华南理工大学

South China University of Technology

图书管理系统

开发说明书

姓　　名　黄泽君

学　　院　设计学院

专　　业　产品设计

学　　号　202130670058

完成时间　2023.11.28

目 录

一、项目规划

1.1 项目背景

图书管理系统是运用信息技术（Information Technology，简称 IT）对图书馆进行电子化管理而构建的信息管理系统[1]。随着信息化的迅速发展，越来越多的图书馆转变传统的图书管理模式，构建信息化的管理系统以提高工作效率。

相较于传统的手工管理模式，图书管理系统具有检索方便快捷、检索速度快、检索结果准确、存储数据量大、成本低、节省资源、且具有良好的人机交互界面等优势[2]。华南理工大学图书馆于 2006 年 9 月建成使用，建筑面积达 4.23 万平方米，设有办公室、文献资源建设部、文献资源服务部、参考咨询服务部、信息技术服务部等 5 个行政业务部门。本项目咨询了华南理工大学大学城校区图书馆的工作人员，并结合图书馆的特点，设计一个具有图书信息管理、读者信息管理以及查询、修改等功能的图书管理系统。

1.2 系统的定位与功能

实验设计并开发了一个简单的图书馆管理数据库系统。该系统主要的数据库中包括了图书馆图书信息、学校师生信息、师生借阅书籍信息等。该系统面向的对象有图书馆图书管理员和读者。

主要实现的功能如下所示：

图书管理员可以根据书籍流动情况对图书完成登记、修改、注销登记，根据读者注册情况对读者完成登记、修改、注销管理。

读者可以对图书借阅情况完成借阅、续借、归还等操作。

1.3 开发进度计划

2023 年 9 月 10 日，确定开发图书管理系统，并拟出管理系统的功能结构设计。

2023 年 9 月 20 日，建立实体关系图（entity relationship diagram，E-R 图）和数据表，在 Mysql 数据库中创建完数据库。

2023 年 10 月 10 日，初步建立用户界面，优化用户界面，完善用户端功能。

2023 年 10 月 30 日，前端应用连接数据库。

2023 年 11 月 20 日，完成最后调试。

二、数据库设计

2.1 功能需求设计

(1) 图书查询功能

(2) 图书入库功能

对图书信息添加、修改。

(3) 借/还图书功能

图书出借时考虑两个个前提：

A. 该书是否在库中；

B. 读者是否已经借满其限额；

如果不存在以上情况，则可以出借。

读者还书的时候可以续借该图书，续借的过程主要是修改借书记录里的还书日期。

(4) 借书证管理功能

对读者的登录账号、密码进行添加、修改、删除。

2.2 概念结构设计 （ER 图）

数据库中的数据信息包括以下几种:

(1) 书籍信息

(2) 借书记录信息

(3) 读者信息

(4)管理员信息

这些数据项之间的关系可以用图 1 表示:

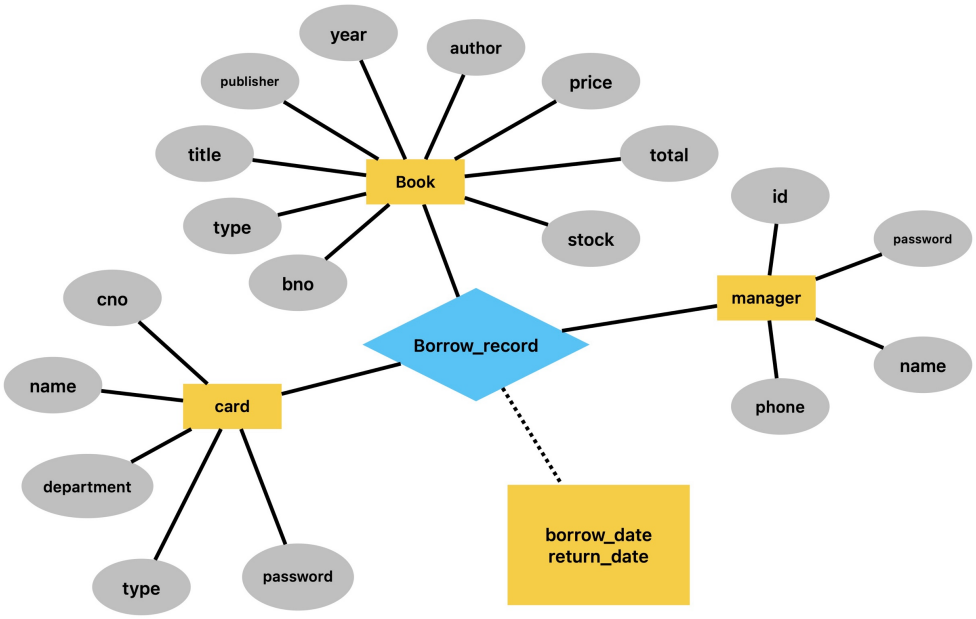

图 1　E-R 关系图

2.3 逻辑结构设计

本系统共设计 4 个表。

针对图书管理信息系统的需求，通过对图书管理工作过程的内容和数据流程分析，设计如下面所示的 4 个数据表:

（1）书库表

属性：图书编号，类别，书名，出版社，年份，作者，价格，总数，库存

主键：图书编号

（2）借书记录表

属性：借书记录编号，借书日期，归还日期，图书编号，读书证 id，经手人编号

主键：借书记录编号

（3）读者表

属性：读者 id，读者姓名，部门（学院），身份，密码

主键：读者 id

（4）管理员表

属性：管理员 id，密码，姓名，联系方式

主键：管理员 id

2.4 数据字典

（1）书库表

书库表：web_book

数据项名	类型	长度	精度	标度	取值范围	默认值	约束
图书编号：bno	VARCHAR	32					PRIMARY KEY
类别：type	VARCHAR	30					NOT NULL
书名：title	VARCHAR	32					NOT NULL

出版社：publisher	VARCHAR	32					NOT NULL
年份：year	INT						NOT NULL
作者：author	VARCHAR	32					NOT NULL
价格：price	DECIMAL	8		2			NOT NULL
总数：total	INT						NOT NULL
库存：stock	INT						NOT NULL

（2）借书记录表

借书记录表：web_borrow_list

数据项名	类型	长度	精度	标度	取值范围	默认值	约束
借书记录编号：id	BIGINT						PRIMARY KEY
借书日期：borrow_time	DATE						NOT NULL
归还日期：return_time	DATE						NOT NULL
图书编号：book_id	VARCHAR	32					FOREIGN KEY

读者 id：card_id	VARCHAR	32					FOREIGN KEY
经手人编号：manage_id	VARCHAR	32					FOREIGN KEY

（3）读者表

读者表：web_card

数据项名	类型	长度	精度	标度	取值范围	默认值	约束
读者 id：cno	VARCHAR	32					PRIMARY KEY
读者姓名：name	VARCHAR	32					NOT NULL
部门（学院）：department	VARCHAR	32					NOT NULL
身份：type	SMALLINT				1 或 2		NOT NULL
密码：password	VARCHAR	32					NOT NULL

（4）管理员表

管理员表：web_manager

数据项名	类型	长度	精度	标度	取值范围	默认值	约束
管理员 id：id	VARCHAR	32					PRIMARY KEY
密码：password	VARCHAR	32					NOT NULL
姓名：name	VARCHAR	32					NOT NULL
联系方式：contact	VARCHAR	20					NOT NULL

三、系统开发与实现

3.1 开发工具与环境

（1）开发工具

本次开发环境为 Visual Studio Code，编程语言采用 python3.9，数据库采用的是 Mysql，数据库的设计和修改使用 Navicat Premium。

