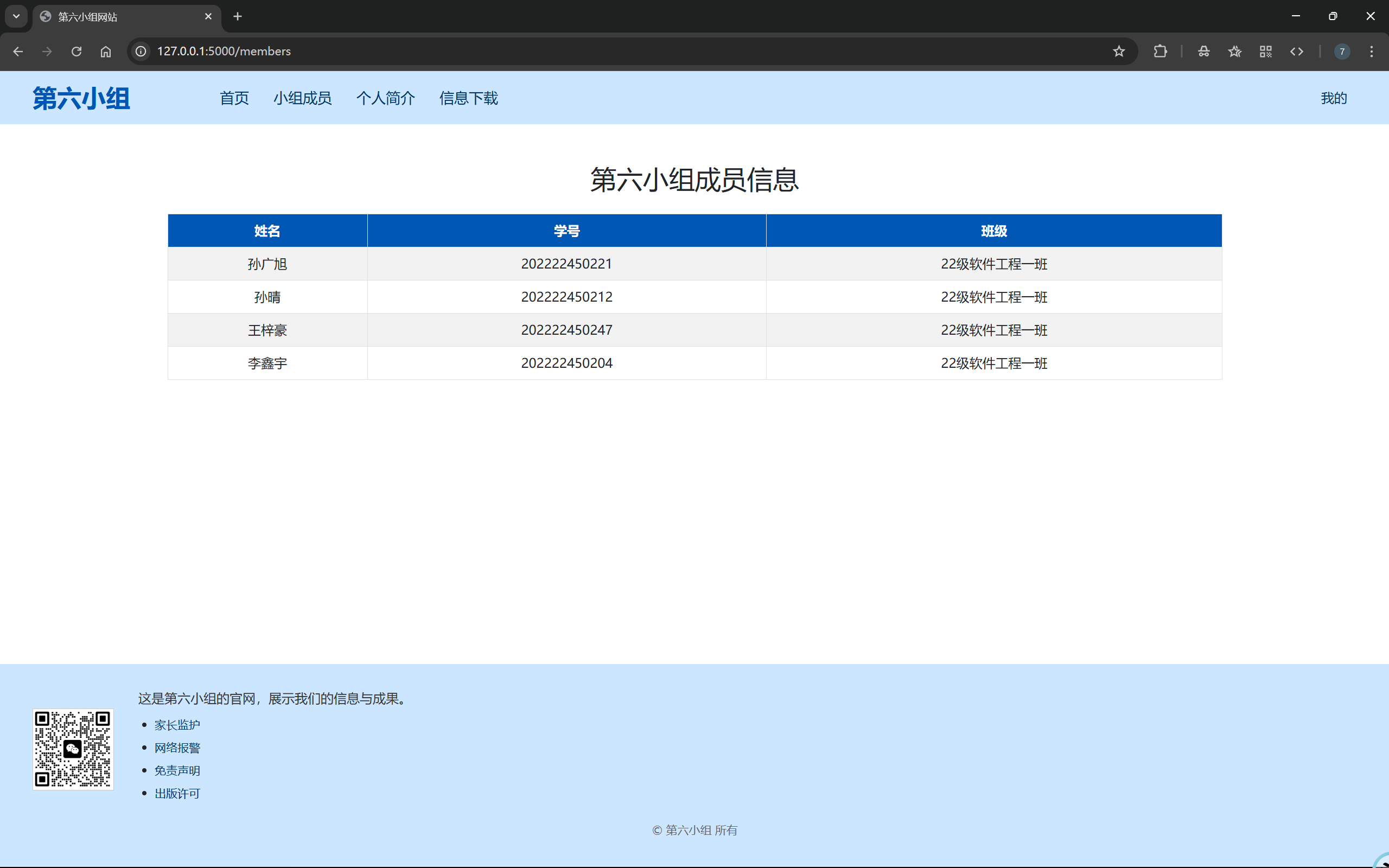


图4.5 用户登录时小组成员页面

关键代码：

{% extends 'base.html' %}

{% block content %}

<div class="container mt-5 extra-space-bottom">

<h2 class="text-center">第六小组成员信息</h2>

{% if current\_user.is\_authenticated %}

<!-- 登录状态下显示表格 -->

<table class="table table-striped table-bordered mt-4">

<thead class="custom-blue text-center">

<tr>

<th>姓名</th>

<th>学号</th>

<th>班级</th>

</tr>

</thead>

<tbody class="text-center">

<tr>

<td>孙广旭</td>

<td>202222450221</td>

<td>22级软件工程一班</td>

</tr>

.........

