江 蘇 大 學

JIANG SU UNIVERSITY

图书馆管理系统

课程设计报告

小组成员	班级	学号	分工
葛康（组长）	软件1401	3140608029	普通用户角色编码
孙伟峰	软件1402	3140608036	管理员角色编码
赵玲玲	软件1402	3140608031	文档撰写
刘耀东	软件1402	3140608037	文档撰写

2017 年 12 月 30 日

目录

1 需求分析

1.1 功能分析

系统开发的总目标是实现内部图书借阅管理的系统化、规范化和自动化。

基本需求如下：

（1）能够对图书进行注册登记，也就是将图书的基本信息（如：书的编号、书名、作者、价格等）预先存入数据库中，供以后检索。

（2）能够对借阅人进行注册登记，包括记录借阅人的姓名、编号、班级、年龄、性别、地址、电话等信息。

（3）提供方便的查询方法。如：以书名、作者、出版社、出版时间（确切的时间、时间段、某一时间之前、某一时间之后）等信息进行图书检索，并能反映出图书的借阅情况；以借阅人编号对借阅人信息进行检索；以出版社名称查询出版社联系方式信息。

（4）提供对书籍进行的预先预订的功能。

（5）提供旧书销毁功能，对于淘汰、损坏、丢失的书目可及时对数据库进行修改。

（6）能够对使用该管理系统的用户进行管理，按照不同的工作职能提供不同的功能授权。

（7）提供较为完善的差错控制与友好的用户界面，尽量避免误操作。

将以上基本需求抽象化，可将图书馆管理系统分为基本信息管理、借阅管理、查询管理三大功能模块，其中基本信息管理模块包含数据导出、图书信息管理、出版社信息管理等功能，借阅管理模块包含查询、借阅等功能，查询管理模块包含简单查询、高级查询等功能。其功能模块图如图1所示：

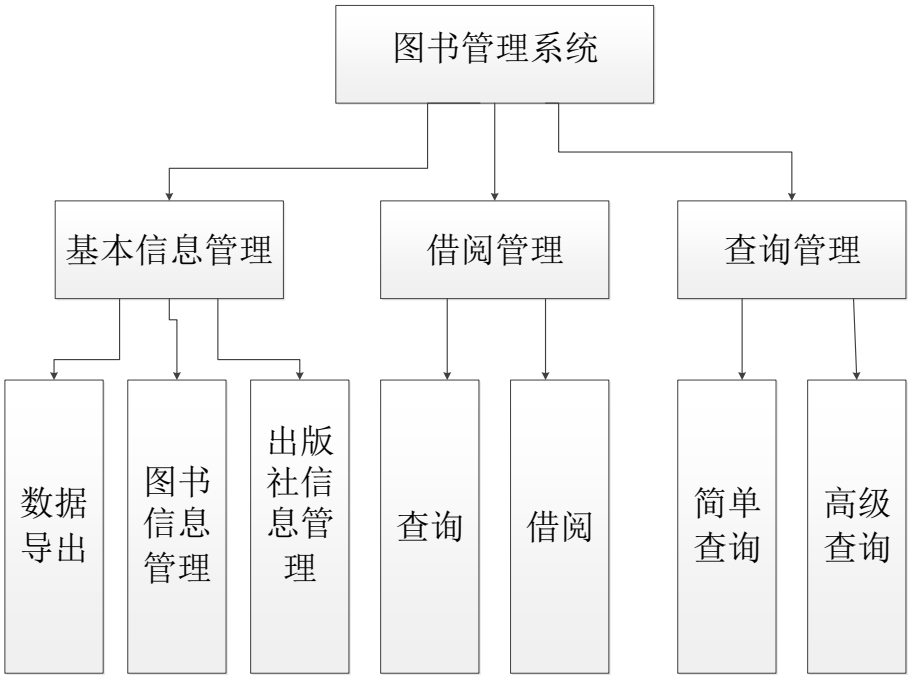


图 1 功能模块图

数据导出：可以将每页的数据，如编码、书名、作者、价格、出版日期、出版社、库存量等信息，以 JSON、XML、CSV、TXT、SQL、Ms-Excel 等格式导出。可以将全页数据导出，也可以只导出选中的数据信息。

图书信息管理：可以新增一条图书信息记录，也可以删除选中的图书记录。另外，也可以通过直接点击某一项直接修改图书相关信息，如通过点击图书名、作者、价格、库存量等直接进行修改。在显示图书相关信息时，也可以自己选择要显示的图书信息，如只显示书名、作者等。

出版社信息管理：可以查看出版社相关信息，如出版社名称、出版社电话、出版社 Email 等，也可以通过快捷键直接向该出版社发送电子邮件。另外，还可以通过点击出版社电话、出版社 Email 等直接对其进行修改。也可以将页面中的所有或选中的出版社相关数据，以 JSON、XML、CSV、TXT、SQL、Ms-Excel 等文件格式导出。也可以新增一条出版社信息记录，同时也可以删除选中的出版社信息记录。在显示出版社相关信息时，也可以自己选择要显示的出版社信息，如只显示出版社名称、出版社电话等。

借阅查询：可以在搜索栏中输入借阅用户名、书名等信息查看借阅的相关信息，其中，显示借阅记录号、用户名、书名、归还日期、状态（欠款/正常）、欠款数等信息。可以将页面数据或选中的数据，以 JSON、XML、CSV、TXT、SQL、Ms-Excel 等文件格式导出。在显示信息时，可以自己选择要显示的借阅信息，如只显示用户名、书名等。

图书借阅：可以新增一条借阅记录，通过图书编号和用户编号确定，其中同一个用户编号最多只能借两本图书，另外若用户欠款，需将欠款补齐才可再次借阅。也可以将选中的借阅记录进行删除。

简单查询：能够根据图书名、出版社名、作者名对图书进行模糊检索，显示编码、书名、作者、价格、出版日期、出版社、库存量等相关信息。

高级查询：通过图书名、出版社名、作者名和出版时间进行综合模糊检索，显示编码、书名、作者、价格、出版日期、出版社、库存量等相关信息。

1.2 角色分析

基于上面抽象出的功能，我们将该系统的角色分为管理员和普通用户。

管理员角色包含登录、图书管理（添加图书信息、删除图书信息、修改图书信息、图书信息导出、查询图书信息等）、借阅登记（添加借阅信息、删除借阅信息、修改借阅信息、查询借阅信息、借阅信息导出）、出版社信息管理（添加出版社信息、删除出版社信息、修改出版社信息、查询出版社信息、出版社信息导出、发送邮件）等功能。

管理员角色其用例图如图 2 所示：

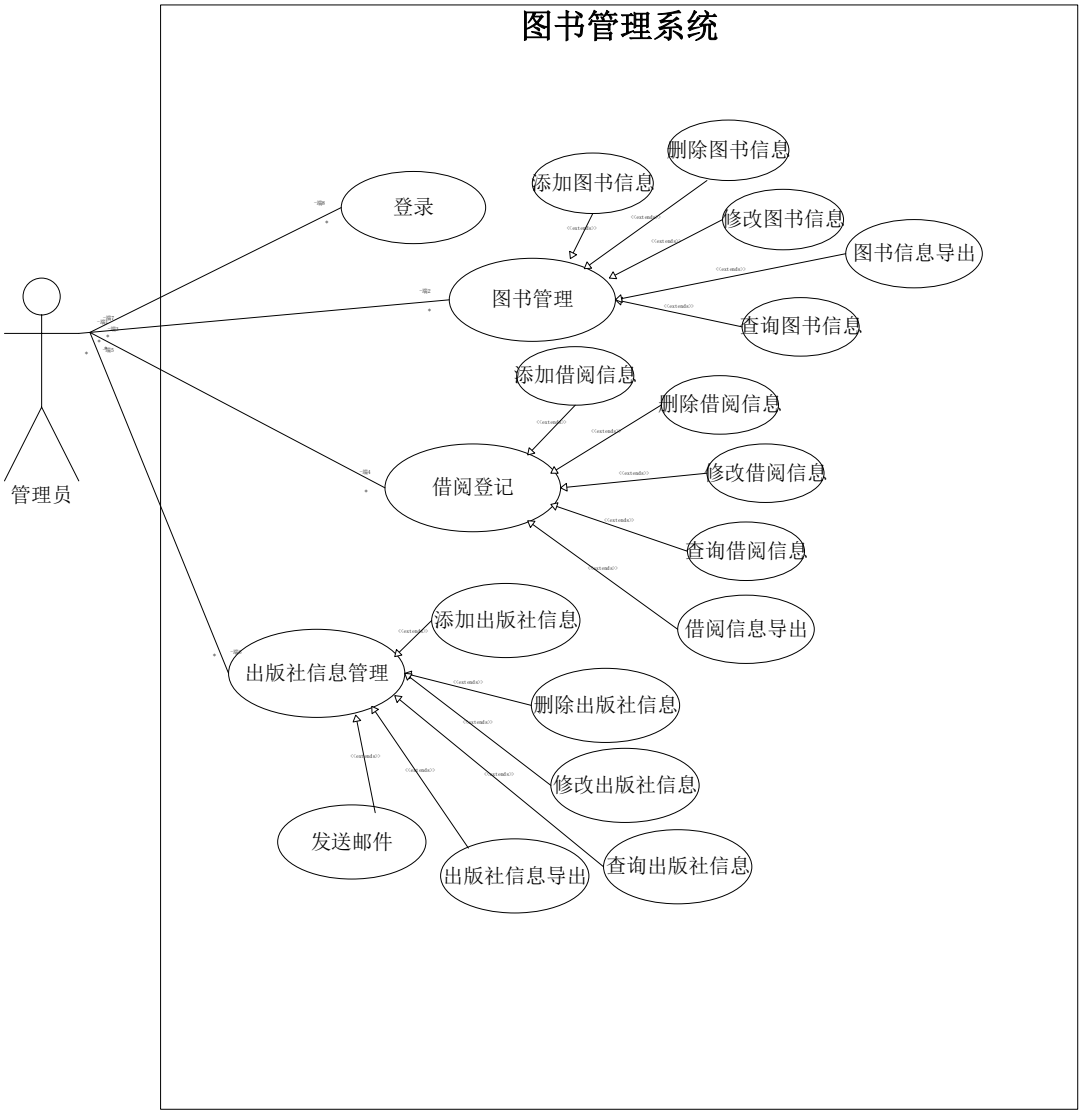


图 2 管理员用例图

普通用户角色包含登录、简单搜索、高级搜索等功能。其用例图如图 3 所示：

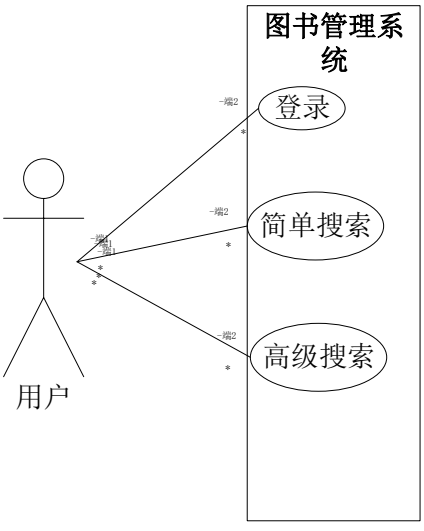


图 3 用户用例图

1.3 存储对象

出版社数据对象包含出版社编号、出版社名称、联系电话、电子邮箱等数据项。其实体图如图 4 所示：

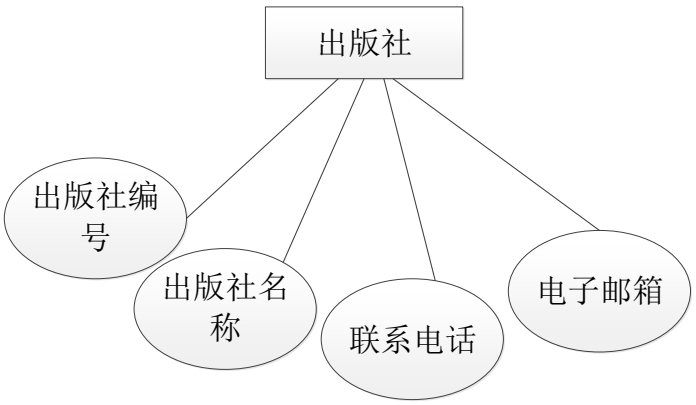


图 4 数据对象-出版社

借阅记录数据对象包含借阅编号、图书编号、用户编号、借阅日期、应还日期等数据项。其实体图如图 5 所示：

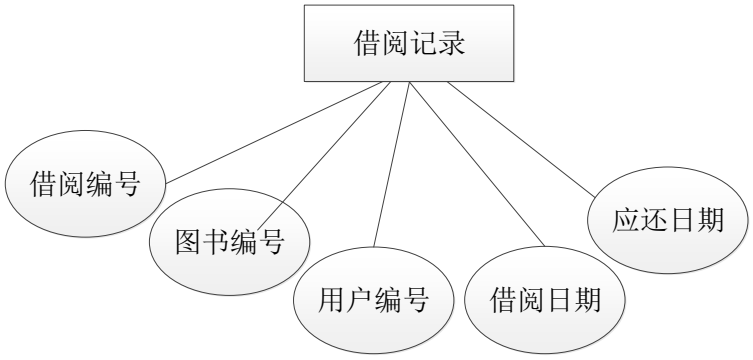


图 5 数据对象-借阅记录

库存量数据对象包含图书编号、余量等数据项。其实体图如图 6 所示：

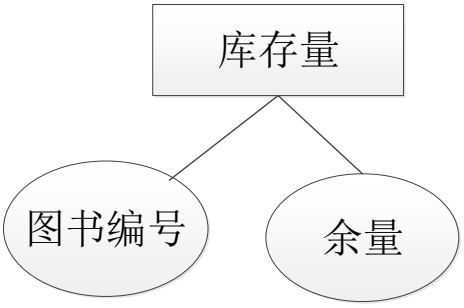


图 6 数据对象-库存量

图书数据对象包含图书编号、图书名称、作者、价格、出版日期、出版社等数据项。其实体图如图 7 所示：

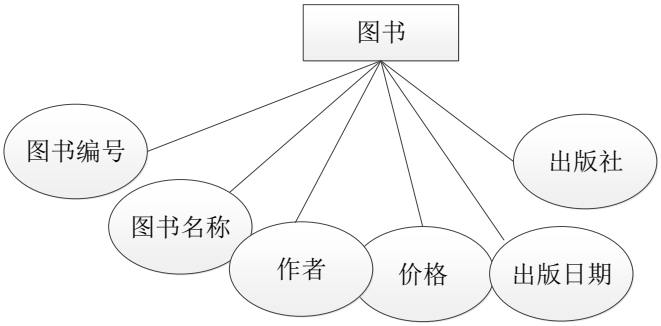


图 7 数据对象-图书

用户数据对象包含用户编号、用户名、登录密码、系别、性别、年龄、地址、联系电话、职称等数据项。其实体图如图 8 所示：

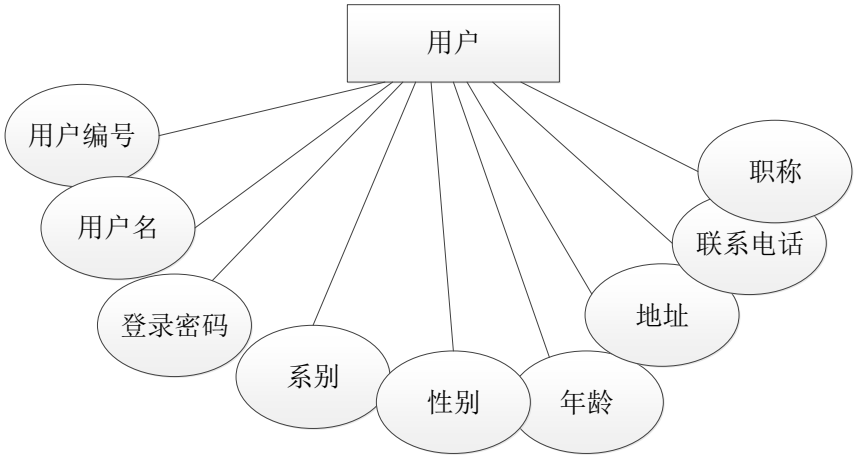


图 8 数据对象-用户

该系统的 E-R 图如图 9 所示：

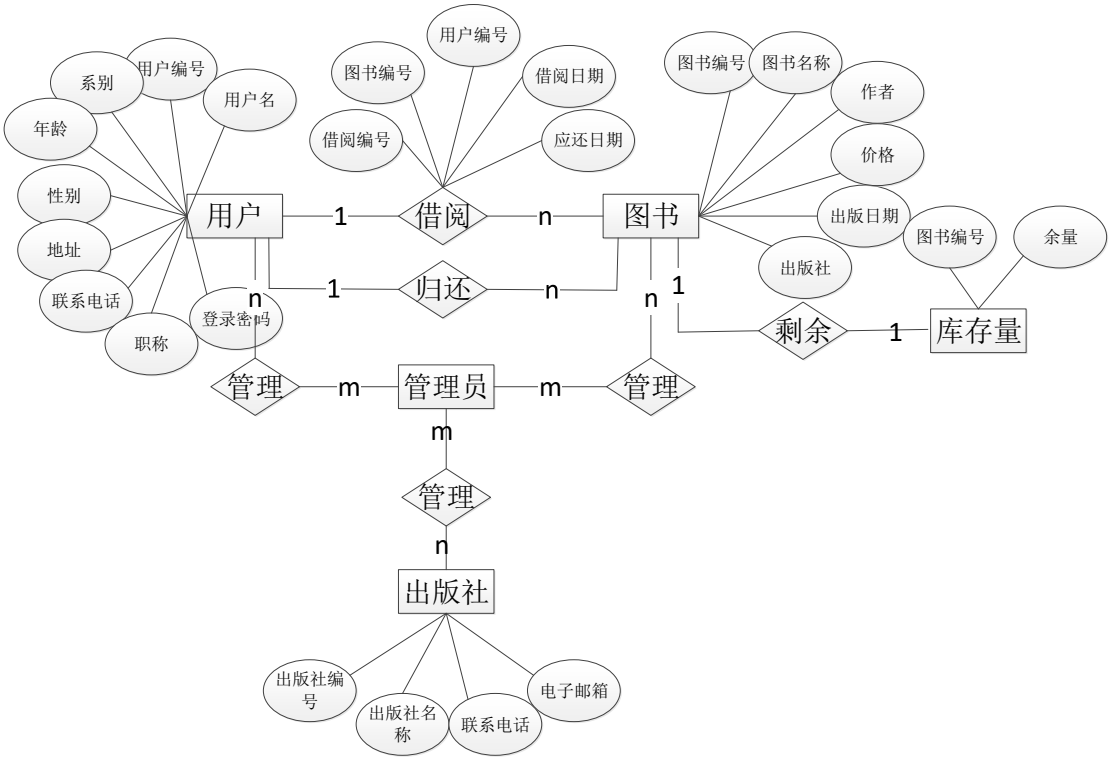


图 9 系统 E-R 图

1.4 系统组件及物理部署

组件图：用来建模系统的各个组件，包括源代码文件，二进制文件，脚本文件，可执行文件之间的关系，他们是通过功能或者文件组织在一起的，使用组件图可以帮助读者了解某个功能位于软件包的那一位置，以及各个版本的软件包含哪些功能。其构成如下：

组件：描述了一个可执行程序，一个库，一个 web 程序等；

接口：接口是组件所提供的的服务，可以理解为一个方法，接口可以有多个，但至少有一个，在 UML 中表示为一个圆形；

实现：就是组件与接口之间的连线，代表了谁实现了这个接口；

依赖：就是指组件使用了另一个组件的接口，依赖于另一个接口的存在。

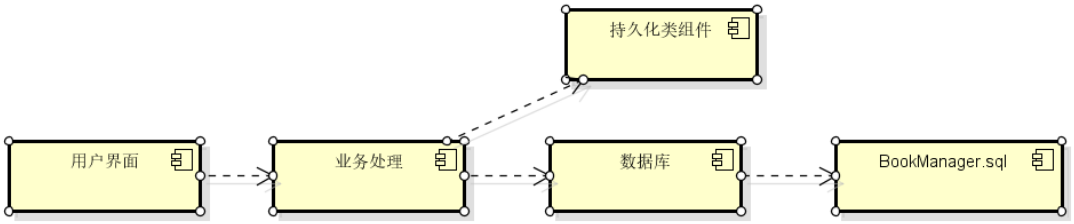


图 10 系统组件图

部署图：是用来显示系统中软件和硬件的物理架构，从部署图中，你可以了解到软件和硬件之间的物理关系，以及处理节点的组件分布情况，使用部署图可以显示运行系统的结构，同时还可传达构成应用程序的硬件和软件元素的配置和部署方案。其元素有：