在创建页面时，采用 Bootstrap(依赖于 jQuery)作为前端框架。

通过 Django 的 ORM 功能实现使用操作对象的方式来操作数据库中的数据。

在本地端口(localhost)生成 web 端的图书管理系统，在同一个局域网的管理员用户可通过地址访问 web 端，进行图书管理。

（2）环境配置

电脑　MacBook AIR

芯片　Apple M1

操作系统版本　macOS Sonoma Version 14.1.1

3.2 建表、插入初始数据

```
SET NAMES utf8mb4;
SET FOREIGN_KEY_CHECKS = 0;

-- ----------------------------
-- Table structure for auth_group
-- ----------------------------
DROP TABLE IF EXISTS `auth_group`;
CREATE TABLE `auth_group` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(150) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `name` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

-- ----------------------------
-- Records of auth_group
-- ----------------------------
BEGIN;
COMMIT;

-- ----------------------------
-- Table structure for auth_group_permissions
-- ----------------------------
DROP TABLE IF EXISTS `auth_group_permissions`;
CREATE TABLE `auth_group_permissions` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `group_id` int NOT NULL,
  `permission_id` int NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `auth_group_permissions_group_id_permission_id_0cd325b0_uniq`
(`group_id`,`permission_id`),
  KEY `auth_group_permissio_permission_id_84c5c92e_fk_auth_perm`
(`permission_id`),
  CONSTRAINT `auth_group_permissio_permission_id_84c5c92e_fk_auth_perm`
FOREIGN KEY (`permission_id`) REFERENCES `auth_permission` (`id`),
  CONSTRAINT `auth_group_permissions_group_id_b120cbf9_fk_auth_group_id`
FOREIGN KEY (`group_id`) REFERENCES `auth_group` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

-- ----------------------------
```

```
-- Records of auth_group_permissions
-- ----------------------------
BEGIN;
COMMIT;

-- ----------------------------
-- Table structure for auth_permission
-- ----------------------------
DROP TABLE IF EXISTS `auth_permission`;
CREATE TABLE `auth_permission` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(255) NOT NULL,
  `content_type_id` int NOT NULL,
  `codename` varchar(100) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `auth_permission_content_type_id_codename_01ab375a_uniq`
(`content_type_id`,`codename`),
  CONSTRAINT `auth_permission_content_type_id_2f476e4b_fk_django_co` FOREIGN
KEY (`content_type_id`) REFERENCES `django_content_type` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

-- ----------------------------
-- Records of auth_permission
-- ----------------------------
BEGIN;
COMMIT;

-- ----------------------------
-- Table structure for auth_user
-- ----------------------------
DROP TABLE IF EXISTS `auth_user`;
CREATE TABLE `auth_user` (
  `id` int NOT NULL AUTO_INCREMENT,
  `password` varchar(128) NOT NULL,
  `last_login` datetime(6) DEFAULT NULL,
  `is_superuser` tinyint(1) NOT NULL,
  `username` varchar(150) NOT NULL,
  `first_name` varchar(150) NOT NULL,
  `last_name` varchar(150) NOT NULL,
  `email` varchar(254) NOT NULL,
  `is_staff` tinyint(1) NOT NULL,
  `is_active` tinyint(1) NOT NULL,
  `date_joined` datetime(6) NOT NULL,
  PRIMARY KEY (`id`),
```

```
  UNIQUE KEY `username` (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

-- ----------------------------
-- Records of auth_user
-- ----------------------------
BEGIN;
COMMIT;

-- ----------------------------
-- Table structure for auth_user_groups
-- ----------------------------
DROP TABLE IF EXISTS `auth_user_groups`;
CREATE TABLE `auth_user_groups` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `user_id` int NOT NULL,
  `group_id` int NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `auth_user_groups_user_id_group_id_94350c0c_uniq`
(`user_id`,`group_id`),
  KEY `auth_user_groups_group_id_97559544_fk_auth_group_id` (`group_id`),
  CONSTRAINT `auth_user_groups_group_id_97559544_fk_auth_group_id` FOREIGN
KEY (`group_id`) REFERENCES `auth_group` (`id`),
  CONSTRAINT `auth_user_groups_user_id_6a12ed8b_fk_auth_user_id` FOREIGN KEY
(`user_id`) REFERENCES `auth_user` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

-- ----------------------------
-- Records of auth_user_groups
-- ----------------------------
BEGIN;
COMMIT;

-- ----------------------------
-- Table structure for auth_user_user_permissions
-- ----------------------------
DROP TABLE IF EXISTS `auth_user_user_permissions`;
CREATE TABLE `auth_user_user_permissions` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `user_id` int NOT NULL,
