

智能图书推荐系统

**组长姓名：张俊彪 学号：33920212204593**

**组员姓名：李涵晨 学号：11920222202381**

**组员姓名：周冠旭 学号：30320212200099**

**完成日期： 2024 年 12 月 20 日**

**指导老师意见：**

目录

[1） 项目背景 3](#_Toc186327308)

[2） 系统概述 3](#_Toc186327309)

[2.1 总体设计 4](#_Toc186327310)

[2.2 系统功能详述 5](#_Toc186327311)

[3）项目设计与实现 6](#_Toc186327312)

[3.1 数据库设计与实现 6](#_Toc186327313)

[3.2 协同过滤：CF 算法 10](#_Toc186327314)

[3.3 前端网页设计与实现 13](#_Toc186327315)

[4）页面成果展示（有演示视频） 22](#_Toc186327316)

[5）项目总结 26](#_Toc186327317)

[5.1 项目成果总结 26](#_Toc186327318)

[5.2 项目不足与未来展望 27](#_Toc186327319)

[6）小组分工 29](#_Toc186327323)

智能图书推荐系统

# 项目背景

随着互联网技术的飞速发展和数字化阅读的普及，图书推荐系统成为了连接读者与书籍的重要桥梁。一个有效的图书推荐系统能够帮助用户在浩瀚的书海中快速找到自己感兴趣的书籍，同时也为图书销售商提供了精准营销的工具。在这样的背景下，智能图书推荐系统项目应运而生，旨在通过先进的算法和技术支持，为用户提供个性化的阅读体验。

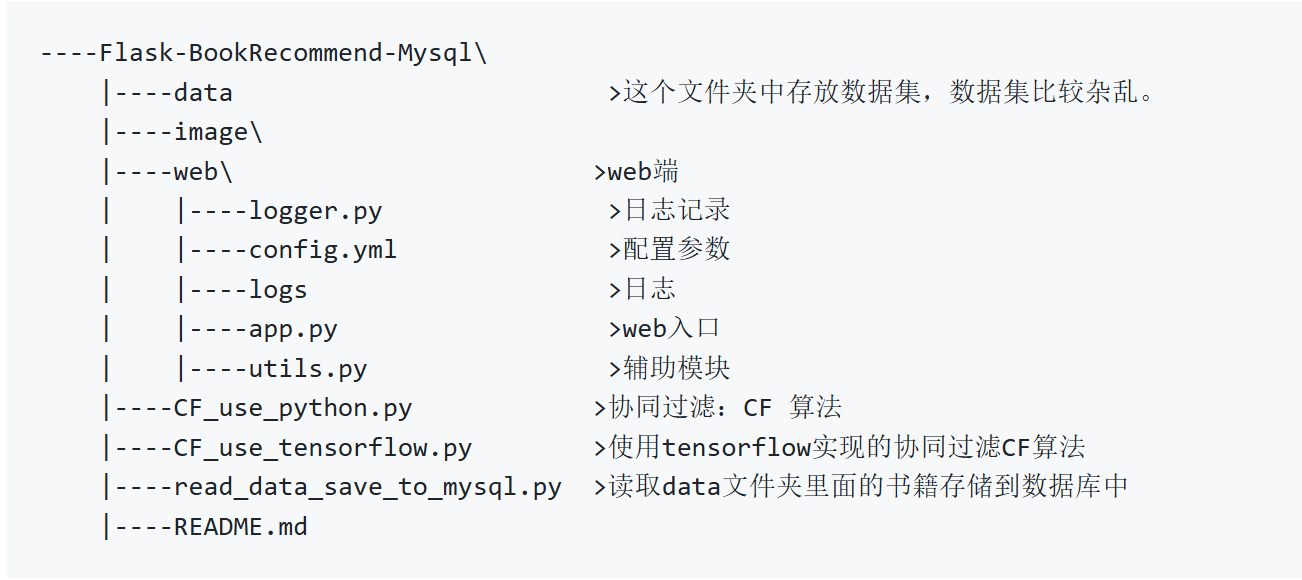
本项目是一个基于互联网的智能图书推荐系统，它通过分析用户的历史评分数据，使用协同过滤算法为用户提供个性化的书籍推荐。系统的主要目标是提高用户满意度，增加用户粘性，并最终推动图书销售。

项目特点：

1. **个性化推荐**：系统通过分析用户的评分历史和浏览行为，为用户提供个性化的书籍推荐，包括“热门书籍”、“猜你喜欢”和“推荐书籍”三个模块。
2. **实时与离线推荐**：系统不仅能够实现实时推荐，还能够通过离线计算好的推荐表提供推荐信息，以适应不同的业务场景。
3. **协同过滤算法**：系统采用了协同过滤算法，这是一种基于用户行为的推荐技术，能够有效地挖掘用户间的相似性，为用户提供更加精准的推荐。
4. **技术实现**：项目使用了Python 3.6作为编程语言，MySQL作为数据库存储，并集成了pandas、Flask、pymysql等库。特别是利用TensorFlow框架和GPU加速，实现了大规模矩阵运算，显著提升了算法的计算速度。
5. **系统架构**：项目采用了B/S架构，用户可以通过浏览器访问系统，实现注册、登录、搜索、评分、查看推荐等功能。管理员则拥有更多的权限，包括用户管理和书籍管理。
6. **环境配置**：项目在高性能的服务器上进行了环境配置，以支持大规模数据集的处理和模型训练。

# 系统概述

本系统是一款基于Flask框架和MySQL数据库的智能图书推荐平台，旨在为用户提供精准且个性化的阅读推荐服务。通过集成协同过滤算法，结合用户的历史评分数据及浏览行为，系统实现了实时推荐和离线推荐功能，从而全面提升用户的阅读体验。



## 2.1 总体设计

本系统的总体设计分为以下几个关键模块：

1. **数据预处理**  
   系统通过read\_data\_save\_to\_mysql.py脚本对原始数据集进行处理，并将其导入MySQL数据库。数据集中的用户评分、书籍信息等被转换为标准化的数据库格式，为后续的数据分析和算法训练提供坚实的数据基础。系统支持大规模数据集处理，但也允许通过切片功能灵活调整处理范围，以适应不同硬件配置。
2. **协同过滤算法实现**  
   系统提供了两种协同过滤算法实现：
   * **Python原生实现**：CF\_use\_python.py基于纯Python编写，算法逻辑清晰易懂，适合用于算法教学、调试和小规模数据处理。
   * **TensorFlow实现**：CF\_use\_tensorflow.py利用TensorFlow框架和GPU加速，极大地提升了协同过滤算法的训练效率。通过构建约27万行乘10万列的矩阵，系统能够处理大规模评分数据。在高性能硬件环境（如64G内存和4块TaiTanXP显卡）下，系统在一天内即可完成60000轮迭代训练。该算法具备较高的扩展性，用户可根据硬件条件调整数据切片参数及训练轮数。
3. **Web端功能模块**  
   系统采用Flask框架实现Web服务，提供了用户友好的交互界面。通过登录主页，用户可以完成注册、登录、书籍检索、评分、推荐查看等操作。基于RESTful API的架构设计，前后端解耦，确保系统的灵活性和可扩展性。
4. **数据库设计**  
   系统采用MySQL数据库，存储包括用户信息、书籍信息、评分数据和推荐结果在内的各类数据。通过合理设计数据表结构和索引优化，系统能够高效地处理复杂查询，确保实时推荐功能的响应速度。
5. **日志记录与配置管理**  
   系统通过logger.py模块实时记录系统操作日志，便于问题排查和运行状态监控。配置管理方面，系统通过config.yml文件集中存储各类参数（如数据库连接信息），提升了系统的灵活性和维护性。

## 2.2 系统功能详述

本系统为用户和管理员分别提供了多项核心功能，具体如下：

**2.2.1 用户功能**

1. **注册与登录**  
   用户可通过注册页面创建新账户，输入基本信息后即可激活账号。登录功能支持已注册用户访问系统，系统通过校验用户名和密码保障账户安全。
2. **书籍搜索**  
   用户可以通过关键词查询感兴趣的书籍，系统将基于数据库中的书籍信息返回匹配结果，并支持排序和筛选功能。
3. **书籍评分**  
   用户可以为已阅读的书籍打分，评分数据将实时存入数据库，并用于协同过滤算法的模型训练，提升推荐结果的准确性。
4. **实时推荐**  
   系统通过SQL查询实时分析用户行为数据，并生成个性化推荐书单。该功能无需预计算，能够即时响应用户需求。
5. **离线推荐**  
   系统定期运行协同过滤算法生成离线推荐结果，并将结果存储在数据库中。用户登录后可直接查看预先计算的推荐书单，获得更高质量的推荐。
6. **购物车管理**  
   用户可以将感兴趣的书籍添加到购物车，支持批量删除与结算功能，方便后续购买。
7. **书单创建与管理**  
   用户可以创建个性化书单，将喜欢的书籍归类收藏，书单支持编辑、删除等操作。
8. **历史评分查询**  
   用户可以通过该功能查看自己过往的评分记录，了解自己的阅读偏好，为推荐算法提供进一步的参考依据。

**2.2.2 管理员功能**

1. **用户管理**  
   管理员可以通过后台管理界面查看所有用户信息，并提供删除用户的权限，确保系统的秩序和安全。
2. **书籍管理**  
   管理员可以添加新书籍到数据库，也可以删除过时或不适合推荐的书籍，保持书籍库的更新和维护。

**2.2.3 辅助功能**

1. **日志记录**  
   系统自动记录用户操作和系统状态信息，帮助开发者排查问题，优化系统性能。
2. **配置管理**  
   管理员可以通过配置文件调整数据库连接参数、日志级别等系统设置，灵活适应不同运行环境。

通过上述功能模块的有机结合，本系统为用户提供了一个高效、智能、全面的图书推荐解决方案，同时也为系统管理者提供了强大的维护工具，为智能图书推荐领域的研究和实践提供了典范。

# 3）项目设计与实现

## 3.1 数据库设计与实现

**3.1.1. 数据库概述**

本系统的数据库以 MySQL 作为数据存储后端，旨在高效管理与操作用户数据、书籍数据、评分数据以及推荐结果等信息。数据库由多个表组成，涵盖了用户、书籍、评分、购物车、收藏等核心功能模块，支持智能图书推荐系统的所有功能。