（1）节点：代表运行时计算机系统中的硬件资源。节点通常拥有一些内存，并具有处理能力。节点的分类：1.处理器：处理器是能执行软件，具有计算能力的节点；2.设备：是没有计算能力的节点。

（2）连接：部署图用连接表示各节点之间通讯路径，连接用一条实线表示，对于企业的计算机系统硬件设备间的关系，我们通常关心的是节点之间是如何连接的，因此描述节点之间的关系一般不使用名称，而是使用构造性描述。

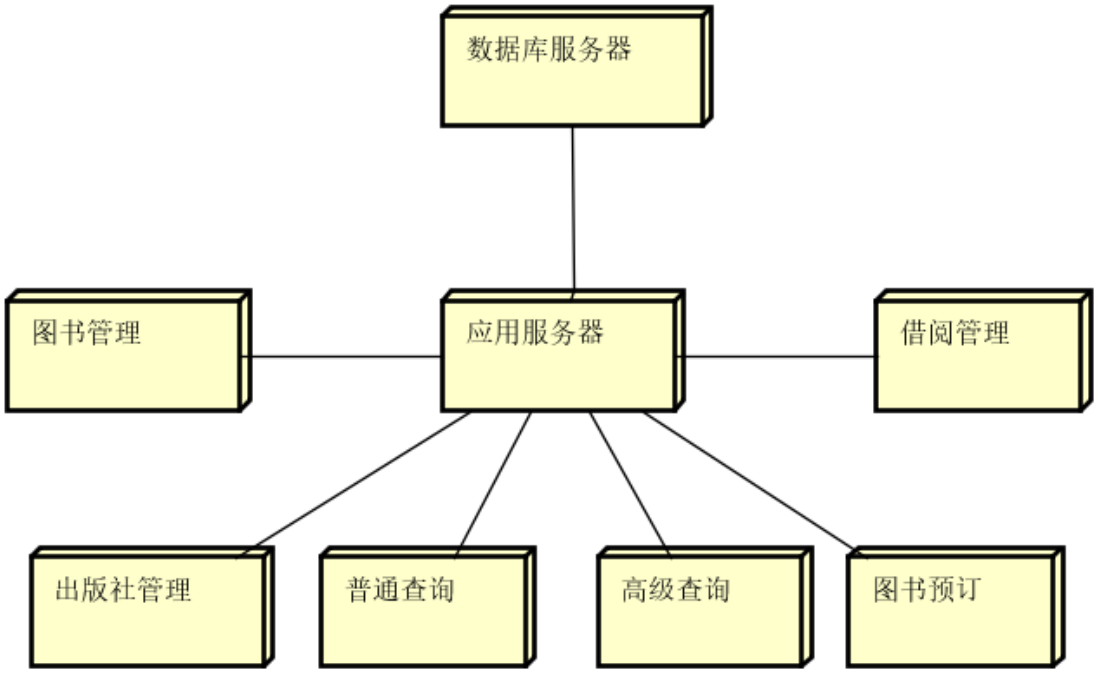


图 11 系统部署图

2 概要设计

2.1 数据流图

管理员：管理员通过读者信息、图书信息、出版社信息等与图书管理系统进行交互。其数据流图如图所示：

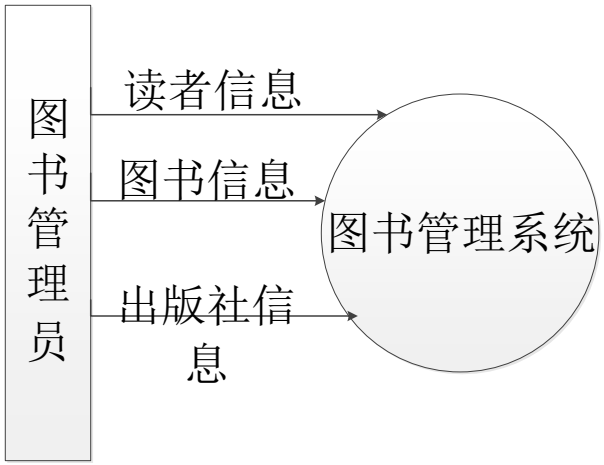


图 12 管理员数据流图

出版社信息管理：包含出版社信息的添加、修改、删除、查询，发送邮件，信息导出等功能，完成与出版社信息表的交互。其数据流图如图所示：

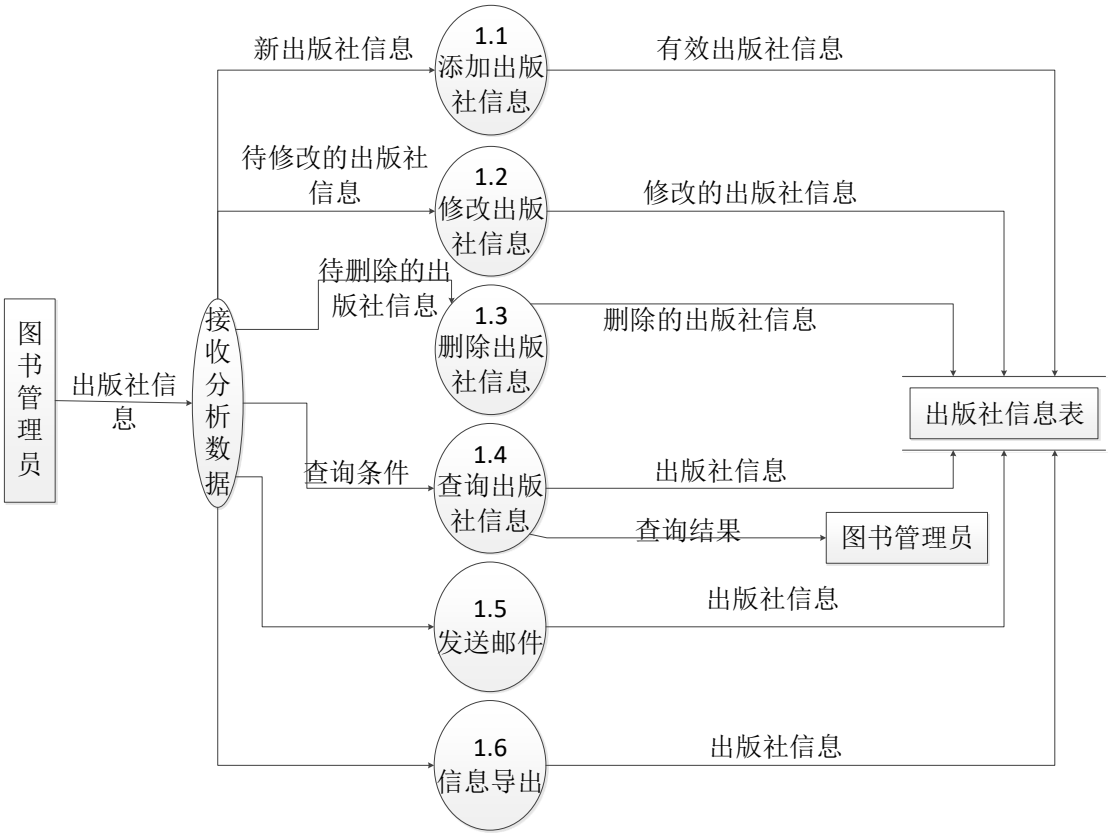


图 13 出版社信息管理数据流图

图书信息管理：图书管理员包含对图书进行添加、修改、删除、查询，信息导出等功能，

通过这些功能完成与图书信息表的交互。另外，图书借阅登记也需要完成对图书信息表的交互。其数据流图如图所示：

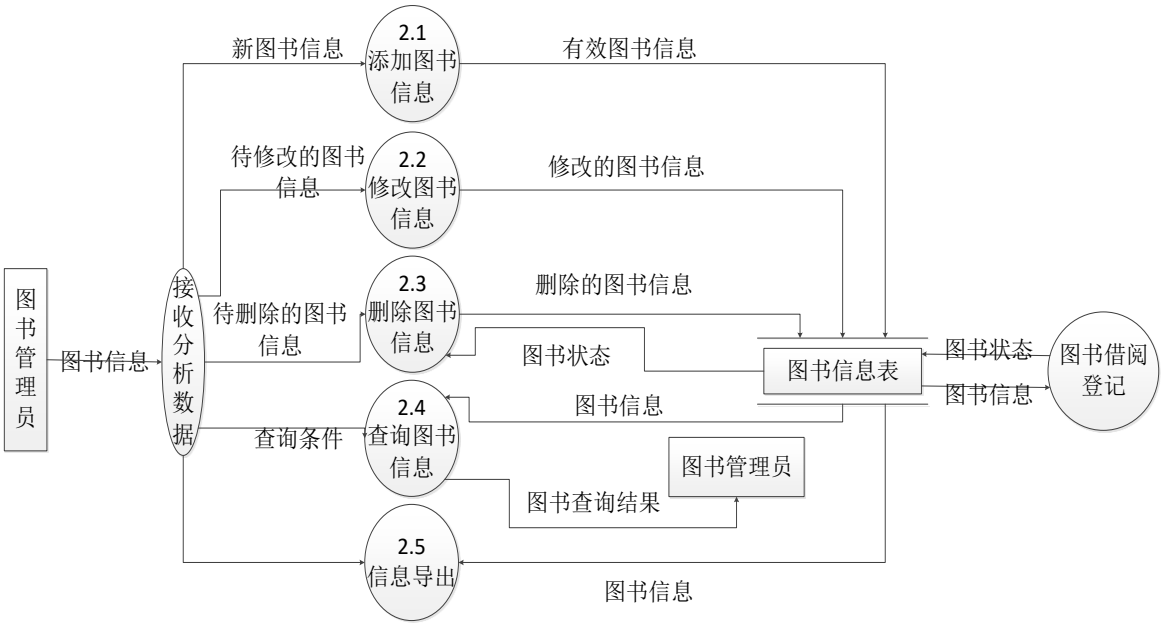


图 14 图书信息管理数据流图

借阅登记管理：当图书管理员收到图书借还信息时，产生两种情况。1.用户借书，管理员输入相关信息，系统根据相关信息自动查阅图书信息表、用户信息表等表，判断是否有违规状况，若状况良好，则产生借阅记录。2.用户还书，管理员删除借阅记录，系统自动修改库存量信息表等表，检查是否有罚款，提示用户及时缴清罚款。其数据流图如图所示：

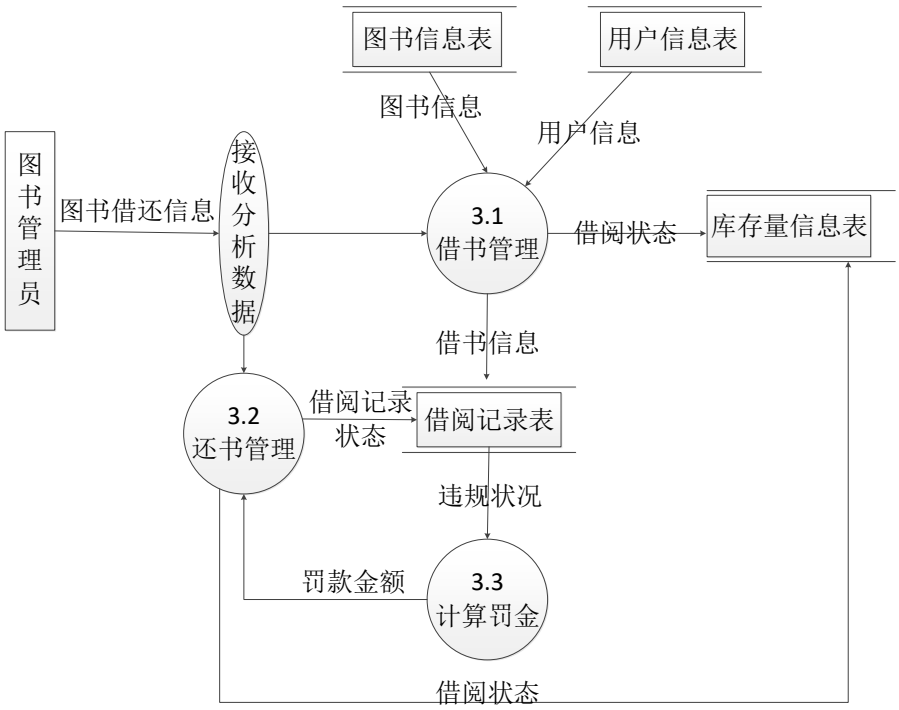


图 15 借阅登记数据流图

2.2 流程图

管理员角色：首先管理员需要输入用户名及密码，完成身份的验证，若验证不通过，则重新登录；否则，进入管理员管理界面。其中包含三个重要功能：图书管理、借阅登记、出版社信息管理。

图书管理：可以新增一条图书信息记录，也可以删除选中的图书记录。另外，也可以通过直接点击某一项直接修改图书相关信息，如通过点击图书名、作者、价格、库存量等直接进行修改。在显示图书相关信息时，也可以自己选择要显示的图书信息，如只显示书名、作者等。

借阅登记：可以新增一条借阅记录，通过图书编号和用户编号确定，其中同一个用户编号最多只能借两本图书，另外若用户欠款，需将欠款补齐才可再次借阅。也可以将选中的借阅记录进行删除。可以在搜索栏中输入借阅用户名、书名等信息查看借阅的相关信息，其中，显示借阅记录号、用户名、书名、归还日期、状态（欠款/正常）、欠款数等信息。可以将页面数据或选中的数据，以 JSON、XML、CSV、TXT、SQL、Ms-Excel 等文件格式导出。在显示信息时，可以自己选择要显示的借阅信息，如只显示用户名、书名等。

出版社信息管理：可以查看出版社相关信息，如出版社名称、出版社电话、出版社 Email 等，也可以通过快捷键直接向该出版社发送电子邮件。另外，还可以通过点击出版社电话、出版社 Email 等直接对其进行修改。也可以将页面中的所有或选中的出版社相关数据，以 JSON、XML、CSV、TXT、SQL、Ms-Excel 等文件格式导出。也可以新增一条出版社信息记录，同时也可以删除选中的出版社信息记录。在显示出版社相关信息时，也可以自己选择要显示的出版社信息，如只显示出版社名称、出版社电话等。

最后，管理员点击退出登录，即可完成注销退出系统。

管理员管理流程图如图所示：

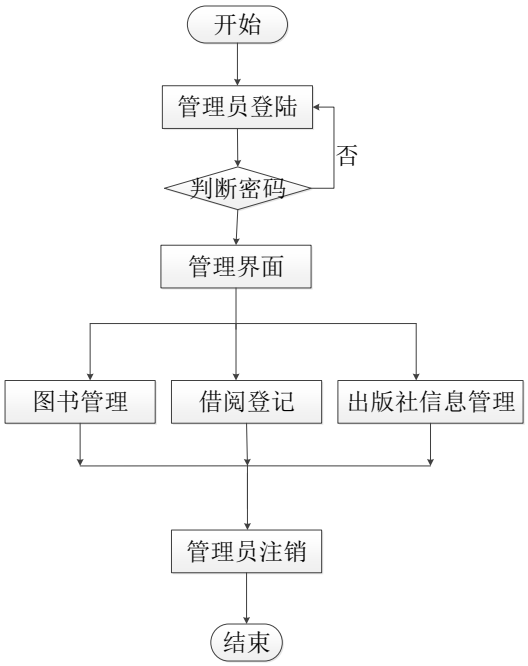


图 16 管理员管理流程图

普通用户角色：首先用户需要输入用户名及密码，完成身份的验证，若验证不通过，则重新登录；否则，进入用户查询界面。其中包含两个重要功能：普通查询、高级查询。

普通查询：能够根据图书名、出版社名、作者名对图书进行模糊检索，显示编码、书名、

作者、价格、出版日期、出版社、库存量等相关信息。

高级查询：通过图书名、出版社名、作者名和出版时间进行综合模糊检索，显示编码、书名、作者、价格、出版日期、出版社、库存量等相关信息。

最后，用户点击退出登录，即可完成注销退出系统。

普通用户查询流程图如图所示：

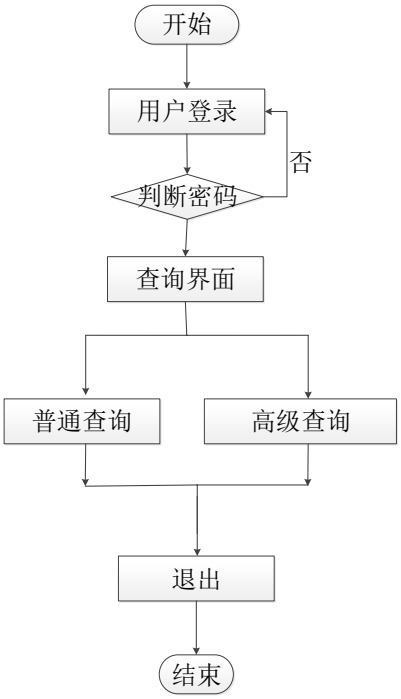


图 17 用户查询流程图

2.3 顺序图

借阅图书顺序：首先管理员根据借阅者提供的图书信息进行查找，若找到该图书，则检查用户状态是否正常，若用户状态正常，且满足借阅图书小于两本，且图书有剩余，那么满足借阅条件，则可以根据用户信息和图书信息生成借阅记录，写入数据库，借书成功。其顺序图如图所示：

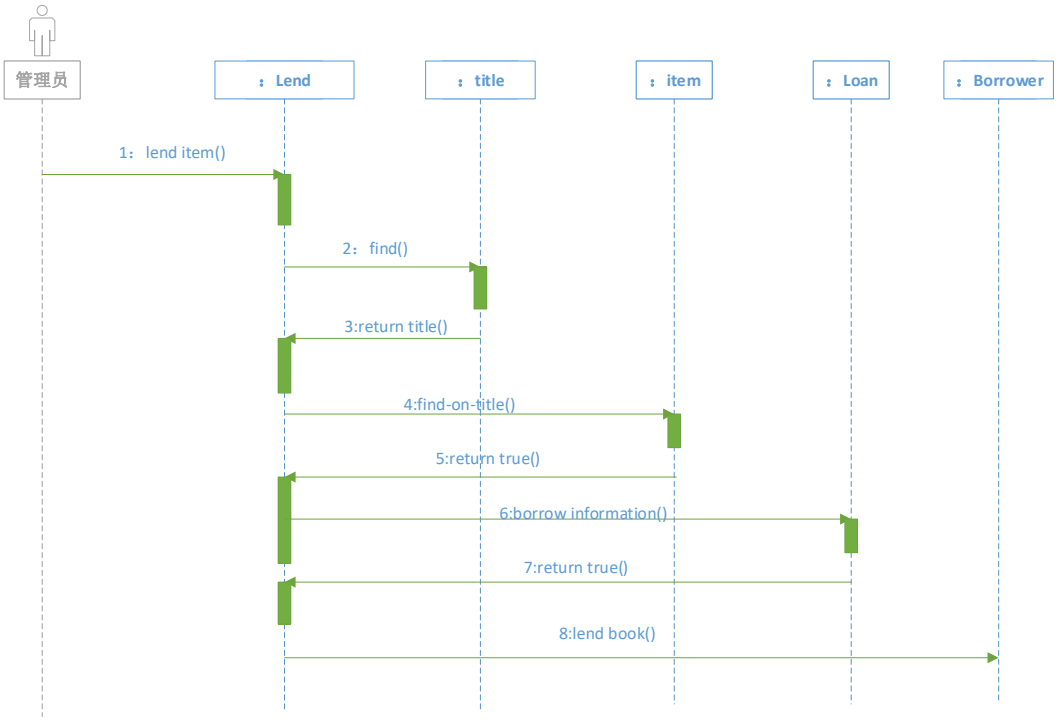


图 18 借阅者借阅图书顺序图

预订图书顺序：首先借阅者登录图书管理系统，然后查找自己要借阅的图书，根据系统返回的图书信息，选择自己要预订的图书，完成预订，退出系统。其顺序图如图所示：

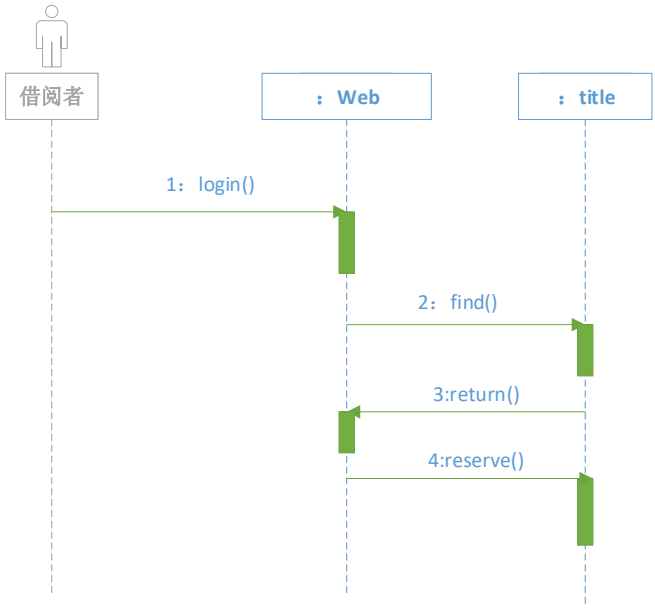


图 19 借阅者预订图书顺序图

归还图书顺序：首先借阅者将所借图书归还给管理员，然后管理员根据图书信息和用户信息查找借阅记录，系统自动返回所找记录，然后管理员删除该条借阅记录，更新数据库，最后完成归还图书活动。其顺序图如图所示：

