

课程设计

题 目 基于 Flask 的文学网站系统设计

学 院 信息科学与工程学院

专 业 计算机科学与技术

班 级

学 生 ff202118

学 号

指导教师

二〇二三 年 六 月 二十六 日

摘 要

随着 Internet 的广泛应用，动态网页技术也应运而生。本文介绍了应用 Flask 动态网页技术和 MySQL 数据库开发文学网站系统的设计与实现。该系统主要旨在为用户提供发表文学作品、浏览作品等功能，并通过 Internet 实现用户间的交流和互动。系统开发设计采用了 Flask 作为动态网页框架，结合 MySQL 数据库作为数据存储和管理工具，从而为用户提供功能丰富且用户友好的文学交流平台。

在系统设计中，本文首先对文学网站系统进行整体分析，明确了系统的可行性和用户需求。根据模块化原理，规划设计了系统的功能模块，包括用户注册、登录、文学作品展示、评论互动等。此外，在数据库设计中，详细说明了系统数据库的结构以及数据的完整性和安全性措施。

针对程序设计，本文采用了面向对象的程序设计思想，提出了系统的程序设计思路，并对前台和后台功能的程序实现进行了详细论述。系统测试部分分析了测试过程中出现的主要问题，并提出了解决方案，以确保系统功能和稳定性的实现。

最后，对系统进行了客观、全面的评价，并提出了进一步改进的建议，以不断提升文学网站系统的用户体验和性能。通过整合 Flask 的灵活性和 MySQL 的强大功能，该文学网站系统为用户提供了一个功能齐全、易于使用的文学交流平台，推动文学创作的发展和传播。

关键词：Flask；MySQL；文学网站；用户交流；数据库设计；程序设计；用户体验

目 录

摘 要	I
1 前言	1
1.1 项目背景	1
1.1.1 市场需求	1
1.1.2 用户期望	1
1.1.3 现有技术体系	错误!未定义书签。
1.2 市场分析	错误!未定义书签。
1.3 开发内容	5
1.4 论文结构介绍	6
2 相关技术简介	8
2.1 Flask 框架	8
2.2 MySQL 数据库	8
2.3 HTML、CSS 和 JavaScript	8
2.4 本章小结	9
3 系统分析	10
3.1 项目可行性分析	10
3.1.1 技术可行性	10
3.1.2 操作可行性	10
3.1.3 社会可行性	10
3.1.4 其他可行性	10
3.2 功能需求分析	11
3.2.1 系统数据流图	11
3.2.2 系统数据字典	12
3.2.3 系统数据模型	13
3.2.4 系统用例图	14
3.2.5 系统类图	17
3.3 非功能需求分析	18
3.3.1 性能需求	18
3.3.2 可用性需求	18
3.3.3 安全性需求	18
3.3.4 可维护性	19

3.4 本章小结	19
4 系统设计	20
4.1 系统架构设计	20
4.2 关键模块设计	22
4.3 数据库设计	31
4.4 界面设计	33
4.5 本章小结	34
5 系统实现与测试	35
5.1 系统实现	35
5.1.1 用户认证模块实现	35
5.1.2 文章发布模块	39
5.1.3 留言管理模块	43
5.1.4 用户管理模块	46
5.1.5 文章管理模块	48
5.1.6 查询模块	50
5.1.7 个人信息	52
5.2 系统测试	54
5.2.1 功能测试	54
5.2.2 出错处理测试	56
6 结语与展望	57
6.1 结语	57
6.2 展望	57
7 参考文献	58

1 前言

1.1 项目背景

1.1.1 市场需求

根据招商证券发布的《2022 年在线阅读行业发展现状分析多元化在线阅读市场需求匹配充分》，2022 年在线阅读行业发展现状分析显示，多元化在线阅读市场需求匹配充分，用户对于不同类型和风格的文学作品有着多样化的需求，同时也渴望与其他文学爱好者进行交流和互动。报告还分析了在线阅读用户的年龄结构、付费意愿、使用习惯、题材偏好等特征，为文学网站系统的设计提供了参考市场需求分析。据此可以得知在文学网站系统领域存在着以下关键需求。

1、文学创作和分享需求

随着互联网的普及，越来越多的人愿意通过在线平台分享自己的文学作品，展示自己的创作才华。用户希望一个文学网站系统能够提供方便的创作工具，支持用户发布、编辑和管理自己的文学作品，以及与其他用户分享、交流和接受反馈。

2、文学交流和互动需求

读者们渴望与其他文学爱好者进行交流和互动，分享阅读心得、讨论作品等。用户期望文学网站系统能够提供评论、点赞、关注等功能，以促进用户之间的互动和社区氛围的建立。

3、多样化的文学作品需求

读者们对于不同类型和风格的文学作品有着多样化的需求。他们希望能够方便地浏览和发现各种类型的文学作品，包括小说、散文、诗歌等。用户期望文学网站系统提供一个丰富多样的作品库，支持按照不同分类、标签或作者进行检索和浏览。

4、便捷的阅读体验需求

用户希望能够在文学网站系统上享受到便捷的阅读体验。他们期望系统提供清晰、美观的排版，支持不同阅读设备和屏幕大小的适配，以及个性化的阅读设置，例如字体大小、夜间模式等。此外，用户也希望系统能够提供书签功能，方便他们记录和管理自己的阅读进度。

1.1.2 用户期望

一个文学网站系统能够满足用户在文学创作、阅读和交流方面的需求。首先，用户期望文学网站系统能够提供方便的创作工具。他们希望能够通过系统轻松地发布、编辑

和管理自己的文学作品。这包括一个简洁而直观的编辑界面，支持格式化文本、插入图片和调整布局等功能。用户希望能够以自己的方式展现创作才华，使其作品更具个性和吸引力。

其次，用户渴望与其他文学爱好者进行交流和互动。他们希望文学网站系统能够提供社区功能，让他们能够与其他读者分享阅读心得、讨论作品等。这包括评论、点赞和分享功能，以使用户能够表达对作品的喜爱和观点。用户还期望能够关注其他用户或作家，以便及时获取其最新作品和动态。通过这种交流和互动，用户能够建立联系、扩展社交圈并从中获得更多的文学灵感和启发。

此外，用户对于文学作品的多样性也有着明显的需求。他们希望能够方便地浏览和发现各种类型和风格的文学作品，还希望能够按照不同的分类、标签或作者来组织和筛选作品，以便快速找到自己感兴趣的内容。

对于阅读体验，用户希望文学网站系统能够提供便捷和愉悦的阅读环境。他们希望系统具有清晰、美观的页面排版，以提升阅读的舒适度和可读性。此外，用户还希望系统能够适配不同的阅读设备和屏幕大小，使其能够在手机、平板和电脑等各种设备上阅读。个性化的阅读设置也是用户的期望，例如调整字体大小、切换夜间模式等，以满足不同用户的阅读习惯和需求。此外，用户还希望系统能够提供书签功能，方便他们记录和管理阅读进度，以便随时回到之前阅读的地方。

综上所述，用户期望一个文学网站系统能够提供方便的创作工具、社区交流和互动功能、多样化的文学作品选择，以及便捷的阅读体验。满足这些关键需求将能够吸引更多的用户，建立一个活跃的文学社区，并为用户提供丰富的文学体验和互动机会。

1.1.3 现有技术体系

现有技术体系是指在文学网站系统开发领域已经存在的技术和工具集合，用于设计、开发和部署文学网站系统。这些技术和工具经过多年的发展和实践，在功能性、可扩展性和可靠性方面得到验证，为开发人员提供了丰富的选择和支持。在本文中，我们将详细探讨文学网站系统开发中的现有技术体系。

1、Flask 框架

Flask 是一个轻量级的 Python Web 框架，它以简洁和灵活的设计理念而闻名。Flask 提供了构建 Web 应用程序所需的核心功能，同时保持了足够的灵活性，允许开发人员根据自己的需求进行定制。Flask 的设计哲学是“微框架”，意味着它只提供了基本的功能，如路由、请求处理和模板引擎，而其他功能如数据库集成、用户认证等可以通过第三方扩展来实现。

Flask 的优势之一是其易于学习和上手。它具有清晰的文档和简洁的 API，使初学者能够快速入门并开始构建 Web 应用程序。另外，Flask 的代码结构简单，易于理解和维

护，使开发人员能够专注于业务逻辑的实现。

在文学网站系统开发中，Flask 框架可以用于处理用户请求和路由，管理用户会话和权限，以及与数据库进行交互。通过 Flask 的扩展库，如 Flask-Login 和 Flask-WTF，开发人员可以方便地实现用户认证和表单验证。Flask 还提供了轻量级的模板引擎 Jinja2，使开发人员能够构建动态的网页模板，以呈现文学作品和用户界面。

2、MySQL 数据库

MySQL 是一种开源的关系型数据库管理系统，广泛用于 Web 应用程序和文学网站系统的数据存储和管理。MySQL 具有可靠性、高性能和可扩展性的特点，能够处理大量的数据并提供高效的查询和事务支持。

在文学网站系统中，MySQL 可以用于存储和管理各种数据，如用户信息、文章内容、评论等。通过使用 MySQL，开发人员可以设计和创建数据库表，定义数据结构和关系，并执行复杂的查询操作来满足用户需求。此外，MySQL 还提供了数据安全性和完整性的控制机制，如用户权限管理和数据备份。

3、HTML、CSS 和 JavaScript

HTML、CSS 和 JavaScript 是构建 Web 页面的核心技术。HTML（超文本标记语言）定义了网页的结构和内容，CSS（层叠样式表）用于定义网页的样式和布局，而 JavaScript 用于实现网页的交互和动态效果。

在文学网站系统中，HTML、CSS 和 JavaScript 的配合使用可以创建吸引人且具有良好用户体验的前端界面。通过 HTML，开发人员可以定义文学作品的展示方式，如标题、段落、图片等。CSS 则负责为网页添加样式，如字体、颜色、布局等，以增强用户界面的美观性和一致性。JavaScript 提供了丰富的功能和库，如 DOM 操作、事件处理、动画效果等，使开发人员能够实现与用户的实时交互，例如评论、点赞和分享功能。

4、RESTful API

RESTful API 是一种基于 HTTP 协议的 Web 服务架构风格，它通过定义一组 URL 和请求方法来实现对资源的操作。RESTful API 可以用于文学网站系统的前后端交互，使前端可以通过 API 发送请求并获取数据或执行操作，实现数据的传输和业务逻辑的处理。

在文学网站系统中，RESTful API 可以用于用户注册、登录、发布作品、评论、点赞等操作。通过 RESTful API，开发人员可以设计和定义合适的 API 接口，确保数据的安全性和一致性，并提供友好的错误处理和响应机制。常见的 RESTful API 工具包括 Flask-RESTful、Django REST framework 等，它们提供了便捷的方式来构建和管理 API。

5、第三方库和工具

在文学网站系统开发过程中，开发人员可以利用各种第三方库和工具来简化开发任务，提高效率和质量。这些库和工具提供了丰富的功能和扩展，可以满足不同的需求。

例如，Flask-Login 是一个用于处理用户认证和会话管理的 Flask 扩展库。它提供了登录、登出、会话管理等功能，使开发人员能够轻松地实现用户身份验证和权限控制；Bootstrap 是一个流行的前端开发框架，提供了各种现成的样式和组件，可以快速构建响应式和美观的界面。通过使用 Bootstrap，开发人员可以节省大量时间和精力，同时确保界面的一致性和可靠性。

此外，还有许多其他有用的工具和库，如 jQuery(JavaScript 库)、SQLAlchemy(Python 数据库工具)、Celery（分布式任务队列）、Redis（内存数据库）等，它们可以在不同方面提供支持，如前端交互、数据库操作、任务处理等。

1.2 市场分析

当涉及到文学交流网站的竞争产品时，存在许多相关或相似的平台和应用。以下是对几个具有代表性的竞争产品进行更详尽和丰富的分析。

1、Medium

Medium 是一个备受关注的在线文学交流平台，它致力于建立作者与读者之间的联系，并提供一个分享和讨论创作的平台。这个平台允许用户发表自己的文章，并与其他用户进行交流和互动。

（1）功能：提供丰富的文章发表和编辑功能，支持富文本格式和多媒体内容，使作者能够以专业和吸引人的方式展示自己的创作；用户可以在文章上留下评论，与作者和其他读者进行交流和互动，为创作者提供反馈和建议；平台通过根据用户的兴趣和关注推荐相关的文章，帮助用户发现新的文学作品和创作。

（2）优势：平台汇集了来自世界各地的写作者，用户可以接触到各种不同风格和主题的文学作品，满足多样化的阅读需求；提供了高质量的编辑和排版工具，使用户的文章以专业和美观的方式呈现，增加作品的吸引力和可读性；社交功能齐全，用户可以关注其他作者，建立关系网络，并获得更多的关注和回应，增加作品的曝光度。

（3）不足：平台上存在大量的文章，用户可能面临信息过载的问题，难以筛选出符合自己兴趣和品味的作品，需要花费时间和精力来发现高质量的内容；由于平台的开放性，存在虚假信息和低质量内容的可能性，用户需要具备辨别真伪的能力，以获取可靠和有价值的阅读体验；部分用户认为平台过于商业化，存在广告和付费内容过多的情况，影响了用户体验和阅读的流畅性。

2、知乎

知乎是一个广泛应用的综合性问答社区，虽然它的主要功能不是文学交流，但在文学领域也存在一定的用户活动和内容分享。

（1）功能：用户可以提出与文学相关的问题，向其他用户寻求答案和建议，与其

他用户共同探讨文学话题；用户可以回答和评论其他用户提出的文学问题，分享自己的知识和见解，为其他用户提供有价值的信息和观点；平台上有一些专栏作家，他们会发布有关文学创作和阅读的专栏文章，为用户提供更深入的文学内容。

（2）优势：平台上的用户群体广泛，用户可以获得来自不同背景和领域的观点和见解，拓宽了对文学的理解和认知；平台上存在一些专业人士和文学爱好者，用户可以从他们的专业知识和经验中获取高质量的文学知识和见解。

（3）不足：知乎的主要功能并非文学交流，文学话题可能在平台上相对较少，用户需要花费时间和精力寻找相关内容；平台上存在大量的回答和评论，用户需要耗费精力筛选和评估其中的信息可信度和质量；文学交流可能与其他领域的内容混杂，用户需要具备一定的鉴别能力。

综上所述，这些竞争产品在文学交流网站领域提供了不同的功能和特点。了解这些竞争产品的优势和不足可以为文学交流网站的设计和优化提供参考，以便更好地满足用户的期望和需求。

1.3 开发内容

文学交流网站的功能模块可以分为以下几个部分：

1、用户认证模块

- （1）实现了用户注册功能，用户可以填写必要的信息进行注册，并进行账户验证。
- （2）实现了用户登录功能，用户可以使用注册时的账号和密码进行登录。

2、文章发布模块

- （1）实现了文章发表功能，用户登录后可以填写文章的标题、内容等信息，并选择文章所属的版块。完成后，提交申请，等待版主的审核。
- （2）同样地，管理员和版主也可以填写文章的标题、内容等信息，并选择文章所属的版块。完成后，无需审核即可发表。

3、留言管理模块

- （1）实现了留言功能，用户登录后可以在文章下方留下评论和反馈。
- （2）实现了留言删除功能，管理员可以删除不当或违规的留言。

4、用户管理模块

- （1）实现了用户权限管理功能，管理员可以设置用户的权限，包括版主、普通用户等。
- （2）实现了用户添加功能，管理员可以手动添加用户，并设置其权限。

5、文章管理模块

- （1）实现了文章标签的添加、删除、编辑功能，管理员可以对文章标签进行统一

管理。

(2) 实现了文章审核功能，版主可以对用户提交的文章进行审核，审核通过的可以发表在网页中，反之，则不能。

6、查询模块

(1) 实现了文章搜索功能，用户可以根据关键词搜索文章标题或内容，以便快速找到感兴趣的文章。

(2) 实现了文章分类查询功能，用户可以按照版块名称或其他分类标准对文章进行筛选和浏览，以便浏览特定类型的文章。

(3) 实现了文章标签查询功能，用户可以点击文章标签，快速找到相关主题的文章。

7、个人信息模块

(1) 实现了个人资料查看功能，用户可以查看自己的个人信息，包括用户名、邮箱、注册时间等。

(2) 实现了个人资料编辑功能，用户可以编辑和更新自己的个人信息。

(3) 实现了个人留言查看功能，用户可以查看自己发布的留言，通过留言可以追踪到文章本身。

(4) 实现了用户密码重置功能，用户可以重新设置自己的密码。

文学交流网站实现了用户认证、文章发布、留言管理、版块管理和用户管理等功能模块，以满足用户在网站上发表文章、交流评论和浏览不同版块的需求。同时，系统也注重安全性和性能优化，以确保用户信息的安全和系统的稳定运行。