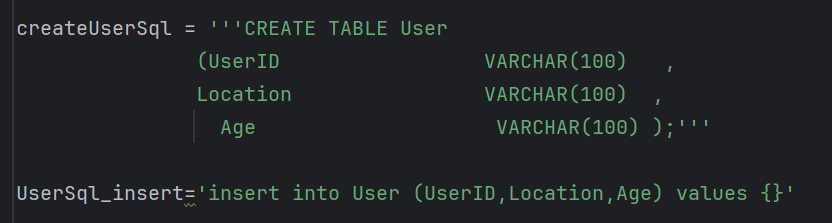
**3.1.2. 数据库表结构设计**

以下是系统的数据库表设计及其字段说明：

**2.1 用户表（User）**

用于存储用户的基本信息。

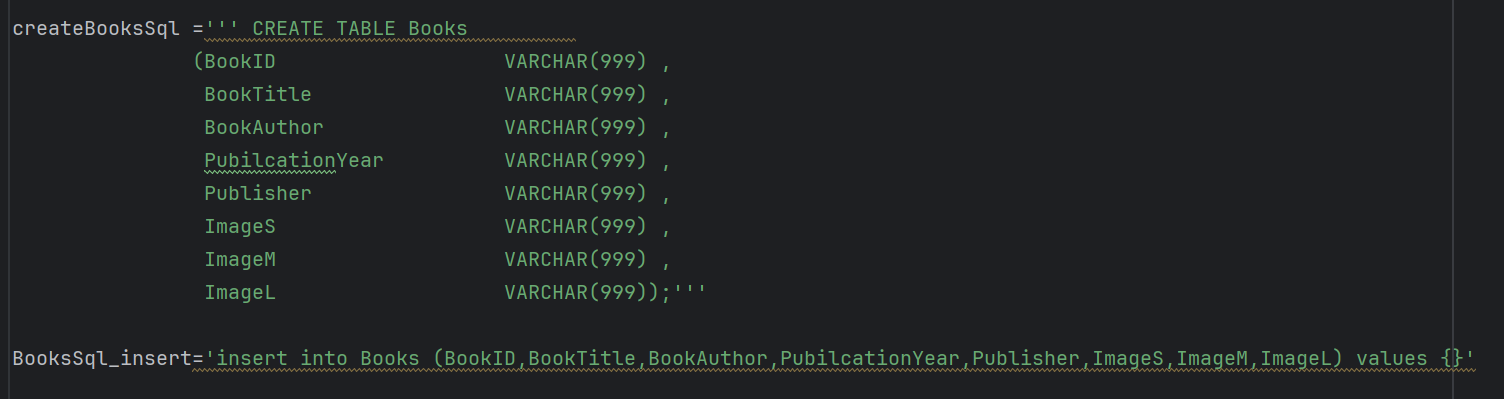
* **字段描述**：
  + UserID：用户 ID，主键（VARCHAR(100)）。
  + Location：用户所在地（VARCHAR(100)）。
  + Age：用户年龄（VARCHAR(100)）。
* **建表语句**：



**2.2 书籍表（Books）**

用于存储书籍的基本信息。

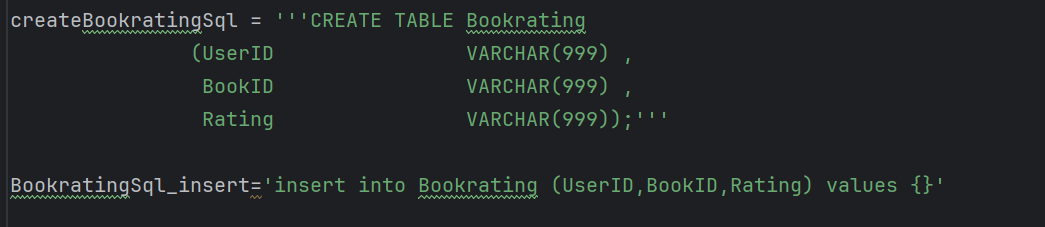
* **字段描述**：
  + BookID：书籍 ID，主键（VARCHAR(999)）。
  + BookTitle：书名（VARCHAR(999)）。
  + BookAuthor：作者（VARCHAR(999)）。
  + PubilcationYear：出版年份（VARCHAR(999)）。
  + Publisher：出版社（VARCHAR(999)）。
  + ImageS：书籍小尺寸封面图片 URL（VARCHAR(999)）。
  + ImageM：书籍中尺寸封面图片 URL（VARCHAR(999)）。
  + ImageL：书籍大尺寸封面图片 URL（VARCHAR(999)）。
* **建表语句**：



**2.3 评分表（Bookrating）**

用于存储用户对书籍的评分信息。

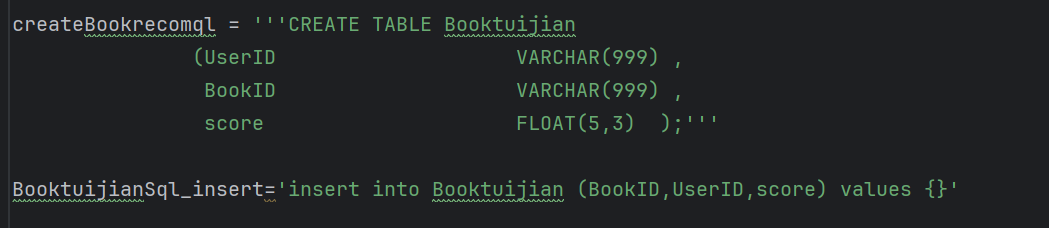
* **字段描述**：
  + UserID：用户 ID（VARCHAR(999)）。
  + BookID：书籍 ID（VARCHAR(999)）。
  + Rating：评分（VARCHAR(999)）。
* **建表语句**：



**2.4 推荐表（Booktuijian）**

用于存储系统生成的书籍推荐结果。

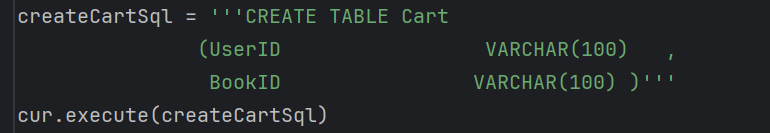
* **字段描述**：
  + UserID：用户 ID（VARCHAR(999)）。
  + BookID：书籍 ID（VARCHAR(999)）。
  + score：推荐分数，浮点数（FLOAT(5,3)）。
* **建表语句**：



**2.5 购物车表（Cart）**

用于存储用户的购物车信息。

* **字段描述**：
  + UserID：用户 ID（VARCHAR(100)）。
  + BookID：书籍 ID（VARCHAR(100)）。
* **建表语句**：



**2.6 收藏表（Collects）**

用于存储用户的收藏信息。

* **字段描述**：
  + UserID：用户 ID（VARCHAR(100)）。
  + BookID：书籍 ID（VARCHAR(100)）。
* **建表语句**：



**3.1.3. 数据导入实现**

各数据表数据导入通过脚本 read\_data\_save\_to\_mysql.py 实现，脚本读取本地 CSV 文件并将其插入到数据库中。以下是关键步骤：

* **1. 读取 CSV 文件**： 使用 pandas 读取数据文件，支持灵活处理不规则数据格式（on\_bad\_lines='skip'）。
* **2. 数据插入**：
  + 创建表结构（如 CREATE TABLE）。
  + 逐行插入数据，使用 pymysql 执行 INSERT 语句。
  + 异常处理机制确保导入过程的稳定性。
* **示例代码**：
  + 数据插入函数：



* + 调用插入函数：



**3.1.4. 数据库初始化**

脚本实现了数据库初始化功能：

1. 删除旧的 Book 数据库（DROP DATABASE IF EXISTS）。
2. 创建新的 Book 数据库及其所有表（CREATE DATABASE 和 CREATE TABLE）。
3. 批量导入数据，为系统运行提供基础。

**3.1.5. 设计特点**

* **模块化**：各表结构清晰，功能分区明确。
* **扩展性**：新增功能时可轻松添加表结构。
* **高效性**：使用批量插入和 GPU 支持的算法减少数据处理时间。
* **安全性**：引入异常处理，确保数据库操作的健壮性。

通过上述设计与实现，本系统构建了一个功能完备、性能优越的数据库，为智能图书推荐系统的核心服务提供了坚实的数据支撑。

## 3.2 协同过滤：CF 算法

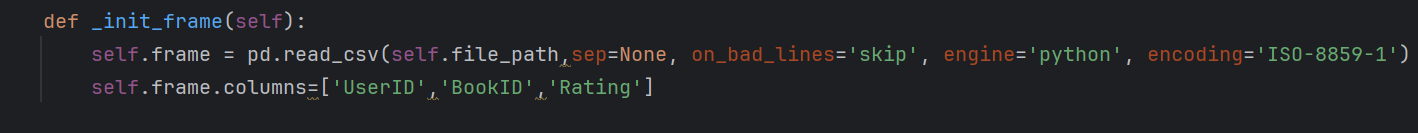
协同过滤（Collaborative Filtering, CF）是一种常用的推荐算法，其主要思想是基于用户或物品之间的相似性，通过历史交互数据预测用户可能感兴趣的内容。本项目实现了基于用户的协同过滤算法（User-CF）和基于深度学习的协同过滤算法，具体实现如下：

**3.2.1 基于用户的协同过滤算法**

在基于用户的协同过滤算法中，我们主要计算用户之间的相似度，根据最相似用户的偏好预测目标用户的推荐内容。以下是算法的关键步骤和实现细节：

**数据加载与预处理**

使用 Pandas 库加载和预处理用户对书籍的评分数据。数据源为 BX-Book-Ratings.csv，包括用户 ID、书籍 ID 和评分。加载完成后，将数据清洗并转化为适合分析的结构。



**相似度计算**

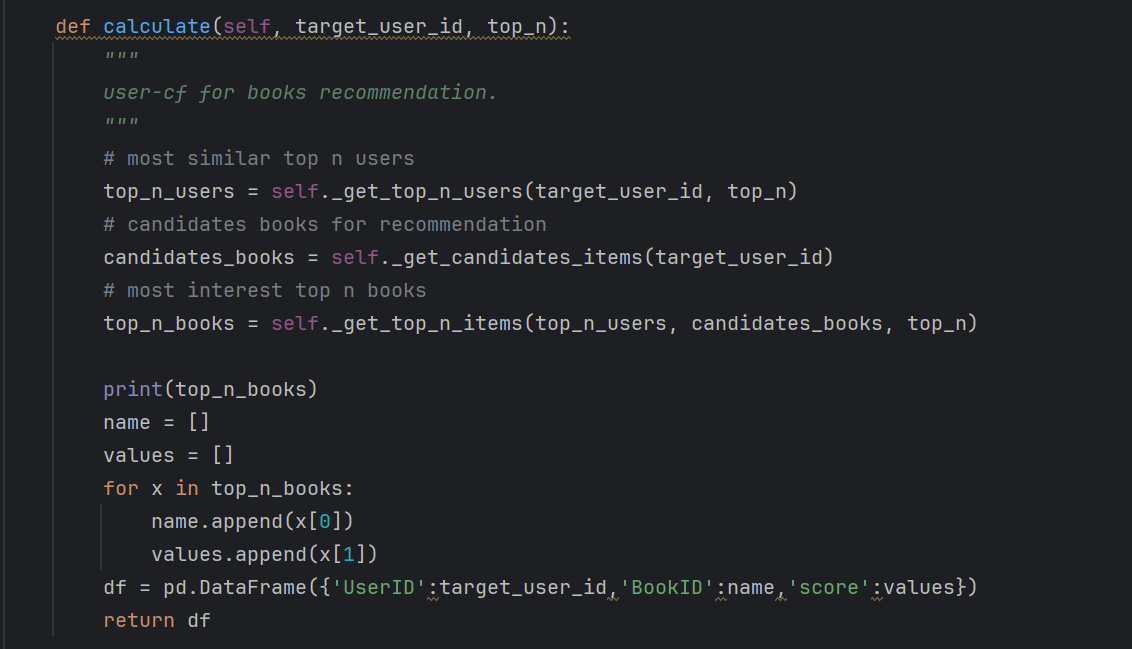
使用余弦相似度作为用户相似度的度量方法。计算两个用户的相似度时，首先统计两者评分过的书籍交集大小，结合评分向量长度计算余弦值，值越高表明用户兴趣越相似。