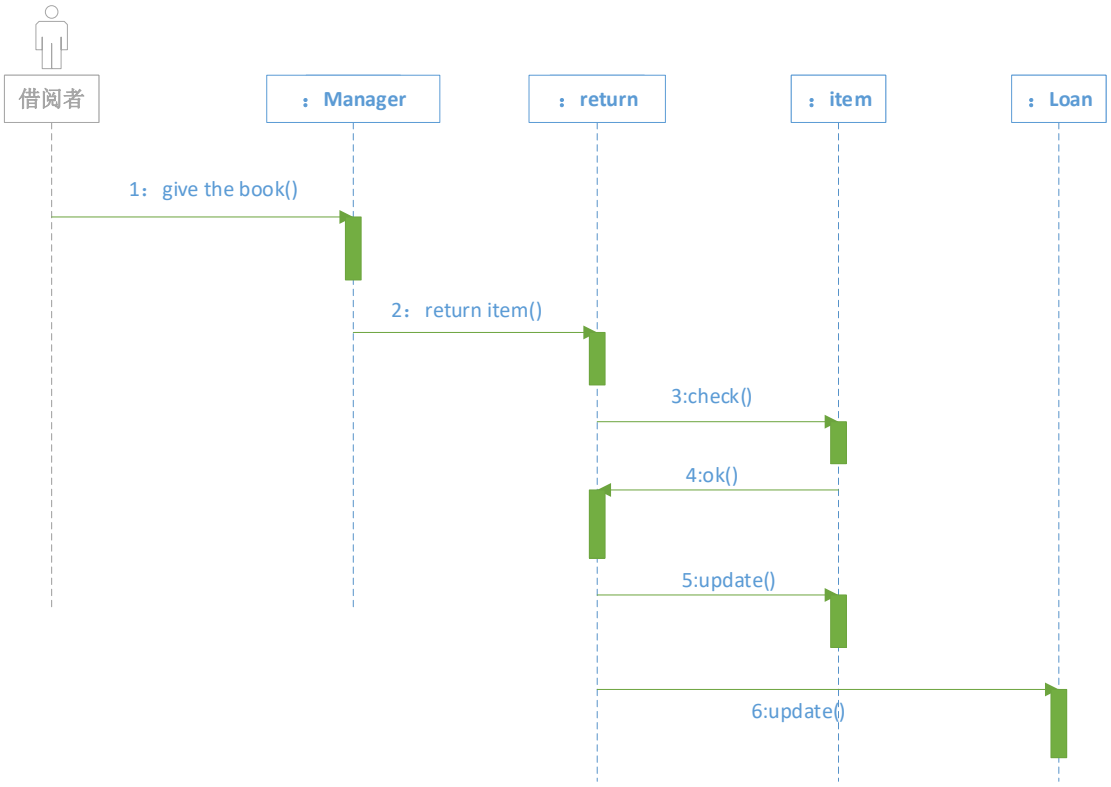


图 20 借阅者归还图书顺序图

2.4 数据存储

本系统数据库存储表主要有五个：用户信息表、图书信息表、出版社信息表、库存量信息表、借阅记录表。

用户信息存储表包含 id、password、name、classmate、sex、age、address、phone、position 等字段。其中 id 作为用户信息的主键。其数据库表的格式如表 1 所示：

表 1 用户信息存储表

列名	数据类型	长度	是否允许为空	其他	列含义
id	Char	30	否	主键	用户编号
password	Char	30	否		登录密码
name	Char	30	否		用户姓名
classmate	Char	30	否		所属班级
sex	Char	30	是		性别
age	Char	30	是		年龄
address	Char	30	是		联系地址
phone	Char	30	是		联系电话
position	Char	30	是		职称

图书信息存储表包含 book_id、book_name、author、price、book_public、book_publicment 等字段。其中 book_id 作为图书信息的主键。其数据库表的格式如表 2 所示：

表 2 图书信息存储表

列名	数据类型	长度	是否允许为空	其他	列含义
book_id	Char	30	否	主键	图书 ISBN 编码
book_name	Char	30	否		图书名称
author	Char	30	是		作者
price	Char	30	是		图书价格
book_public	Char	30	是		出版日期
book_publicment	Char	30	是		出版社

出版社信息存储表包含 public_id、public_name、public_phone、public_email 等字段。其中 public_id 作为出版社信息的主键。其数据库表的格式如表 3 所示：

表 3 出版社信息存储表

列名	数据类型	长度	是否允许为空	其他	列含义
public_id	Char	30	否	主键	出版社编码
public_name	Char	30	否		出版社名称
public_phone	Char	30	是		联系电话
public_email	Char	30	否		出版社邮箱

库存量信息存储表包含 book_id、size 等字段。其中 book_id 作为库存量信息的主键。其数据库表的格式如表 4 所示：

表 4 库存量信息存储表

列名	数据类型	长度	是否允许为空	其他	列含义
book_id	Char	30	否	主键	图书编号
size	Char	30	否		剩余数量

借阅记录存储表包含 loaning_record_id、book_id、id、borrow_date、return_date 等字段。其中 loaning_record_id 作为借阅记录的主键，book_id、id 作为外键。其数据库表的格式如表 5 所示：

表 5 借阅记录存储表

列名	数据类型	长度	是否允许为空	其他	列含义
loaning_record_id	Char	30	否	主键	借阅记录标号
book_id	Char	30	否	外键	图书编号
id	Char	30	否	外键	用户编号
borrow_date	Char	30	否		借出日期
return_date	Char	30	否		应还日期

3 详细设计

3.1 类图

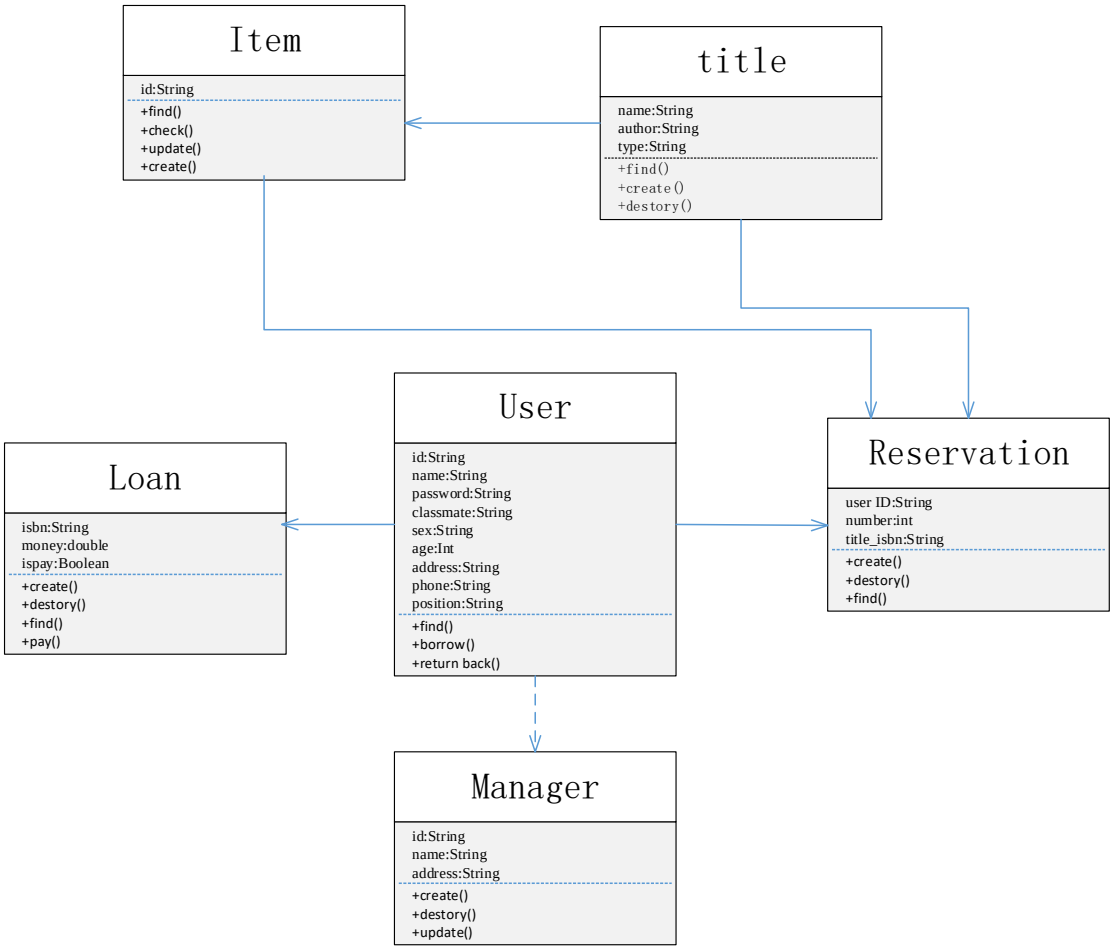


图 21 系统主要类图及其关系

3.2 活动图

管理员图书管理：首先管理员需要输入用户名和密码，进行合法验证，若不合法，则重新登录；否则，登录成功，进入主界面。在图书管理界面，可以进行添加、删除、修改图书信息等操作，从而完成数据库的更改。最后，操作完毕后，退出系统。其活动图如图所示：

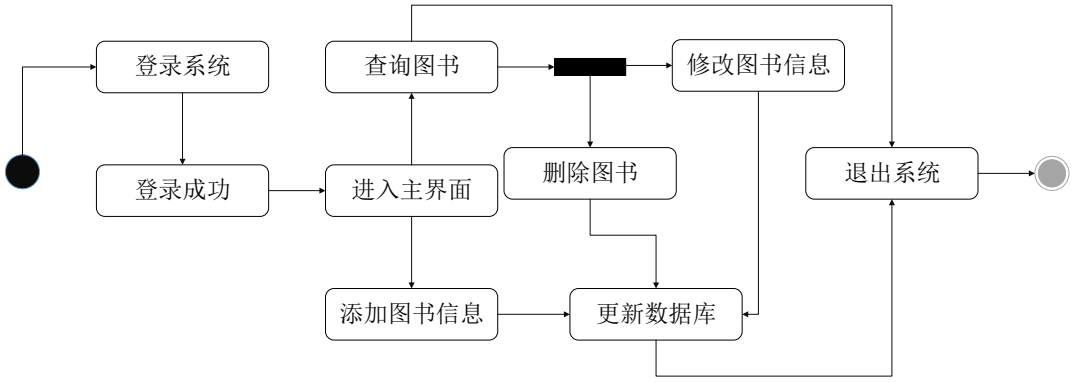


图 22 管理员图书管理活动图

管理员处理借阅及归还活动：首先管理员需要输入用户名和密码，进行合法验证，若不合法，则重新登录；否则，登录成功，进入主界面。在借阅登记界面，主要完成借书处理和还书处理两个功能。借书时，要判断借书量（不能超过两本）和是否有欠款未缴清，若状态正常，则可以借阅，更新借阅信息；否则，不予借阅。还书时，也要判断是否有欠款，若有，则及时提醒借阅者缴清欠款，然后更新借阅信息。最后，退出系统，完成借阅或归还活动。其活动图如图所示：

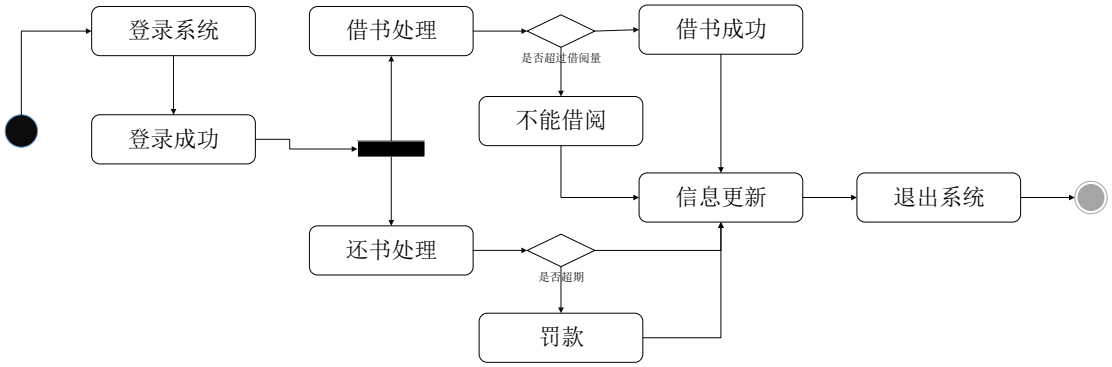


图 23 管理员处理借阅及归还活动图

管理员出版社信息管理：首先管理员需要输入用户名和密码，进行合法验证，若不合法，则重新登录；否则，登录成功，进入主界面。在出版社信息管理界面，可以进行添加、删除、修改出版社信息等操作；还可以点击发送邮件，直接向某一个出版社发送邮件信息。最后，完成出版社信息修改或发送邮件成功，退出系统。其活动图如图所示：

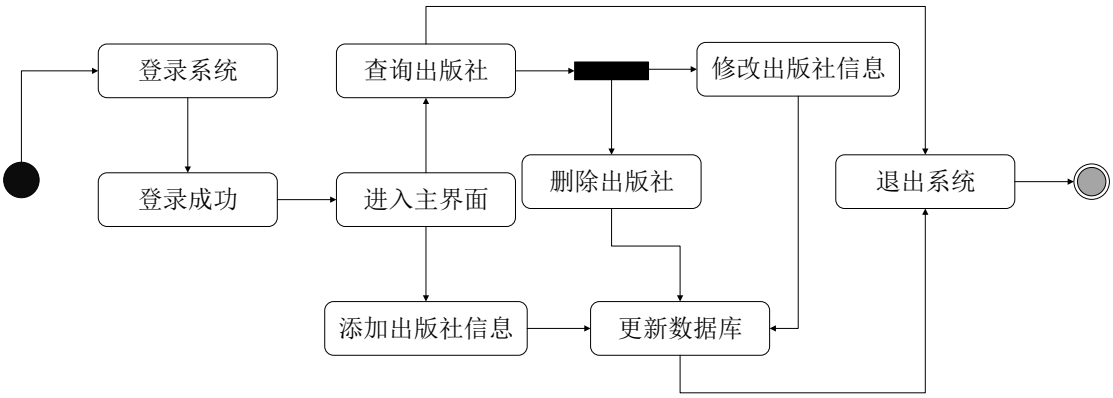


图 24 管理员出版社管理活动图

借阅者查询预订：首先借阅者需要输入用户名和密码，进行合法验证，若不合法，则重新登录；否则，登录成功，进入主界面。在查询界面，有两种查询机制。一个是普通查询，一个是高级查询。普通查询，能够根据图书名、出版社名、作者名对图书进行模糊检索，显示编码、书名、作者、价格、出版日期、出版社、库存量等相关信息。高级查询，通过图书名、出版社名、作者名和出版时间进行综合模糊检索，显示编码、书名、作者、价格、出版日期、出版社、库存量等相关信息。通过查询检索出的结果，可以直接对图书进行预订。其活动图如图所示：

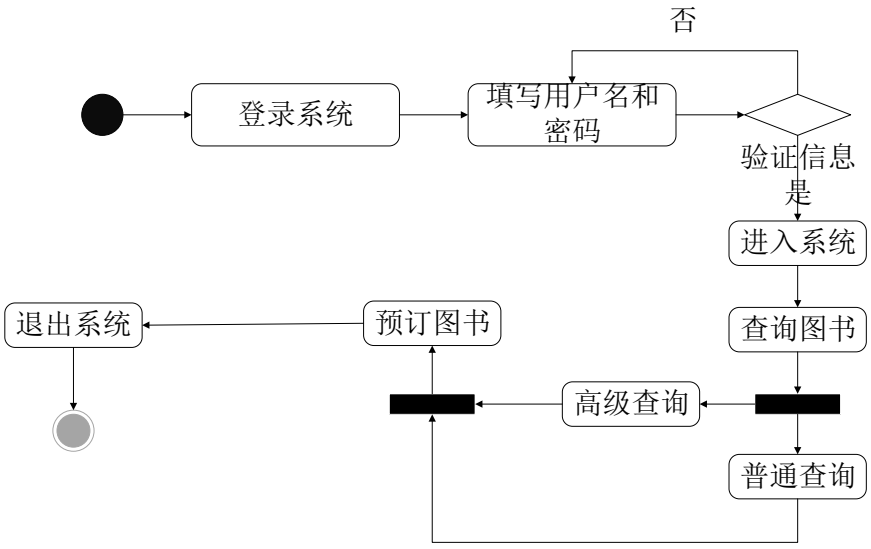


图 25 借阅者查询预订活动图

3.3 数据库 SQL 语句

（1）图书信息表

```
CREATE TABLE `book` (
    `book_id` varchar(30) NOT NULL,
    `book_name` varchar(100) DEFAULT NULL,
    `author` varchar(100) DEFAULT NULL,
    `price` double DEFAULT NULL,
    `book_public` date DEFAULT NULL,
    `book_publicment` varchar(100) DEFAULT NULL,
    PRIMARY KEY (`book_id`)
)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

（2）借阅登记信息表

```
CREATE TABLE `loaning_record` (
    `Loaning_record_id` int(11) NOT NULL,
    `book_id` varchar(30) DEFAULT NULL,
    `id` varchar(30) DEFAULT NULL,
    `borrow_date` date DEFAULT NULL,
    `return_date` date DEFAULT NULL,
    PRIMARY KEY (`Loaning_record_id`)
)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

（3）出版社信息表

```
CREATE TABLE `public` (
    `public_id` int(11) NOT NULL,
    `public_name` varchar(100) DEFAULT NULL,
    `public_phone` varchar(100) DEFAULT NULL,
    `public_email` varchar(100) DEFAULT NULL,
```

```
        PRIMARY KEY (`public_id`)
    )
     ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

（4）库存量信息表

```
    CREATE TABLE `stock_size` (
        `book_id` varchar(30) NOT NULL,
        `size` int(11) DEFAULT NULL,
        PRIMARY KEY (`book_id`),
        CONSTRAINT `FK_ID` FOREIGN KEY (`book_id`) REFERENCES
          `book`(`book_id`)
    )
     ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