1.4 论文结构介绍

论文共分为七个章节，分别为前言、相关技术简介、系统分析、系统设计、系统实现与测试、结语与展望、参考文献。

1、前言

本章介绍了论文的背景，包括项目背景、市场需求、用户期望和现有技术体系。还介绍了市场分析和开发内容，并简要介绍了论文的结构。

2、相关技术简介

本章主要介绍了文学交流网站开发所使用的关键技术，包括 Flask 框架、MySQL 数据库以及 HTML、CSS 和 JavaScript 等。每个技术都有简单的介绍，为后续章节的系统分析和设计打下基础。

3、系统分析

本章对文学交流网站进行了系统分析。首先进行了项目可行性分析，包括技术可行性、操作可行性、社会可行性和其他可行性。接下来进行了功能需求分析，包括系统数

据流图、数据字典、数据模型、用例图和类图的设计。最后进行了非功能需求分析，包括性能需求、可用性需求、安全性需求和可维护性。

4、系统设计

本章详细介绍了文学交流网站的系统设计。包括系统架构设计、关键模块设计、数据库设计和界面设计。每个设计方面都进行了详细的阐述，确保系统能够满足功能需求和非功能需求。

5、系统实现与测试

本章描述了文学交流网站的实现和测试过程。具体包括用户认证模块实现、文章发布模块、留言管理模块、用户管理模块、文章管理模块、查询模块和个人信息模块的实现。同时介绍了系统测试的方法和结果。

6、结语与展望

本章对整个论文进行了总结，并展望了未来文学交流网站的发展方向。结语部分对项目的成果进行总结，展望部分提出了进一步改进和扩展的建议。

7、参考文献

列出了论文中引用的相关文献，供读者进一步查阅。

2 相关技术简介

2.1 Flask 框架

Flask 是一个轻量级的 Python Web 框架，它以简洁和灵活的设计理念而闻名。Flask 提供了构建 Web 应用程序所需的核心功能，同时保持了足够的灵活性，允许开发人员根据自己的需求进行定制。Flask 的设计哲学是“微框架”，意味着它只提供了基本的功能，如路由、请求处理和模板引擎，而其他功能如数据库集成、用户认证等可以通过第三方扩展来实现。

Flask 的优势之一是其易于学习和上手。它具有清晰的文档和简洁的 API，使初学者能够快速入门并开始构建 Web 应用程序。另外，Flask 的代码结构简单，易于理解和维护，使开发人员能够专注于业务逻辑的实现。

在文学网站系统开发中，Flask 框架可以用于处理用户请求和路由，管理用户会话和权限，以及与数据库进行交互。通过 Flask 的扩展库，如 Flask-Login 和 Flask-WTF，开发人员可以方便地实现用户认证和表单验证。Flask 还提供了轻量级的模板引擎 Jinja2，使开发人员能够构建动态的网页模板，以呈现文学作品和用户界面。

2.2 MySQL 数据库

MySQL 是一种开源的关系型数据库管理系统，广泛用于 Web 应用程序和文学网站系统的数据存储和管理。MySQL 具有可靠性、高性能和可扩展性的特点，能够处理大量的数据并提供高效的查询和事务支持。

在文学网站系统中，MySQL 可以用于存储和管理各种数据，如用户信息、文章内容、评论等。通过使用 MySQL，开发人员可以设计和创建数据库表，定义数据结构和关系，并执行复杂的查询操作来满足用户需求。此外，MySQL 还提供了数据安全性和完整性的控制机制，如用户权限管理和数据备份。

2.3 HTML、CSS 和 JavaScript

HTML、CSS 和 JavaScript 是构建 Web 页面的核心技术。HTML（超文本标记语言）定义了网页的结构和内容，CSS（层叠样式表）用于定义网页的样式和布局，而 JavaScript 用于实现网页的交互和动态效果。

在文学网站系统中，HTML、CSS 和 JavaScript 的配合使用可以创建吸引人且具有良好用户体验的前端界面。通过 HTML，开发人员可以定义文学作品的展示方式，如标题、段落、图片等。CSS 则负责为网页添加样式，如字体、颜色、布局等，以增强用户

界面的美观性和一致性。JavaScript 提供了丰富的功能和库，如 DOM 操作、事件处理、动画效果等，使开发人员能够实现与用户的实时交互，例如评论、点赞和分享功能。

2.4 本章小结

本章概述了在文学网站系统开发中使用的关键技术和工具。Flask 框架作为一个轻量级的 Python Web 框架，以其简洁和灵活的设计理念而闻名，提供了构建 Web 应用程序所需的核心功能。MySQL 数据库是一种可靠性高、性能优秀的关系型数据库管理系统，适用于文学网站系统中的数据存储和管理。同时，HTML、CSS 和 JavaScript 作为构建 Web 页面的核心技术，通过定义网页结构、样式和实现交互，为文学网站系统提供了吸引人且具有良好用户体验的前端界面。综合使用这些技术和工具，开发人员可以构建出功能完善、用户友好的文学网站系统。

3 系统分析

3.1 项目可行性分析

3.1.1 技术可行性

技术可行性分析了系统所使用的技术是否能够满足开发需求和实现系统功能。在本文学交流网站系统的开发中,使用了 Flask 作为后端框架, jinja2 作为模板引擎, Bootstrap 和 Vue 作为前端技术, 以及 MySQL 作为数据库。这些技术都是成熟且广泛应用的技术, 具备稳定性、可靠性和安全性, 能够满足系统的需求。此外, Flask 框架的灵活性和可扩展性使得开发人员能够根据需求进行定制和扩展, 满足系统的功能和性能要求。因此, 从技术角度来看, 该文学交流网站系统具备良好的可行性。

3.1.2 操作可行性

操作可行性分析了系统在实际操作和使用中的便捷性和可操作性。该文学交流网站系统通过使用简洁明了的界面设计和用户友好的交互方式, 使用户能够轻松上手并操作系统。用户可以通过简单的操作进行文章的发表、留言、浏览和管理, 同时系统提供了搜索、分类和标签等功能, 使用户能够方便地找到感兴趣的文章。此外, 个人信息的管理和编辑也经过了简化和优化, 使用户能够轻松更新和维护个人资料。因此, 从操作角度来看, 该文学交流网站系统具备良好的可行性。

3.1.3 社会可行性

社会可行性分析了系统对社会环境和用户需求的适应程度。文学交流网站作为一个在线社区平台, 满足了用户分享和交流文学作品的需求。它为文学爱好者提供了一个互动和交流的平台, 促进了文学创作和阅读文化的传播。用户可以通过该系统展示自己的创作才华, 与其他文学爱好者交流和讨论, 激发了更多人参与文学创作和阅读的热情。因此, 从社会角度来看, 该文学交流网站系统具备良好的可行性。

3.1.4 其他可行性

此处分析了系统在经济、法律、环境等方面的可行性。从经济角度来看, 文学交流网站系统可以通过广告投放、会员收费或其他商业模式获得收益, 具备一定的商业可行性。从法律角度来看, 系统需要符合相关的法律法规, 保护用户隐私和版权, 遵循合规性要求。从环境角度来看, 该系统基于互联网平台, 使用虚拟化技术, 减少了对实体资源的消耗, 具备环保可行性。

3.2 功能需求分析

3.2.1 系统数据流图

顶层数据流图（如图 3.1）对系统主要的外部实体和数据流进行了大体的设计。

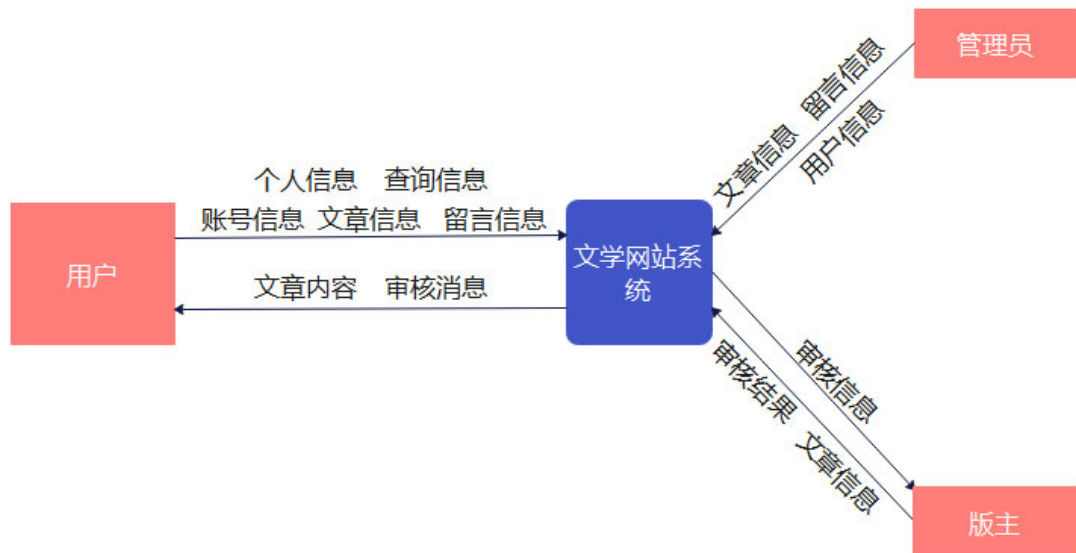


图 3.1 顶层数据流图

数据流图（如图 3.2）根据顶层数据流图，对数据流、处理过程和数据存储进行了详细的描述，主要针对于系统的功能模块进行设计。

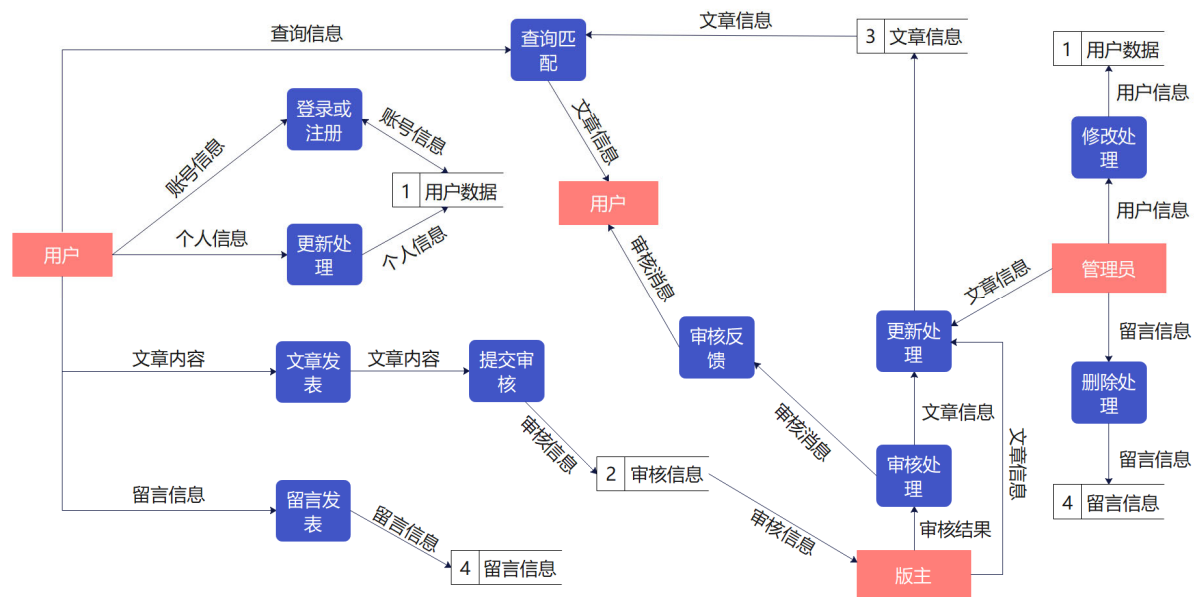


图 3.2 数据流图

3.2.2 系统数据字典

1、文章信息数据字典

表 3.1 文章信息数据字典

字段	类型	是否允许为空	说明
id	int	否	主键
title	varchar(128)	否	文章标题
desc	varchar(200)	是	文章简介
has_type	enum('draft', 'show')	否	文章类型（草稿,发布）
category_id	int	否	归属分类 id
content	longtext	否	文章内容
add_date	datetime	否	添加时间
pub_date	datetime	否	更新时间
user_id	int	否	发表文章的用户 id

2、用户数据数据字典

表 3.2 用户数据数据字典

字段	类型	是否允许为空	说明
inspect	int	是	版主所管理的分类 id
id	int	否	主键
username	varchar(128)	否	用户名
password	varchar(320)	否	密码
desc	varchar(150)	是	个人描述
avatar	varchar(200)	是	头像
gexing	varchar(100)	是	个性签名
gender	varchar(30)	是	性别
email	varchar(10)	是	邮箱
address	varchar(150)	是	住址
is_super_user	boolean	是	是否为超级用户版主
is_first_user	boolean	是	是否为茶余饭后版主
is_second_user	boolean	是	是否为风花雪月版主
is_third_user	boolean	是	是否为校园故事版主
is_fourth_user	boolean	是	是否为以诗会友版主
is_active_user	boolean	是	是否为活跃用户
is_staff	boolean	是	账户是否锁定
add_date	datetime	否	添加时间
pub_date	datetime	否	更新时间

3、留言信息数据字典

表 3.3 留言信息数据字典

字段	类型	是否允许为空	说明
id	int	否	主键
content	varchar(300)	否	留言内容
user_id	int	否	用户 id
post_id	int	否	文章 id
add_date	datetime	否	添加时间
pub_date	datetime	否	更新时间

4、审核信息数据字典

表 3.4 审核信息数据字典

字段	类型	是否允许为空	说明
id	int	否	主键
title	varchar(128)	否	留言内容
desc	int	是	文章描述
content	int	否	文章 id
category_id	varchar(200)	否	分类 id
user_id	in	否	用户 id
add_date	datetime	否	添加时间
pub_date	datetime	否	更新时间

3.2.3 系统数据模型

1、实体-属性图

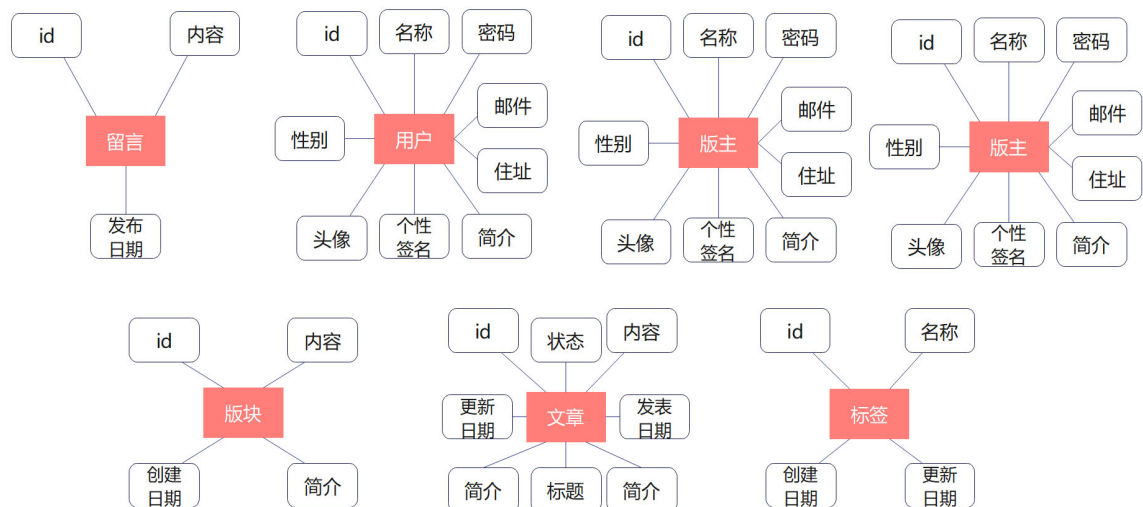


图 3.3 实体-属性图

2、实体-联系图

下面对实体间的联系类型进行说明。

- ①一个用户、管理员或版主可以在一篇文章下发表多个留言。
- ②每篇文章都属于一个特定的版块，而每个版块下可以有多篇文章。
- ③每篇文章可以被赋予多个标签，同样一个标签也可以对应多篇文章。
- ④一个用户、管理员或版主可以发表多篇文章。
- ⑤一个版主可以审核多篇文章，一篇文章也可以有多个版主进行审核。