**推荐算法核心流程**

1. **获取相似用户**  
   为目标用户找到前 N 个最相似的用户，并记录他们的评分信息。
2. **生成候选书籍列表**  
   获取目标用户未评分的书籍作为候选书籍。
3. **计算候选书籍的推荐度**  
   候选书籍的推荐度由相似用户的评分和用户之间的相似度加权计算得到，推荐度高的书籍将优先推荐。

完整算法实现如下：

****

**实验结果**

通过从百万数据集中随机抽取的用户样本，计算推荐书籍并输出结果，示例如下：



**3.2.2 基于 TensorFlow 的协同过滤算法**

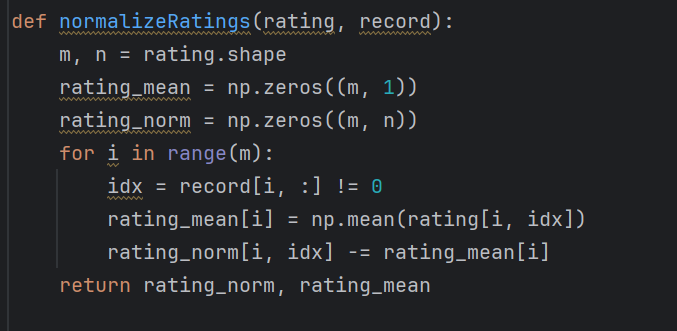
为进一步提升推荐算法性能，使用 TensorFlow 实现基于矩阵分解的协同过滤算法，主要包括以下步骤：

**数据加载与矩阵构建**

加载用户评分数据后，构建用户-物品评分矩阵 rating，以及用户与物品的交互记录矩阵 record，用于后续模型训练。

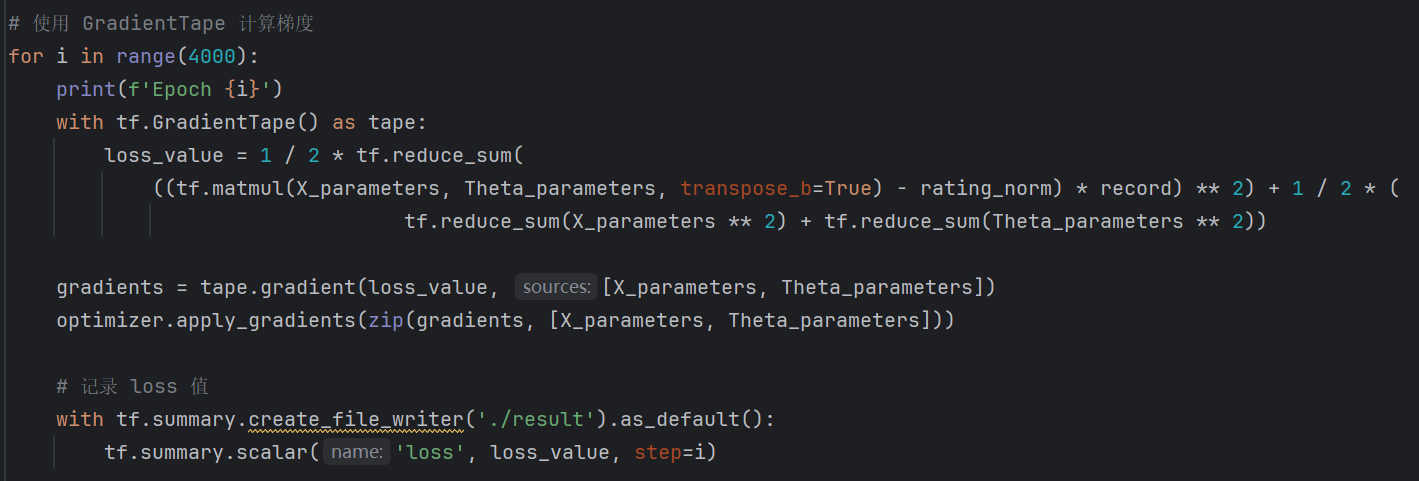
**数据归一化**

对评分矩阵进行归一化处理，减去每一行的平均值，使得模型训练更高效。



**模型定义与训练**

通过 TensorFlow 定义矩阵分解模型，将评分矩阵分解为用户特征矩阵和物品特征矩阵。使用梯度下降优化器迭代更新参数，最小化损失函数。



**预测与推荐**

通过模型预测用户对书籍的评分，并结合实际评分的均值还原为原始评分，用于生成推荐结果。

**3.2.3 总结**

本部分实现了两种协同过滤算法，分别基于用户相似度和矩阵分解，提供了高效的推荐系统解决方案。实验表明，基于深度学习的算法在大规模数据上更具优势，而传统协同过滤算法适用于小规模数据集。

## 3.3 前端网页设计与实现

**3.3.1 管理员界面**

管理员界面是图书后台管理系统的重要组成部分，它提供了用户和书籍信息的管理功能。以下是管理员界面的关键设计和实现细节：

**设计理念**

管理员界面的设计遵循简洁、直观的原则，旨在提供高效的操作流程和良好的用户体验。界面采用了Bootstrap和Flat-UI框架，确保了响应式布局和现代化的视觉效果。

**结构布局**

管理员界面由以下几个部分组成：

1. **导航栏（Navbar）**：提供一个静态的顶部导航栏，包含系统名称“图书后台管理系统”和导航链接，以及用户信息和注销按钮。
2. **容器（Container）**：主要内容容器，用于展示不同的管理模块。
3. **缩略图（Thumbnail）**：用于展示模块标题和内容，增加视觉层次感。
4. **表单（Form）**：用于输入和管理数据，如添加书籍和用户信息。
5. **列表组（List Group）**：展示用户和书籍的列表，每个项目都可以进行操作，如删除。

**样式定制**

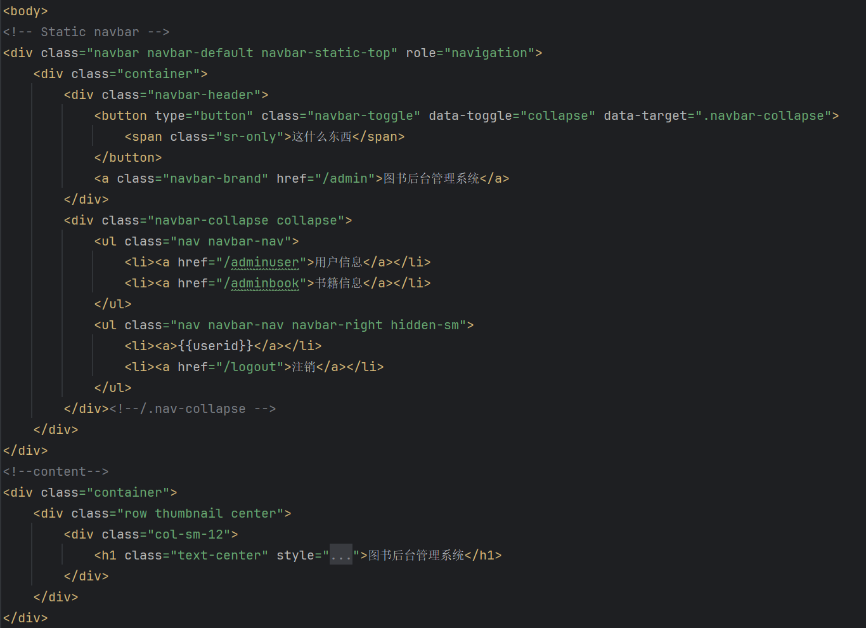
管理员界面的样式通过自定义CSS实现，包括：

* 行（Row）和段落（P）的样式，确保文本的简洁显示。
* 列表组项（List Group Item）的悬停效果，提供视觉反馈。
* 按钮（Button）的定制样式，包括添加书籍按钮和删除用户按钮的视觉效果。