（5）用户信息表

```
    CREATE TABLE `user` (
        `id` varchar(30) NOT NULL,
        `password` varchar(30) DEFAULT NULL,
        `name` varchar(100) DEFAULT NULL,
        `classmate` varchar(100) DEFAULT NULL,
        `sex` varchar(10) DEFAULT NULL,
        `age` int(11) DEFAULT NULL,
        `address` varchar(100) DEFAULT NULL,
        `phone` varchar(100) DEFAULT NULL,
        `position` varchar(100) DEFAULT NULL,
        PRIMARY KEY (`id`)
    )
    ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

4 编码

4.1 系统开发环境

基于以上分析，本系统采用 B/S 结构，将系统分为数据服务层、业务逻辑层、表示层三层架构。使用 MySQL 作为数据库，使用 Java、JSP、J2EE 语言，Struts2、Hibernate、jQuery 等框架及 eclipse 开发工具进行开发。

Struts2 以 WebWork 优秀的设计思想为核心，吸收了 Struts 框架的部分优点，提供了一个更加整洁的 MVC 设计模式实现的 Web 应用程序框架。 Struts2 引入了几个新的框架特性：从逻辑中分离出横切关注点的拦截器、减少或者消除配置文件、贯穿整个框架的强大表达式语言、支持可变更和可重用的基于 MVC 模式的标签 API， Struts2 充分利用了从其它 MVC 框架学到的经验和教训，使得 Struts2 框架更加清晰灵活。Struts2 框架是 MVC 流程框架，适合分层开发。框架应用实现不依赖于 Servlet，使用大量的拦截器来处理用户请求，属于无侵入式的设计。

在数据持久层，我们通常使用 JDBC 连接来读写数据库，最常见的就是打开数据库连接、使用复杂的 SQL 语句进行读写、关闭连接，获得的数据又需要在转换或封装后向外传，这是一个非常繁琐的过程。这时出现了 Hibernate 框架，它需要用户创建一系列持久化类，每个类的属性都可以简单地看作和一张数据库表的属性一一对应，当然，也可以实现关系数据库中的各种关系表与类之间的一一对应。这样，当我们需要相关操作时，不必再关注数据库表。不用再去一行行地查询数据库，只需要持久化类就可以完成增删改查的功能。这使得我们的软件开发真正面向对象，而不是面向混乱的代码。

Hibernate 是连接应用程序和数据库的一个中间件，在应用程序中通过创建持久化类来使用 Hibernate，这样应用程序不再关心后台所用的是什么数据库，实现了应用程序的业务逻辑与数据库之间的解耦。在 Hibernate 框架内需要两个文件，一个是 hibernate.cfg.xml，该文件用于配置 Hibernate 和数据库的连接信息；另一个是 XML 映射文件，该文件用来描述持久化类和数据库表、数据列之间的对应关系。

jQuery 是一个快速、简洁的 JavaScript 框架，是继 Prototype 之后又一个优秀的 JavaScript 代码库（或 JavaScript 框架）。jQuery 设计的宗旨是“write Less，Do More”，即倡导写更少的代码，做更多的事情。它封装 JavaScript 常用的功能代码，提供一种简便的 JavaScript 设计模式，优化 HTML 文档操作、事件处理、动画设计和 Ajax 交互。jQuery 的核心特性可以总结为：具有独特的链式语法和短小清晰的多功能接口；具有高效灵活的 css 选择器，并且可对 CSS 选择器进行扩展；拥有便捷的插件扩展机制和丰富的插件。jQuery 兼容各种主流浏览器，如 IE 6.0+、FF 1.5+、Safari 2.0+、Opera 9.0+等。

4.2 系统核心代码

管理员：

Bookcontrol.java 核心代码

```
public class Bookcontrol {
    public static List<Stock_Size> queryAll(){
        Stock_Size stock_size=new Stock_Size();
        SessionFactory sessionFactory = null;
        sessionFactory = new Configuration().configure().buildSessionFactory();
        Session session = sessionFactory.openSession();
        Transaction transaction = session.beginTransaction();
```

```
        Query q=session.createQuery("from Stock_Size stock_size");
        List<Stock_Size> list=q.list();
        transaction.commit();
        session.close();
        sessionFactory.close();
        return list;
    }
    public static String modify(Stock_Size stock){
        SessionFactory sessionFactory = null;
        sessionFactory = new Configuration().configure().buildSessionFactory();
        Session session = sessionFactory.openSession();
        Transaction transaction = session.beginTransaction();
            Stock_Size temp=session.load(Stock_Size.class, stock.getBook().getBook_id());
            temp.setBook(stock.getBook());
            temp.setSize(stock.getSize());
            session.clear();
            session.update(temp);
            transaction.commit();
            session.close();
            sessionFactory.close();
            return "success";
    }
    public static String delete(ArrayList<String> list){
        SessionFactory sessionFactory = null;
        sessionFactory = new Configuration().configure().buildSessionFactory();
        Session session = sessionFactory.openSession();
         Transaction transaction = session.beginTransaction();
          String hql = "";
            for(int i=0;i<list.size();i++) {
                Stock_Size stock=session.load(Stock_Size.class, list.get(i));
                session.delete(stock);

            }
            transaction.commit();
            session.close();
            sessionFactory.close();
            return "success";
    }
    public static List<Stock_Size> querySimple(String bookName){
        Stock_Size stock_size=new Stock_Size();
        SessionFactory sessionFactory = null;
        sessionFactory = new Configuration().configure().buildSessionFactory();
        // 2. 创建一个 Session 对象
        Session session = sessionFactory.openSession();
```

```
        // 3. 开启事务
        Transaction transaction = session.beginTransaction();
        Query q=session.createQuery("from Stock_Size as stock_size where stock_size.book.book_name like :name or stock_size.book.author like :author"
                + " or stock_size.book.book_publicment like :publicment");
        System.out.print(bookName);
        q.setString("name", "%" + bookName + "%");
        q.setString("author", "%" + bookName + "%");
        q.setString("publicment", "%" + bookName + "%");
        List<Stock_Size> list=q.list();
        transaction.commit();
        // 6. 关闭 Session
        session.close();
        // 7. 关闭 SessionFactory 对象
        sessionFactory.close();
        return list;
    }
    public static List<Stock_Size> queryHighLevel(String book_name,String book_author,String book_publish,String book_time_start,
            String book_time_end){
        book_time_start = book_time_start + "-1-1";
        book_time_end = book_time_end + "-12-31";
        System.out.println(book_time_start);
        java.sql.Date beginDate = java.sql.Date.valueOf(book_time_start);
        java.sql.Date endDate = java.sql.Date.valueOf(book_time_end);
        Stock_Size stock_size=new Stock_Size();
        SessionFactory sessionFactory = null;
        sessionFactory = new Configuration().configure().buildSessionFactory();
        // 2. 创建一个 Session 对象
        Session session = sessionFactory.openSession();
        // 3. 开启事务
        Transaction transaction = session.beginTransaction();
        Query q=session.createQuery("from Stock_Size as stock_size where stock_size.book.book_name like :name "
                + "and stock_size.book.author like :author and stock_size.book.book_publicment like :publish "
                + "and stock_size.book.book_public >=:beginDate and stock_size.book.book_public <=:endDate");
        //q.setString(0, bookName);
        System.out.println(book_name);
        q.setString("name", "%" + book_name + "%");
        q.setString("author", "%" + book_author + "%");
        q.setString("publish", "%" + book_publish + "%");
        q.setDate("beginDate", beginDate);
```

```
            q.setDate("endDate", endDate);
            List<Stock_Size> list=q.list();
            transaction.commit();
            // 6. 关闭 Session
            session.close();
            // 7. 关闭 SessionFactory 对象
            sessionFactory.close();
            return list;
        }
        public static List<Book> queryNull(String bookName){
            Book book = new Book();
            SessionFactory sessionFactory = null;
            sessionFactory = new Configuration().configure().buildSessionFactory();
            // 2. 创建一个 Session 对象
            Session session = sessionFactory.openSession();
            // 3. 开启事务
            Transaction transaction = session.beginTransaction();
            Query q=session.createQuery("from Book book where book.book_name=?");
            q.setString(0, "null");
            List<Book> list=q.list();
            transaction.commit();
            // 6. 关闭 Session
            session.close();
            // 7. 关闭 SessionFactory 对象
            sessionFactory.close();
            return list;
        }
}
```

Loaning_Recordcontrol.java 核心代码

```
public class Loaning_Recordcontrol {
        public static ArrayList<User_loan> queryAll() {
            SessionFactory sessionFactory = null;
            sessionFactory = new Configuration().configure().buildSessionFactory();
            // 2. 创建一个 Session 对象
            Session session = sessionFactory.openSession();
            // 3. 开启事务
            Transaction transaction = session.beginTransaction();
            Query query = session.createQuery(
                    "select
user.name,book.book_name,loaning_record.return_date,loaning_record.Loaning_record_id "
                            + "from Book as book,User as user,Loaning_Record as
loaning_record "
                            + "where loaning_record.id=user.id and
loaning_record.book_id=book.book_id");
```

```
            List<Object> list = query.list();
            transaction.commit();
            // 6. 关闭 Session
            // System.out.println("提示"+list.size());
            session.close();
            // 7. 关闭 SessionFactory 对象
            sessionFactory.close();
            ArrayList<User_loan> userlist = new ArrayList<>();
            User_loan user_loan = null;
            Date d = new Date();
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
            String dateNowStr = sdf.format(d);
            // System.out.println(dateNowStr);
            for (int i = 0; i < list.size(); i++) {
                user_loan = new User_loan();
                Object[] object = (Object[]) list.get(i);// 每行记录不在是一个对象 而是一个数
                // System.out.println("tishi "+(int)object[3] );
                user_loan.setUsername((String) object[0]);
                user_loan.setBookname((String) object[1]);
                user_loan.setReturn_date((String) object[2]);
                user_loan.setId((int) object[3]);
                if (Method.compare_date(dateNowStr, (String) object[2]) == 1) {
                    user_loan.setState("正常");
                    user_loan.setMoney("0");
                } else {
                    user_loan.setState("欠款");
                    user_loan.setMoney(String.valueOf(Method.differentDays(dateNowStr,
(String) object[2]) * 0.1));
                }
                userlist.add(user_loan);
            }
            return userlist;
        }
        public static String addloan(String book_id, String user_id) throws ParseException {
            SessionFactory sessionFactory = null;
            sessionFactory = new Configuration().configure().buildSessionFactory();
            // 2. 创建一个 Session 对象
            Session session = sessionFactory.openSession();
            // 3. 开启事务
            Transaction transaction = session.beginTransaction();
            User user = session.load(User.class, user_id);
            // System.out.println("添加记录查询"+user.getPosition());
            // 4. 执行保存操作
            Loaning_Record loan = new Loaning_Record();
```

```
            loan.setBook_id(book_id);
            loan.setId(user_id);
            Date date = new Date();
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
            String dateNowStr = sdf.format(date);
            loan.setBorrow_date(dateNowStr);
            if (user.getPosition().equals("本科生")) {
                loan.setReturn_date(Method.addDate(dateNowStr, 30));
            } else if (user.getPosition().equals("研究生")) {
                loan.setReturn_date(Method.addDate(dateNowStr, 40));
            }
            // System.out.println("哈哈"+loan.getBook_id()+" "
            // +loan.getBorrow_date()+" "+
            // loan.getReturn_date()+" "+loan.getId());
            session.save(loan);
            Stock_Size stock = session.load(Stock_Size.class, book_id);
            stock.setSize(stock.getSize() - 1);
            session.update(stock);
            // 5. 提交时务
            transaction.commit();
            // 6. 关闭 Session
            session.close();
            // 7. 关闭 SessionFactory 对象
            sessionFactory.close();
            return "success";
        }
        public static int queryInfo(String user_id) {
            // TODO 自动生成的方法存根
            SessionFactory sessionFactory = null;
            sessionFactory = new Configuration().configure().buildSessionFactory();
            // 2. 创建一个 Session 对象
            Session session = sessionFactory.openSession();
            // 3. 开启事务
            Transaction transaction = session.beginTransaction();
            Query query = session.createQuery("from Loaning_Record as loaning_record " + "where
loaning_record.id=?");
            query.setString(0, user_id);
            List<Loaning_Record> list = query.list();
            transaction.commit();
            // 6. 关闭 Session
            // System.out.println("提示"+list.size());
            session.close();
            // 7. 关闭 SessionFactory 对象
            sessionFactory.close();
```

```
            return list.size();
        }
}
```

Placecontrol.java 核心代码

```
public class Placecontrol {
    public static List<Province> queryProvince() {
        // TODO 自动生成的方法存根
        SessionFactory sessionFactory = null;
        sessionFactory = new Configuration().configure().buildSessionFactory();
        // 2. 创建一个 Session 对象
        Session session = sessionFactory.openSession();
        // 3. 开启事务
        Transaction transaction = session.beginTransaction();
        Query query = session.createQuery("from Province province");
        List<Province> list = query.list();
        transaction.commit();
        // 6. 关闭 Session
        session.close();
        // 7. 关闭 SessionFactory 对象
        sessionFactory.close();
        return list;
    }
}
```

Publiccontrol.java 核心代码

```
public class Publiccontrol {
    public static String addpublic(String public_name,String public_phone,String public_email){
        SessionFactory sessionFactory = null;
        sessionFactory = new Configuration().configure().buildSessionFactory();
        // 2. 创建一个 Session 对象
        Session session = sessionFactory.openSession();
        // 3. 开启事务
        Transaction transaction = session.beginTransaction();
        // 4. 执行保存操作
          Public pub=new Public();
            pub.setPublic_name(public_name);
            pub.setPublic_phone(public_phone);
            pub.setPublic_email(public_email);
            session.save(pub);
            // 5. 提交时务
            transaction.commit();
            // 6. 关闭 Session
            session.close();
            // 7. 关闭 SessionFactory 对象
            sessionFactory.close();
```

```
            return "success";
        }
        public static String deletepublic(ArrayList<String> list){
            SessionFactory sessionFactory = null;
            sessionFactory = new Configuration().configure().buildSessionFactory();
            // 2. 创建一个 Session 对象
            Session session = sessionFactory.openSession();
            // 3. 开启事务
            Transaction transaction = session.beginTransaction();
            for(int i=0;i<list.size();i++){
                Public pub=new Public();
                pub.setPublic_id(Integer.parseInt(list.get(i)));
                session.delete(pub);
            }
            transaction.commit();
            // 6. 关闭 Session
            session.close();
            // 7. 关闭 SessionFactory 对象
            sessionFactory.close();
            return "success";
        }
        public static String modify(Public pub){
        SessionFactory sessionFactory = null;
        sessionFactory = new Configuration().configure().buildSessionFactory();
        // 2. 创建一个 Session 对象
        Session session = sessionFactory.openSession();
        // 3. 开启事务
        Transaction transaction = session.beginTransaction();
        // 4. 执行更新操作
            session.update(pub);
            // 5. 提交时务
            transaction.commit();
            // 6. 关闭 Session
            session.close();
            // 7. 关闭 SessionFactory 对象
            sessionFactory.close();
            return "success";
        }
}
```

Stockcontrol.java 核心代码

```
public class Stockcontrol {
        public static int quertsome(String book_id){
            SessionFactory sessionFactory = null;
            sessionFactory = new Configuration().configure().buildSessionFactory();
```

```
            // 2. 创建一个 Session 对象
            Session session = sessionFactory.openSession();
            // 3. 开启事务
             Transaction transaction = session.beginTransaction();
             Stock_Size u=(Stock_Size)session.get(Stock_Size.class, book_id);
             transaction.commit();

            // 6. 关闭 Session
            session.close();
            // 7. 关闭 SessionFactory 对象
            sessionFactory.close();
            return u.getSize();
    }
}
```

Usercontrol.java 核心代码

```
public class Usercontrol {
    public static String addUser(String id, String name, String classmate, String sex, int age, String
address, String phone,String password, String position) {
            SessionFactory sessionFactory = null;
            sessionFactory = new Configuration().configure().buildSessionFactory();
            // 2. 创建一个 Session 对象
            Session session = sessionFactory.openSession();
            // 3. 开启事务
            Transaction transaction = session.beginTransaction();
            // 4. 执行保存操作
                User people = new User();
                people.setId(id);
                people.setAddress(address);
                people.setAge(age);
                people.setClassmate(classmate);
                people.setName(name);
                people.setPhone(phone);
                people.setSex(sex);
                people.setPassword(password);
                people.setPosition(position);
                session.save(people);
                // 5. 提交时务
                transaction.commit();
                // 6. 关闭 Session
                session.close();
                // 7. 关闭 SessionFactory 对象
                sessionFactory.close();
                return "success";
    }
```

```
        public static User isIN(String id){
          SessionFactory sessionFactory = null;
          sessionFactory = new Configuration().configure().buildSessionFactory();
          // 2. 创建一个 Session 对象
          Session session = sessionFactory.openSession();
          // 3. 开启事务
           Transaction transaction = session.beginTransaction();
           User u=(User)session.get(User.class, id);
          transaction.commit();
                // 6. 关闭 Session
          session.close();
                // 7. 关闭 SessionFactory 对象
          sessionFactory.close();
          return u;
    }
        public static List<User> queryAll(){
                Stock_Size stock_size=new Stock_Size();
                SessionFactory sessionFactory = null;
                sessionFactory = new Configuration().configure().buildSessionFactory();
                // 2. 创建一个 Session 对象
                Session session = sessionFactory.openSession();
                // 3. 开启事务
                Transaction transaction = session.beginTransaction();
                Query q=session.createQuery("from User user where position=? or position=?");
                q.setString(0, "本科生");
                q.setString(1, "研究生");
                List<User> list=q.list();
                transaction.commit();
                // 6. 关闭 Session
                session.close();
                // 7. 关闭 SessionFactory 对象
                sessionFactory.close();
                return list;
          }
}
```

普通用户：

普通查询 QuerySimple.java 核心代码

```
    public void querySimple() throws IOException{
        this.bookName= new String(bookName.getBytes("iso-8859-1"),"utf-8");
        System.out.println(pageSize+" "+pageNumber);
        System.out.println(bookName);
        ArrayList<Stock_Size>                                    list=(ArrayList<Stock_Size>)
Bookcontrol.querySimple(bookName);
           ArrayList<Stock_Size> temp=new ArrayList<>();
```

```
            int compare=0;
            int size=Integer.parseInt(pageSize); //单页的个数
            int number=Integer.parseInt(pageNumber); //当前的页码
            int remain=list.size()-(number-1)*size;
            if(remain>=size){
                compare=size;
            }else{
                compare=list.size();
            }
            System.out.println("i="+(number-1)*size+"    compare="+compare);
            for(int i=(number-1)*size;i<compare;i++){
                temp.add(list.get(i));
            }
            Map<String,Object> map = new HashMap<String,Object>();
            map.put("total", list.size());
            map.put("rows",temp);
            ServletActionContext.getResponse().setContentType("application/json;charset=UTF-
8");
            PrintWriter out = ServletActionContext.getResponse().getWriter();
            JSONObject json = JSONObject.fromObject(map);
            System.out.println(json.toString());
            out.println(json);
            out.flush();
            out.close();
    }
    public void queryNull() throws IOException{
        this.bookName= new String(bookName.getBytes("iso-8859-1"),"utf-8");
        System.out.println(pageSize+" "+pageNumber);
        System.out.println(bookName);
        ArrayList<Book> list=(ArrayList<Book>) Bookcontrol.queryNull(bookName);
          ArrayList<Book> temp=new ArrayList<>();
          int compare=0;
          int size=Integer.parseInt(pageSize); //单页的个数
          int number=Integer.parseInt(pageNumber); //当前的页码
          int remain=list.size()-(number-1)*size;
          if(remain>=size){
              compare=size;
          }else{
              compare=list.size();
          }
          System.out.println("i="+(number-1)*size+"    compare="+compare);
          for(int i=(number-1)*size;i<compare;i++){
              temp.add(list.get(i));
          }
```