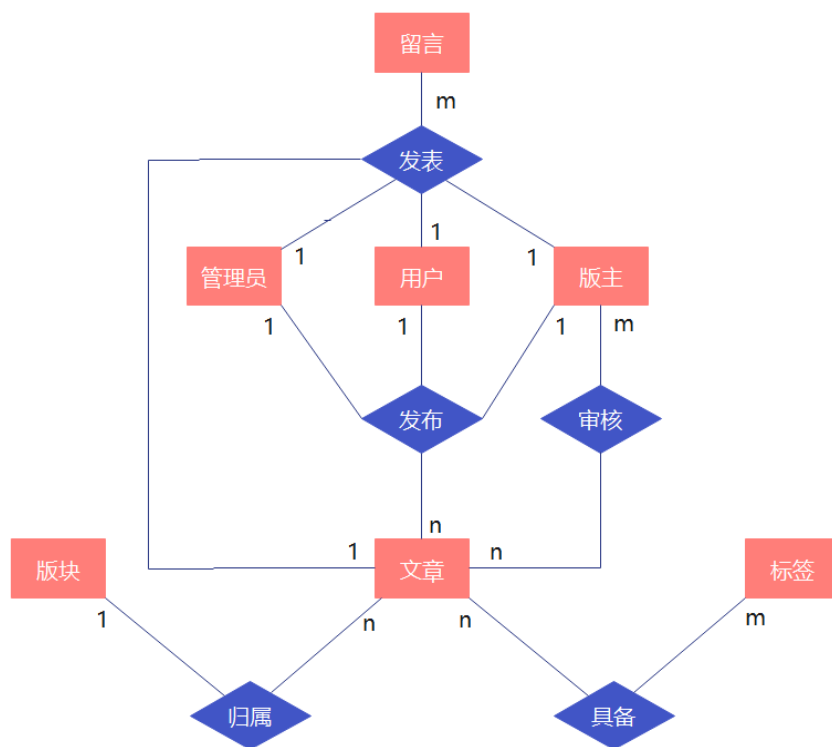


图 3.4 实体-联系图

3、基本 E-R 图

由于图 3.3 和图 3.4 已经详细说明了各实体、属性和联系，而基本 E-R 图是对实体、属性和联系的综合表达，因此，我们可以依据这些图形来建立一个全面而准确的基本 E-R 图。在这种情况下，再次绘制一个基本 E-R 图可能会导致重复和冗余，而且无法提供更多的信息。因此，我们可以直接参考图 3.3 和图 3.4，以获得关于实体、属性和联系之间关系的深入理解，并基于这些信息进行数据库设计。

3.2.4 系统用例图

1、用例图

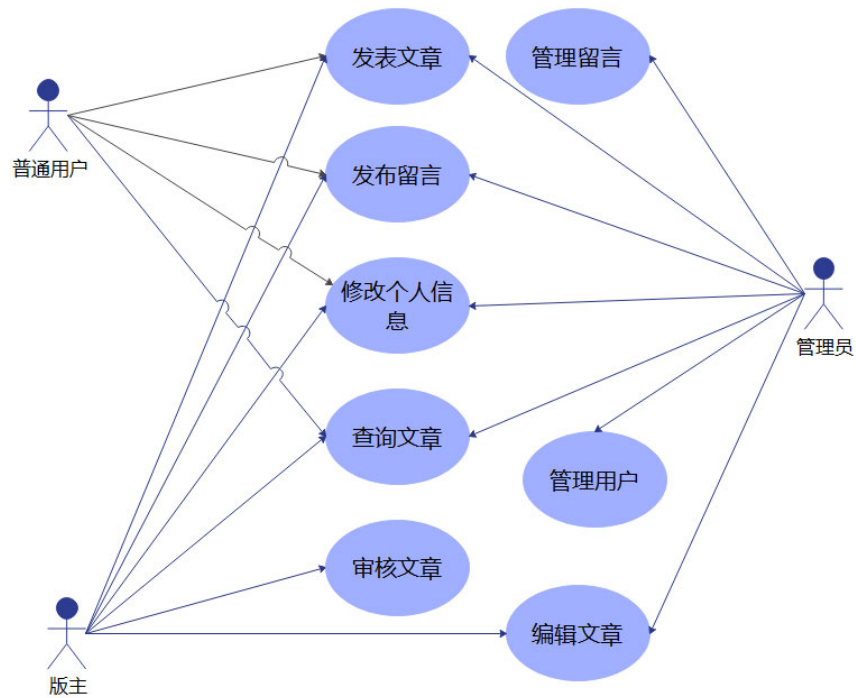


图 3.6 用例图

2、用例规约

用例名称	发表文章
用例编号	1
参与者	普通用户、版主、管理员
前置条件	用户登录系统
后置条件	文章提交成功并进入审核状态
基本流程	1、用户登录系统 2、用户进入文章发表界面 3、用户编辑文章信息 4、用户提交文章进行审核

用例名称	发布留言
用例编号	2
参与者	普通用户、版主、管理员
前置条件	用户登录系统
后置条件	留言提交成功并显示在对应文章下
基本流程	1、用户登录系统 2、用户进入文章页面 3、用户在对应文章下编辑留言信息 4、用户提交留言

用例名称	修改个人信息
用例编号	3
参与者	普通用户、版主、管理员
前置条件	用户登录系统
后置条件	个人信息成功修改并显示
基本流程	1、用户登录系统 2、用户进入个人中心 3、用户编辑个人信息 4、用户保存个人信息

用例名称	查询文章
参与者	普通用户、版主、管理员
用例编号	4
前置条件	用户进入文章搜索界面
后置条件	搜索成功并显示出搜索结果
基本流程啊	1、用户登录系统 2、用户进入文章查询界面 3、用户填写查询内容 4、用户提交搜索请求

用例名称	审核文章
用例编号	5
参与者	版主
前置条件	版主登录系统
后置条件	文章成功发布
基本流程	1、版主登录系统 2、版主进入文章审核界面 3、版主对文章进行合格与不合格评定 4、版主提交评定结果

用例名称	管理留言
用例编号	6
参与者	管理员
前置条件	管理员登录系统
后置条件	管理成功删除
基本流程	1、管理员登录系统 2、管理员进入管理留言界面 3、管理员对不当言论进行删除 4、管理员点击删除按钮

用例名称	管理用户
用例编号	7
参与者	管理员
前置条件	管理员登录系统
后置条件	用户信息成功修改
基本流程	1、管理员登录系统 2、管理员进入管理用户页面 3、管理员修改用户信息 4、用户保存用户信息

用例名称	编辑文章
参与者	管理员、版主
用例编号	8
前置条件	用户登录系统
后置条件	文章信息成功保存
基本流程	1、用户登录系统 2、用户进入文章编辑界面 3、用户填写文章编辑信息 4、用户保存文章信息

3.2.5 系统类图

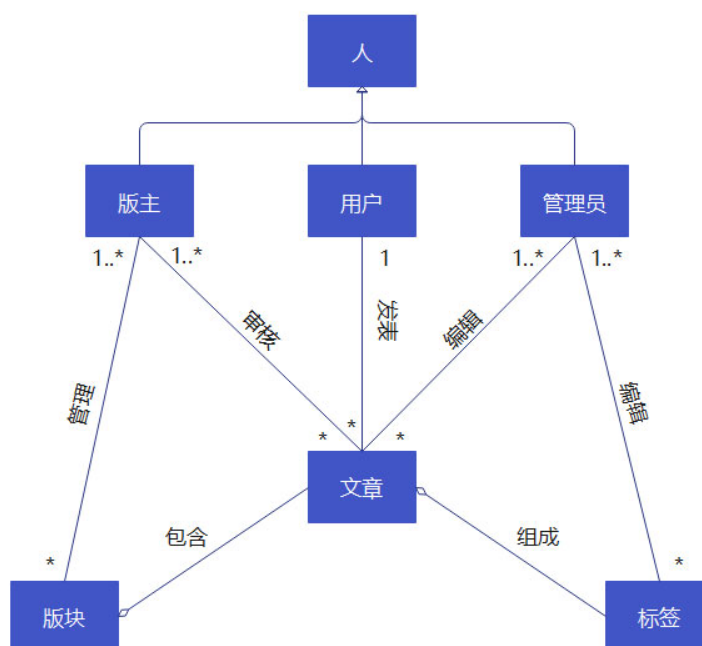


图 3.7 类图

3.3 非功能需求分析

3.3.1 性能需求

1、响应时间

- (1) 文章浏览页面加载时间不超过 2 秒。
- (2) 留言提交后，留言板显示更新时间不超过 1 秒。

2、并发性能

- (1) 能够同时处理至少 100 个并发用户请求，而不出现系统崩溃或明显延迟。

3、数据库性能

- (1) 数据库查询响应时间不超过 200 毫秒。
- (2) 数据库读取吞吐量达到每秒至少 100 次。

4、可扩展性

- (1) 能够支持至少 1000 个注册用户和 10000 篇文章的存储和管理。
- (2) 能够水平扩展，以适应用户量的增加，例如每增加 1000 个用户，系统性能不受明显影响。

5、页面加载时间

- (1) 文章列表页面加载时间不超过 1 秒。
- (2) 文章详情页面加载时间不超过 2 秒。

6、并发留言处理

- (1) 能够同时处理至少 50 个并发留言请求，确保留言板的实时更新。

3.3.2 可用性需求

- (1) 用户界面友好：系统的用户界面应设计简洁、直观，符合用户的习惯和预期，使用户能够轻松上手和操作系统。
- (2) 易学性和易用性：系统应提供清晰的操作指导和帮助文档，以使用户快速学习和掌握系统的使用方法。
- (3) 错误处理和恢复：系统应能够捕获和处理用户的错误输入，并给出相应的提示和建议，同时能够进行数据备份和恢复，以保证系统数据的完整性和可靠性。

3.3.3 安全性需求

- (1) 用户身份验证：系统应提供用户身份验证机制，确保只有合法用户才能进行文章发表、留言等操作，并防止未经授权的访问和操作。
- (2) 数据保护和隐私：系统应采取合适的措施保护用户的个人信息和文章内容的

隐私安全，确保数据不被非法获取或篡改。

(3) 防止恶意攻击：系统应具备防止常见的网络攻击，如 SQL 注入、跨站脚本等，确保系统的安全性和稳定性。

3.3.4 可维护性

(1) 可扩展性和模块化设计：系统应采用模块化的设计和良好的架构，使得系统的各个模块之间解耦合，方便后续的功能扩展和维护。

(2) 易于修改和维护：系统的代码应具备良好的可读性和可维护性，使得开发人员能够方便地进行代码的修改和维护，提高开发效率和系统的稳定性。

3.4 本章小结

本章主要对系统的可行性进行了分析和评估。首先，从技术可行性、操作可行性、社会可行性和其他可行性等方面进行了评估，确保系统的开发和实施是可行的。接下来，针对非功能需求进行了分析，包括性能需求、安全性需求、可用性需求等。在性能需求方面，对系统的响应时间、并发性能、数据库性能和页面加载时间等进行了量化的指标设定。最后，通过本章的分析和评估，为后续的系统设计和开发提供了指导和依据。

综上所述，本章的目标是确保系统在技术、操作、社会和其他方面的可行性，并明确定义了系统的非功能需求，特别是性能需求。这为后续的系统开发和实施奠定了基础，确保系统能够满足用户的期望和要求。下一章将进入系统设计和开发的具体阶段，根据本章的分析结果进行具体的功能设计和实现。

4 系统设计

4.1 系统架构设计

系统采用分层架构设计，主要分为用户界面层、业务逻辑层、数据访问层、数据存储层。

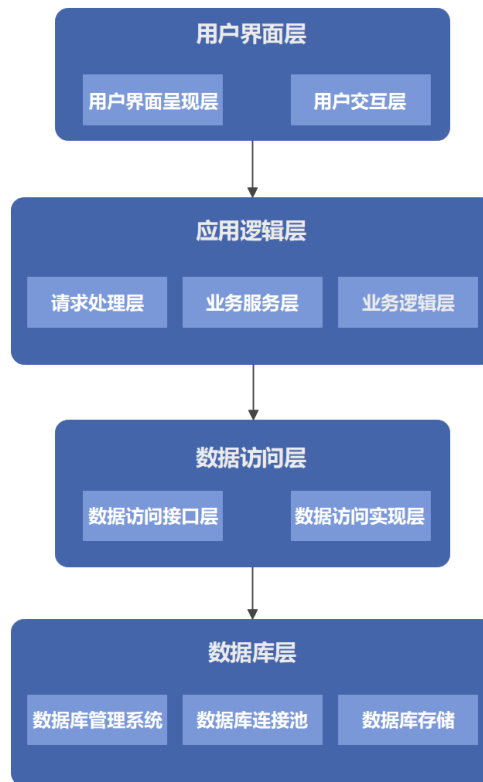


图 4.1 系统架构图

1、用户界面层

（1）用户界面呈现层

负责将后端数据呈现为用户可见的页面，使用 HTML、CSS 和模板引擎等技术实现页面的布局、样式和动态渲染。

设计理由：用户界面呈现层的设计目的是将系统的数据以用户友好的方式呈现给用户。使用 HTML、CSS 和模板引擎可以实现灵活的页面布局和样式，以及动态渲染的效果。这样的设计能够提升用户的体验，并让用户对系统的操作和内容更加直观和易于理解。

（2）用户交互层

处理用户的交互行为，包括前端验证、表单提交、AJAX 请求等，使用 JavaScript 和前端框架来实现交互逻辑。

设计理由：用户交互层的设计旨在响应用户的操作，并与后端进行数据交互。通过 JavaScript 和前端框架，可以实现实时验证用户输入、处理表单提交、异步请求等交互逻辑，以提供更加友好和灵活的用户交互方式。这样的设计可以增强系统的用户体验，使用户能够与系统进行有效的互动。

2、业务逻辑层

（1）请求处理层

接收和处理用户请求，进行参数解析、权限验证等操作。

设计理由：请求处理层负责接收和解析用户的请求，以及进行必要的验证和准备工作。这一层的设计可以使系统能够准确获取用户的请求并提供相应的数据或服务。参数解析和权限验证等操作有助于确保请求的合法性和安全性。

（2）业务逻辑层

实现系统的具体业务逻辑，包括用户认证、数据处理、权限管理等。

设计理由：业务逻辑层是系统的核心，负责实现具体的业务逻辑和功能。将业务逻辑独立为一层的设计可以提高系统的可维护性和扩展性，方便对业务规则的修改和调整。这样的设计使得系统的业务逻辑能够被清晰地组织和管理，易于理解和维护。

（3）业务服务层

封装复杂的业务逻辑，提供服务接口给上层调用，实现业务逻辑的复用和解耦。

设计理由：业务服务层的设计旨在提供可复用的业务逻辑功能，并将其封装为服务接口供上层调用。这样的设计有助于实现业务逻辑的解耦和复用，避免在上层重复编写相同的业务逻辑代码。通过提供清晰的服务接口，业务服务层可以促进团队的协作和开发效率。

3、数据访问层

（1）数据访问接口层

定义对数据的访问接口和规范，包括数据查询、写入、更新等操作。

设计理由：数据访问接口层的设计旨在规范和定义对数据的访问方式和操作。通过抽象出数据访问接口，可以实现数据的独立性和封装性，降低其他层对底层数据库的依赖性。这样的设计使得数据的访问操作更加统一和可控，有利于代码的维护和数据库的切换。

（2）数据访问实现层

具体实现数据访问接口，与数据库进行交互，执行 SQL 语句或使用 ORM（对象关系映射）工具。

设计理由：数据访问实现层的设计目的是将数据访问接口的定义具体实现。通过与数据库进行交互，执行 SQL 语句或使用 ORM 工具，可以有效地操作和管理数据。这样的设计能够将数据的访问操作与具体的数据库实现解耦，提高系统的灵活性和可维护性。

4、数据库层

（1）数据库管理系统

选择和配置适合系统需求的数据库管理系统，如 MySQL、PostgreSQL 等。

设计理由：数据库管理系统的选择和配置对系统的数据存储和管理起着重要的作用。不同的数据库管理系统具有不同的特性和性能，根据系统的需求选择合适的数据库管理系统可以提高数据的存储效率和可靠性。

（2）数据库存储

创建和管理数据库表、字段，确保数据结构的一致性和完整性。

设计理由：数据库存储的设计目的是创建和管理数据库表和字段，确保数据结构的一致性和完整性。通过定义和维护良好的数据库结构，可以保证数据的准确性和可靠性，并提供高效的数据查询和管理能力。

（3）数据库连接池

管理数据库连接的池化资源，提高数据库访问的性能和效率。

设计理由：数据库连接池的设计旨在管理数据库连接的池化资源，以提高数据库访问的性能和效率。通过使用连接池，可以减少数据库连接的创建和销毁开销，重复利用已经建立的连接，提高数据库操作的响应速度和并发处理能力。