  `permission_id` int NOT NULL,
  PRIMARY KEY (`id`),
```

```
  UNIQUE KEY
`auth_user_user_permissions_user_id_permission_id_14a6b632_uniq`
(`user_id`,`permission_id`),
  KEY `auth_user_user_permi_permission_id_1fbb5f2c_fk_auth_perm`
(`permission_id`),
  CONSTRAINT `auth_user_user_permi_permission_id_1fbb5f2c_fk_auth_perm`
FOREIGN KEY (`permission_id`) REFERENCES `auth_permission` (`id`),
  CONSTRAINT `auth_user_user_permissions_user_id_a95ead1b_fk_auth_user_id`
FOREIGN KEY (`user_id`) REFERENCES `auth_user` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

-- ----------------------------
-- Records of auth_user_user_permissions
-- ----------------------------
BEGIN;
COMMIT;

-- ----------------------------
-- Table structure for django_admin_log
-- ----------------------------
DROP TABLE IF EXISTS `django_admin_log`;
CREATE TABLE `django_admin_log` (
  `id` int NOT NULL AUTO_INCREMENT,
  `action_time` datetime(6) NOT NULL,
  `object_id` longtext,
  `object_repr` varchar(200) NOT NULL,
  `action_flag` smallint unsigned NOT NULL,
  `change_message` longtext NOT NULL,
  `content_type_id` int DEFAULT NULL,
  `user_id` int NOT NULL,
  PRIMARY KEY (`id`),
  KEY `django_admin_log_content_type_id_c4bce8eb_fk_django_co`
(`content_type_id`),
  KEY `django_admin_log_user_id_c564eba6_fk_auth_user_id` (`user_id`),
  CONSTRAINT `django_admin_log_content_type_id_c4bce8eb_fk_django_co`
FOREIGN KEY (`content_type_id`) REFERENCES `django_content_type` (`id`),
  CONSTRAINT `django_admin_log_user_id_c564eba6_fk_auth_user_id` FOREIGN KEY
(`user_id`) REFERENCES `auth_user` (`id`),
  CONSTRAINT `django_admin_log_chk_1` CHECK ((`action_flag` >= 0))
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

-- ----------------------------
-- Records of django_admin_log
-- ----------------------------
```

```sql
BEGIN;
COMMIT;

-- ----------------------------
-- Table structure for django_content_type
-- ----------------------------
DROP TABLE IF EXISTS `django_content_type`;
CREATE TABLE `django_content_type` (
  `id` int NOT NULL AUTO_INCREMENT,
  `app_label` varchar(100) NOT NULL,
  `model` varchar(100) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `django_content_type_app_label_model_76bd3d3b_uniq`
(`app_label`,`model`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

-- ----------------------------
-- Records of django_content_type
-- ----------------------------
BEGIN;
COMMIT;

-- ----------------------------
-- Table structure for django_migrations
-- ----------------------------
DROP TABLE IF EXISTS `django_migrations`;
CREATE TABLE `django_migrations` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `app` varchar(255) NOT NULL,
  `name` varchar(255) NOT NULL,
  `applied` datetime(6) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=19 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

-- ----------------------------
-- Records of django_migrations
-- ----------------------------
BEGIN;
INSERT INTO `django_migrations` (`id`, `app`, `name`, `applied`) VALUES (1,
'contenttypes', '0001_initial', '2023-11-28 08:23:17.183717');
INSERT INTO `django_migrations` (`id`, `app`, `name`, `applied`) VALUES (2,
'auth', '0001_initial', '2023-11-28 08:23:17.310423');
```

```
INSERT INTO `django_migrations` (`id`, `app`, `name`, `applied`) VALUES (3,
'admin', '0001_initial', '2023-11-28 08:23:17.334945');
INSERT INTO `django_migrations` (`id`, `app`, `name`, `applied`) VALUES (4,
'admin', '0002_logentry_remove_auto_add', '2023-11-28 08:23:17.337675');
INSERT INTO `django_migrations` (`id`, `app`, `name`, `applied`) VALUES (5,
'admin', '0003_logentry_add_action_flag_choices', '2023-11-28
08:23:17.340260');
INSERT INTO `django_migrations` (`id`, `app`, `name`, `applied`) VALUES (6,
'contenttypes', '0002_remove_content_type_name', '2023-11-28
08:23:17.355685');
INSERT INTO `django_migrations` (`id`, `app`, `name`, `applied`) VALUES (7,
'auth', '0002_alter_permission_name_max_length', '2023-11-28
08:23:17.366193');
INSERT INTO `django_migrations` (`id`, `app`, `name`, `applied`) VALUES (8,
'auth', '0003_alter_user_email_max_length', '2023-11-28 08:23:17.374108');
INSERT INTO `django_migrations` (`id`, `app`, `name`, `applied`) VALUES (9,
'auth', '0004_alter_user_username_opts', '2023-11-28 08:23:17.378542');