```
            Map<String,Object> map = new HashMap<String,Object>();
            map.put("total", list.size());
            map.put("rows",temp);
            ServletActionContext.getResponse().setContentType("application/json;charset=UTF-
8");
            PrintWriter out = ServletActionContext.getResponse().getWriter();
            JSONObject json = JSONObject.fromObject(map);
            System.out.println(json.toString());
            out.println(json);
            out.flush();
            out.close();
    }
```

高级查询 QueryHighLevel.java 核心代码

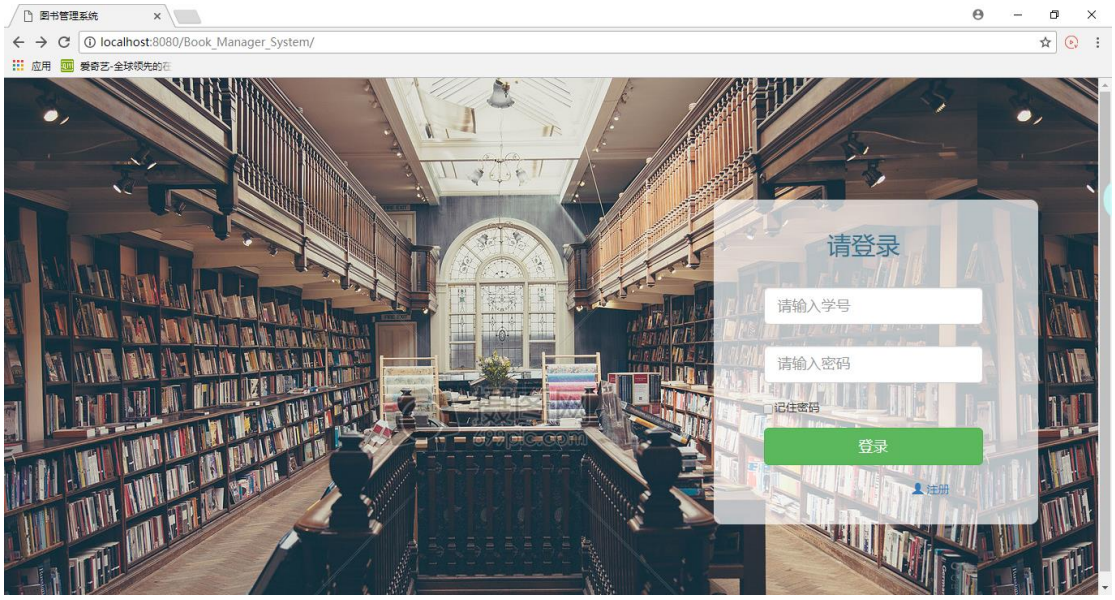
```
    public void queryHighLevel() throws IOException{
        this.book_name = new String(book_name.getBytes("iso-8859-1"),"utf-8");
        this.book_author = new String(book_author.getBytes("iso-8859-1"),"utf-8");
        this.book_publish = new String(book_publish.getBytes("iso-8859-1"),"utf-8");
        this.book_time_start = new String(book_time_start.getBytes("iso-8859-1"),"utf-8");
        this.book_time_end = new String(book_time_end.getBytes("iso-8859-1"),"utf-8");
        ArrayList<Stock_Size>                                    list=(ArrayList<Stock_Size>)
Bookcontrol.queryHighLevel(book_name, book_author,
                book_publish, book_time_start, book_time_end);
            ArrayList<Stock_Size> temp=new ArrayList<>();
            int compare=0;
            int size=Integer.parseInt(pageSize); //单页的个数
            int number=Integer.parseInt(pageNumber); //当前的页码
            int remain=list.size()-(number-1)*size;
            if(remain>=size){
                compare=size;
            }else{
                compare=list.size();
            }
            System.out.println("i="+(number-1)*size+"    compare="+compare);
            for(int i=(number-1)*size;i<compare;i++){
                temp.add(list.get(i));
            }
            Map<String,Object> map = new HashMap<String,Object>();
            map.put("total", list.size());
            map.put("rows",temp);
            ServletActionContext.getResponse().setContentType("application/json;charset=UTF-
8");
            PrintWriter out = ServletActionContext.getResponse().getWriter();
            JSONObject json = JSONObject.fromObject(map);
            System.out.println(json.toString());
```

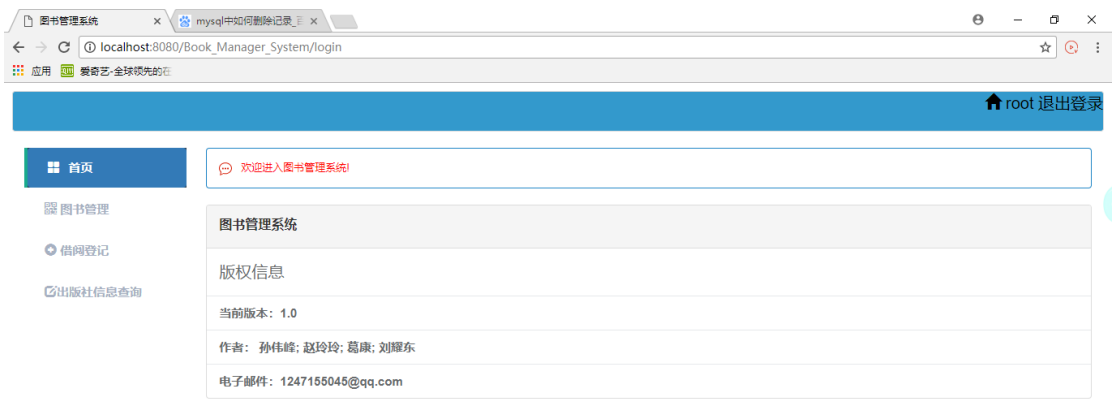
```
            out.println(json);
            out.flush();
            out.close();
    }
```

5 系统测试

5.1 管理员登录用例测试

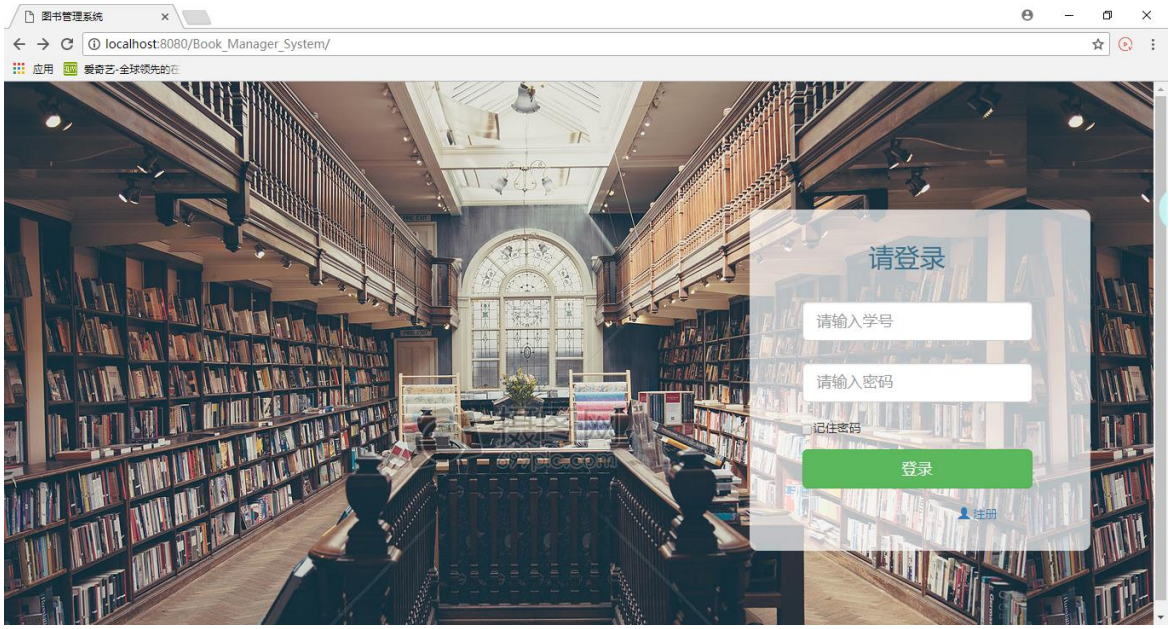
用例编号	001	用例标识	logon_admin
项目名称	图书管理系统	模块名称	管理员登录
用例作者	刘耀东	测试人员	刘耀东
测试类型	功能测试	设计日期	2018-01-01
测试方法	黑盒	测试日期	2018-01-01
用例描述	管理员若合法，则能正常进入管理员界面。		
前置条件	管理员由学校相关部门指定，不可自己注册。		
描述/输入/操作	期望结果	真实结果	备注
某管理员已被指定，且被录入数据库，且用户名和密码输入正确	跳转到管理员界面	跳转到管理员界面	该功能已完善
某管理员未被指定，或尚未录入数据库，或用户名和密码输入错误	未跳转到管理员界面，重新登录	未跳转到管理员界面，重新登录	

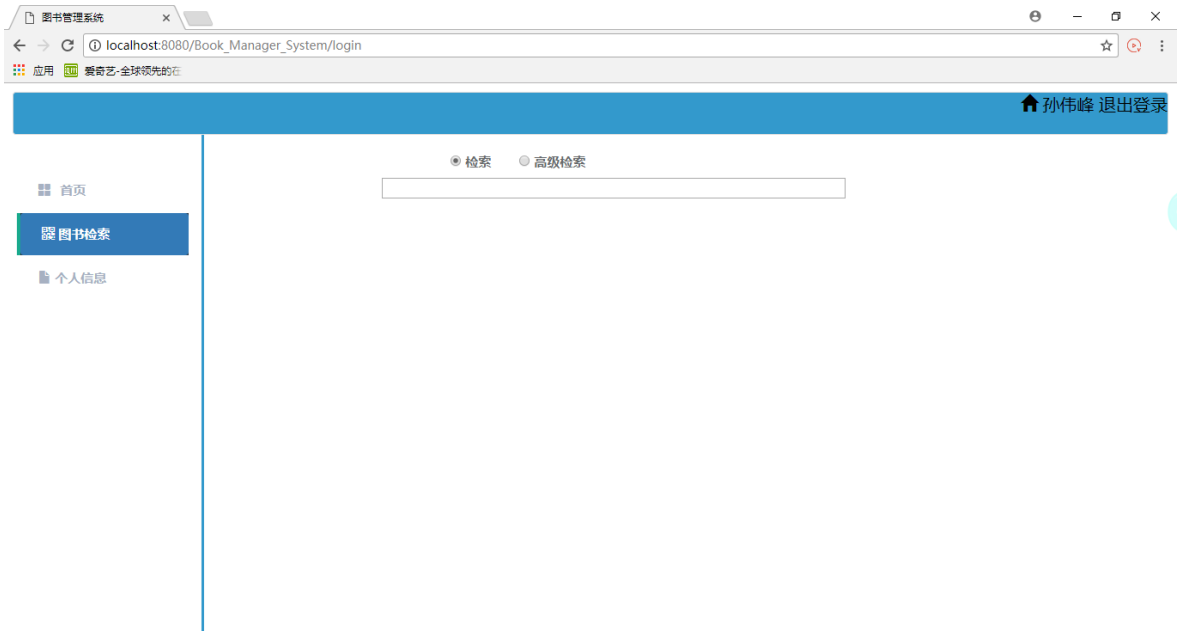




5.2 普通用户登录用例测试

用例编号	002	**用例标识**	logon_user
项目名称	图书管理系统	**模块名称**	普通用户登录
用例作者	刘耀东	**测试人员**	刘耀东
测试类型	功能测试	**设计日期**	2018-01-01
测试方法	黑盒	**测试日期**	2018-01-01
用例描述	用户若合法，则能正常进入用户查询界面。		
前置条件	无		
描述/输入/操作	**期望结果**	**真实结果**	**备注**
某用户已注册成功，且用户名和密码输入正确	跳转到用户查询界面	跳转到用户查询界面	该功能已完善
某用户尚未注册，或注册失败，或用户名和密码输入错误	未跳转到用户查询界面，重新登录	未跳转到用户查询界面，重新登录	

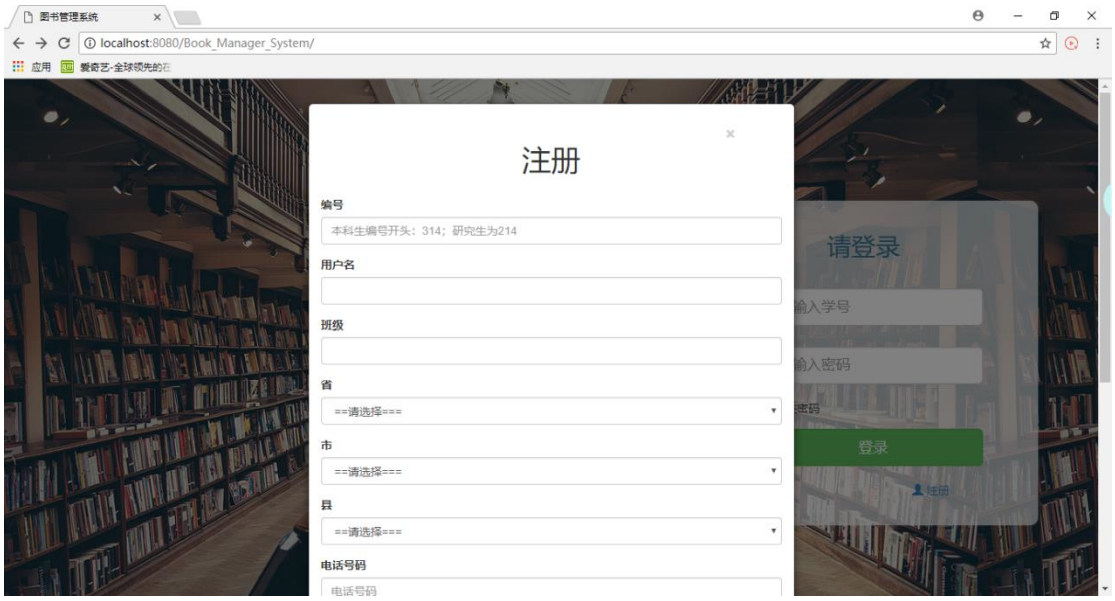


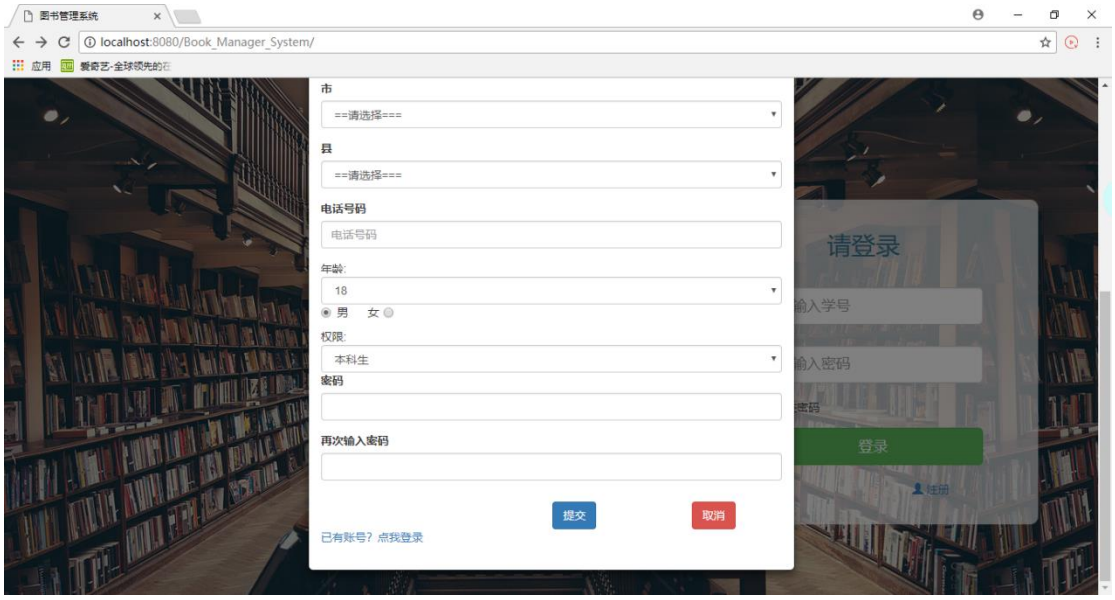