4.2 关键模块设计

1、用户认证模块设计流程

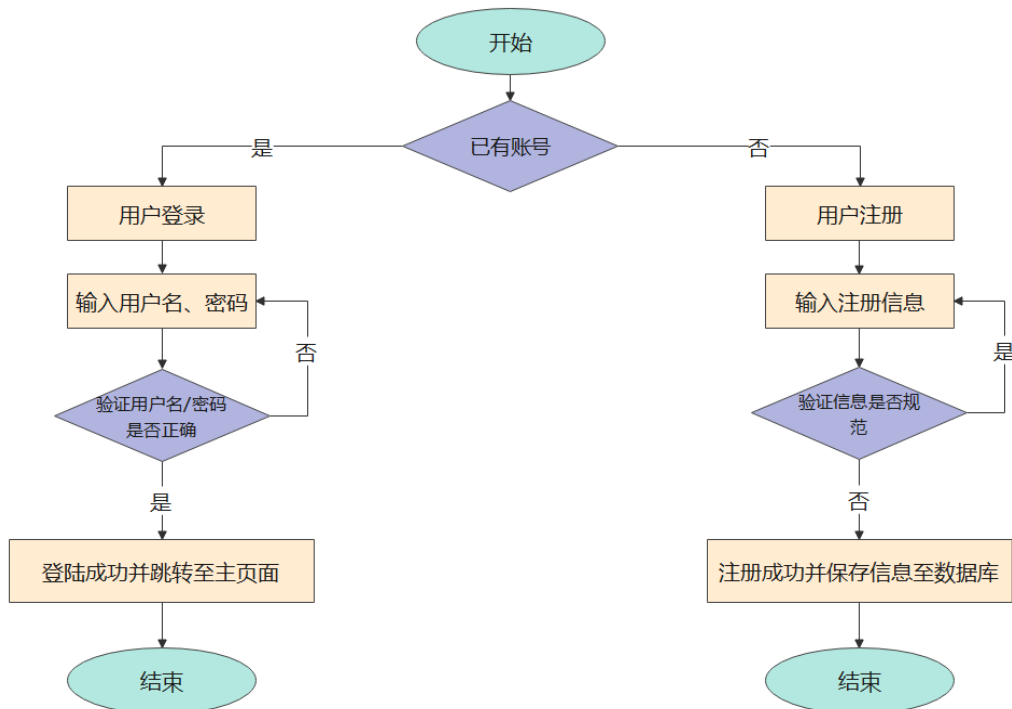


图 4.2 用户认证模块流程图

(1) 用户注册功能设计流程

- ①用户访问注册页面，填写必要的信息，包括用户名、密码等。
- ②后端接收注册请求，验证用户输入的信息是否符合要求，例如用户名是否已被使用，密码是否符合规范等。
- ③如果验证通过，将用户信息存储到数据库中，并生成用户的唯一标识符（例如用户 ID）。
- ④返回注册成功的消息给用户，提示其可以进行登录操作。

(2) 用户登录功能设计流程

- ①用户访问登录页面，输入用户名和密码。
- ②后端接收登录请求，查询数据库验证用户输入的用户名和密码是否匹配。
- ③如果验证通过，生成一个身份验证的令牌（例如用户的登录凭证或者访问令牌），并将其存储在用户的浏览器 cookie 或者 session 中。
- ④返回登录成功的消息给用户，提示其登录成功，并跳转到主页或者之前访问的页面。

2、文章发布模块设计流程

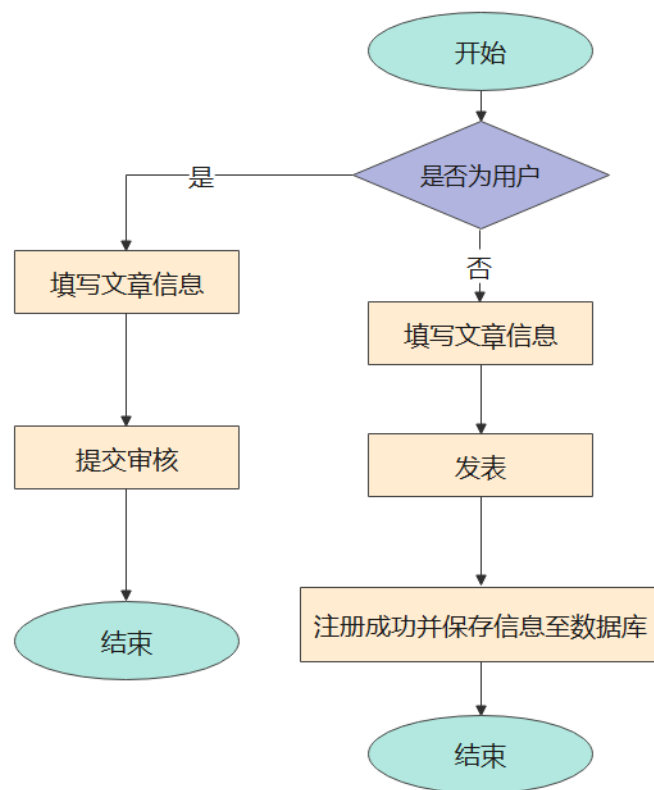


图 4.3 文章发表功能流程图

(1) 文章发表功能设计流程

- ①用户登录后，进入文章发表页面。

- ②用户填写文章的标题、内容等信息，并选择文章所属的版块。
- ③用户提交文章发布请求，后端接收请求，验证用户身份和权限。
- ④如果审核通过，则将文章的信息存储到数据库中，反之，则不存入信息。

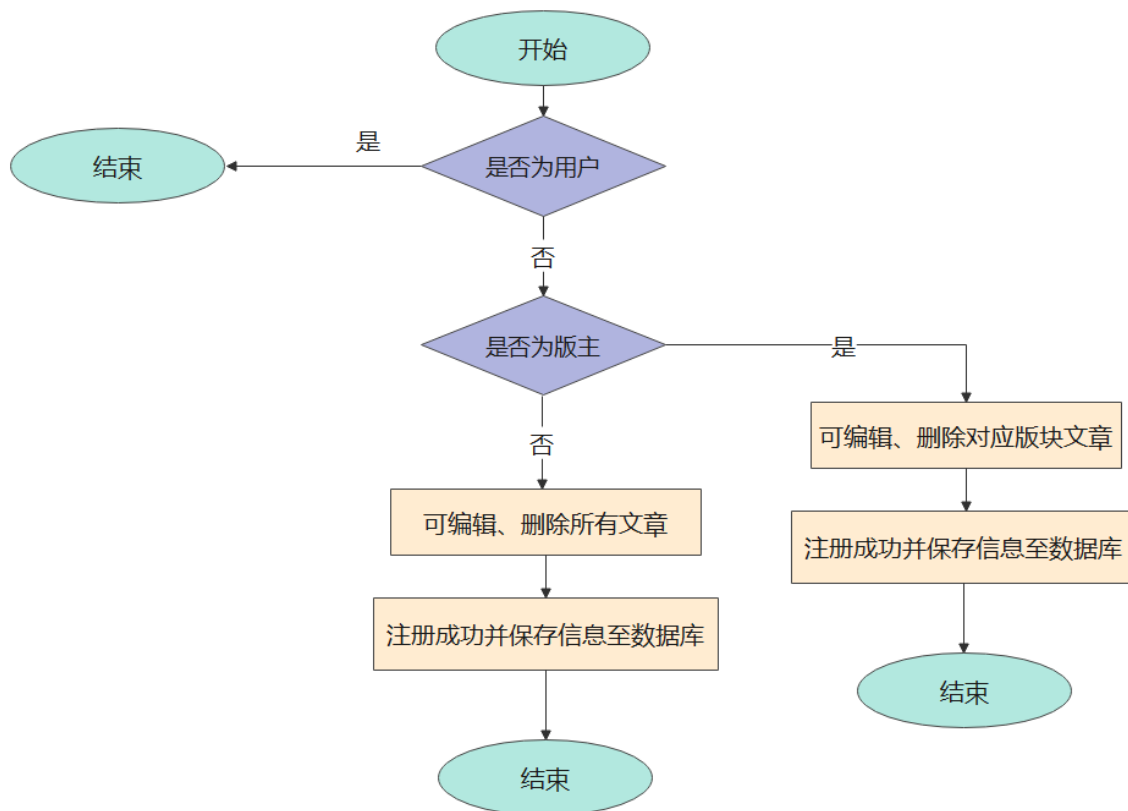


图 4.4 文章编辑、删除功能流程图

(2) 文章编辑功能设计流程

- ①管理员登录后，进入已发布文章的编辑页面。
- ②管理员可以修改文章的标题、内容等信息，并保存修改。
- ③管理员提交文章编辑请求，后端接收请求，验证管理员身份和权限。
- ④如果验证通过，更新数据库中相应的文章信息。
- ⑤返回编辑成功的消息给管理员，提示文章已成功更新。

(3) 文章删除功能设计流程

- ①管理员登录后，进入已发布文章的管理页面。
- ②管理员可以选择要删除的文章，并确认删除操作。
- ③管理员提交文章删除请求，后端接收请求，验证管理员身份和权限。
- ④如果验证通过，从数据库中删除相应的文章信息。
- ⑤返回删除成功的消息给管理员，提示文章已成功删除。

3、留言管理模块设计流程

(1) 留言发表设计流程

- ①用户登录后，在文章页面下方可以看到留言输入框和已有留言列表。
- ②用户填写留言内容并提交留言请求。
- ③后端接收留言请求，验证用户身份和权限。
- ④如果验证通过，将留言信息存储到数据库中，并与对应的文章关联。
- ⑤返回留言成功的消息给用户，提示留言已成功提交。

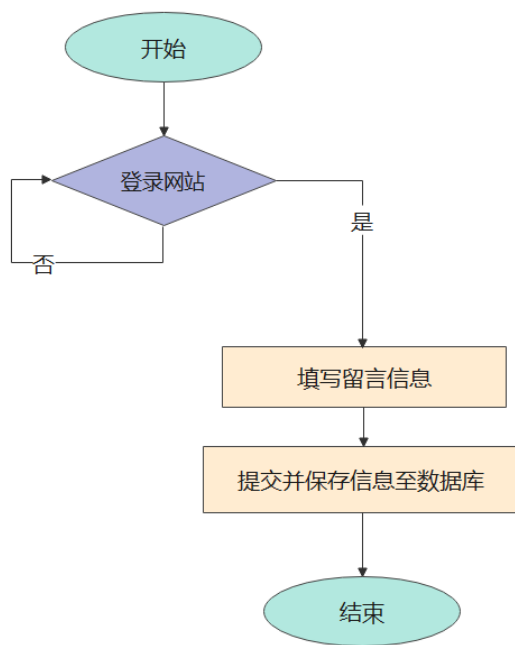


图 4.5 留言发表功能流程图

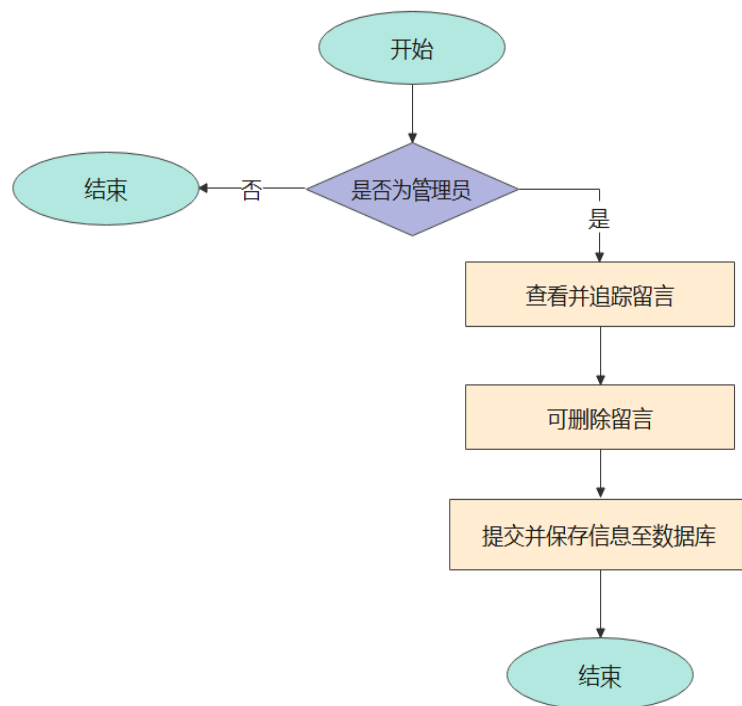


图 4.6 留言删除功能流程图

(2) 留言删除功能设计流程

- ①管理员登录后，进入留言管理页面。
- ②管理员可以查看文章下的留言列表，并选择要删除的留言。
- ③管理员提交留言删除请求，后端接收请求，验证管理员身份和权限。
- ④如果验证通过，从数据库中删除相应的留言信息。
- ⑤返回删除成功的消息给管理员，提示留言已成功删除。

4、用户管理模块设计流程

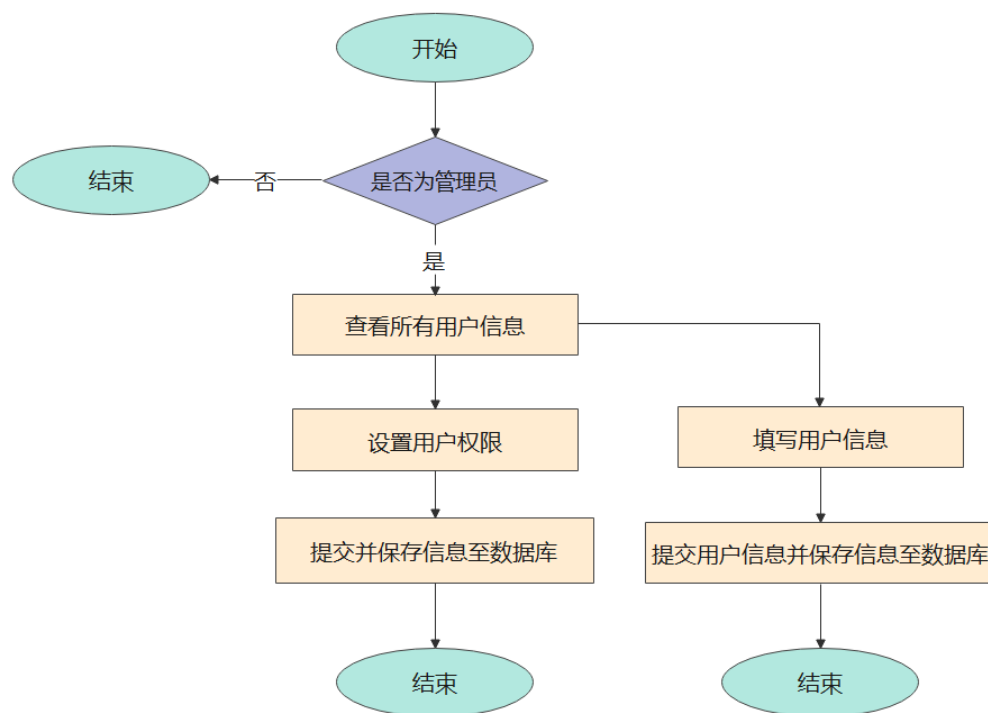


图 4.7 用户管理模块流程图

(1) 用户权限管理功能设计流程

- ①管理员登录后，进入用户管理页面。
- ②管理员可以查看已注册的用户列表，并选择要管理的用户。
- ③管理员可以设置用户的权限，包括普通用户、版主等级别。
- ④后端接收请求，验证管理员身份和权限。
- ⑤如果验证通过，更新数据库中相应用户的权限字段。
- ⑥返回操作成功的消息给管理员，提示权限设置成功。

(2) 用户添加功能设计流程

- ①管理员登录后，进入用户管理页面。
- ②管理员可以点击添加用户按钮，进入用户添加页面。
- ③管理员填写新用户的相关信息，包括用户名、密码等。

- ④管理员可以设置新用户的权限级别。
- ⑤后端接收请求，验证管理员身份和权限。
- ⑥如果验证通过，将新用户的信息保存到数据库中。
- ⑦返回操作成功的消息给管理员，提示用户添加成功。

5、文章管理模块设计流程

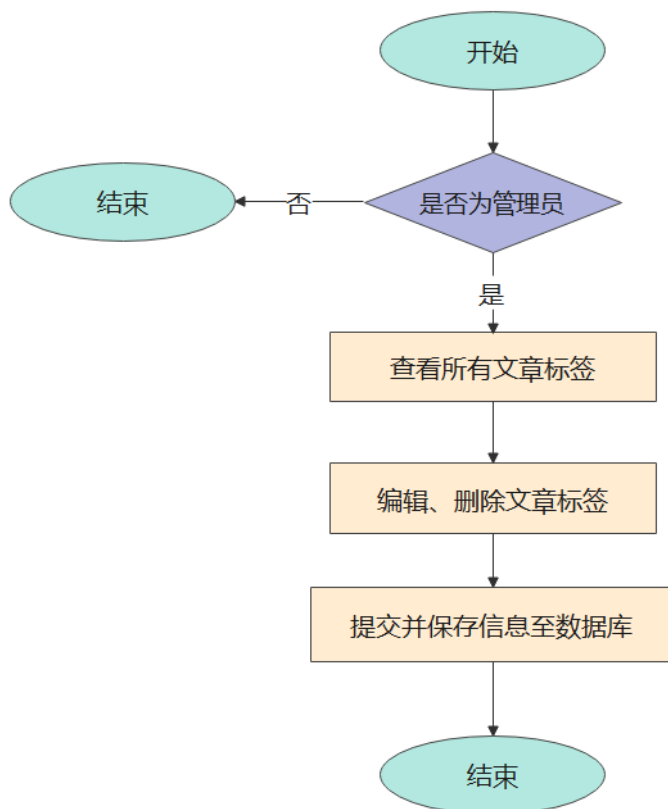


图 4.8 文章标签管理功能流程图

(1) 文章标签管理功能设计流程

- ①管理员进入文章标签管理页面，可以添加、删除和编辑文章标签。
- ②前端向后端发送添加、删除和编辑文章标签的请求。
- ③后端接收请求并对文章标签进行相应的处理，包括添加新的标签、删除已有标签或更新标签信息。
- ④后端将更新后的文章标签信息保存到数据库中，并返回结果给前端进行展示和提示。