**功能实现**

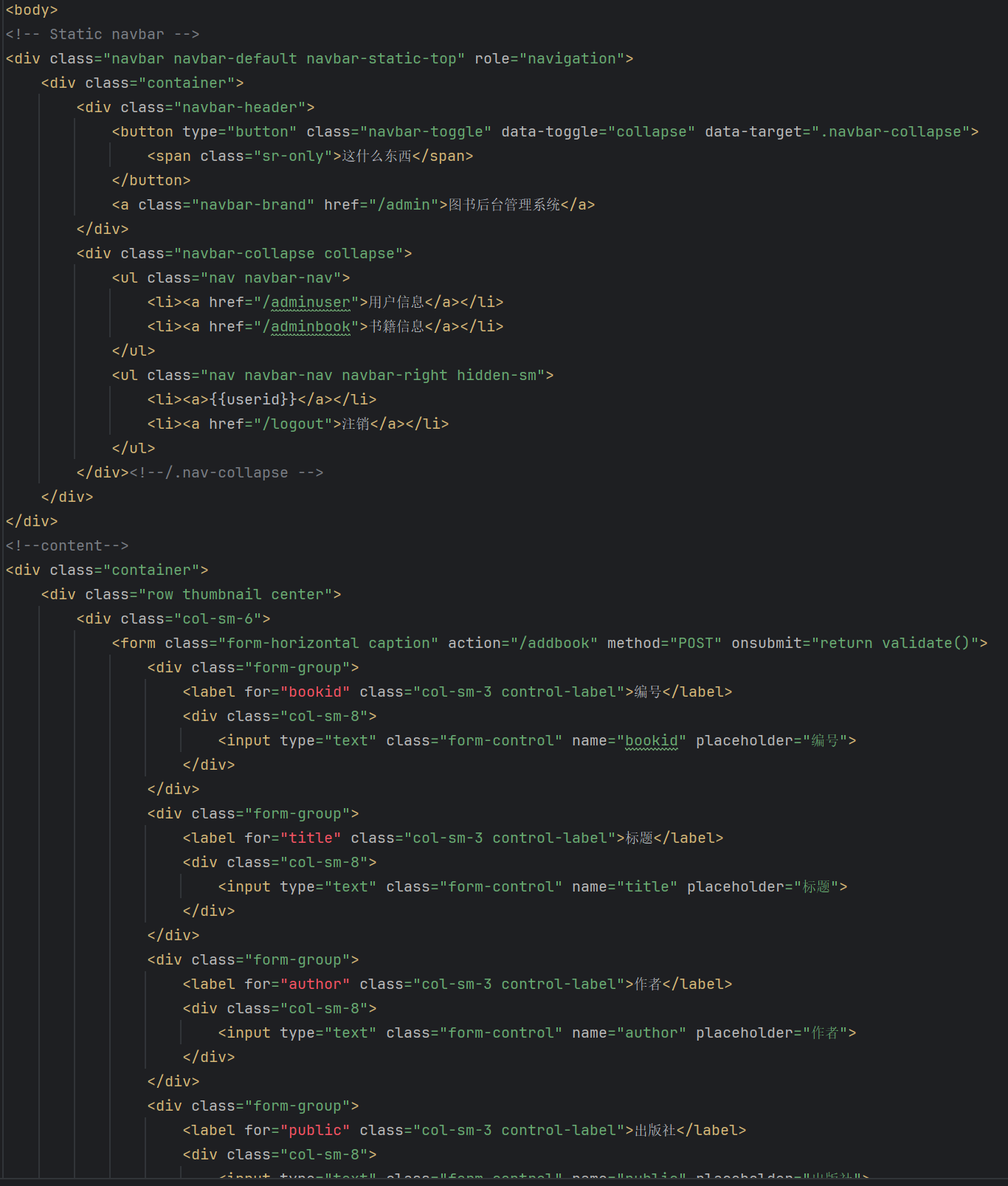
**Admin.html**

管理员主页面提供了一个概览界面，显示系统名称和两个主要的管理模块：用户信息和书籍信息。页面顶部的导航栏允许管理员快速切换不同的管理模块。



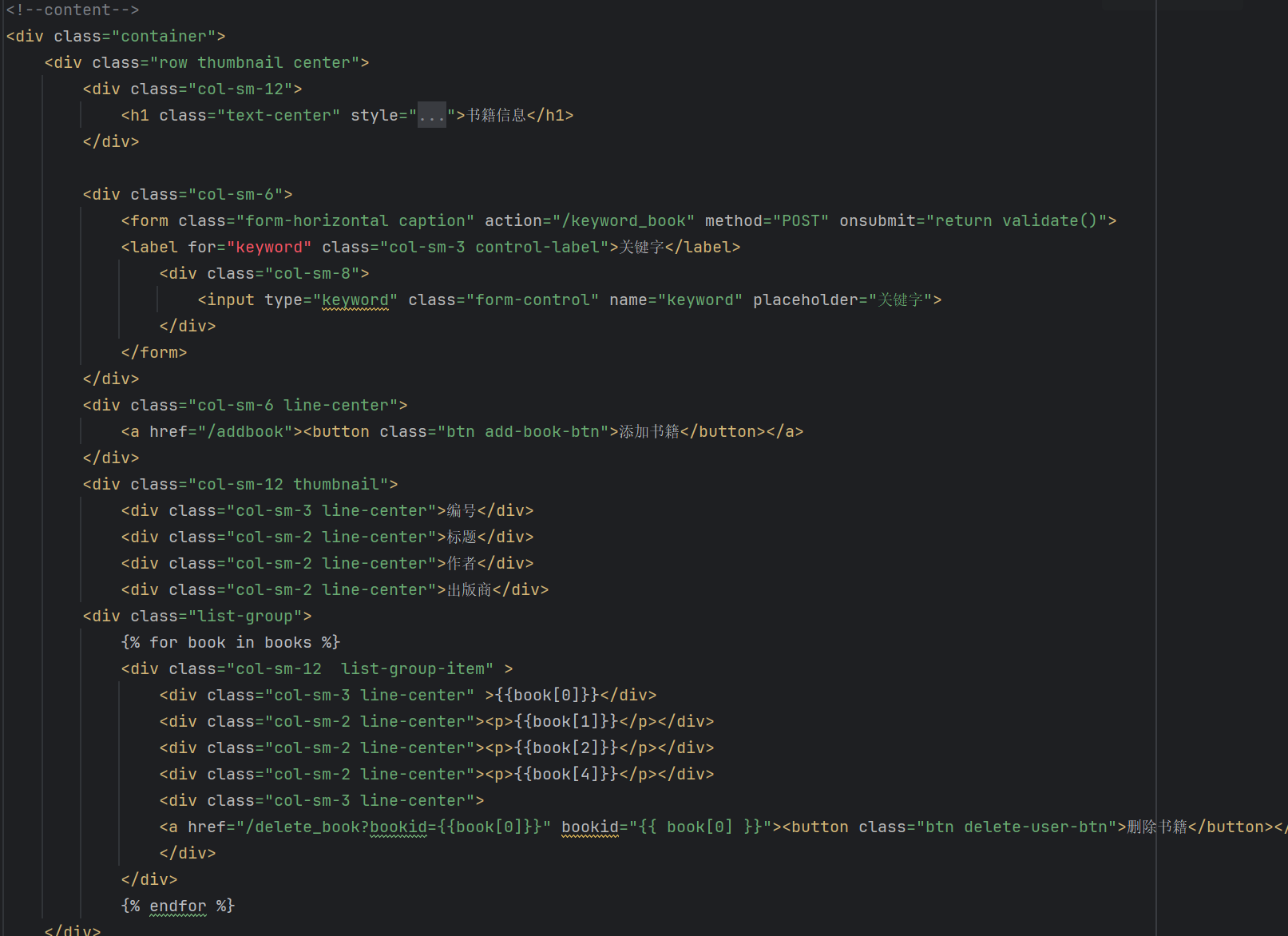
**AdminAddBook.html**

添加书籍页面提供了一个表单，管理员可以输入书籍的编号、标题、作者和出版社信息。表单提交后，新的书籍信息将被添加到系统中。



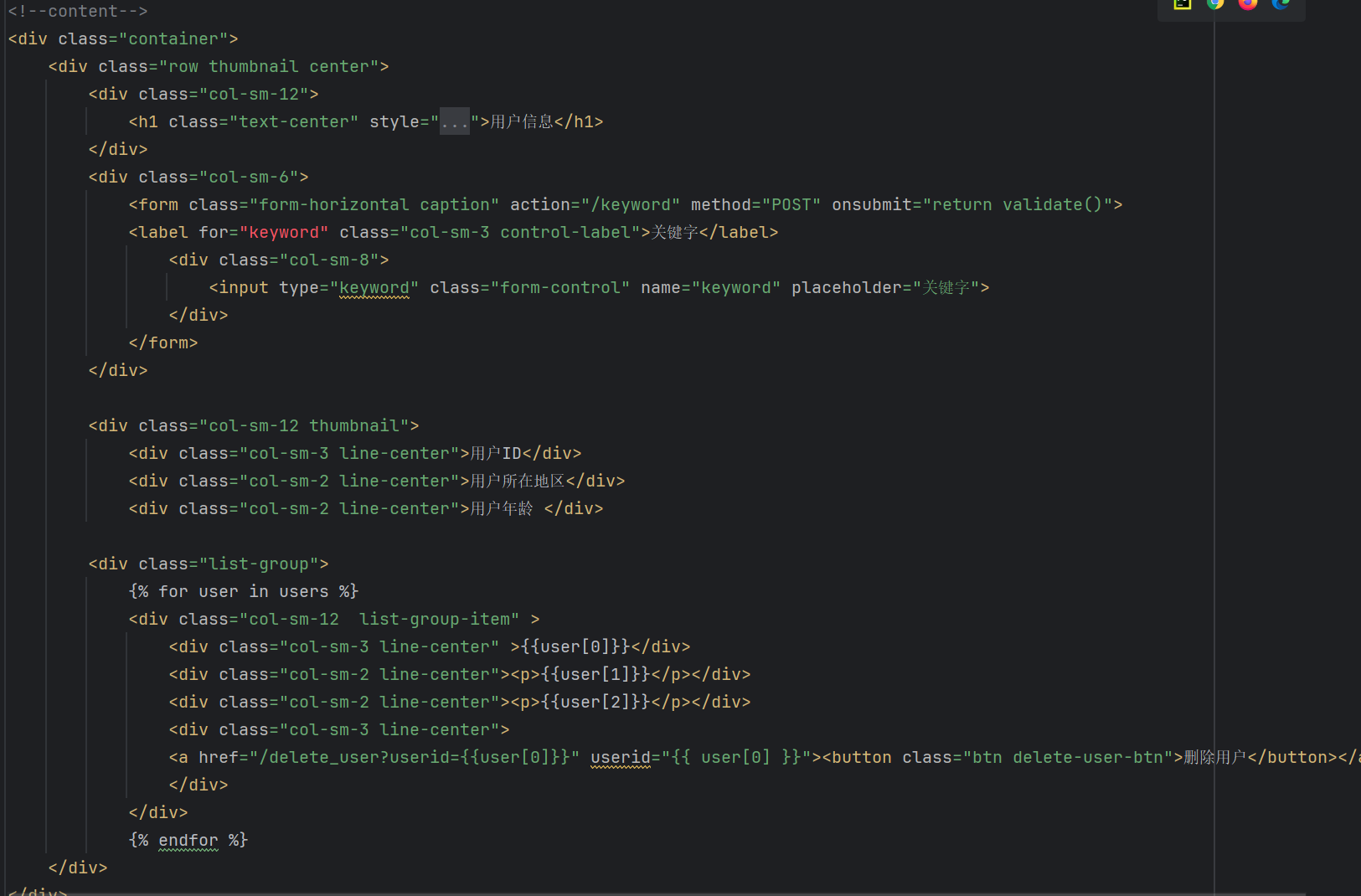
**AdminBook.html**

书籍信息页面展示了系统中所有书籍的列表，包括编号、标题、作者和出版社。管理员可以通过关键字搜索特定的书籍，并且可以添加新的书籍或删除已有的书籍。



**AdminUser.html**

用户信息页面展示了系统中所有用户的列表，包括用户ID、所在地区和年龄。管理员可以通过关键字搜索特定的用户，并且可以删除用户信息。



**交互设计**

管理员界面的交互设计简洁直观：

* **悬停效果**：列表组项在鼠标悬停时改变背景色，提供视觉反馈。
* **按钮反馈**：删除按钮在悬停时改变颜色，提示用户这是一个可操作的按钮。
* **表单验证**：在添加书籍和用户信息时，表单提交前进行验证，确保数据的准确性。

**总结**

管理员界面的设计和实现考虑了管理员的实际操作需求，提供了直观的导航、清晰的布局和高效的数据管理功能。通过定制的样式和交互设计，管理员界面不仅功能全面，而且易于使用，极大地提高了后台管理的效率和用户体验。

**3.3.2 用户界面**

用户界面是图书推荐系统与最终用户直接交互的窗口，它直接影响用户的体验和满意度。以下是用户界面的关键设计和实现细节：

**设计理念**

用户界面的设计以简洁、直观和用户友好为核心，旨在提供流畅的用户体验和便捷的操作流程。界面采用了Bootstrap和Flat-UI框架，确保了响应式布局和现代化的视觉效果。