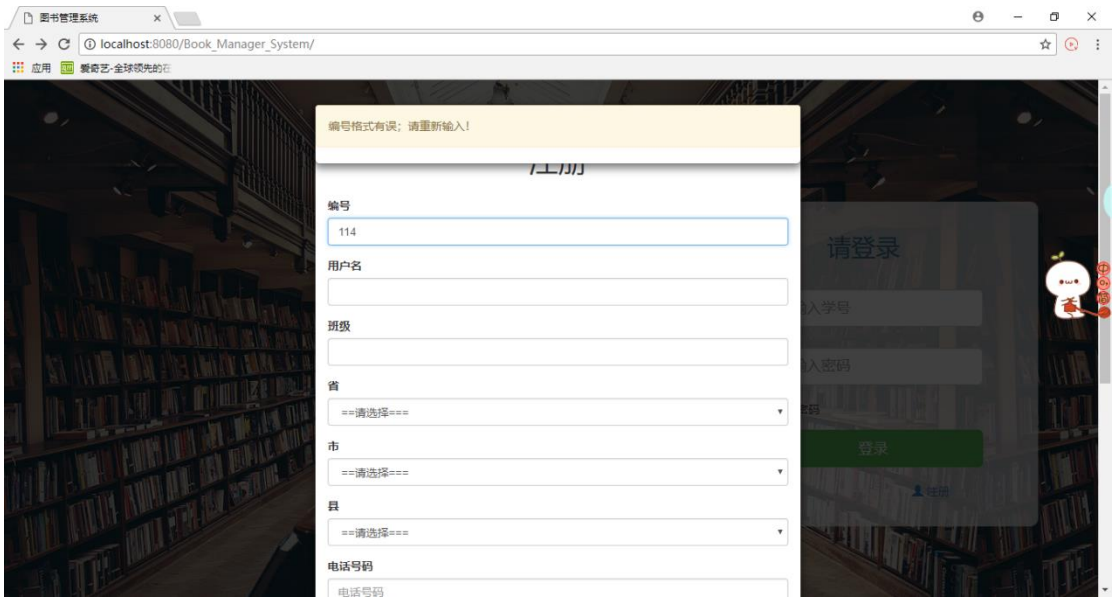
5.3 普通用户注册用例测试

用例编号	003	用例标识	register_user
项目名称	图书管理系统	模块名称	普通用户注册
用例作者	刘耀东	测试人员	刘耀东
测试类型	功能测试	设计日期	2018-01-01
测试方法	黑盒	测试日期	2018-01-01
用例描述	用户第一次使用该系统，需要先注册。		
前置条件	该用户以前未注册过。		
描述/输入/操作	期望结果	真实结果	备注
用户按注册提示正确注册	提示注册成功，重新登录	提示注册成功，重新登录	该功能已完善

用户未按照注册提示正确注册，如编号输入以“114”开头	提示“编号格式有误；请重新输入！”	提示“编号格式有误；请重新输入！”	







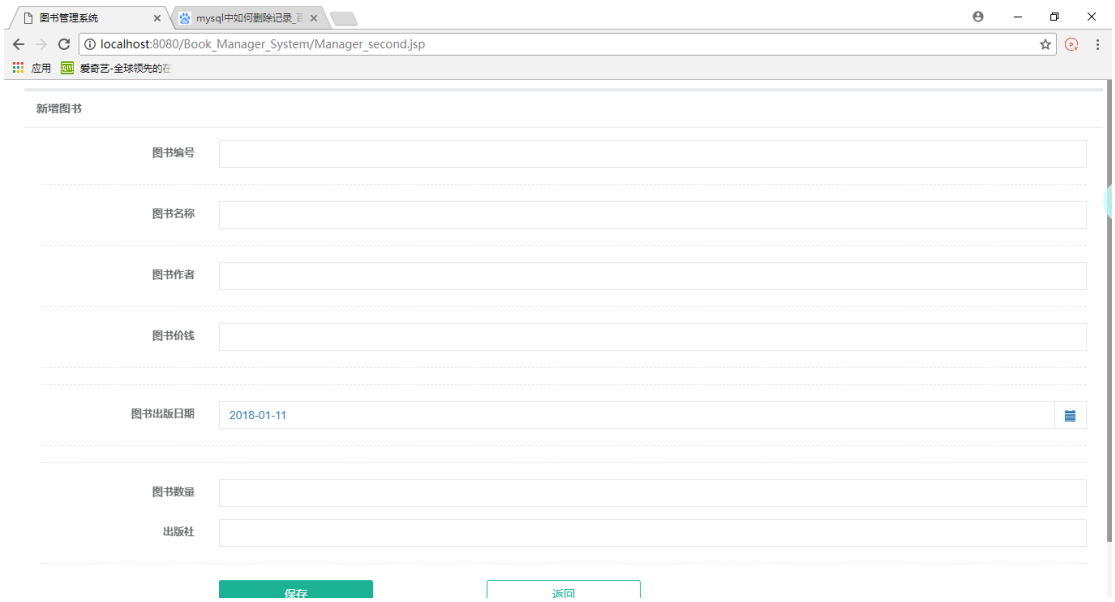
5.4 图书管理用例测试

用例编号	004	**用例标识**	book_management
项目名称	图书管理系统	**模块名称**	图书管理
用例作者	刘耀东	**测试人员**	刘耀东
测试类型	功能测试	**设计日期**	2018-01-01
测试方法	黑盒	**测试日期**	2018-01-01
用例描述	管理员可对图书信息增删改查，并可导出。		
前置条件	管理员合法，并进入该系统。		
描述/输入/操作	**期望结果**	**真实结果**	**备注**
管理员修改一条图书记录	刷新页面后，该记录已被修改	刷新页面后，该记录已被修改	该功能已完善
管理员新增一条图书记录	刷新页面后，该记录已被存入数据库	刷新页面后，该记录已被存入数据库	
管理员删除一条图书记录	刷新页面后，该记录已被删除	刷新页面后，该记录已被删除	
管理员导出图书信息，以导出 Excel 文件为例	在保存位置可以找到该文件，且打开后为保存的图书信息	在保存位置可以找到该文件，且打开后为保存的图书信息	

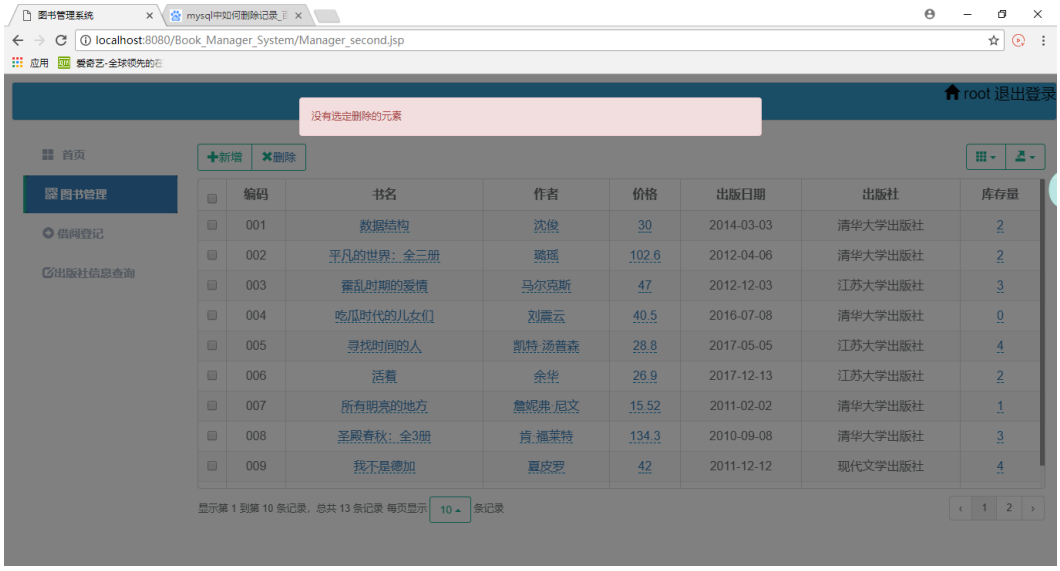

图书管理系统
mysql中如何删除记录
localhost:8080/Book_Manager_System/Manager_second.jsp
root 退出登录
首页
图书管理
借阅登记
出版社信息查询
新增
删除
编码 书名 作者 价格 出版日期 出版社 库存量
001 数据结构 沈俊 30 2014-03-03 清华大学出版社 2
002 平凡的世界：全三册 璐瑶 102.6 2012-04-06 清华大学出版社 2
003 霍乱时期的爱情 马尔克斯 47 2012-12-03 江苏大学出版社 3
004 吃瓜时代的儿女们 刘震云 40.5 2016-07-08 清华大学出版社 0
005 寻找时间的人 凯特·汤普森 28.8 2017-05-05 江苏大学出版社 4
006 活着 余华 26.9 2017-12-13 江苏大学出版社 2
007 所有明亮的地方 詹妮弗·尼文 15.52 2011-02-02 清华大学出版社 1
008 圣殿春秋：全3册 肯·福莱特 134.3 2010-09-08 清华大学出版社 3
009 我不是德加 夏皮罗 42 2011-12-12 现代文学出版社 4
显示第 1 到第 10 条记录，总共 13 条记录 每页显示 10 条记录
1 2


图书管理系统
mysql中如何删除记录
localhost:8080/Book_Manager_System/Manager_second.jsp
root 退出登录
首页
图书管理
借阅登记
出版社信息查询
新增
删除
编码 书名 作者 价格 出版日期 出版社 库存量
001 数据结构 沈俊 30 2014-03-03 清华大学出版社 2
002 平凡的世界：全三册 璐瑶 102.6 2012-04-06 清华大学出版社
003 霍乱时期的爱情 马尔克斯 47 2012-12-03 江苏大学出版社 3
004 吃瓜时代的儿女们 刘震云 40.5 2016-07-08 清华大学出版社 0
005 寻找时间的人 凯特·汤普森 28.8 2017-05-05 江苏大学出版社 4
006 活着 余华 26.9 2017-12-13 江苏大学出版社 2
007 所有明亮的地方 詹妮弗·尼文 15.52 2011-02-02 清华大学出版社 1
008 圣殿春秋：全3册 肯·福莱特 134.3 2010-09-08 清华大学出版社 3
009 我不是德加 夏皮罗 42 2011-12-12 现代文学出版社 4
显示第 1 到第 10 条记录，总共 13 条记录 每页显示 10 条记录
1 2

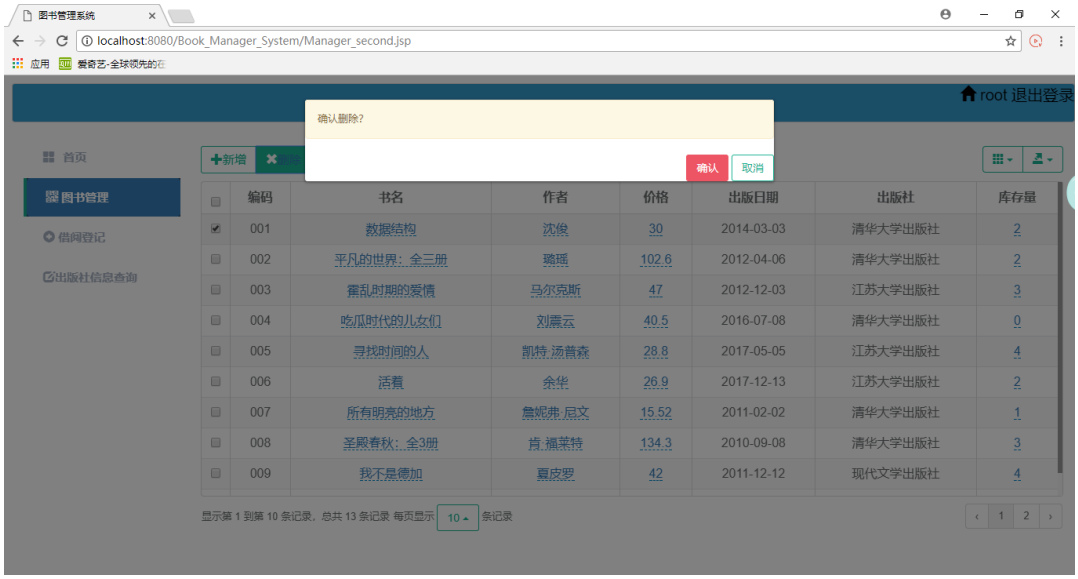
点击“新增”后：


图书管理系统
mysql中如何删除记录
localhost:8080/Book_Manager_System/Manager_second.jsp
新增图书
图书编号
图书名称
图书作者
图书价钱
图书出版日期
2018-01-11
图书数量
出版社
保存
返回

当未选择图书前面的复选框就点击“删除”按钮时，会提示下面的错误语句：



当选中编码为“001”前面的复选框，并点击“删除”按钮时，页面提示如下：

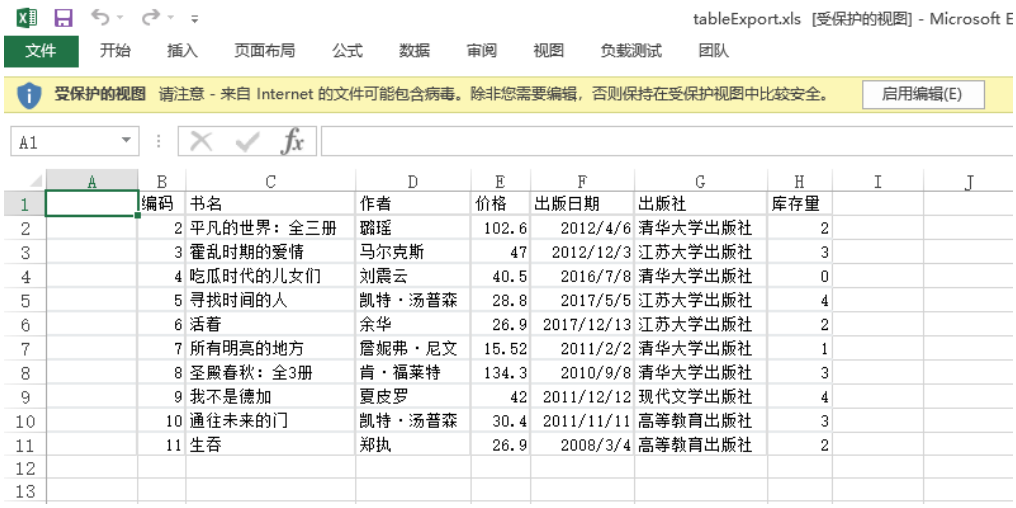


点击“确认”后，可发现数据库中没有了编码为“001”的记录：

通过选择右上角的第一个按钮，可以改变图书查看的样式：

通过选择右上角的第二个按钮，可以将图书相关数据导出为不同格式的文件：

以导出 Excel 文件为例：



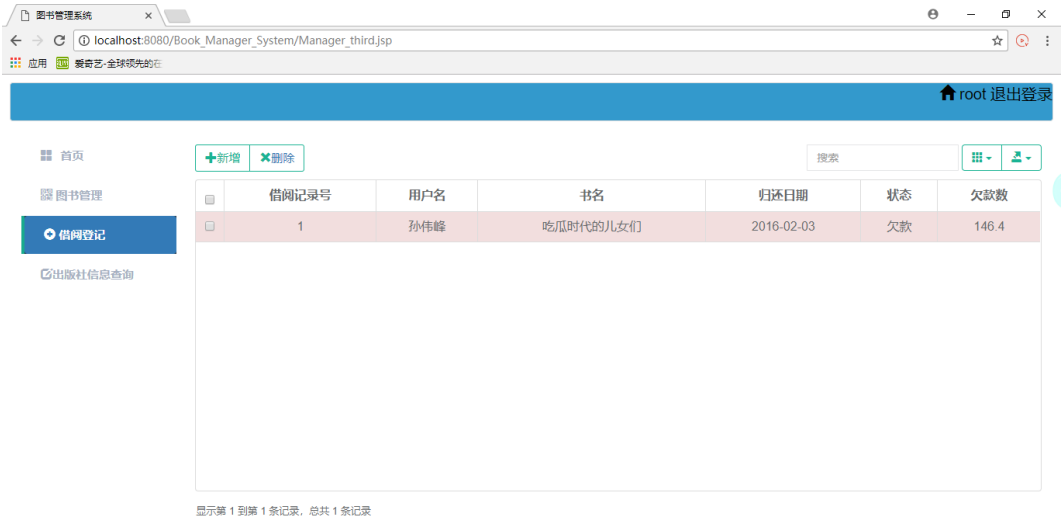
	A	编码	书名	作者	价格	出版日期	出版社	库存量
2		2	平凡的世界：全三册	璐瑶	102.6	2012/4/6	清华大学出版社	2
3		3	霍乱时期的爱情	马尔克斯	47	2012/12/3	江苏大学出版社	3
4		4	吃瓜时代的儿女们	刘震云	40.5	2016/7/8	清华大学出版社	0
5		5	寻找时间的人	凯特·汤普森	28.8	2017/5/5	江苏大学出版社	4
6		6	活着	余华	26.9	2017/12/13	江苏大学出版社	2
7		7	所有明亮的地方	詹妮弗·尼文	15.52	2011/2/2	清华大学出版社	1
8		8	圣殿春秋：全3册	肯·福莱特	134.3	2010/9/8	清华大学出版社	3
9		9	我不是德加	夏皮罗	42	2011/12/12	现代文学出版社	4
10		10	通往未来的门	凯特·汤普森	30.4	2011/11/11	高等教育出版社	3
11		11	生吞	郑执	26.9	2008/3/4	高等教育出版社	2
12								
13								

5.5 借阅管理用例测试

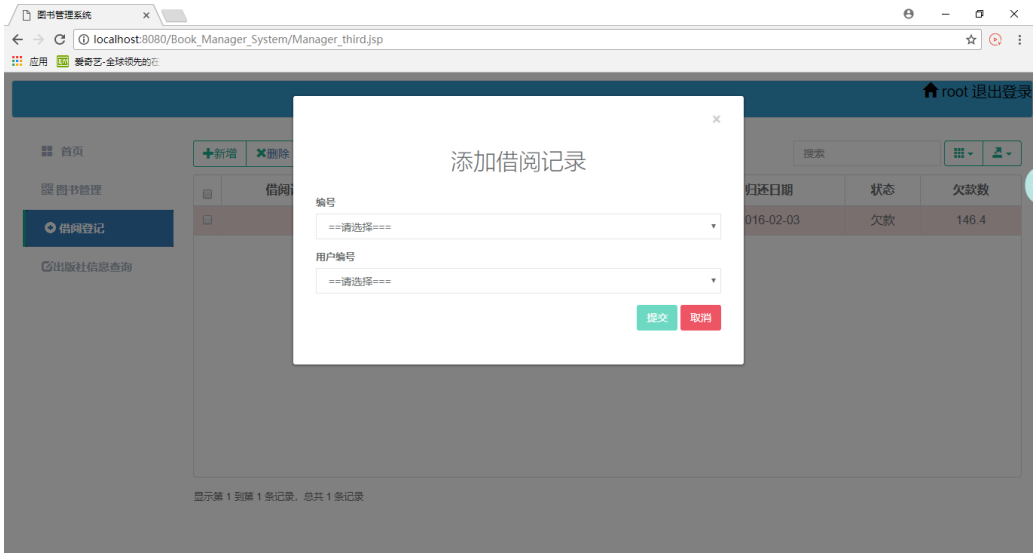
5.5.1 黑盒测试

用例编号	005	用例标识	borrow_management
项目名称	图书管理系统	模块名称	借阅管理
用例作者	刘耀东	测试人员	刘耀东
测试类型	功能测试	设计日期	2018-01-01
测试方法	黑盒	测试日期	2018-01-01
用例描述	管理员可新增、删除、查看借阅记录，并可导出借阅信息。		
前置条件	管理员合法，并进入该系统。		
描述/输入/操作	期望结果	真实结果	备注
管理员新增借阅记录时，若所选书籍已被借光	无法正常借阅	无法正常借阅	该功能已完善
管理员新增借阅记录时，若用户处于欠款状态	无法正常借阅	无法正常借阅	