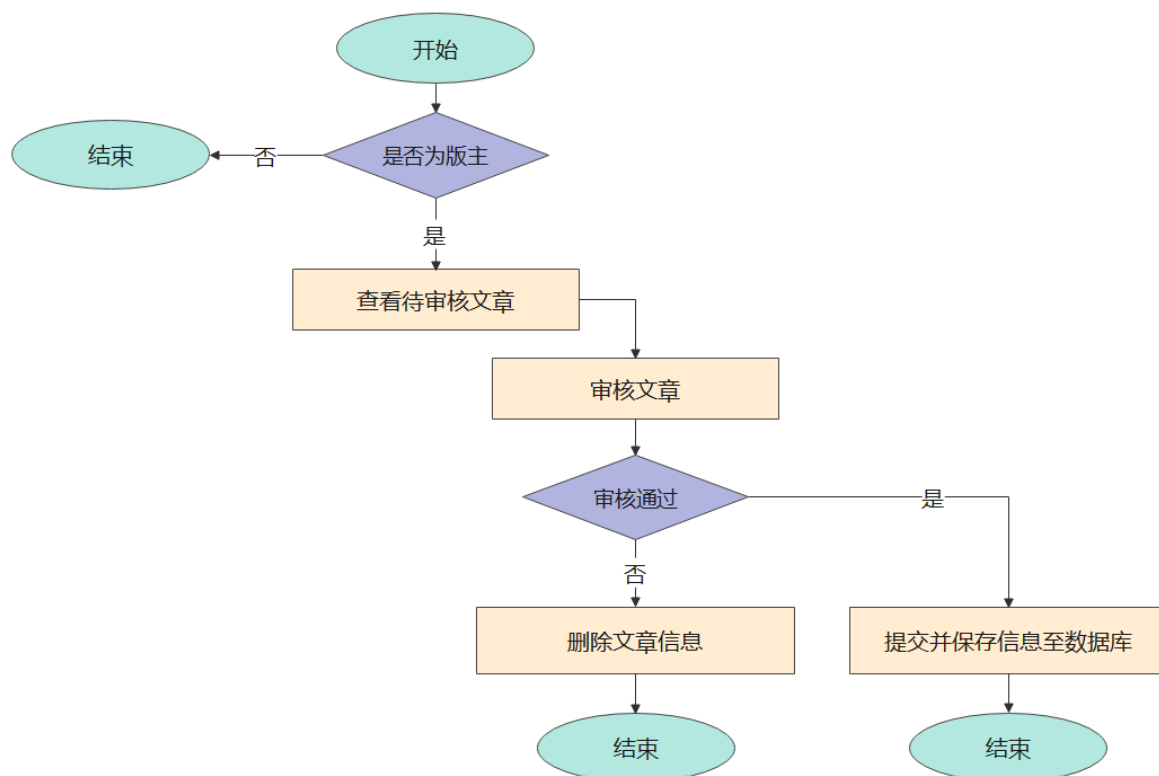


图 4.9 文章审核功能流程图

(2) 文章审核功能设计流程

- ①版主进入文章审核页面，可以查看待审核的文章列表。
- ②前端向后端发送获取待审核文章列表的请求。
- ③后端接收请求并从数据库中获取待审核文章的信息。
- ④后端将待审核文章列表返回给前端进行展示。
- ⑤版主可以浏览待审核文章的内容，然后决定是否通过审核。
- ⑥版主对审核通过的文章可以选择发布，将其展示在网页中。
- ⑦前端向后端发送文章发布请求。
- ⑧后端接收请求并将已审核通过的文章状态更新为已发布状态。
- ⑨后端将更新后的文章状态返回给前端进行展示和提示。

6、查询模块设计流程

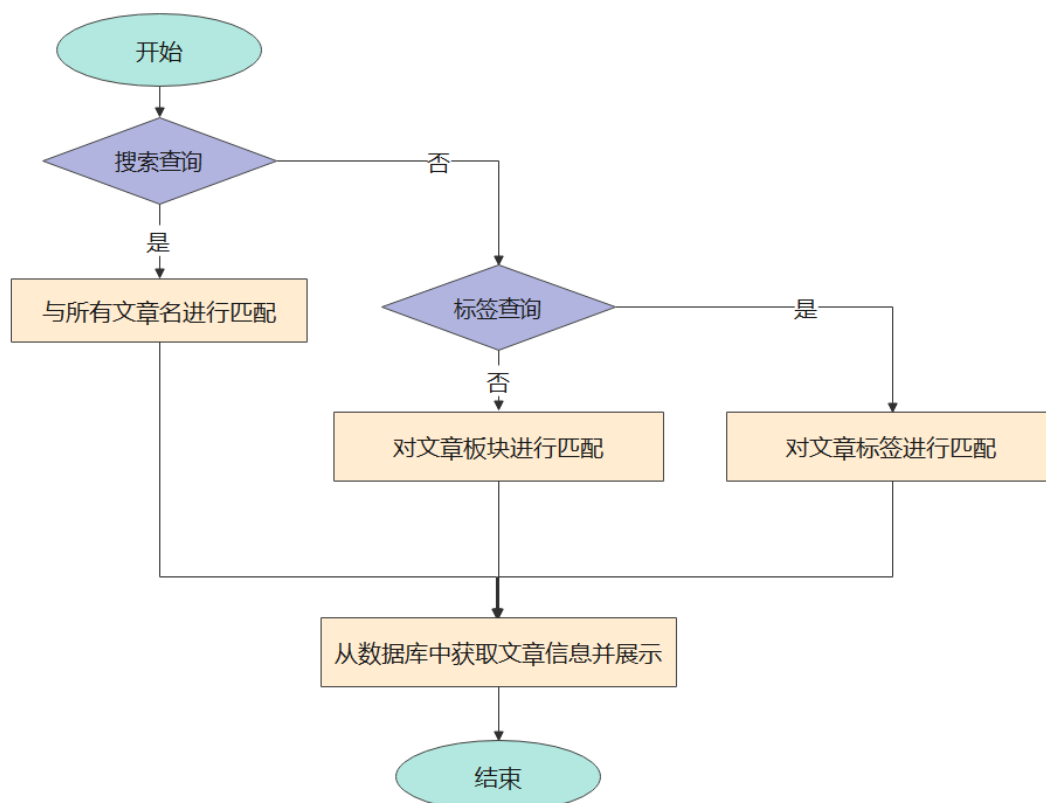


图 4.10 文章查询模块流程图

(1) 文章搜索功能设计流程

- ①用户进入搜索页面或在导航栏中输入搜索关键词。
- ②前端将搜索关键词发送到后端。
- ③后端接收搜索关键词并处理。
- ④后端根据关键词在数据库中进行文章标题或内容的搜索。
- ⑤匹配到的文章结果返回给前端展示给用户。

(2) 文章分类查询功能设计流程

- ①用户进入文章分类页面或选择特定的版块进行浏览。
- ②前端将选定的版块名称或其他分类标准发送到后端。
- ③后端接收请求并根据选择的版块名称或分类标准从数据库中获取相应的文章列表。
- ④返回符合条件的文章列表给前端展示。

(3) 文章标签查询功能设计流程

- ①用户浏览文章页面，每篇文章都标有相应的标签。
- ②用户点击文章标签，前端将所选标签发送到后端。
- ③后端接收请求并根据标签从数据库中获取相关的文章列表。
- ④返回符合标签的文章列表给前端展示。

7、个人信息模块设计流程

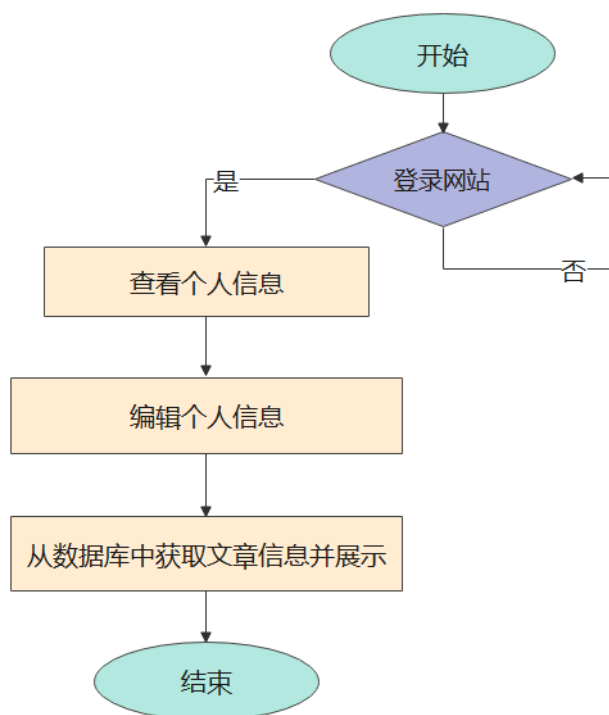


图 4.11 个人信息模块流程图

(1) 个人资料查看功能设计流程

- ①用户登录后，可以通过导航栏或个人信息页面查看自己的个人资料。
- ②前端向后端发送获取个人资料的请求。
- ③后端接收请求并从数据库中获取用户的个人信息。
- ④将用户的个人信息返回给前端进行展示。

(2) 个人资料编辑功能设计流程

- ①用户进入个人信息编辑页面，可以修改个人资料，如用户名、邮箱等。
- ②前端将修改后的个人资料发送到后端。
- ③后端接收请求并更新数据库中用户的个人信息。
- ④返回更新后的个人资料给前端进行展示，并提示用户修改成功。

(3) 个人留言查看功能设计流程

- ①用户进入个人留言页面，可以查看自己发布的留言。
- ②前端向后端发送获取个人留言的请求。
- ③后端接收请求并从数据库中获取用户发布的留言信息。
- ④返回用户的留言信息给前端进行展示。

(4) 用户密码重置功能设计流程

- ①用户进入密码重置页面，填写需要重置的密码和确认密码。

- ②前端将密码重置请求发送到后端。
- ③后端接收请求，对用户的密码进行更新，并将更新结果返回给前端。
- ④前端展示密码重置成功的提示信息，提示用户密码已成功重置。

4.3 数据库设计

1、分类模型（category）

表 4.1 分类表

字段	类型	是否允许为空	说明
id	int	否	主键
name	varchar(128)	否	分类名称
icon	varchar(128)	是	分类描述
add_date	datetime	否	添加时间
pub_date	datetime	否	更新时间

2、文章模型（post）

表 4.2 文章表

字段	类型	是否允许为空	说明
id	int	否	主键
title	varchar(128)	否	文章标题
desc	varchar(200)	是	文章简介
has_type	enum('draft', 'show')	否	文章类型（草稿,发布）
category_id	int	否	归属分类 id
content	longtext	否	文章内容
add_date	datetime	否	添加时间
pub_date	datetime	否	更新时间
user_id	int	否	发布文章的用户 id

3、文章标签（tag）

表 4.3 文章标签表

字段	类型	是否允许为空	说明
id	int	否	主键
name	varchar(128)	否	分类名称
add_date	datetime	否	添加时间
pub_date	datetime	否	更新时间

4、文章与文章标签多对多关系 (tags)

表 4.4 多对多标签表

字段	类型	是否允许为空	说明
tag_id	int	否	文章标签 id
post_id	int	否	文章 id

5、用户关系 (user)

表 4.5 用户表

字段	类型	是否允许为空	说明
inspect	int	是	版主所管理的分类 id
id	int	否	主键
username	varchar(128)	否	用户名
password	varchar(320)	否	密码
desc	varchar(150)	是	个人描述
avatar	varchar(200)	是	头像
gexing	varchar(100)	是	个性签名
gender	varchar(30)	是	性别
email	varchar(10)	是	邮箱
address	varchar(150)	是	住址
is_super_user	boolean	是	是否为超级用户
is_first_user	boolean	是	是否为茶余饭后版主
is_second_user	boolean	是	是否为风花雪月版主
is_third_user	boolean	是	是否为校园故事版主
is_fourth_user	boolean	是	是否为以诗会友版主
is_active_user	boolean	是	是否为活跃用户
is_staff	boolean	是	账户是否锁定
add_date	datetime	否	添加时间
pub_date	datetime	否	更新时间

6、留言模型 (comment)

表 4.6 留言表

字段	类型	是否允许为空	说明
id	int	否	主键
content	varchar(300)	否	留言内容
user_id	int	否	用户 id
post_id	int	否	文章 id
add_date	datetime	否	添加时间
pub_date	datetime	否	更新时间

7、审核文章模型（inspect_post）

表 4.7 审核文章表

字段	类型	是否允许为空	说明
id	int	否	主键
title	varchar(128)	否	留言内容
desc	int	是	文章描述
content	int	否	文章 id
category_id	varchar(200)	否	分类 id
user_id	in	否	用户 id
add_date	datetime	否	添加时间
pub_date	datetime	否	更新时间
is_inspect	bool	否	是否通过审核

4.4 界面设计

1、用户认证界面设计

（1）注册页面：包括用户名、密码、邮箱等注册信息的输入框，以及注册按钮和跳转到登录页面的链接。

（2）登录页面：包括用户名和密码的输入框，以及登录按钮和跳转到注册页面的链接。

2、文章发布界面设计

（1）文章编辑页面：包括文章标题、内容和标签的输入框，以及发布按钮和取消按钮。

（2）文章编辑页面提供富文本编辑器，使用户能够方便地编辑和格式化文章内容。

3、留言管理界面设计

（1）文章页面下方应提供留言输入框，用户可以在此处输入留言内容。

（2）用户留言应显示在留言区域，包括留言内容、留言时间和用户信息等。

（3）对于管理员，留言管理页面应提供删除留言的功能，管理员可以选择删除不当或违规的留言。

4、文章管理界面设计

（1）版块浏览页面：展示各个版块的名称、描述，点击版块可进入该版块下的文章列表。

（2）版块权限管理页面：管理员可以指定版主，并设置版主的权限，如发布、编辑和删除文章等。

(3) 标签管理页面：展示所有的文章标签，管理员可以编辑和删除相应的标签。

5、用户管理界面设计

(1) 用户列表页面：展示系统中的用户列表，包括用户名、权限和注册时间等信息。

(2) 用户添加页面：管理员可以手动添加用户，并设置其权限。

6、查询模块界面设计

(1) 搜索框：提供关键词搜索文章的输入框，用户可以输入关键词进行搜索。

(2) 分类查询页面：展示不同版块的文章列表，用户可以按照版块名称或其他分类标准进行筛选和浏览。

(3) 标签查询页面：显示各个标签，用户可以点击标签快速找到相关主题的文章。

7、个人信息界面设计

(1) 个人资料查看页面：用户可以查看自己的个人信息，包括用户名、邮箱、注册时间等。

(2) 个人资料编辑页面：用户可以编辑和更新个人信息，如修改密码、上传头像等。

(3) 文章发布页面：用户可以编辑文章内容并提交审核，待审核完毕即可知道自己的文章发布详情。

(4) 实现了个人留言查看功能，用户可以查看自己发布的留言，通过留言可以追踪到文章本身。

4.5 本章小结

文学网站系统设计涵盖了用户认证、文章管理、留言管理、版块管理、用户管理、查询和个人信息等关键模块。通过这些模块的设计，各类用户可以进行注册、登录和修改，发布、编辑和删除文章，进行留言和留言删除，管理版块和用户权限，进行文章搜索和分类查询，以及查看和编辑个人信息等操作。系统旨在提供用户友好的界面和功能，保障系统的安全性和数据的完整性，促进用户之间的交流和互动，以及提供个性化的用户体验。