**结构布局**

用户界面由以下几个部分组成：

1. **导航栏（Navbar）**：提供一个固定的顶部导航栏，包含系统名称“图书推荐系统”和导航链接，以及用户信息和注销按钮。
2. **内容容器（Container）**：主要内容容器，用于展示不同的用户功能模块。
3. **图书信息展示（Book Info Display）**：用于展示图书的详细信息，包括封面图片、书名、作者、出版日期等。
4. **用户操作按钮（User Action Buttons）**：包括“加入购物车”、“收藏”和“评分”等操作按钮，方便用户进行互动。
5. **列表组（List Group）**：展示用户的历史评分、收藏的图书列表，每个项目都可以进行操作，如删除。

**样式定制**

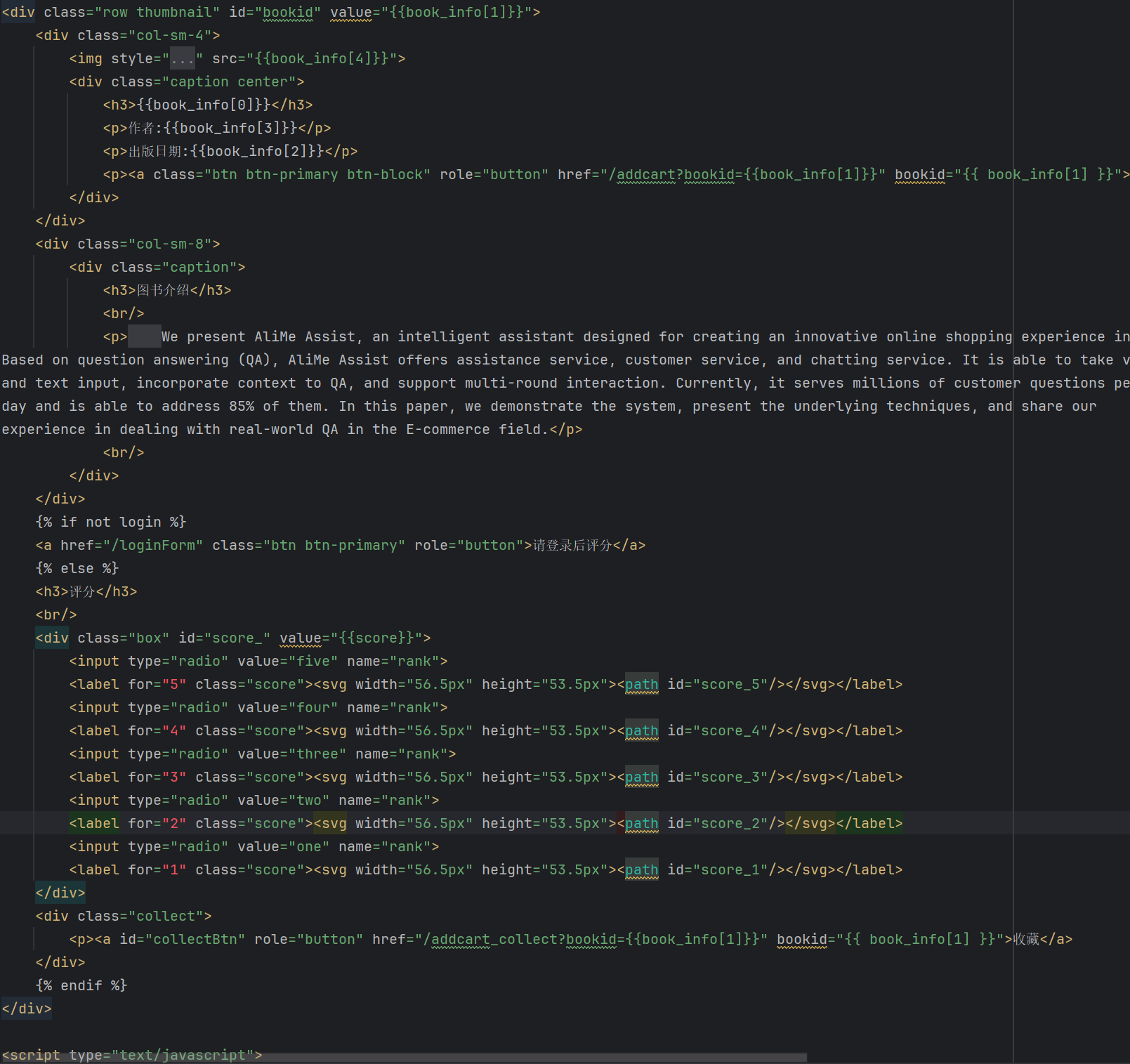
用户界面的样式通过自定义CSS实现，包括：

* 导航栏的背景色、字体色和悬停效果。
* 图书信息展示区域的布局和样式。
* 用户操作按钮的定制样式，包括悬停时的视觉效果。
* 列表组项的悬停效果，提供视觉反馈。

**功能实现**

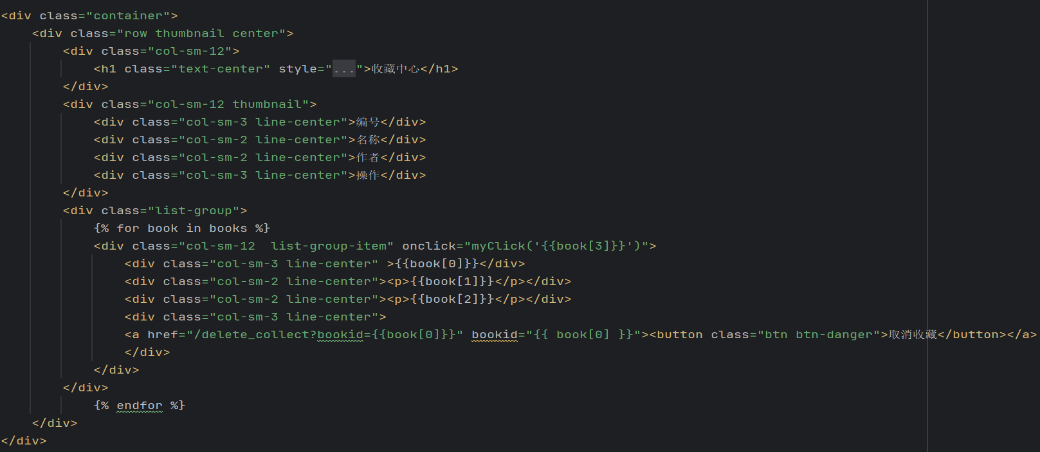
**BookInfo.html**

图书信息页面提供了图书的详细信息，包括封面图片、书名、作者、出版日期等。用户可以对图书进行评分、加入购物车和收藏操作。页面顶部的导航栏允许用户快速切换不同的功能模块。