</tbody>

</table>

{% else %}

<!-- 未登录状态显示提示信息 -->

<div class="alert alert-info text-center mt-4" role="alert">

神秘的小组成员信息，你想看嘛~ 请先登录解锁哦~

</div>

{% endif %}

</div>

{% endblock %}

routes.py

# 小组成员

@main.route('/members')

def members():

return render\_template('members.html')

4.6 修改密码模块设计与实现

修改密码模块旨在为已登录用户提供安全、高效的密码更新功能。用户需输入旧密码、新密码及确认新密码，系统使用 Flask-WTF 表单验证器进行输入校验。在后端逻辑中，首先通过 current\_user.check\_password() 验证旧密码是否正确，确保操作的用户身份合法。新密码在提交前需满足复杂度要求：长度为6-12位，且必须包含数字、大写字母、小写字母中的至少两种字符类型，避免设置弱密码。密码存储采用哈希加盐加密处理，使用 Werkzeug 提供的 generate\_password\_hash() 对新密码进行加密，确保用户密码在数据库中的安全性。该模块不仅保障了用户数据的隐私性，也提升了整体系统的安全性。



图4.6 用户修改密码页面



图4.7 用户修改密码出错情况

关键代码：

{% extends 'base.html' %}

{% block content %}

<div class="login-container">

<h2>修改密码</h2>

<!-- 错误提示消息 -->

{% with messages = get\_flashed\_messages() %}

{% if messages %}

<div id="flash-message" class="alert alert-danger" role="alert">

{{ messages[0] }}

</div>

{% endif %}

{% endwith %}

<form method="POST" class="login-form">

{{ form.hidden\_tag() }}

<div>

<label for="old\_password">{{ form.old\_password.label }}</label>

{{ form.old\_password(size=32, class\_='form-control') }}

{% for error in form.old\_password.errors %}

<span class="error-message">{{ error }}</span>

{% endfor %}

</div>

<div>

<label for="new\_password">{{ form.new\_password.label }}</label>

{{ form.new\_password(size=32, class\_='form-control') }}

{% for error in form.new\_password.errors %}

<span class="error-message">{{ error }}</span>

{% endfor %}

</div>

<div>

<label for="confirm\_password">{{ form.confirm\_password.label }}</label>

{{ form.confirm\_password(size=32, class\_='form-control') }}

{% for error in form.confirm\_password.errors %}

<span class="error-message">{{ error }}</span>

{% endfor %}

</div>

<div>

{{ form.submit(class\_='login-button btn btn-primary') }}

</div>

</form>

</div>

{% endblock %}

routes.py

@main.route('/change\_password', methods=['GET', 'POST'])

@login\_required

def change\_password():

form = ChangePasswordForm()

if form.validate\_on\_submit():

if current\_user.check\_password(form.old\_password.data):

current\_user.set\_password(form.new\_password.data)

try:

db.session.commit()

flash('密码修改成功！请重新登录~ ٩(๑•̀ω•́๑)۶')

logout\_user()

return redirect(url\_for('main.home')) # 返回未登录首页

except Exception as e:

db.session.rollback()

flash('密码修改失败，请稍后再试')

else:

flash('原密码错误')

return render\_template('change\_password.html', form=form)

4.7 退出登录模块设计与实现

退出登录模块主要实现用户会话的安全终止功能。当用户点击“退出登录”按钮时，系统通过 Flask-Login 提供的 logout\_user() 方法，清除当前用户的登录状态和会话信息，防止未授权访问用户专属页面。该操作通常绑定在导航栏中的“退出登录”链接上，并通过 Flask 的 @login\_required 装饰器确保只有已登录用户才可执行。退出后，系统使用 redirect(url\_for('main.login')) 跳转回登录页面，引导用户重新认证。该模块设计简洁明了，配合 Flask-Login 实现用户身份的快速切换与会话管理，确保用户数据的私密性与系统的安全性。



图4.8 用户退出登录页面

关键代码：

routes.py

@main.route('/logout')

@login\_required

def logout():

logout\_user()

return redirect(url\_for('main.login'))

1. 小结

本项目以 Flask 为核心框架，构建了一个具备基本用户认证功能的 Web 应用系统，实现了用户注册、登录、密码修改、退出登录以及权限控制等模块功能。通过集成 Flask-WTF 实现前后端表单交互与校验，增强了系统的安全性和用户体验。在用户登录模块中，采用哈希加盐方式加密用户密码，保障用户信息的机密性；配合 Flask-Login 实现了用户会话管理与“记住我”自动登录功能，提升了用户使用的便捷性。同时，为了增强用户交互体验，首页设计了轮播图和欢迎界面，并通过判断用户登录状态展示不同内容，实现了个性化界面展示。

项目前端采用 Jinja2 模板引擎与 Bootstrap 组件相结合，确保页面风格整洁美观且具备响应式特性。在模块分工方面，系统遵循 MVC 架构划分视图、表单、模型与业务逻辑，结构清晰、易于维护与拓展。

本系统虽然功能简单，但涵盖了 Web 开发中常用的用户认证流程和前后端联动机制，具备良好的学习与实践价值。后续可拓展用户注册功能、数据库信息管理、头像上传等内容，使系统功能更加完善。

参考文献

[1] 王伟，李强。基于 Flask 框架的学生信息管理系统设计与实现 [J]. 信息与电脑，2023, 35(04): 88-90.

[2] 吴迪，张峰。校园二手交易平台的用户体验提升路径探讨 [J]. 数字技术与应用，2022, 40(11): 134-136.

[3] 陈晨。基于 Python 的网站用户登录认证系统设计 [J]. 数字通信世界，2023(08): 112-114.

[4] 张敏，刘洋。浅析Web前端与后端技术的融合开发 [J]. 计算机技术与发展，2021, 31(10): 56-59.

[5] 赵玲。浅谈Flask框架在Web开发中的应用 [J]. 软件导刊，2022, 21(05): 132-134.