5 系统实现与测试

5.1 系统实现

5.1.1 用户认证模块实现

1、实际系统截图

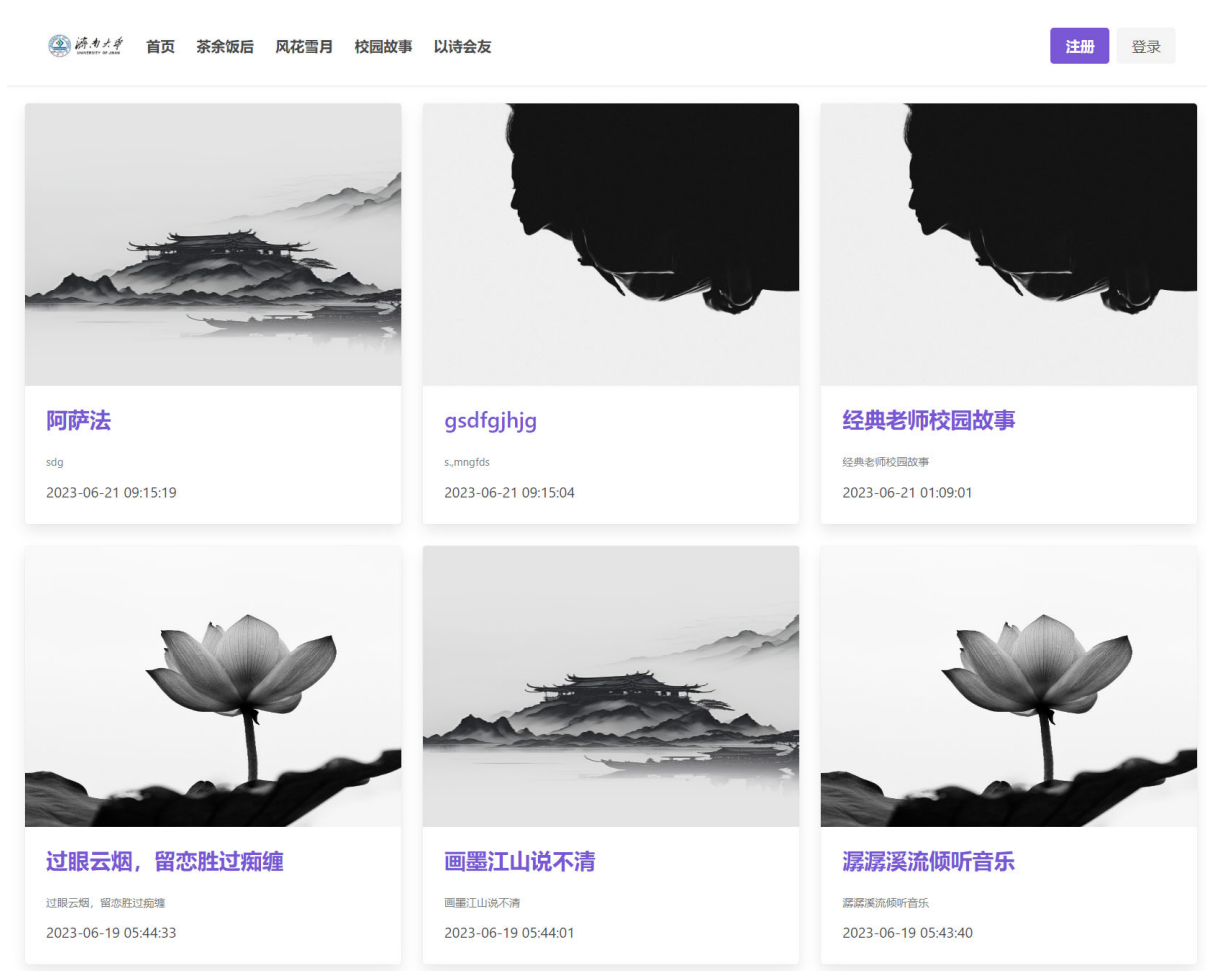


图 5.1 游客界面

登录
文学网站

用户名或密码不正确!

admin

Password

登录

图 5.2 登录验证界面

注册
文学网站

该用户名称已存在!

admin

Password

Password1

注册

图 5.3 注册验证界面

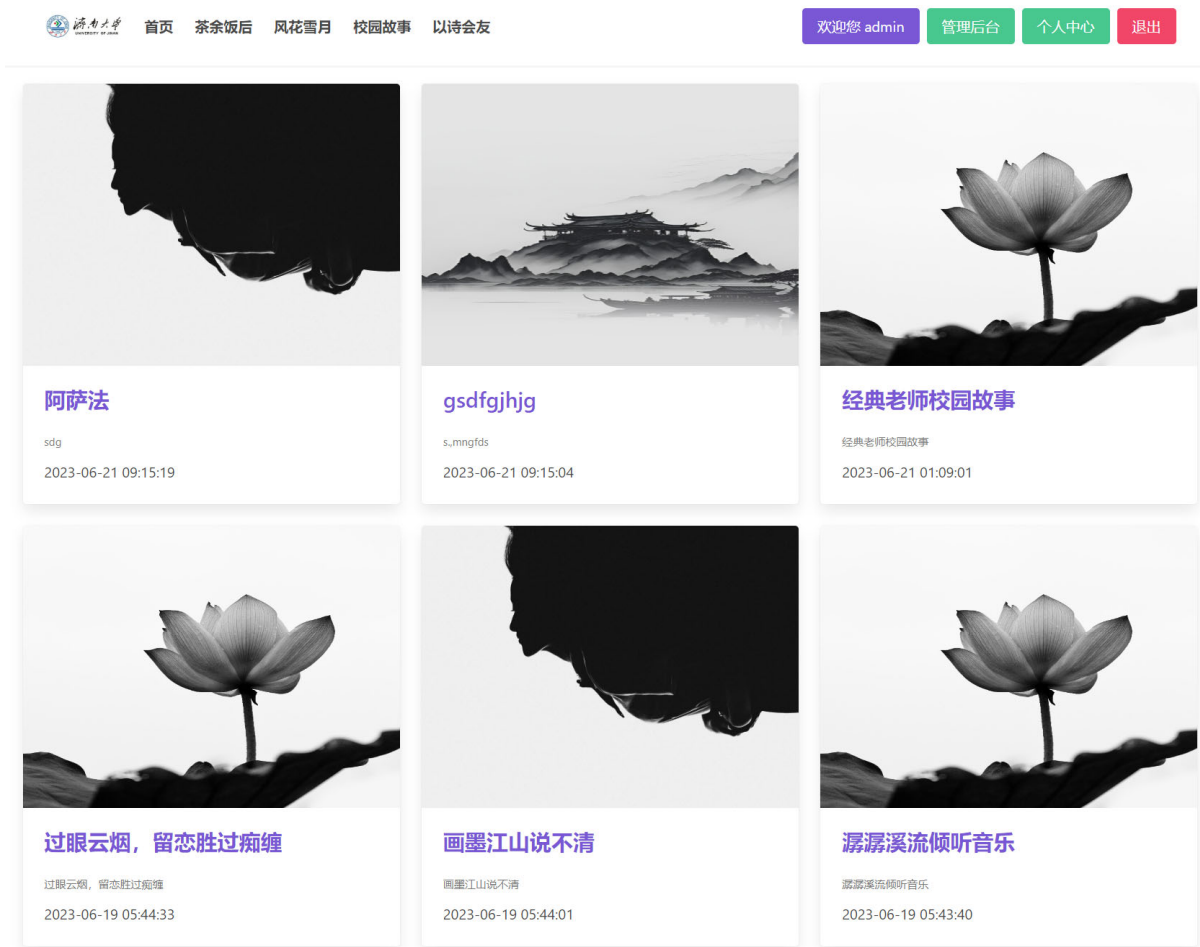


图 5.4 用户界面

2、关键代码

```
class LoginForm(FlaskForm):
    # 登陆表单

    def qs_username(username):
        # 对该字段进行在传递之前处理
        u = f'{username}'
        print(u)
        return username

    username = StringField('username', validators=[
        DataRequired(message="不能为空"),
        Length(max=32, message="不符合字数要求！"),
    ], filters=(qs_username,))
    password = PasswordField('password', validators=[
        DataRequired(message="不能为空"),
        Length(max=32, message="不符合字数要求！"),
    ])
```

```

def validate_username(form, field):
    user = User.query.filter_by(username=field.data).first()
    if user is None:
        error = '该用户不存在！'
        raise ValidationError(error)
    elif not check_password_hash(user.password, form.password.data):
        raise ValidationError('用户名或密码不正确!')

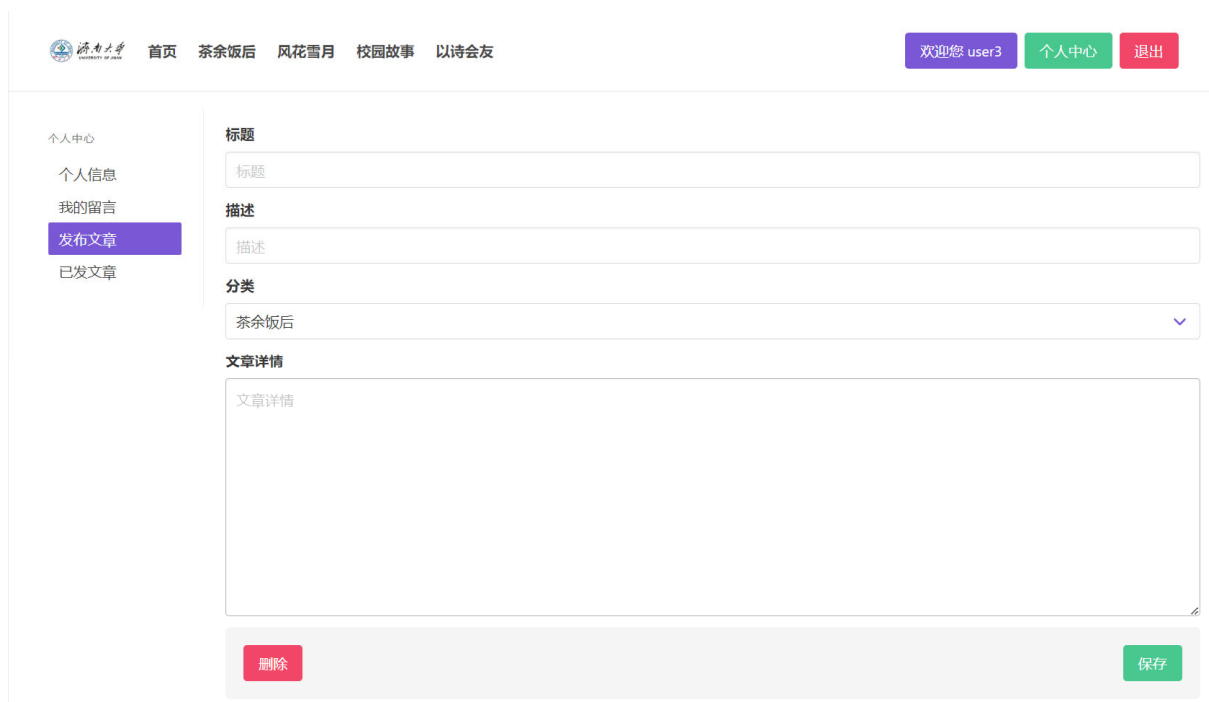
class RegisterForm(FlaskForm):
    # 注册表单
    username = StringField('username', validators=[
        DataRequired(message="不能为空"),
        Length(min=2, max=32, message="超过限制字数！")
    ])
    password = PasswordField('password', validators=[
        DataRequired(message="不能为空"),
        Length(min=2, max=32, message="超过限制字数！"),
        EqualTo('password1', message='两次密码输入不一致！')
    ])
    password1 = PasswordField('password1')

def validate_username(form, field):
    user = User.query.filter_by(username=field.data).first()
    if user is not None:
        error = '该用户名称已存在！'
        raise ValidationError(error)

```

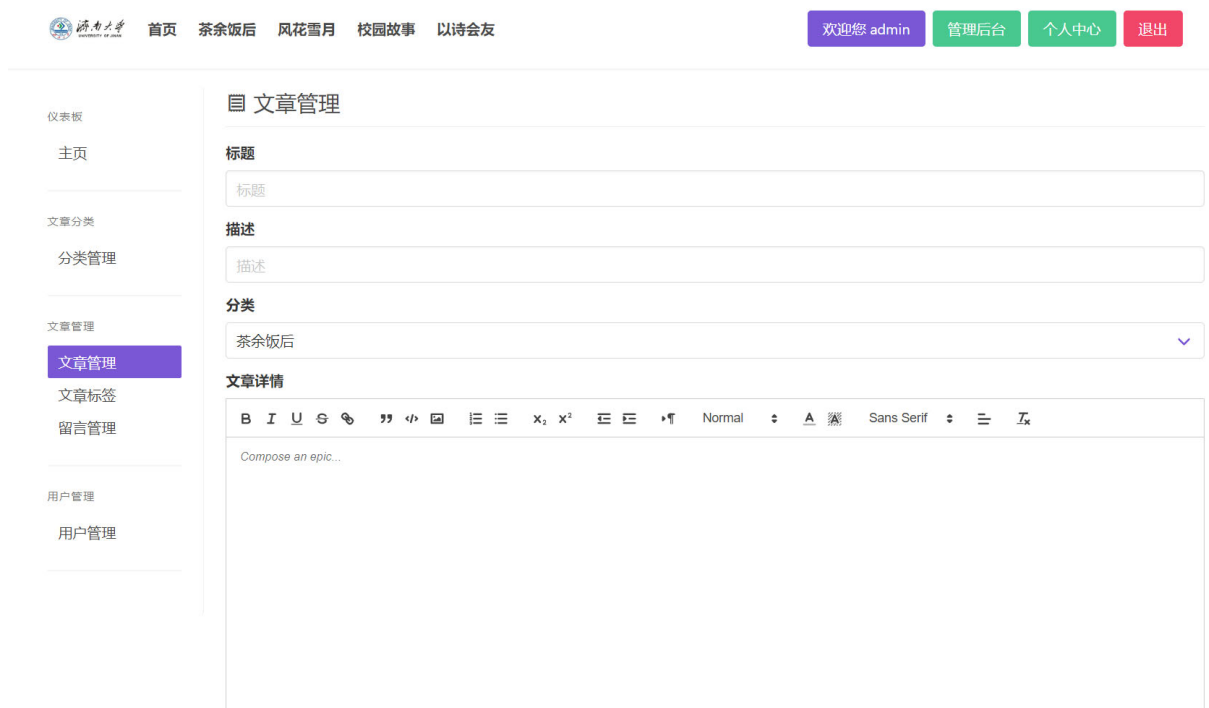
5.1.2 文章发布模块

1、实际系统截图



The screenshot shows the 'User Article Release Interface'. At the top, there is a navigation bar with the university logo, links to '首页', '茶余饭后', '风花雪月', '校园故事', and '以诗会友', and user controls for '欢迎您 user3', '个人中心', and '退出'. On the left, a sidebar menu includes '个人中心', '个人信息', '我的留言', '发布文章' (highlighted), and '已发文章'. The main content area is titled '标题' and contains a text input field. Below it is a '描述' section with another text input field. A '分类' dropdown menu is set to '茶余饭后'. The '文章详情' section is a large text area. At the bottom, there are '删除' and '保存' buttons.

图 5.5 用户文章发布界面



The screenshot shows the 'Admin Article Release Interface'. The top navigation bar is similar to the user interface but includes a '管理后台' button. The left sidebar menu includes '仪表盘', '主页', '文章分类', '分类管理', '文章管理' (highlighted), '文章标签', '留言管理', '用户管理', and '用户管理'. The main content area is titled '文章管理'. It features the same '标题', '描述', and '分类' (set to '茶余饭后') fields as the user interface. The '文章详情' section includes a rich text editor with a toolbar containing various formatting options like bold, italic, underline, link, unlink, text color, background color, font family, font size, and text alignment. The text area contains the placeholder 'Compose an epic...'. At the bottom, there are '删除' and '保存' buttons.