管理员新增借阅记录时，若用户已借阅图书达到两本以上	无法正常借阅	无法正常借阅	
管理员删除借阅记录	刷新页面，删除的记录不会再显示，已从数据库删除	刷新页面，删除的记录不会再显示，已从数据库删除	
管理员查询借阅记录，如输入“葛康”	显示用户葛康的所有借阅记录	显示用户葛康的所有借阅记录	

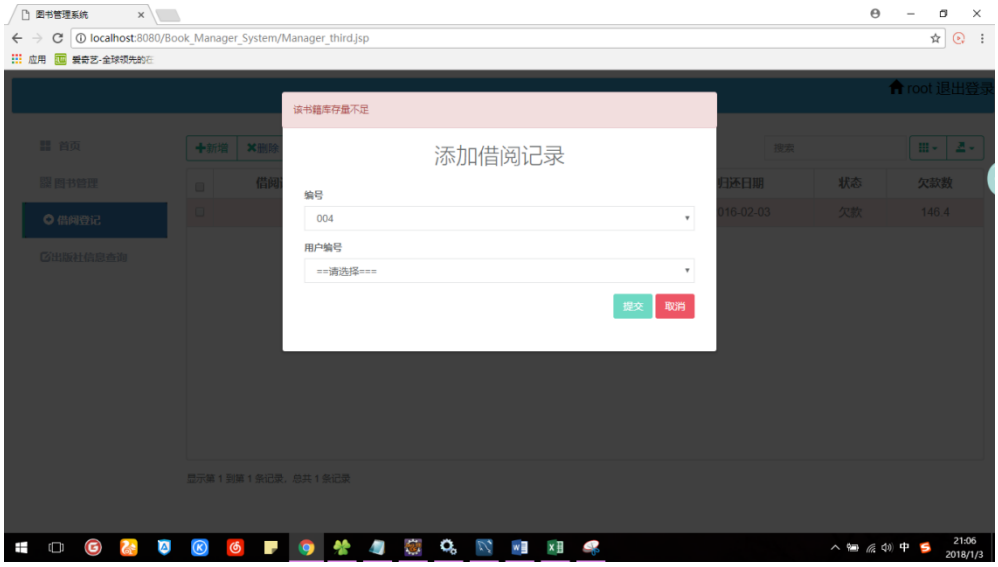
“借阅登记”：



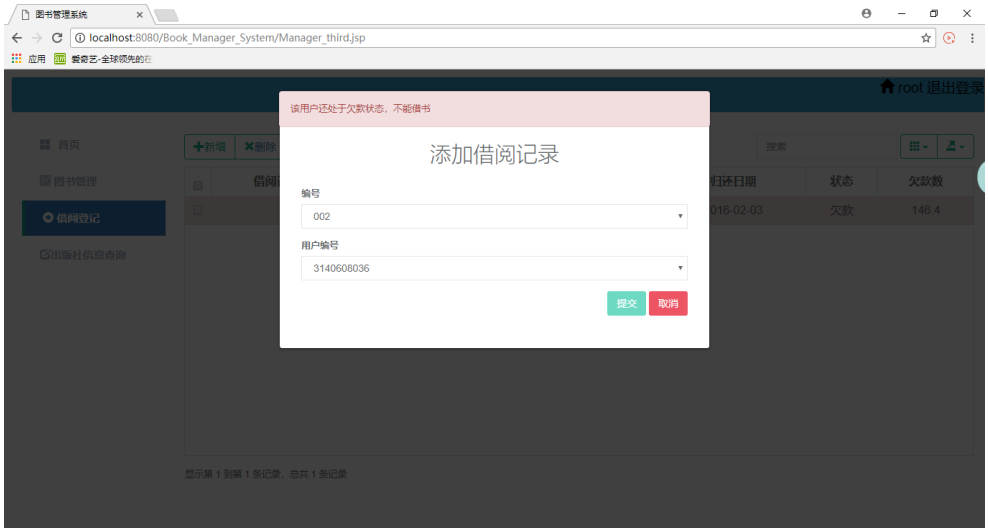
点击“新增”按钮后：



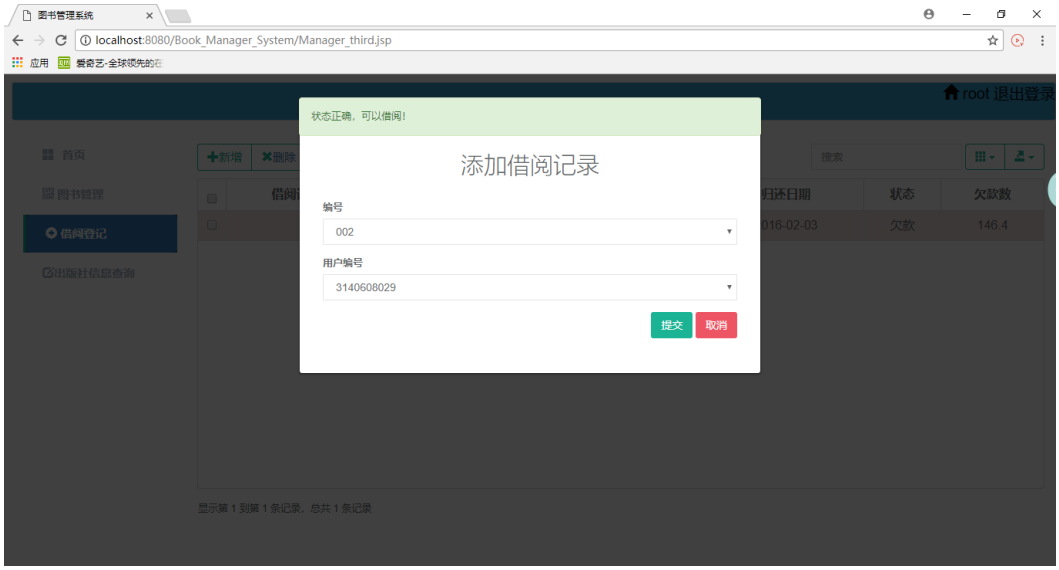
当借阅编码为“004”的图书时，由于该书籍已被借光，因此提示“该书籍库存量不足”：



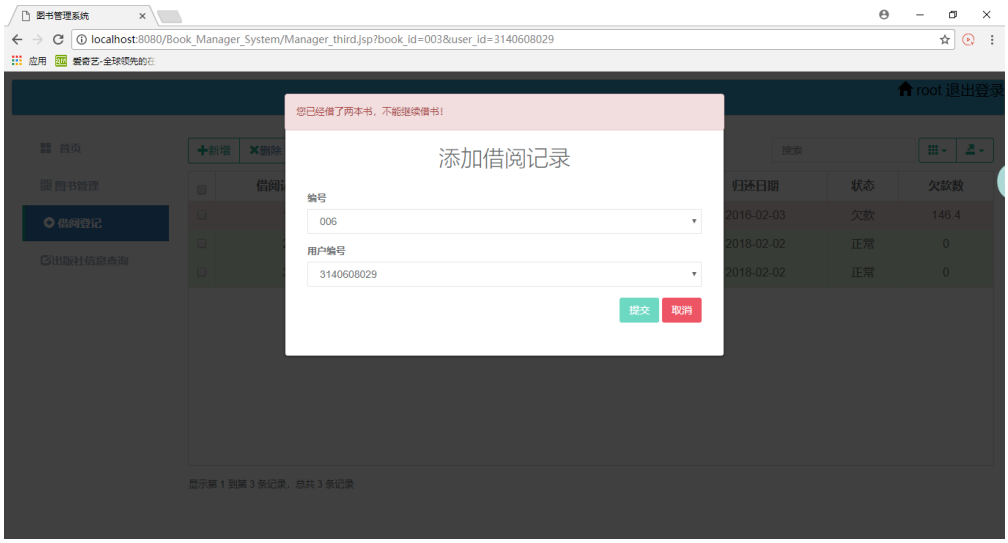
当选择的用户编号为已欠款的同学，则提示“该用户还处于欠款状态，不能借书”：



当借阅的图书还有库存并且借书的用户状态良好时，则提示“状态正确，可以借阅！”：



当一个用户借阅两本书后，就不能继续借阅了。若再次借阅图书，则提示“您已经借了两本书，不能继续借书！”：



右上角的搜索栏可对借阅记录进行检索，如检索用户“葛康”的借阅记录，显示如下：

5.5.2 白盒测试

5.5.2.1 借书

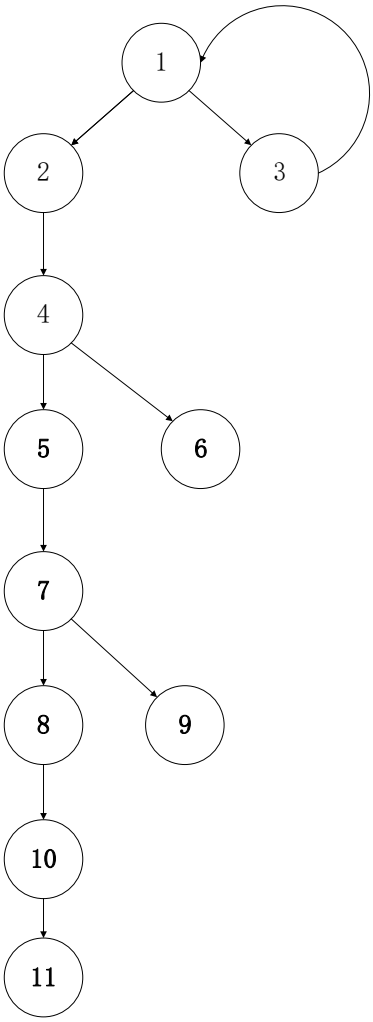


图 26 借书白盒测试路径图

结点说明：1.管理员登录；2.用户名合法；3.用户名非法；4.输入密码；5.密码正确；6.密码错误；7.输入借阅者信息；8.借阅者信息合法；9.借阅者信息非法；10.扫描图书信息；11.借阅成功。

采用基本路径法。测试路径有：

（1）1、3、1

（2）1、2、4、6

（3）1、2、4、5、7、9

（4）1、2、4、5、7、8、10、11

借阅者借书测试用例	输入	预期结果	实际结果
路径 1	录入管理员账户：111111	无法进入系统	错误—提示：该用户不存在
路径 2	录入管理员账户：123456 录入密码：111111	无法进入系统	错误—提示：密码错误
路径 3	录入管理员账户：	进入系统该借阅者不	错误—提示：该用户

	123456 录入密码：123456 录入借阅者用户名：3140608011	存在	不存在
路径 4	录入管理员账户：123456 录入密码：123456 录入借阅者用户名：3140608036	进入系统，扫描图书，借阅成功	借阅成功

5.5.2.2 还书

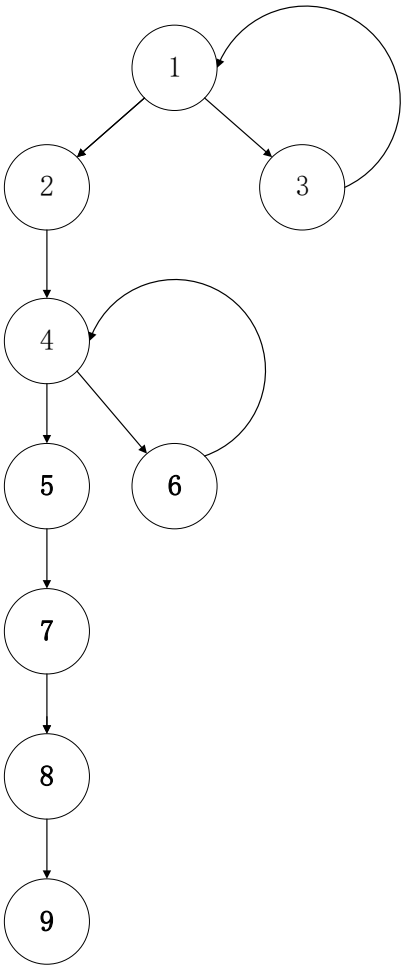


图 27 还书白盒测试路径图

结点说明：1.管理员登录；2.用户名合法；3.用户名非法；4.输入密码；5.密码正确；6.密码错误；7 输入借阅者信息；8.输入图书信息；9.归还成功。

采用基本路径法。测试路径有：

（1）1、3、1

（2）1、2、4、6

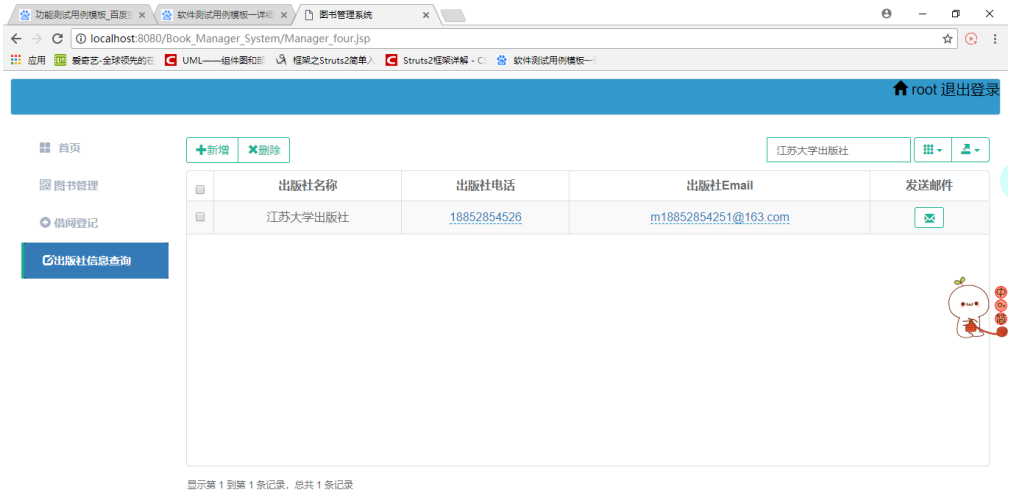
（3）1、2、4、5、7、8、9

借阅者还书测试用例	输入	预期结果	实际结果
路径 1	录入管理员账户：111111	无法进入系统	错误—提示：该用户不存在

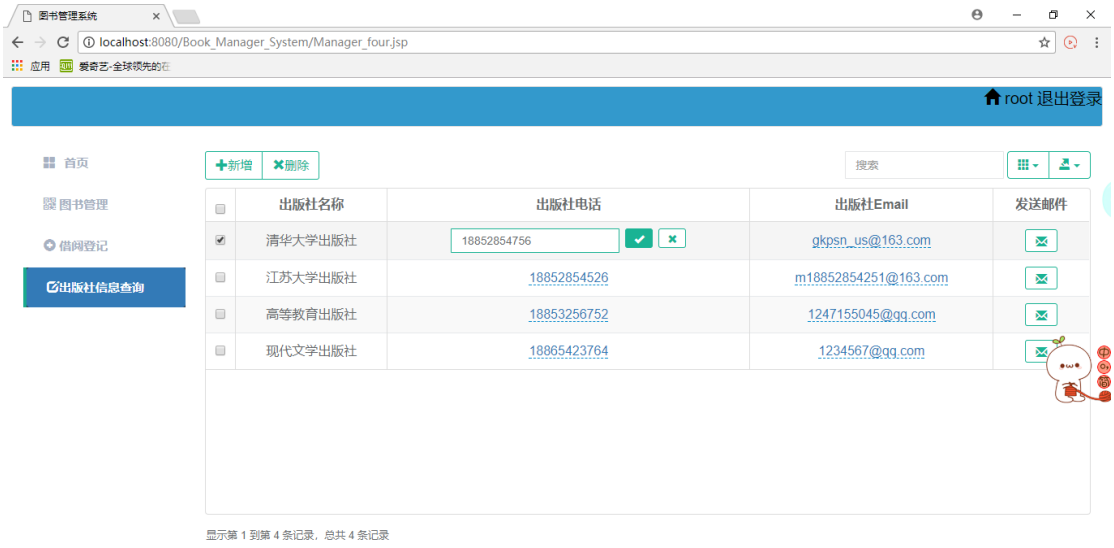
路径 2	录入管理员账户：123456 录入密码：111111	无法进入系统	错误—提示：密码错误
路径 3	录入管理员账户：123456 录入密码：123456 录入借阅者用户名：3140608011 录入图书名称：高数 18 讲	进入系统，归还成功	归还成功

5.6 出版社管理用例测试

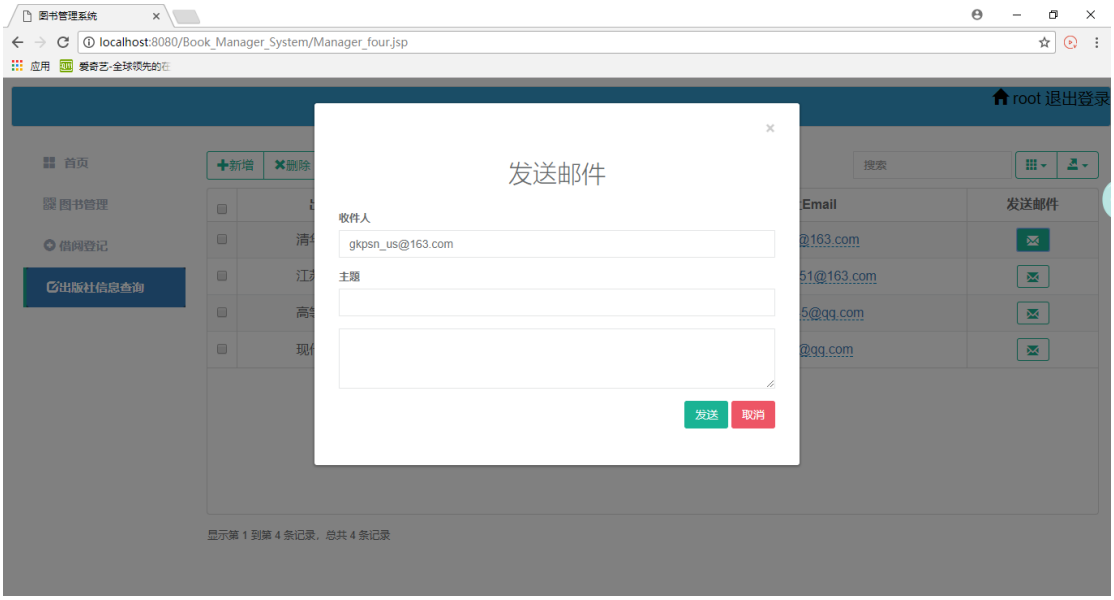
用例编号	006	**用例标识**	press_management
项目名称	图书管理系统	**模块名称**	出版社管理
用例作者	刘耀东	**测试人员**	刘耀东
测试类型	功能测试	**设计日期**	2018-01-01
测试方法	黑盒	**测试日期**	2018-01-01
用例描述	管理员可以对出版社信息增删改查，可以导出出版社信息，还可以直接向出版社发送邮件。		
前置条件	管理员合法，并进入该系统。		
描述/输入/操作	**期望结果**	**真实结果**	**备注**
管理员查询某出版社信息，如在搜索栏输入“江苏大学出版社”	显示江苏大学出版社的相关信息	显示江苏大学出版社的相关信息	该功能已完善
管理员新增一条出版社记录	刷新页面，显示新增的出版社信息	刷新页面，显示新增的出版社信息	
管理员删除一条或几条出版社记录	刷新页面，删除的出版社记录不再存在	刷新页面，删除的出版社记录不再存在	
管理员修改一条出版社记录	刷新页面，可以看到刚才的记录已被修改	刷新页面，可以看到刚才的记录已被修改	
管理员导出出版社信息	在保存的位置可以看到保存的文件，且打开后内容是关于出版社信息的	在保存的位置可以看到保存的文件，且打开后内容是关于出版社信息的	
管理员向某一出版社发送邮件，如“江苏大学出版社”	邮件发送成功	邮件发送成功	



可以直接修改出版社电话、出版社 Email 等信息：



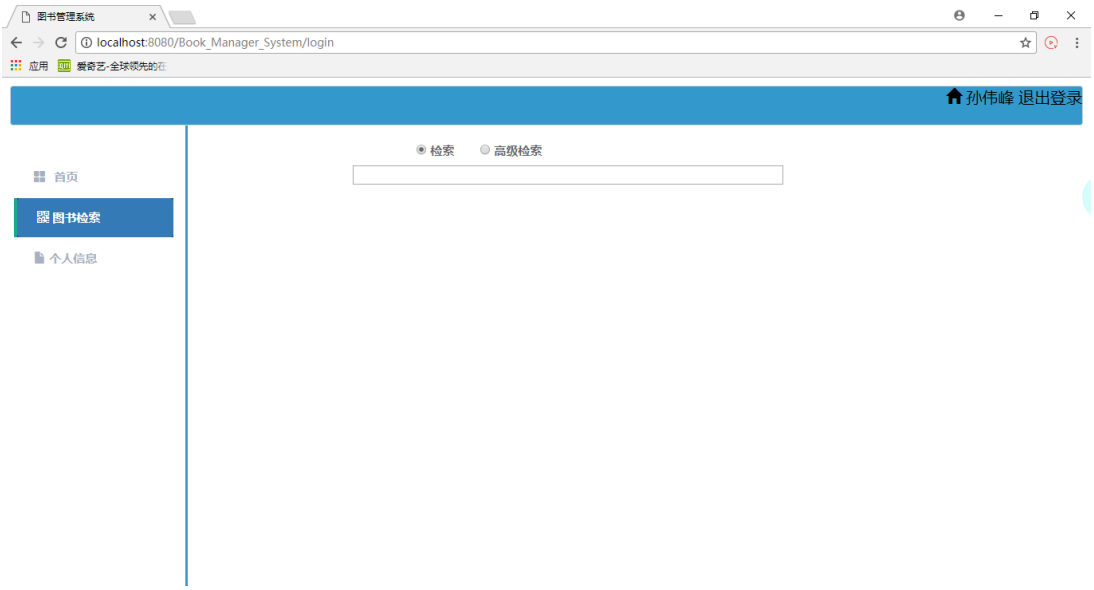
也可以通过点击右面的邮件图标，直接向出版社发送邮件：

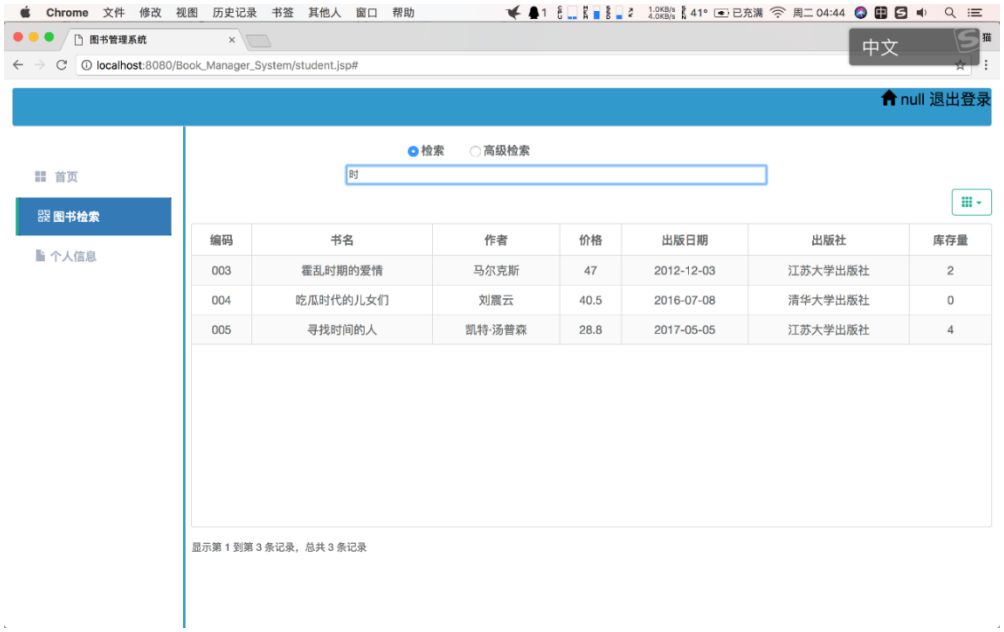


5.7 普通查询用例测试

5.7.1 黑盒测试

用例编号	007	用例标识	query_simple
项目名称	图书管理系统	模块名称	普通查询
用例作者	刘耀东	测试人员	刘耀东
测试类型	功能测试	设计日期	2018-01-01
测试方法	黑盒	测试日期	2018-01-01
用例描述	能够根据图书名、出版社名、作者名对图书进行模糊检索，显示编码、书名、作者、价格、出版日期、出版社、库存量等相关信息。		
前置条件	用户合法，且进入该系统。		
描述/输入/操作	期望结果	真实结果	备注
用户在搜索栏输入“时”	显示所有图书名、出版社名、作者名中包含“时”字的图书	显示所有图书名、出版社名、作者名中包含“时”字的图书	该功能已完善





5.7.2 白盒测试

5.7.2.1 查询图书

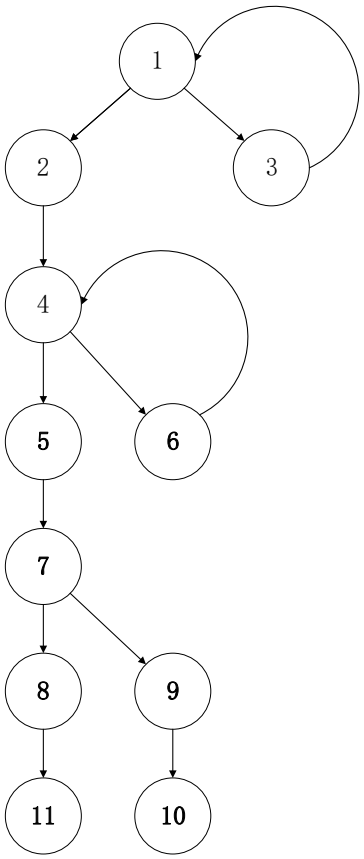


图 28 查询图书白盒测试路径图

结点说明：1.用户登录；2.用户名合法；3.用户名非法；4.输入密码；5.密码正确；6.密码错误；7.输入图书名；8.图书存在；9.图书不存在；10.查询失败；11.查询成功。

采用基本路径法。测试路径有：

（1）1、3、1
（2）1、2、4、6
（3）1、2、4、5、7、9、10
（4）1、2、4、5、7、8、11

借阅者查询测试用例	输入	预期结果	实际结果
路径 1	录入借阅者用户名：111111	无法进入系统	错误—提示：该用户不存在
路径 2	录入借阅者用户名：3140608036 录入密码：111111	无法进入系统	错误—提示：密码错误
路径 3	录入借阅者用户名：3140608036 录入密码：123456 录入图书名称：C#设计模式	进入系统，无法查询	系统提示所查询图书不存在，查询失败
路径 4	录入借阅者用户名：3140608036 录入密码：123456 录入图书名称：高数 18 讲	进入系统，查询成功	系统返回图书信息

5.7.2.2 预订图书

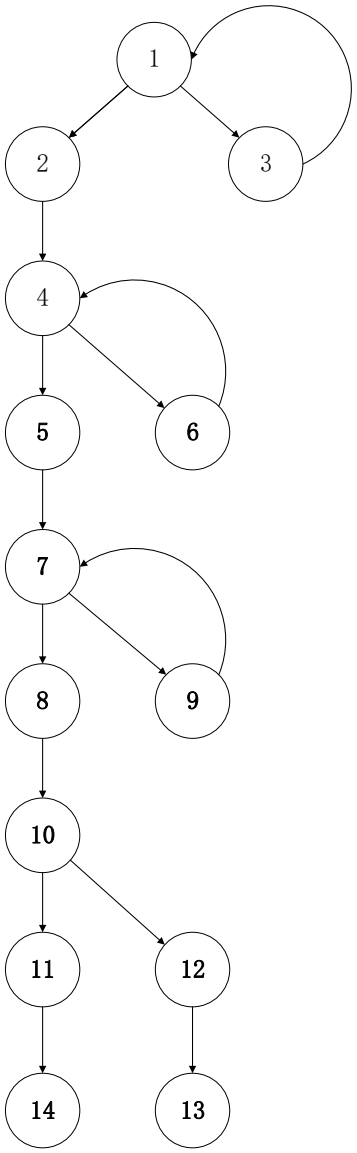


图 29 预订图书白盒测试路径图