**Collect.html**

收藏中心页面展示了用户收藏的图书列表，包括编号、名称、作者等信息。用户可以取消收藏特定的图书。页面设计简洁，便于用户管理和查看收藏的图书。



**Historicalscore.html**

历史评分页面展示了用户的历史评分记录，包括编号、名称、作者和历史评论分数。用户可以查看每本图书的评分详情。页面提供了清晰的列表视图，方便用户回顾和查找。



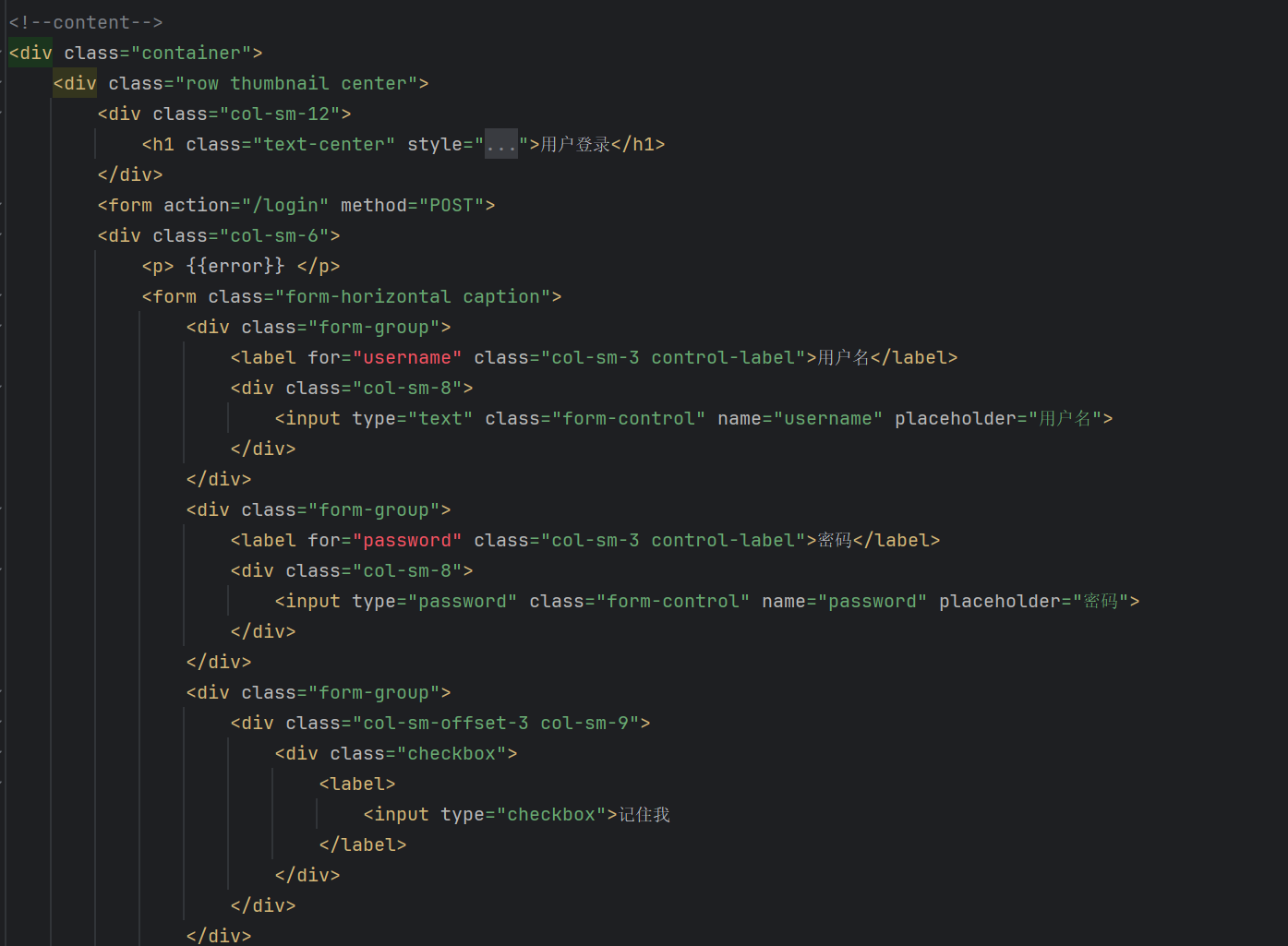
**Index.html**

首页提供了图书推荐系统的主要入口，包括热门书籍、猜你喜欢和推荐书籍等模块。页面设计注重用户体验，通过合理的布局和视觉引导，鼓励用户探索和发现新的书籍。



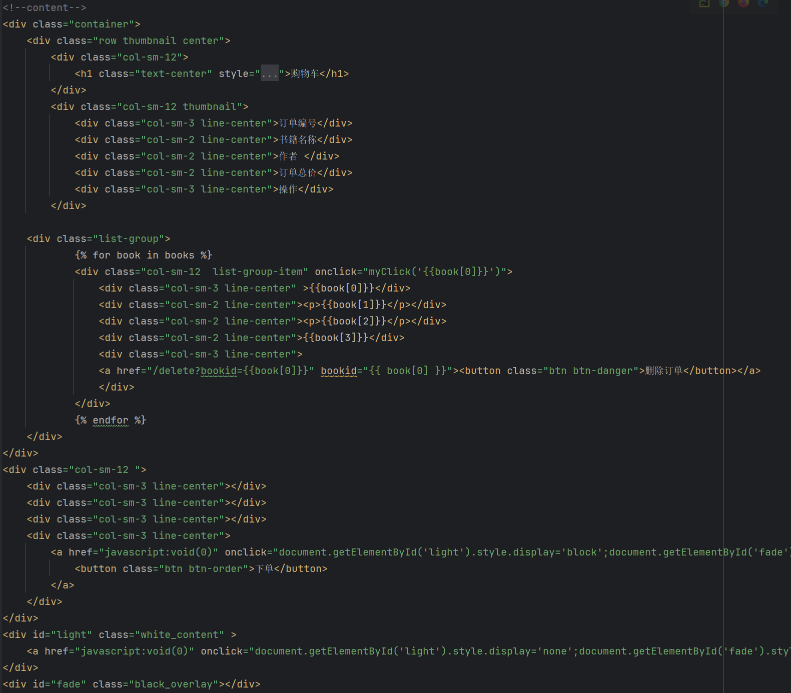
**Login.html**

登录页面提供了用户登录的表单，包括用户名和密码输入框。页面设计简洁，便于用户快速登录系统。



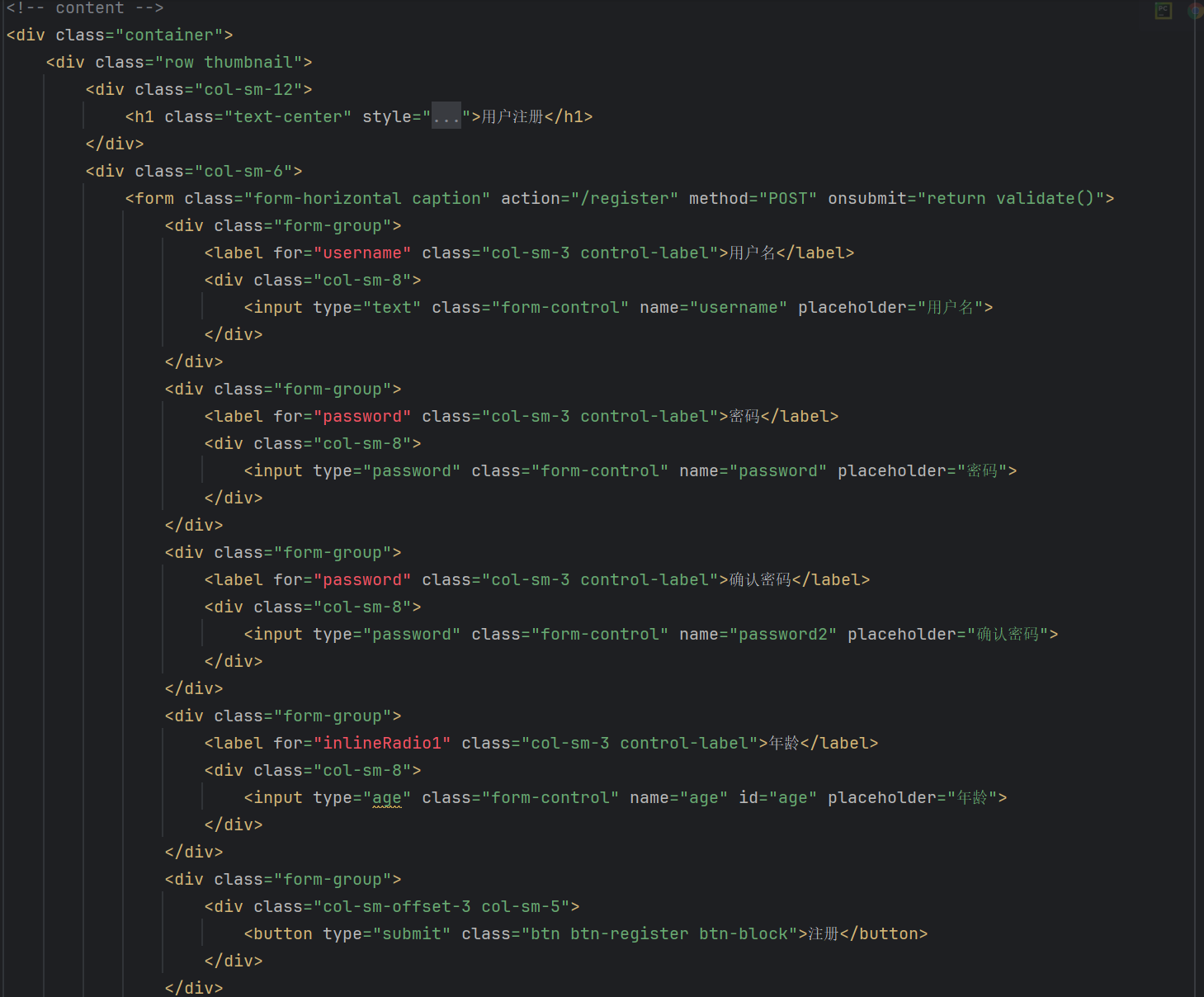
**Order.html**

购物车页面展示了用户的购物车内容，包括订单编号、书籍名称、作者和订单总价。用户可以删除订单或进行下单操作。页面设计注重操作的便捷性，方便用户管理购物车。



**Register.html**

注册页面提供了用户注册的表单，包括用户名、密码、确认密码和年龄输入框。页面设计简洁，便于用户创建新账户。



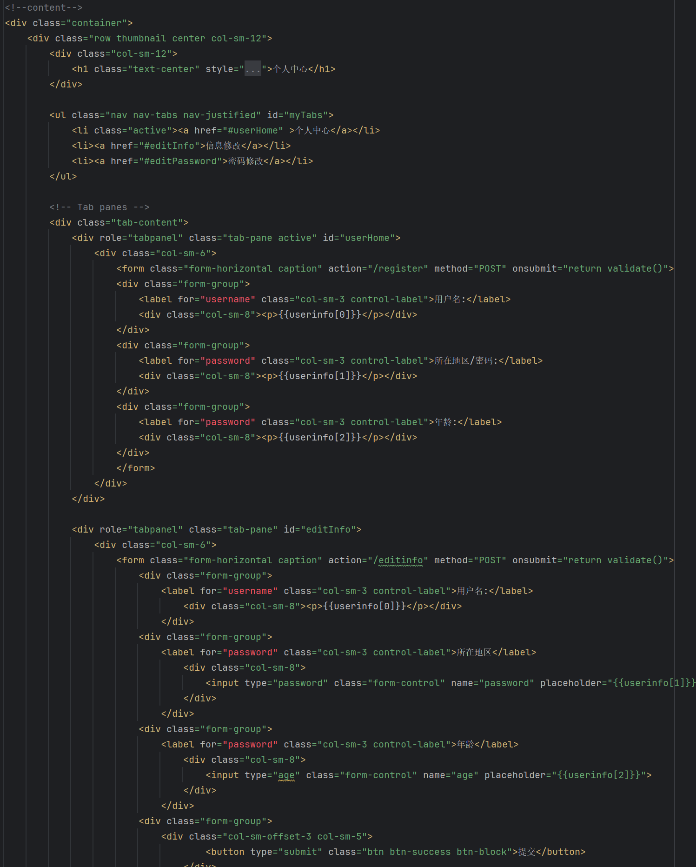
**Search.html**

搜索页面提供了图书搜索的功能，用户可以通过关键词搜索图书。页面设计注重搜索体验，提供了清晰的搜索框和搜索结果展示。



**UserInfo.html**

个人中心页面提供了用户的个人信息展示和编辑功能，包括用户名、所在地区和年龄。用户可以修改个人信息和密码。页面设计注重个人信息的安全性和易用性。



**总结**

用户界面的设计和实现考虑了用户的实际操作需求，提供了直观的导航、清晰的布局和便捷的用户操作功能。通过定制的样式和交互设计，用户界面不仅功能全面，而且易于使用，极大地提高了图书推荐系统的用户体验和满意度。

# 4）项目成果展示（有演示视频）

**项目启动方式，具体见 Readme 文档**

使用python3.6作为编程语言。使用mysql作为数据库存储，需要安装pandas,flask，pymysql

首先建立自己的mysql数据库

运行read\_data\_save\_to\_mysql.py文件 将数据导入到mysql中。

  注意mysql的链接参数.修改为自己的连接参数（id,code,端口）需要修改read\_data\_save\_to\_mysql和web/config.yml文件下的mysql的配置参数。

进入web文件夹,运行app.py

在浏览器上访问 127.0.0.1:8080

使用下载数据中的UserID和其对应的Location作为账号密码登录网站，或者自己注册登录。

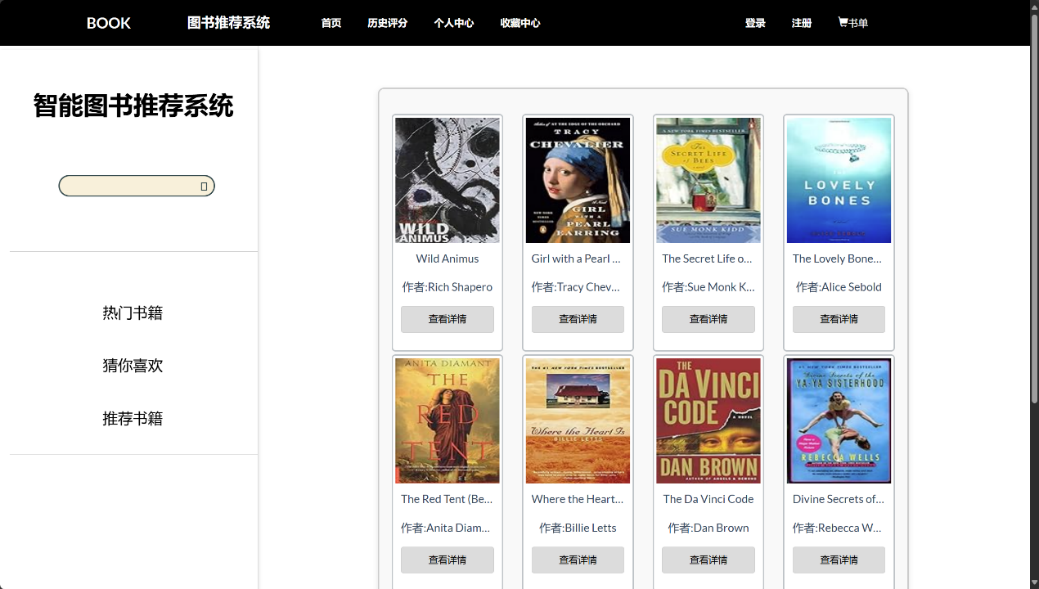
系统管理员的账号：admin 密码：admin 通过这个账号密码进入后台管理

example:

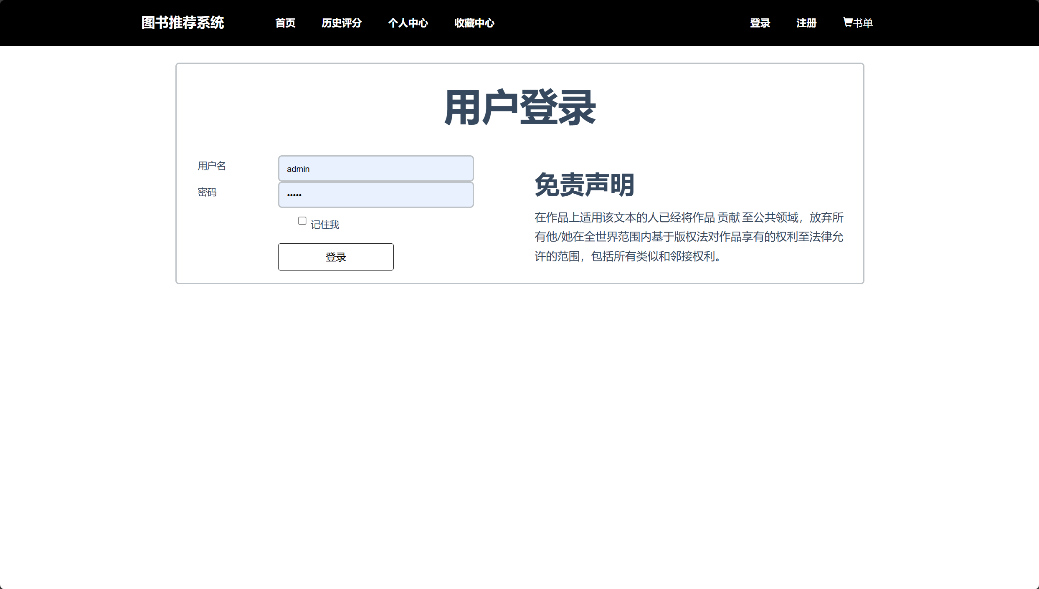
  id：nyc

  code: new york

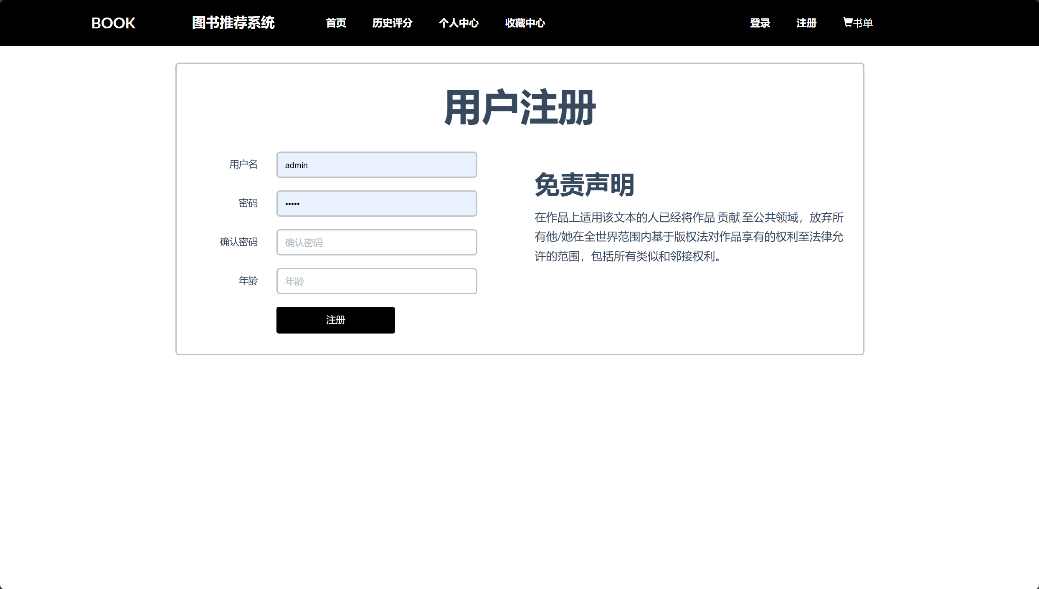
**主页面**

****

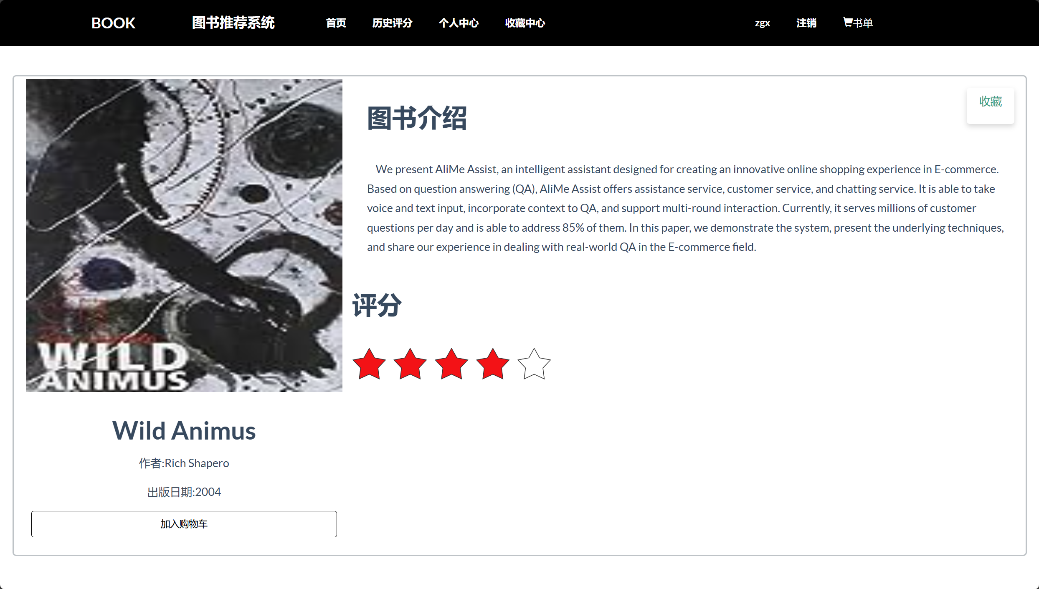
**登陆页面**



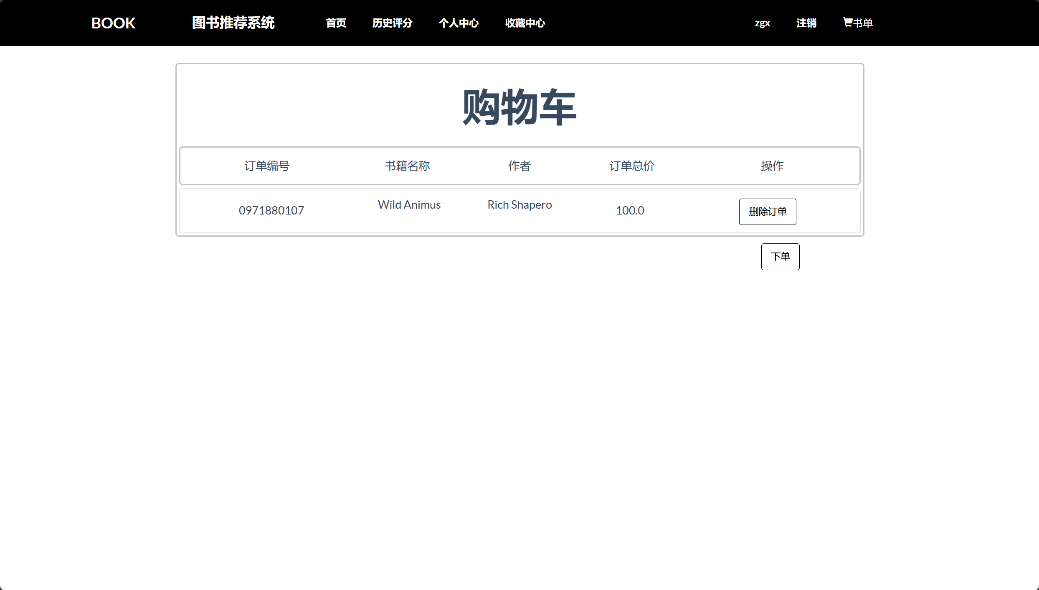
**注册页面**



**书籍信息详情（包含评分，收藏，加入购物车功能）**



**购物车信息**



**收藏中心**



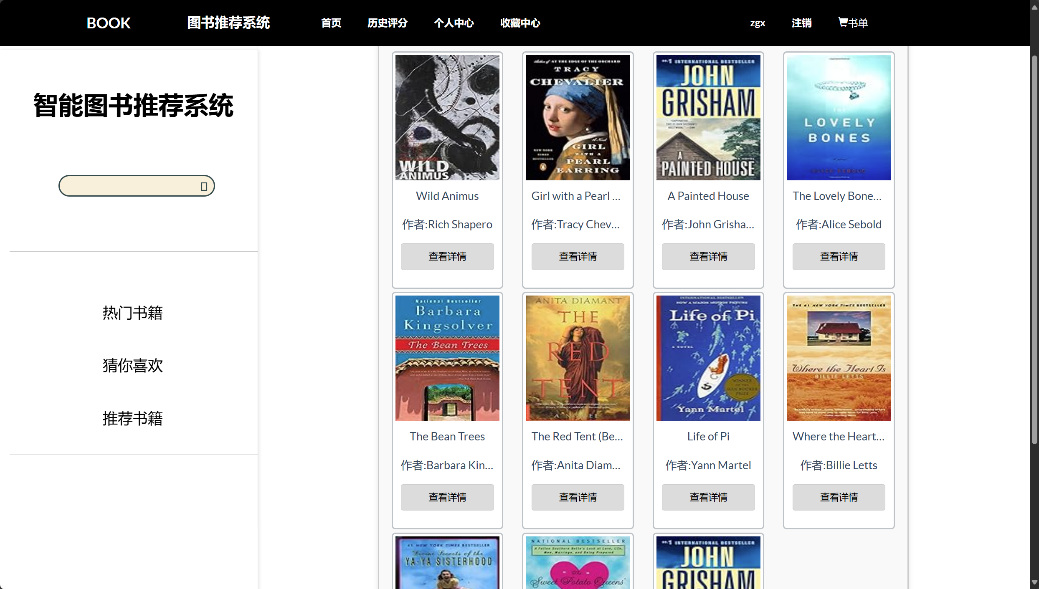
**历史评分**



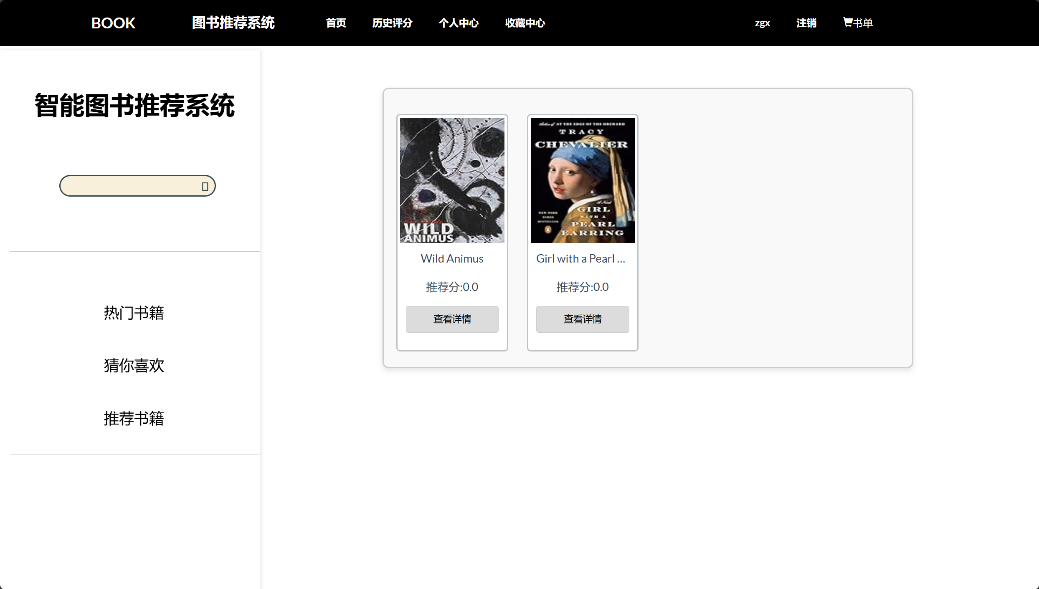
**个人中心**



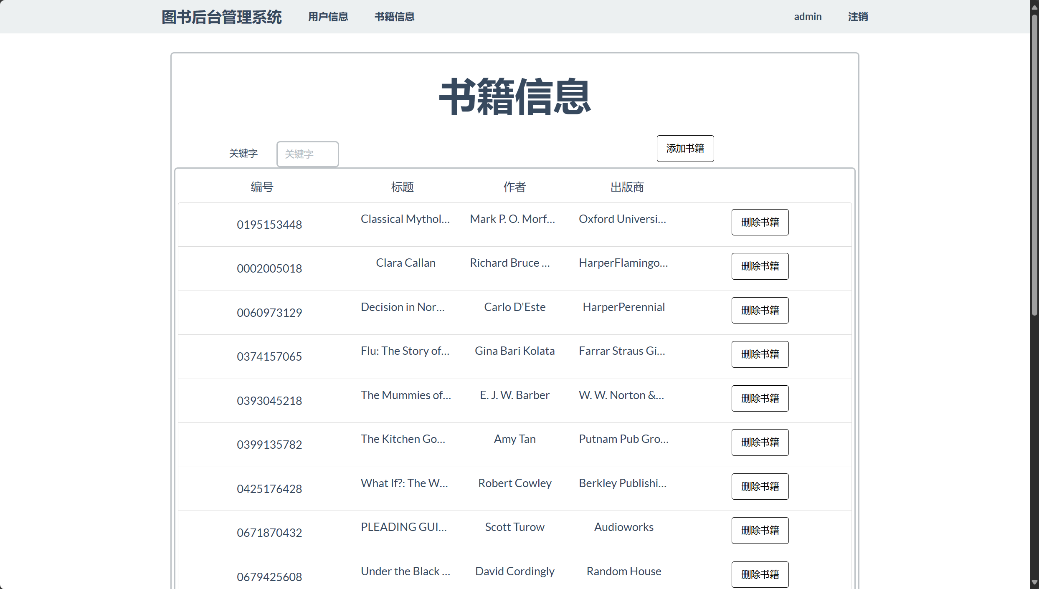
**猜你喜欢**

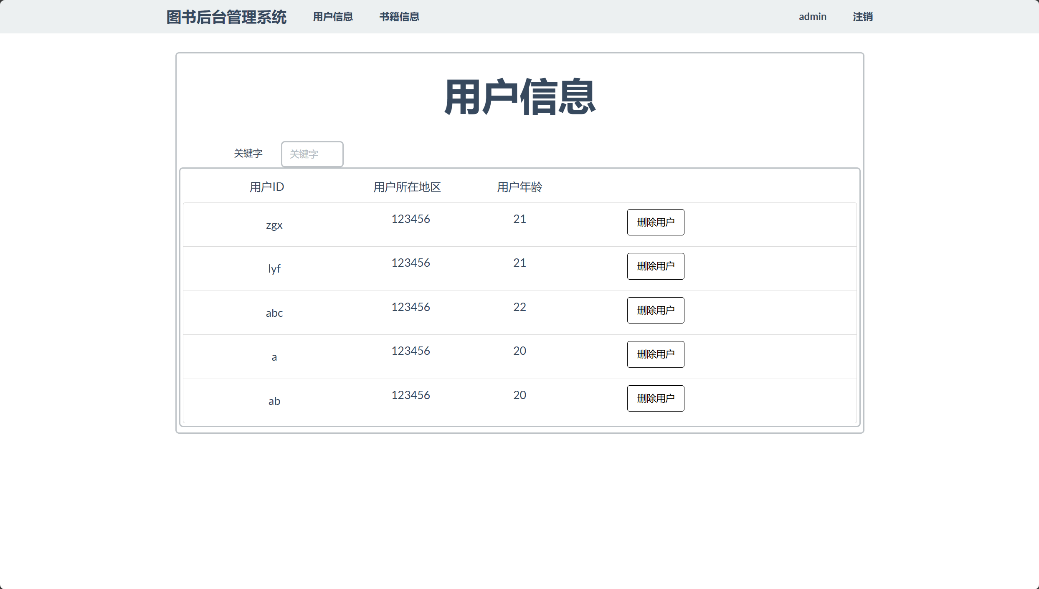


**推荐书籍**



**后端管理页面（用户信息、书籍信息）**





# 5）项目总结

本项目的完成标志着一个基于协同过滤算法的智能图书推荐系统成功落地，其设计和实现展示了我们团队在算法应用、系统集成和前端开发方面的能力。总结如下：

## 5.1 项目成果总结

1. **技术学习与运用**
   * 学习并成功应用了协同过滤算法，包括基于用户的协同过滤和基于深度学习的矩阵分解模型。通过这些算法，系统实现了较为精准的图书推荐功能。
   * 掌握了Python与Flask框架的结合使用，完成了系统的后端服务开发，支持注册、登录、评分、搜索、推荐等功能。
   * 学会了MySQL数据库的设计与管理，设计了包含用户表、书籍表、评分表等核心功能模块的高效数据库结构，并实现了数据的预处理和批量导入。
2. **系统功能实现**
   * 成功实现了用户界面的全功能，包括注册登录、书籍搜索、评分、购物车管理、书单创建与管理等。
   * 开发了管理员界面，实现了用户管理和书籍管理功能，保证了系统的灵活性和可维护性。
   * 系统支持实时推荐与离线推荐，用户可以在前端网页上查看推荐结果并进行操作，如评分或收藏，显著提升了用户体验。
3. **系统架构与性能优化**
   * 系统采用B/S架构设计，结合RESTful API，为未来扩展提供了基础。