图 5.6 管理员文章发布界面

2、关键代码

```
{% extends 'admin/inspect.html' %}
{% block member %}
<div class="is-block">
  <div class=" is-pulled-left">
    <h1 class=" is-size-4">
      <span class="icon">
        <i class="mdi mdi-receipt-outline"></i>
      </span>
      文章管理
    </h1>
  </div>

  {% block button %}
  <div class=" is-pulled-right">
    <a href="{{ url_for('admin.article_add') }}" class=" button is-primary is-light">
      <span class="icon"><i class="mdi mdi-plus-thick"></i></span>
      <span>添加文章</span>
    </a>
  </div>
  {% endblock button %}
</div class="is-clearfix"></div>
<div class=" dropdown-divider"></div>

<!-- 消息闪现 -->
{% with messages = get_flashed_messages() %}
<b-message type="is-success">
  {% if messages %}
  <ul class="flashes">
    {% for message in messages %}
    <li>{{ message }}</li>
    {% endfor %}
  </ul>
  {% endif %}
</b-message>
{% endwith %}
</div>

{% block table_content %}
<div class="table-container">
  <table class="table is-fullwidth is-hoverable is-striped">
    <thead>
```

```

<tr>
  <th>ID</th>
  <th>标题</th>
  <th>发布状态</th>
  <th>归属分类</th>
  <th>标签</th>
  <th>发布日期</th>
  <th>操作</th>
</tr>
</thead>
<tbody>

  {% for post in post_list %}
  {% if post.category_id == inspect%}
  <tr>
    <td>{{ post.id }}</td>
    <td>{{ post.title }}</td>
    <td>{{ post.has_type.name }}</td>
    <td>{{ post.category.name }}</td>
    <td>{{ post.tags|join(',') }}</td>
    <td>{{ post.add_date }}</td>
    <td>
      <div class="tags">
        <a href="{{ url_for('admin.article_edit', post_id=post.id) }}" class="tag
is-success is-light">
          <span class="icon">
            <i class="mdi mdi-square-edit-outline"></i>
          </span>
          编辑
        </a>
        <a href="{{ url_for('admin.article_del', post_id=post.id) }}" class="tag
is-danger is-light" onclick="return confirmAction();">
          <span class="icon">
            <i class="mdi mdi-trash-can-outline"></i>
          </span>
          删除
        </a>
      <script>
        function confirmAction() {
          return confirm("确定要执行此操作吗? ");
        }
      </script>
    </td>
  </tr>
  {% endfor %}

```

```

        </div>
    </td>
</tr>
{% endif %}
{% endfor %}

</tbody>
</table>
</div>
<nav class="pagination is-small" role="navigation" aria-label="pagination">
    {% if pagination.has_prev %}
    <a href="{{ url_for('admin.article_alter') }}"?page={{ pagination.prev_num }}" class="pagination-上
    一页" title="This is the first page">上一页</a>
    {% endif %}
    {% if pagination.has_next %}
    <a href="{{ url_for('admin.article_alter') }}"?page={{ pagination.next_num }}"
    class="pagination-next">下一页</a>
    {% endif %}

    <ul class="pagination-list">
        {% for page in pagination.iter_pages() %}
            {% if page %}
                {% if page != pagination.page %}
                    <li>
                        <a href="{{ url_for('admin.article_alter') }}"?page={{ page }}"
                        class="pagination-link" aria-label="Page 1" aria-current="page">{{ page }}</a>
                    </li>
                {% else %}
                    <li>
                        <a class="pagination-link is-current" aria-label="Page 1"
                        aria-current="page">{{ page }}</a>
                    </li>
                {% endif %}
            {% else %}
                <span class="pagination-ellipsis">&hellip;</span>
            {% endif %}
        {% endfor %}
    </ul>
</nav>

{% endblock table_content %}
{% endblock member %}

```

5.1.3 留言管理模块

1、系统实际截图

留言内容

发表留言

请文明发言，为了方便联系作者，可以留下你的email。（为避免刷屏，留言最多200字）

提交

图 5.7 留言发表界面

[首页](#)
[茶余饭后](#)
[风花雪月](#)
[校园故事](#)
[以诗会友](#)

[欢迎您 admin](#)
[管理后台](#)
[个人中心](#)
[退出](#)

[仪表盘](#)
[主页](#)
[文章分类](#)
[分类管理](#)
[文章管理](#)
[文章管理](#)
[文章标签](#)
[留言管理](#)
[用户管理](#)
[用户管理](#)

留言管理

admin于2023-06-21 09:32:38对文章《牡丹亭前，人间天堂》发表如下留言:	查看该文章	删除
vd		
user2于2023-06-21 01:50:30对文章《牡丹亭前，人间天堂》发表如下留言:	查看该文章	删除
翁你哦我		
admin于2023-06-19 06:50:49对文章《诗酒共孤影》发表如下留言:	查看该文章	删除
无法访问QAQ		
admin于2023-06-19 06:50:11对文章《诗酒共孤影》发表如下留言:	查看该文章	删除
诗酒趁年华，共孤影为伴。独坐幽篁里，弹琴复长啸。深林人不知，明月来相照。高山流水处，白云常陪伴。咏古怀今者，意气自如霄。浩气凌苍穹，心境超尘表。豁然开朗间，万象皆为好。		

图 5.8 留言管理界面

2、关键代码

```
{% extends 'admin/index.html' %}

{% block member %}
<div class="is-block">
  <div class=" is-pulled-left">
    <h1 class=" is-size-4">
      <span class="icon">
        <i class="mdi mdi-receipt-outline"></i>
      </span>
    </h1>
  </div>
</div>
```

留言管理

```

</h1>
</div>

{% block button %}
{% endblock button %}
<div class="is-clearfix"></div>
<div class="dropdown-divider"></div>

<!-- 消息闪现 -->
{% with messages = get_flashed_messages() %}
<b-message type="is-success">
    {% if messages %}
    <ul class="flashes">
        {% for message in messages %}
        <li>{{ message }}</li>
        {% endfor %}
    </ul>
    {% endif %}
</b-message>
{% endwith %}
</div>

{% block table_content %}
{% if comment_list %}

{% for comment in comment_list %}
<article class="media">
    <div class="media-content">
        <div class="content">
            <p>
                <strong>{{ comment.user.username }}于{{ comment.add_date }}</strong>对文章
            <small>《 {{ comment.post.title }} 》</small>
            <small>发表如下留言:</small>
            <br>
            {{ comment.content }}
        </p>
    </div>
</div>
<div class="media-right">
    <a href="{{ url_for('blog.detail', cate_id=comment.post.category.id,
post_id=comment.post.id) }}"#comment"

```



```

class="tag is-success is-light">查看该文章</a>

<a href="{{ url_for('admin.comment_del', comment_id=comment.id) }}" class="tag is-danger
is-light" onclick="return confirmAction();">
    <span class="icon">
        <i class="mdi mdi-trash-can-outline"></i>
    </span>
    删除
</a>
<script>
    function confirmAction() {
        return confirm("确定要执行此操作吗? ");
    }
</script>
</div>
</article>
{% endfor %}

{% else %}
暂无任何留言
{% endif %}

<div class="dropdown-divider"></div>
<nav class="pagination is-small mt-3" role="navigation" aria-label="pagination">
    {% if pagination.has_prev %}
    <a href="{{ url_for('admin.comment') }}"?page={{ pagination.prev_num }}" class="pagination-上一
    页"
        title="This is the first page">上一页</a>
    {% endif %}
    {% if pagination.has_next %}
    <a href="{{ url_for('admin.comment') }}"?page={{ pagination.next_num }}" class="pagination-next">
    下一页</a>
    {% endif %}

    <ul class="pagination-list">
        {% for page in pagination.iter_pages() %}
        {% if page %}
        {% if page != pagination.page %}
        <li>
            <a href="{{ url_for('admin.comment') }}"?page={{ page }}" class="pagination-link"
            aria-label="Page 1"
                aria-current="page">{{ page }}</a>

```

```

        </li>
        {% else %}
        <li>
            <a class="pagination-link is-current" aria-label="Page 1"
aria-current="page">{{ page }}</a>
        </li>
        {% endif %}
        {% else %}
        <span class=pagination-ellipsis>&hellip;</span>
        {% endif %}
        {% endfor %}
    </ul>
</nav>

{% endblock table_content %}
{% endblock member %}

```

5.1.4 用户管理模块

1、系统实际截图

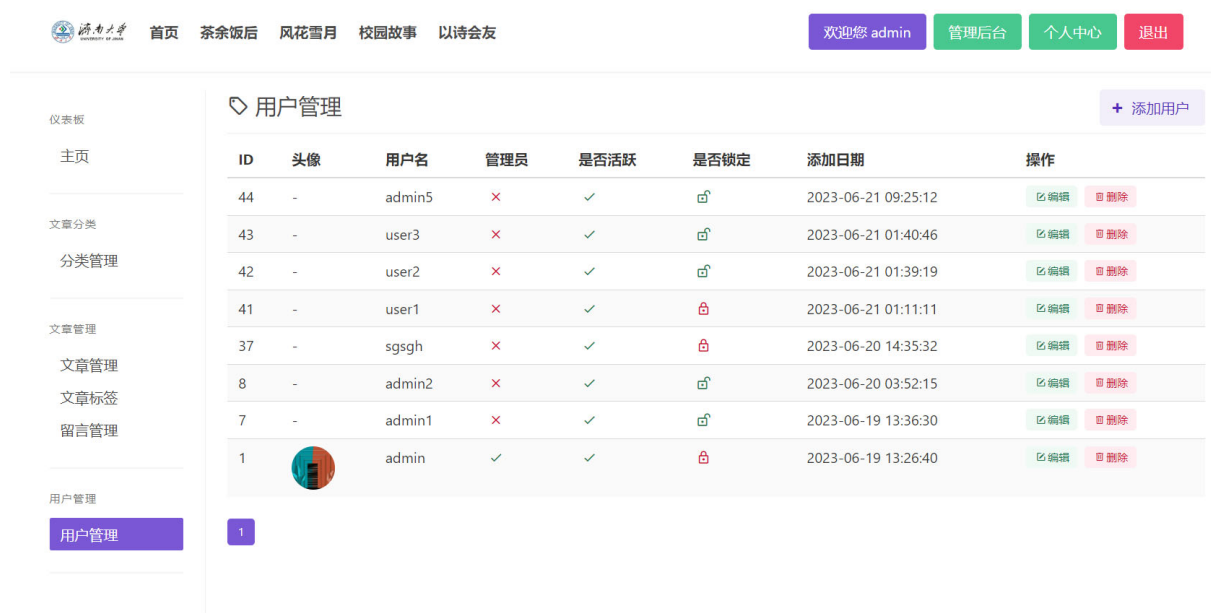


图 5.9 用户管理界面

2、关键代码

```

{% block member %}
<div class="tile is-ancestor">
    <div class="tile is-parent">
        <article class="tile is-child notification is-info is-light">

```

```

        <div class="content">
            <p class="title">{{ post_count }}</p>
            <p class="subtitle">文章数</p>
            <div class="content">
                <!-- Content -->
            </div>
        </div>
    </div>
</div>
<div class="tile is-parent">
    <article class="tile is-child notification is-success is-light">
        <div class="content">
            <p class="title">{{ user_count }}</p>
            <p class="subtitle">用户数</p>
            <div class="content">
                <!-- Content -->
            </div>
        </div>
    </article>
</div>
<div class="tile is-parent">
    <article class="tile is-child notification is-warning is-light">
        <div class="content">
            <p class="title">0</p>
            <p class="subtitle">留言数</p>
            <div class="content">
                <!-- Content -->
            </div>
        </div>
    </article>
</div>
</div>
{% endblock member %}

```

5.1.5 文章管理模块

1、系统实际截图

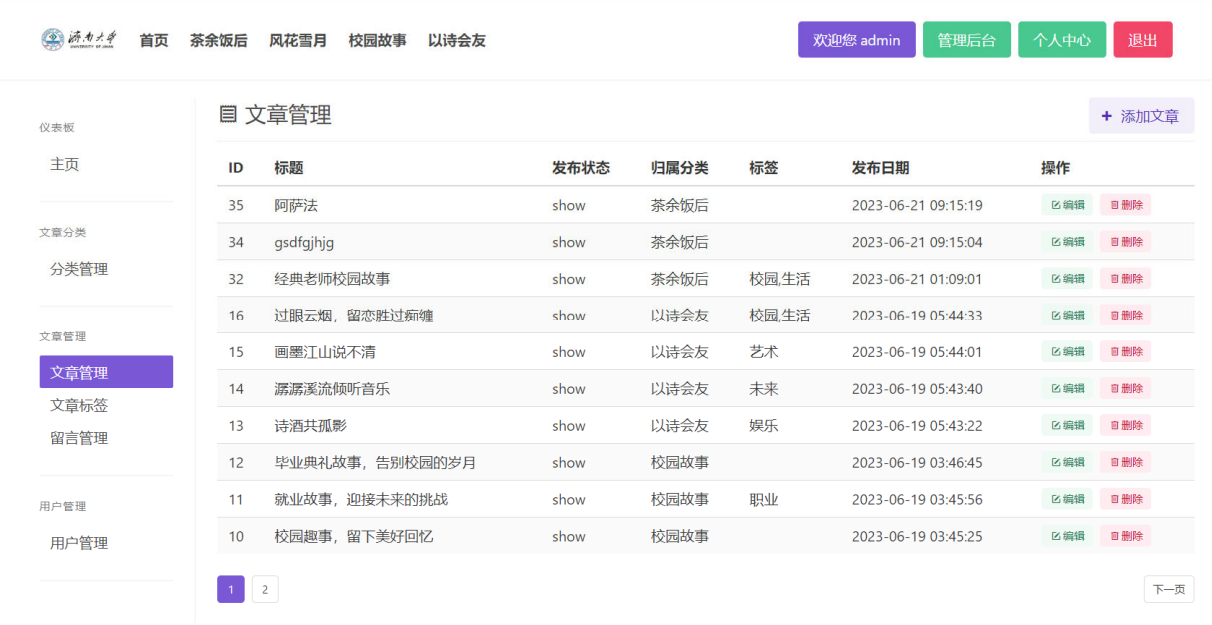


图 5.10 文章管理界面

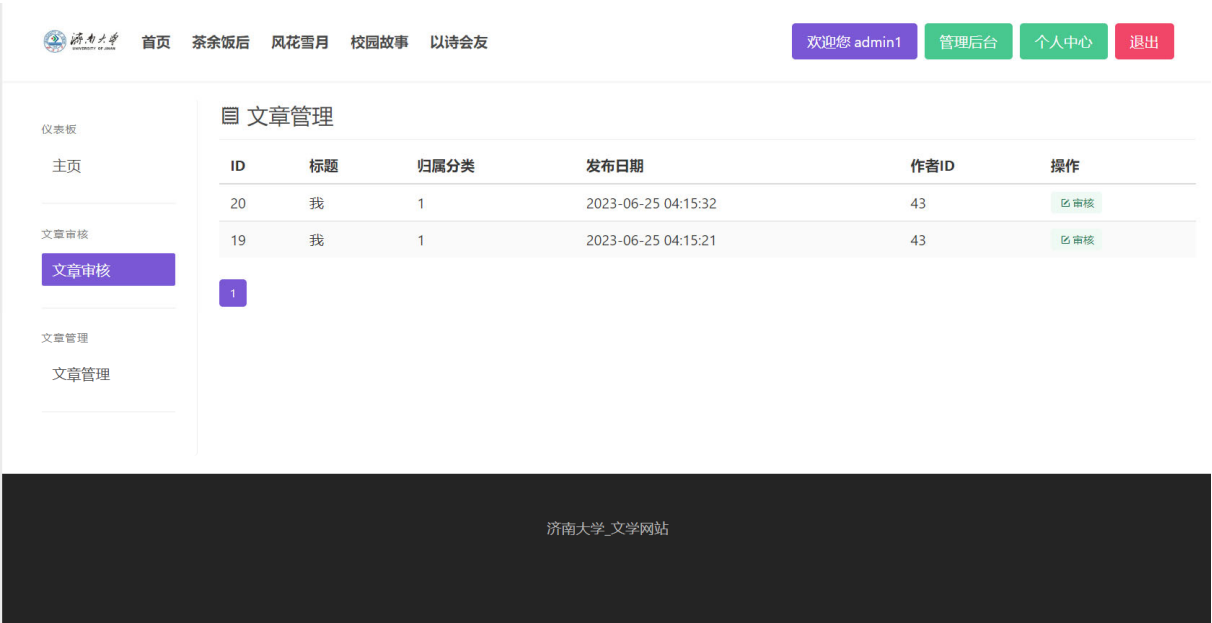


图 5.11 文章审核界面

2、关键代码

```
{% block member %}
<div class="tile is-ancestor">
  <div class="tile is-parent">
    <article class="tile is-child notification is-info is-light">
      <div class="content">
        <p class="title">{{ post_count }}</p>
      </div>
    </article>
  </div>
</div>
```

```

        <p class="subtitle">文章数</p>
        <div class="content">
            <!-- Content -->
        </div>
    </div>
</article>
</div>
<div class="tile is-parent">
    <article class="tile is-child notification is-success is-light">
        <div class="content">
            <p class="title">{{ user_count }}</p>
            <p class="subtitle">用户数</p>
            <div class="content">
                <!-- Content -->
            </div>
        </div>
    </article>
</div>
<div class="tile is-parent">
    <article class="tile is-child notification is-warning is-light">
        <div class="content">
            <p class="title">0</p>
            <p class="subtitle">留言数</p>
            <div class="content">
                <!-- Content -->
            </div>
        </div>
    </article>
</div>
</div>
{% endblock member %}