结点说明：1.用户登录；2.用户名合法；3.用户名非法；4.输入密码；5.密码正确；6.密码错误；7.输入图书名；8.图书存在；9.图书不存在；10.预订图书；11.图书有剩余；12.图书库存量不足；13.预订失败；14.预订成功。

采用基本路径法。测试路径有：

（1）1、3、1

（2）1、2、4、6

（3）1、2、4、5、7、9

（4）1、2、4、5、7、8、10、12、13

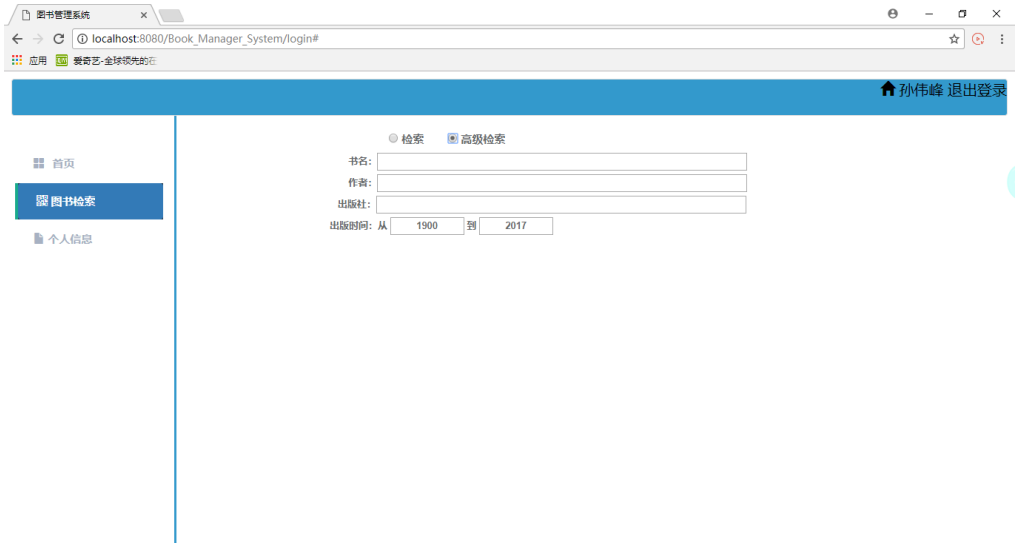
（5）1、2、4、5、7、8、10、11、14

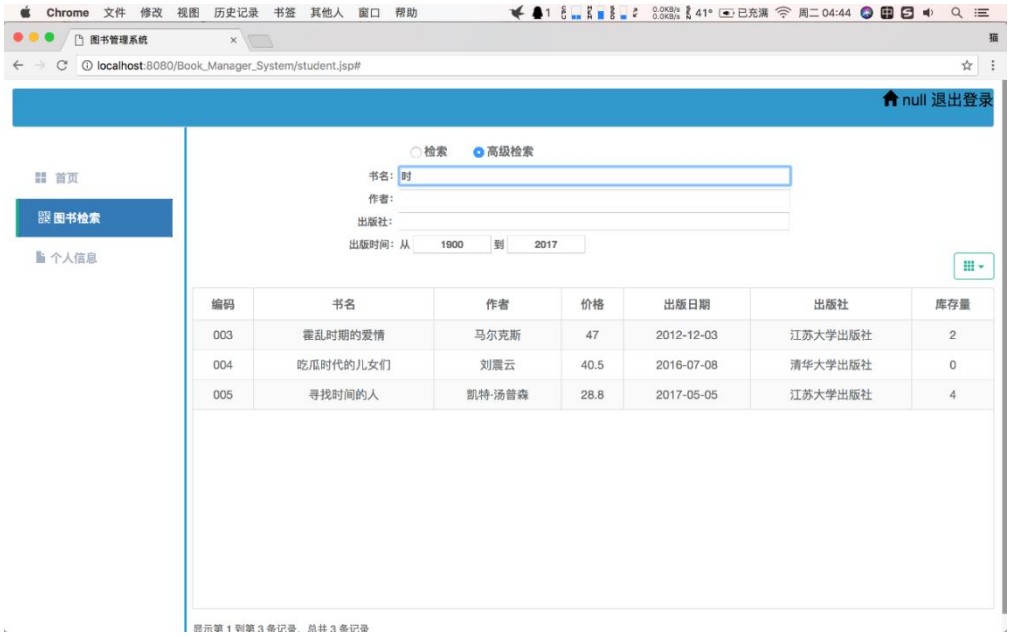
借阅者预订测试用例	输入	预期结果	实际结果
路径 1	录入借阅者用户名：111111	无法进入系统	错误—提示：该用户不存在
路径 2	录入借阅者用户	无法进入系统	错误—提示：密码错

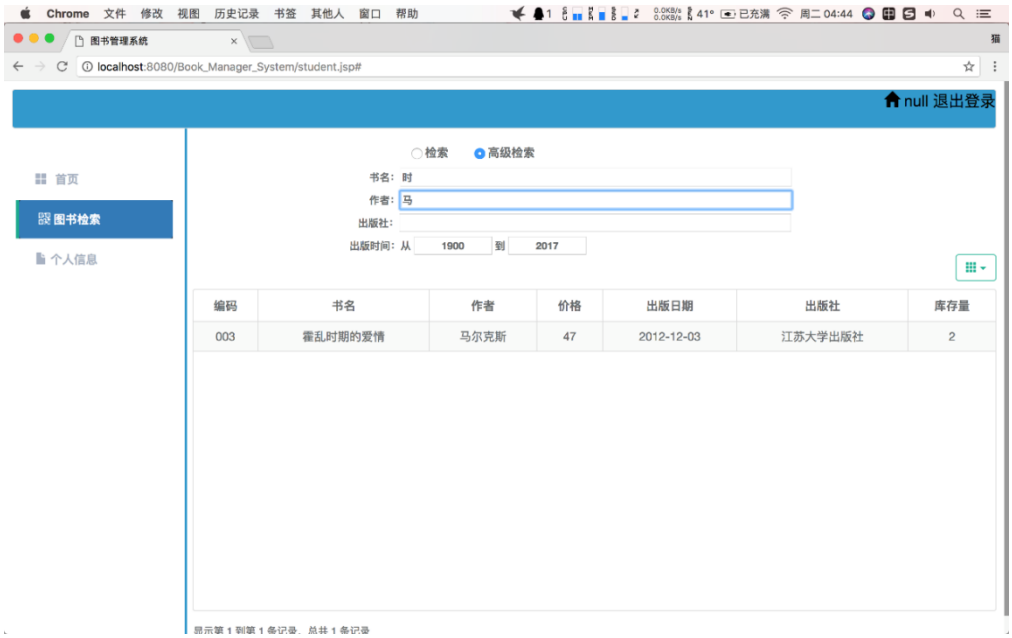
	名:3140608036 录入密码：111111		误
路径 3	录入借阅者用户名:3140608036 录入密码：123456 录入图书名称：C#设计模式	进入系统，图书查找失败	错误—提示：图书不存在
路径 4	录入借阅者用户名:3140608036 录入密码：123456 录入图书名称：高数 18 讲	进入系统，图书存在	错误—提示：该书库存量不足，无法预订
路径 5	录入借阅者用户名:3140608036 录入密码：123456 录入图书名称：线代 9 讲	进入系统，图书存在，预订成功	预订成功

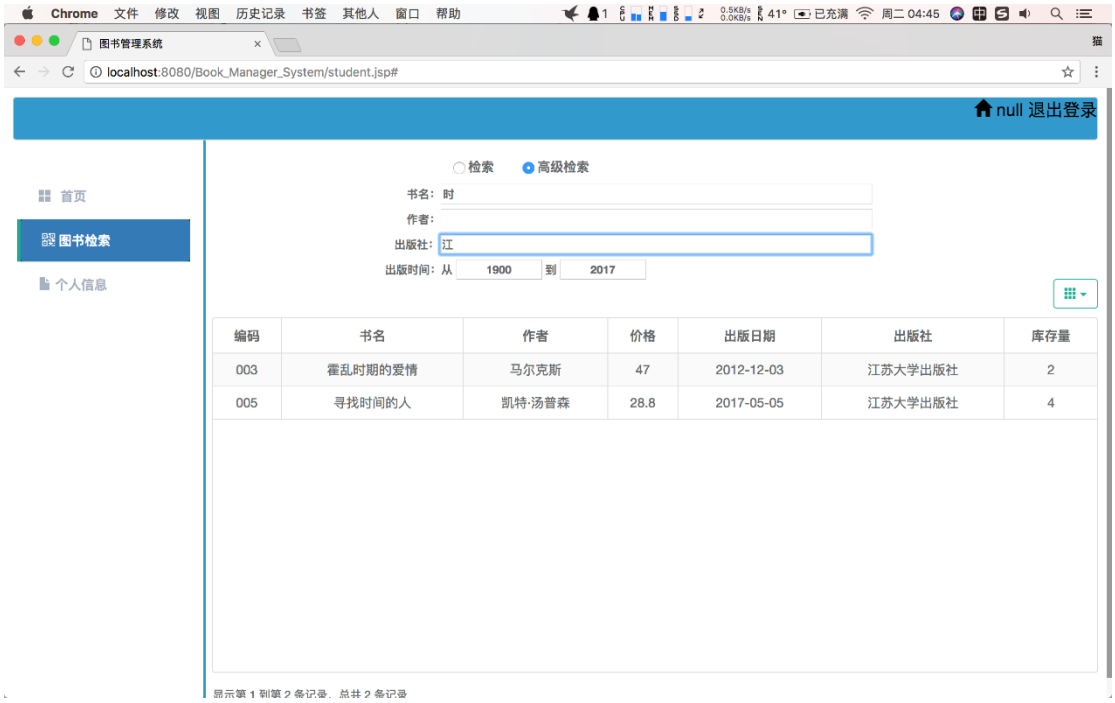
5.8 高级查询用例测试

用例编号	008	**用例标识**	query_highlevel
项目名称	图书管理系统	**模块名称**	高级查询
用例作者	刘耀东	**测试人员**	刘耀东
测试类型	功能测试	**设计日期**	2018-01-01
测试方法	黑盒	**测试日期**	2018-01-01
用例描述	通过图书名、出版社名、作者名和出版时间进行综合模糊检索，显示编码、书名、作者、价格、出版日期、出版社、库存量等相关信息。		
前置条件	用户合法，且进入该系统。		
描述/输入/操作	**期望结果**	**真实结果**	**备注**
用户在书名搜索栏中输入“时”	显示所有书名中包含“时”字的图书	显示所有书名中包含“时”字的图书	该功能已完善
用户在书名搜索栏中输入“时”，在作者搜索栏中输入“马”	显示所有书名中含“时”字，且作者名中含“马”字的图书	显示所有书名中含“时”字，且作者名中含“马”字的图书	
用户在书名搜索栏中输入“时”，在出版社搜索栏中输入“江”	显示所有书名中含“时”字，且出版社名中含“江”字的图书	显示所有书名中含“时”字，且出版社名中含“江”字的图书	

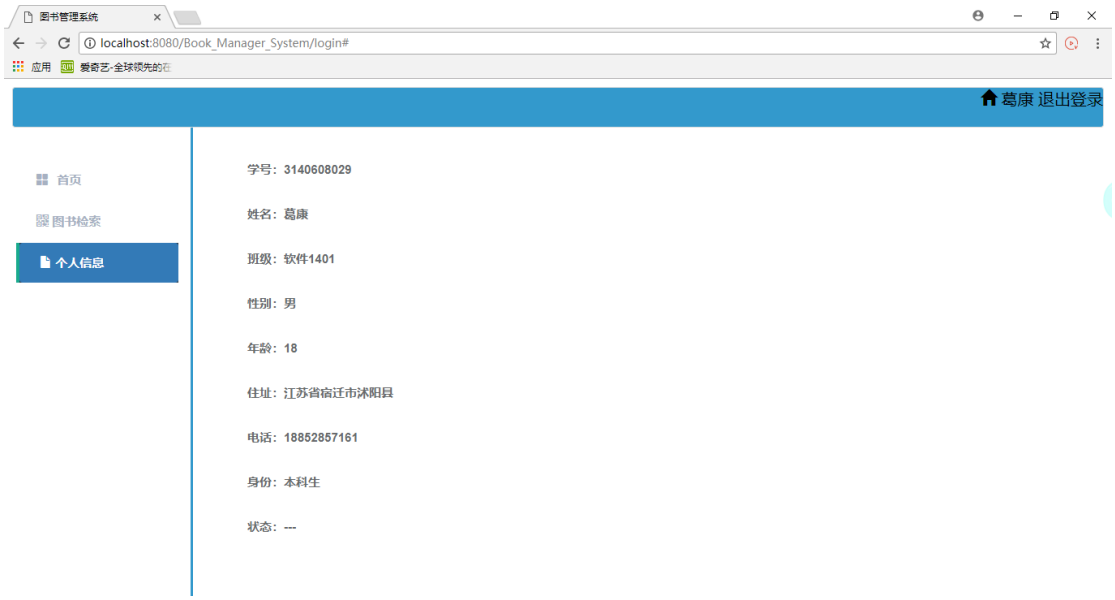

图书管理系统
localhost:8080/Book_Manager_System/login#
孙伟峰 退出登录
首页
图书检索
个人信息
检索
高级检索
书名:
作者:
出版社:
出版时间: 从 1900 到 2017


Chrome 文件 修改 视图 历史记录 书签 其他人 窗口 帮助
图书管理系统
localhost:8080/Book_Manager_System/student.jsp#
null 退出登录
首页
图书检索
个人信息
检索
高级检索
书名: 时
作者:
出版社:
出版时间: 从 1900 到 2017
编码 书名 作者 价格 出版日期 出版社 库存量
003 霍乱时期的爱情 马尔克斯 47 2012-12-03 江苏大学出版社 2
004 吃瓜时代的儿女们 刘震云 40.5 2016-07-08 清华大学出版社 0
005 寻找时间的人 凯特·汤普森 28.8 2017-05-05 江苏大学出版社 4


Chrome 文件 修改 视图 历史记录 书签 其他人 窗口 帮助
图书管理系统
localhost:8080/Book_Manager_System/student.jsp#
null 退出登录
首页
图书检索
个人信息
检索
高级检索
书名: 时
作者: 马
出版社:
出版时间: 从 1900 到 2017
编码 书名 作者 价格 出版日期 出版社 库存量
003 霍乱时期的爱情 马尔克斯 47 2012-12-03 江苏大学出版社 2


Chrome 文件 修改 视图 历史记录 书签 其他人 窗口 帮助
图书管理系统
localhost:8080/Book_Manager_System/student.jsp#
null 退出登录
首页
图书检索
个人信息
检索
高级检索
书名：时
作者：
出版社：江
出版时间：从 1900 到 2017
编码 书名 作者 价格 出版日期 出版社 库存量
003 霍乱时期的爱情 马尔克斯 47 2012-12-03 江苏大学出版社 2
005 寻找时间的人 凯特·汤普森 28.8 2017-05-05 江苏大学出版社 4

用户“个人信息”：


图书管理系统
localhost:8080/Book_Manager_System/login#
葛康 退出登录
首页
图书检索
个人信息
学号：3140608029
姓名：葛康
班级：软件1401
性别：男
年龄：18
住址：江苏省宿迁市沭阳县
电话：18852857161
身份：本科生
状态：---

6 小结

这次的课程设计我们选择了图书管理系统这个项目，考虑到相对于 C/S 架构，目前还是 B/S 架构更为流行，而且使用 B/S 架构部署项目更加容易，浏览使用时也更加方便，所以选择用 B/S 架构来完成这次的项目。数据库方面使用 MySQL，服务器则用 Tomcat 进行部署。文档部分主要由赵玲玲和刘耀东完成，代码部分则是孙伟峰和我完成，大部分的代码由孙伟峰完成。

我们做的图书管理系统功能上主要分为两个部分，一是管理员的部分，二是普通用户的部分。管理员主要是对图书信息、用户信息等进行维护，也就是对数据进行维护，还有就是考虑到学校的图书馆在借书时是人工登记的，所以借书部分的操作也由管理员完成，除此之外，管理员还要根据用户状态对借书做出限制，包括能够借多少本书，以及借书时长等。普通用户则是能够查看个人信息，还有就是对图书进行检索，能够根据图书名、出版社名、作者名对图书进行模糊检索，这是一般检索，还能够根据这三者，以及出版时间同时对图书进行限制来模糊检索。

考虑到课设时间有限，所以我们没有自己写框架，而是选择了一个开源的 Hibernate 框架，在此基础之上再进行相关功能的开发。Hibernate 是一个开放源代码的对象关系映射框架，它对 JDBC 进行了非常轻量级的对象封装，它将 POJO 与数据库表建立映射关系，是一个全自动的 orm 框架，hibernate 可以自动生成 SQL 语句，自动执行，这样我们就可以随心所欲的使用对象编程思维来操纵数据库。

在开发过程中，我们遇到的主要问题就是前期对数据库中的表没有设计好，导致后期频繁的修改表的属性，给开发带来了不便。还有就是在对管理员和用户的功能分配上没有明确，往往是在讨论之后发现管理员或者是用户需要新增一部分功能，也有一部分原因是在设计分析时对整个项目分析的不够细致，不够透彻。如果在对整个项目进行分析时能够考虑到方方面面，那就能够省很多不必要的功夫。所以说对整个软件项目的设计是很重要的，在这上面花的功夫越多，之后在进行编写代码时遇到的麻烦就越少，整体看来花费的时间和精力更少，也可以避免对代码、数据库的修改。