   * 利用TensorFlow和GPU加速训练协同过滤模型，大幅提升了训练效率，为大规模数据处理提供了支持。
4. **实践能力与团队合作**
   * 项目通过明确分工和高效协作，确保了任务的顺利推进。每位成员都在数据处理、算法优化、系统开发等方面做出了贡献。
   * 项目期间积累了丰富的开发经验，从需求分析到最终实现，团队完成了一次完整的软件开发流程。

## 5.2 项目不足与未来展望

1. **项目不足**
   * **个性化推荐不足**：尽管系统实现了基本的推荐功能，但在推荐结果的多样性和个性化方面还有待改进。例如，对于新用户，冷启动问题依然存在，导致推荐结果不够精准。
   * **算法优化空间大**：目前协同过滤算法的训练轮数受限于硬件条件，无法充分挖掘数据中的潜在价值。算法在处理大规模数据时的计算成本较高，存在优化空间。
   * **前端功能不够完善**：用户界面在交互设计上还有提升空间，例如增强用户书单的个性化管理功能，增加更多的交互动画和可视化推荐结果展示。
   * **系统性能问题**：在大规模用户和数据场景下，系统的响应速度和推荐算法的实时计算能力可能会面临挑战。
2. **未来展望**
   * **算法优化**：未来可以引入更多先进的推荐算法，如基于深度学习的神经网络协同过滤（NCF）或基于图神经网络（GNN）的推荐算法，以提升推荐结果的准确性和多样性。
   * **用户行为数据挖掘**：扩展用户行为的捕获范围，例如增加对用户浏览记录、停留时间等数据的分析，进一步提升推荐的个性化程度。
   * **增强系统功能**：在用户界面中加入个性化书籍推荐的可视化展示，同时增加用户的社交互动功能，例如书评分享、好友推荐等。
   * **系统性能优化**：通过分布式计算框架（如Hadoop或Spark）处理海量数据，提升系统的并发处理能力和响应速度。
   * **部署与推广**：将系统部署在云服务器上，通过容器化技术（如Docker）实现高效的资源管理，并支持多终端访问（如移动端）。

本项目的完成不仅让我们掌握了理论知识的实际应用，更为未来深入研究和开发提供了方向和动力。在后续的优化与拓展中，我们期待将智能图书推荐系统进一步完善，为用户带来更优质的服务体验。

# 6）小组分工

组长 张俊彪：数据库搭建，ppt制作，视频剪辑，撰写手册

组员 李涵晨：前端项目设计，录制视频，修改手册

组员 周冠旭：页面美化，CF算法，录制视频，修改手册