```

5.1.6 查询模块

1、实际系统截图



图 5.12 版块查询界面

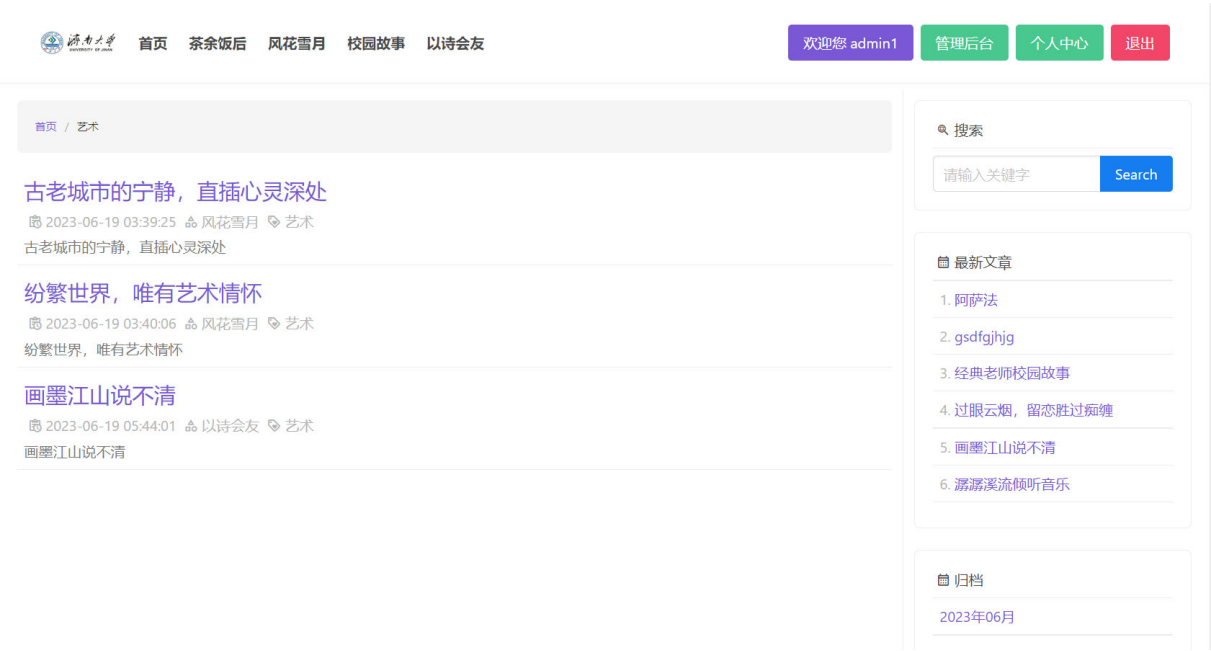


图 5.13 标签查询界面

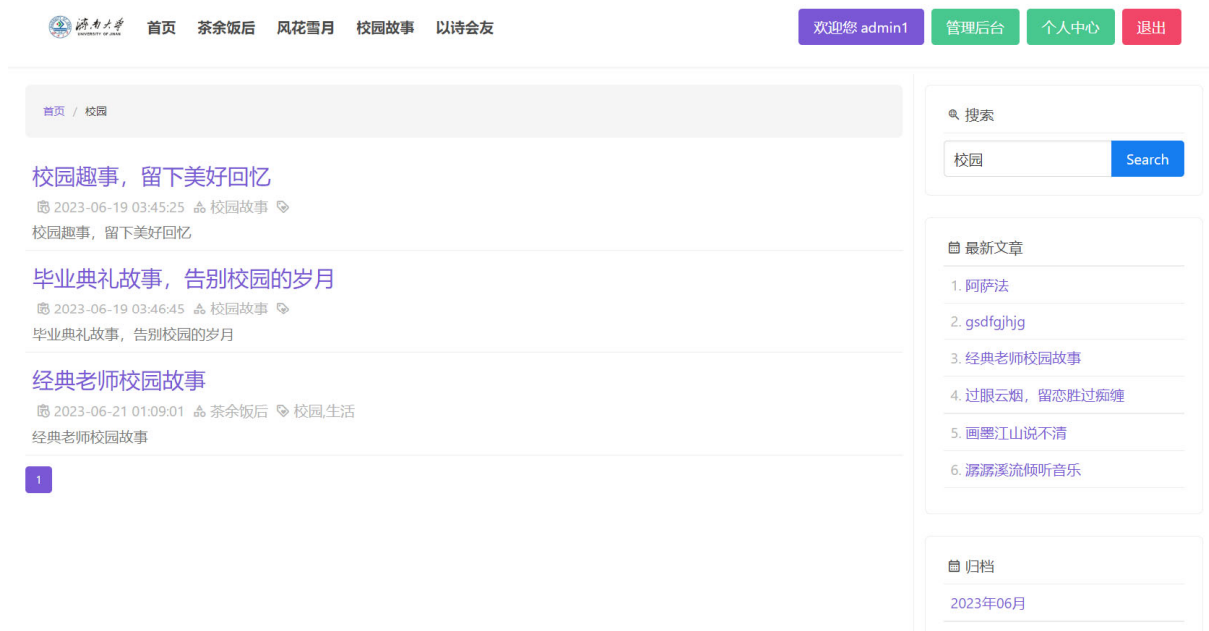


图 5.14 关键字查询界面

2、关键代码

```
@bp.route('/tags/<int:tag_id>')
def tags(tag_id):
    # 标签页
    tag = Tag.query.get(tag_id)
    return render_template('tags.html', post_list=tag.post, tag=tag)

@bp.route('/search')
def search():
    # 搜索页
    words = request.args.get("words")
    page = request.args.get('page', 1, type=int)
    pagination = Post.query.filter(
        Post.title.like("%"+words+"%")).paginate(
            page, per_page=10, error_out=False)
    post_list = pagination.items
    return render_template('search.html', post_list=post_list, words=words, pagination=pagination)
```

5.1.7 个人信息模块

1、实际系统截图

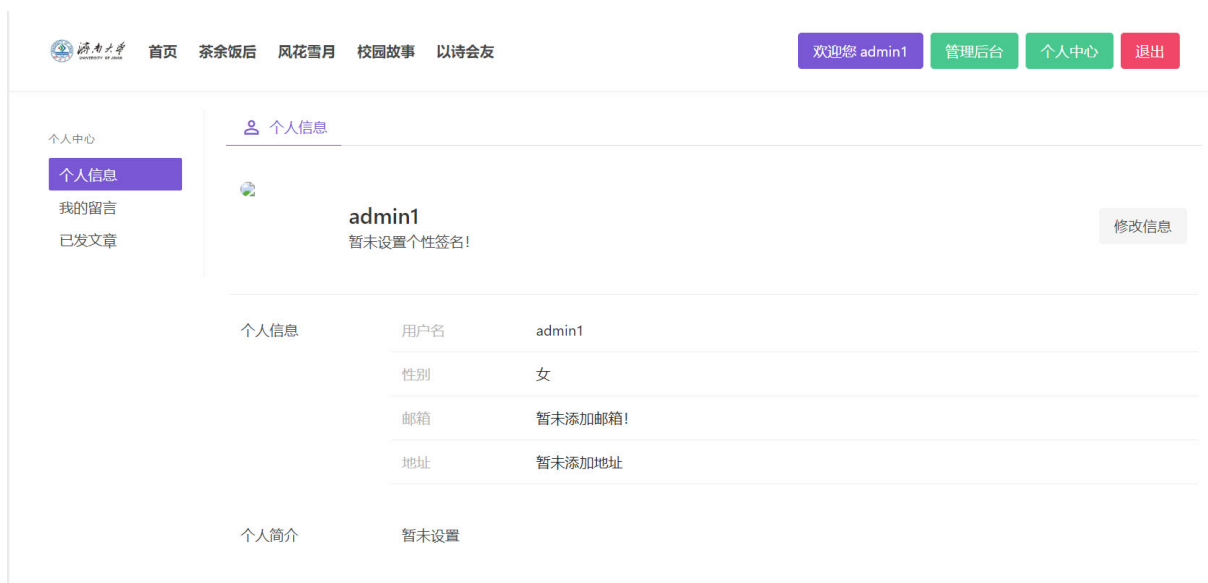


图 5.15 个人信息界面

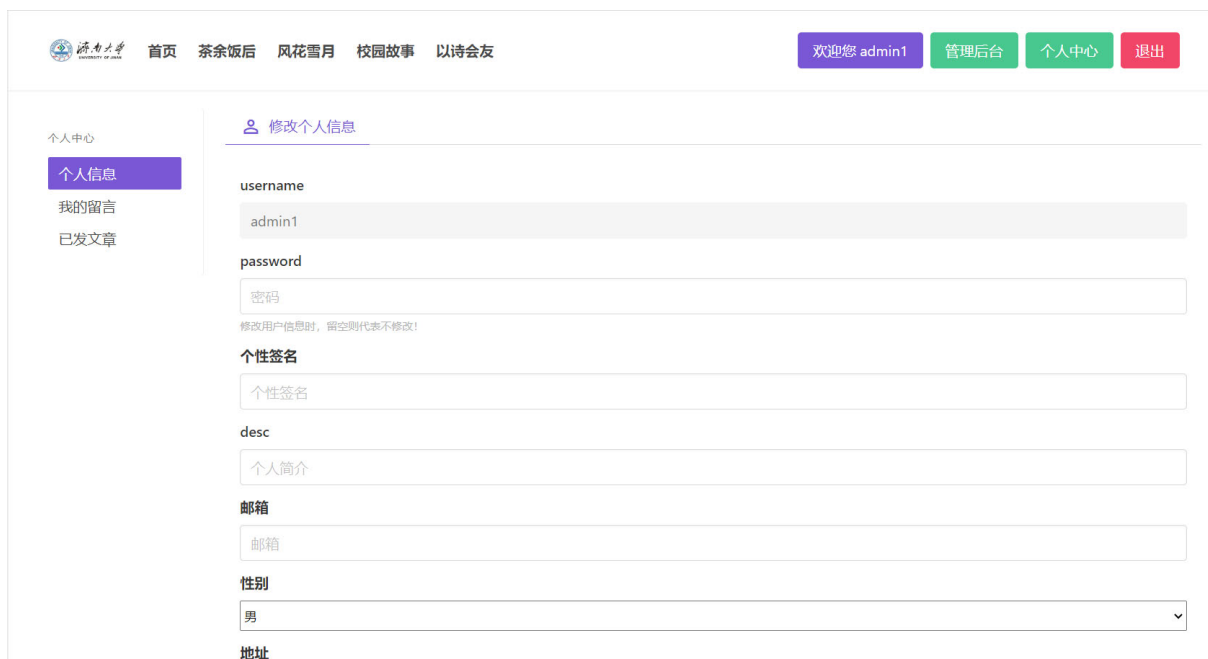


图 5.16 个人信息编辑界面



图 5.17 留言查看界面

2、关键代码

```
@bp.route('/userinfo/edit', methods=['GET', 'POST'])
@login_required
def userinfo_edit():
    # 修改个人信息
    user = User.query.get(g.user.id)
    form = CreateUserForm(
        username=user.username,
        password=user.password,
        avatar=user.avatar,
        gexing=user.gexing,
        desc=user.desc,
        email=user.email,
        gender=user.gender,
        address=user.address
    )
    if form.validate_on_submit():
        # user.username = form.username.data
        # 修改如下则代表不允许修改用户名
        user.username = g.user.username
        if not form.password.data:
            user.password = user.password
        else:
            user.password = generate_password_hash(form.password.data)
        # 考虑，这里如果修改了密码就应该清理 session 中的用户信息，使其重新登录
        session.clear()
    from app.admin.utils import upload_file_path
    f = form.avatar.data
    if user.avatar == f:
```

```

        user.avatar = user.avatar
    else:
        avatar_path, filename = upload_file_path('avatar', f)
        f.save(avatar_path)
        user.avatar = f'avatar/{filename}'

    # 不允许修改自己的状态
    user.gexing=form.gexing.data
    user.desc=form.desc.data
    user.email=form.email.data
    user.gender=form.gender.data
    user.address=form.address.data

    db.session.add(user)
    db.session.commit()
    flash(f'{user.username} 修改成功! ')
    return redirect(url_for('auth.userinfo'))
return render_template('userform.html', form=form)

```

5.2 系统测试

5.2.1 功能测试

功能测试主要涉及到添加用户、发布文章、发表留言、添加标签、修改个人信息等。

1、添加用户

user4添加成功!		
ID	头像	用户名
46	-	user4

图 5.18 用户添加成功提示

2、发布文章

茶添加成功	
ID	标题
36	茶

图 5.19 发布文章成功提示

3、发表留言

留言内容

留言成功!

用户 **admin** 于2023-06-25 05:54:42 发布留言:

茶馆喝茶的日子真的很快乐!

图 5.20 发表留言成功提示

4、添加标签

回忆添加成功

ID	标签名称
14	回忆

图 5.21 添加标签成功提示

5、修改个人信息

admin1修改成功!



admin1
你好

个人信息	用户名	admin1
	性别	男
	邮箱	aiga@nimuns.buzz
	地址	济南大学

图 5.22 个人信息修改成功提示

5.2.2 出错处理测试

1、用户名重复测试

分类管理

admin重复的用户名!

username

用户名

admin重复的用户名!

图 5.23 重复用户提示

2、标签重复测试

分类管理

往事 重复标签名!

标签

标签名称

图 5.24 重复标签提示

3、分类重复测试

以诗会友 分类名重复

分类名称

分类名称

分类描述

分类描述

图 5.25 重复分类提示

6 结语与展望

6.1 结语

通过我开发的基于 Flask 和 MySQL 的文学网站系统设计，我成功实现了一个用户友好、功能丰富的平台，供用户浏览、搜索和发布文学作品，并与其他用户进行交流和互动。在这个过程中，我运用了软件工程的原理和方法，从需求分析、设计、开发到测试和部署，确保系统的质量和可维护性。

在软件工程的框架下，我注重了系统的可扩展性和可维护性。通过采用 Flask 框架，我实现了一个模块化的系统结构，使得功能的添加和修改更加灵活和高效。同时，我合理设计了数据库结构，以支持系统数据的可靠存储和高效访问。

6.2 展望

尽管我已经实现了一个可行的文学网站系统，但在软件工程的实践中，仍始终有进一步改进和提升的空间。未来，我将继续关注系统的可维护性，采用合适的软件工程方法和工具进行代码管理和版本控制，来更好地维护系统并支持功能的持续迭代开发。

此外，我也将继续注重系统的质量保证，进一步完善测试策略和流程，以确保系统的稳定性和安全性。我将积极收集用户反馈和需求，进行持续的用户体验改进和功能优化，从而提供更好的服务。

在软件工程的指导下，我将持续关注行业的发展动态和新兴技术，如人工智能、大数据等，结合这些技术和趋势，不断拓展系统的功能和创新性，为用户带来更加丰富和多样化的文学交流体验。

总的来说，我对我所开发的基于 Flask 和 MySQL 的文学网站系统设计感到满意，并对未来充满信心。我将继续在软件工程的框架下不断改进和优化系统，致力于打造一个高质量、可靠且用户满意的文学交流平台。

7 参考文献

- [1] 常佳宁,李阳齐.基于 Django 的个人博客系统设计开发[J].中国科技信息,2021,No.644(02):75-77.
- [2] 涂远杰,郑剑.基于 Flask 的博客网站设计与实现[J].电脑知识与技术,2020,16(15):109-111.DOI:10.14004/j.cnki.ckt.2020.1762.
- [3] 刘磊.基于 Web 框架的博客管理系统设计与实现[J].计算机时代,2017,No.299(05):20-23.DOI:10.16644/j.cnki.cn33-1094/tp.2017.05.006.
- [4] 钟怡旻,郭昱君.基于 Springboot 的博客管理系统设计与实现[J].现代信息技术,2021,5(07):18-20+24.DOI:10.19850/j.cnki.2096-4706.2021.07.005.
- [5] 王妍.博客系统的概要设计[J].硅谷,2011(16):47-48.
- [6] 王向东. Blog 系统的设计与实现[D].电子科技大学,2009.
- [7] 黄文旭,杨艳红.“全映苏应校园说”校园博客系统数据库的设计[J].现代信息技术,2020,4(10):101-103.DOI:10.19850/j.cnki.2096-4706.2020.10.033.
- [8] 杨嘉群.基于 JSP 的博客系统[J].电子制作,2013(09):68-69+32.DOI:10.16589/j.cnki.cn11-3571/tn.2013.09.189.
- [9] 谢作想.知识博客系统的开发与研究[D].北京化工大学,2008.
- [10] 薛晋炜.属性基加密在博客系统中的研究与应用[D].山西大学,2015.
- [11] 杨帆,林勇,胡秀兵.基于 MVC 模式个人博客系统的设计与实现[J].计算机时代,2014(01):23-26.
- [12] 谭凯.博客系统(个人信息共享和思想感情交流平台)[J].数字技术与应用,2013(06):212.DOI:10.19695/j.cnki.cn12-1369.2013.06.147.