INSERT INTO `django_migrations` (`id`, `app`, `name`, `applied`) VALUES (10,
'auth', '0005_alter_user_last_login_null', '2023-11-28 08:23:17.388952');
INSERT INTO `django_migrations` (`id`, `app`, `name`, `applied`) VALUES (11,
'auth', '0006_require_contenttypes_0002', '2023-11-28 08:23:17.389819');
INSERT INTO `django_migrations` (`id`, `app`, `name`, `applied`) VALUES (12,
'auth', '0007_alter_validators_add_error_messages', '2023-11-28
08:23:17.392467');
INSERT INTO `django_migrations` (`id`, `app`, `name`, `applied`) VALUES (13,
'auth', '0008_alter_user_username_max_length', '2023-11-28
08:23:17.403961');
INSERT INTO `django_migrations` (`id`, `app`, `name`, `applied`) VALUES (14,
'auth', '0009_alter_user_last_name_max_length', '2023-11-28
08:23:17.415908');
INSERT INTO `django_migrations` (`id`, `app`, `name`, `applied`) VALUES (15,
'auth', '0010_alter_group_name_max_length', '2023-11-28 08:23:17.422165');
INSERT INTO `django_migrations` (`id`, `app`, `name`, `applied`) VALUES (16,
'auth', '0011_update_proxy_permissions', '2023-11-28 08:23:17.424889');
INSERT INTO `django_migrations` (`id`, `app`, `name`, `applied`) VALUES (17,
'auth', '0012_alter_user_first_name_max_length', '2023-11-28
08:23:17.436529');
INSERT INTO `django_migrations` (`id`, `app`, `name`, `applied`) VALUES (18,
'sessions', '0001_initial', '2023-11-28 08:23:17.443469');
COMMIT;

-- ----------------------------
-- Table structure for django_session
-- ----------------------------
```

```sql
DROP TABLE IF EXISTS `django_session`;
CREATE TABLE `django_session` (
  `session_key` varchar(40) NOT NULL,
  `session_data` longtext NOT NULL,
  `expire_date` datetime(6) NOT NULL,
  PRIMARY KEY (`session_key`),
  KEY `django_session_expire_date_a5c62663` (`expire_date`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

-- ----------------------------
-- Records of django_session
-- ----------------------------
BEGIN;
INSERT INTO `django_session` (`session_key`, `session_data`, `expire_date`)
VALUES ('i0r26yq3b0z7gpgdnmghhrusydpp1fqt',
'.eJxlyjEKgCAABdC7_NnhayXlWQQJUnBIIxEK8e7V3Pp4DTGFDNMQNxgoKjlQk5QKAmnd_au2Tk
HR1tFz-
fRfu4ArvpSYk_PXEc8bRnOcyf4ACTca5A:1r8FNz:d1Nx8q3PkqqPD8S6GqgOFUj9_pfXRxFhvV_
Sfj33CDA', '2023-12-06 07:53:31.683929');
COMMIT;

-- ----------------------------
-- Table structure for web_book
-- ----------------------------
DROP TABLE IF EXISTS `web_book`;
CREATE TABLE `web_book` (
  `bno` varchar(32) NOT NULL,
  `type` varchar(20) NOT NULL,
  `title` varchar(32) NOT NULL,
  `publisher` varchar(32) NOT NULL,
  `year` int NOT NULL,
  `author` varchar(32) NOT NULL,
  `price` decimal(8,2) NOT NULL,
  `total` int NOT NULL,
  `stock` int NOT NULL,
  PRIMARY KEY (`bno`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

-- ----------------------------
-- Records of web_book
-- ----------------------------
BEGIN;
```

```sql
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`) VALUES ('1098765432109', '社会学', '社会学概论', '高等教育出版社', 2009, '陈寅恪', 66.00, 10, 9);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`) VALUES ('1209345657213', '文学', '午夜惊魂', '上海译文出版社', 2010, '诶里', 56.00, 20, 20);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`) VALUES ('1234534655322', '文学', '汤姆·索亚历险记', '上海译文出版社', 1992, '马克·吐温', 56.00, 20, 18);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`) VALUES ('1235466442', '恐怖小说', '鬼上身', '物料架', 1967, '诶里', 45.00, 10, 9);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`) VALUES ('124553674', '恐怖小说', '贞子', '物料架', 1976, '诶里', 45.00, 20, 20);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`) VALUES ('2145642675346', '艺术', '设计学概论', '湖南科学技术出版社', 1999, '杨林', 88.00, 10, 9);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`) VALUES ('2345678909876', '经济学', '资本论', '人民出版社', 1867, '马克思', 65.00, 10, 9);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`) VALUES ('2435675432', '恐怖小说', '314', '物料架', 1344, '诶里', 134.00, 3, 3);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`) VALUES ('3210987654321', '地理学', '人文地理学', '浙江大学出版社', 2003, '董作民', 55.00, 20, 20);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`) VALUES ('3234768765434', '自动化技术、计算技术', '现代操作系统', '机械工业出版社', 2013, '陈向群', 89.00, 10, 10);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`) VALUES ('34567543214', '恐怖小说', '354', '物料架', 1343, '134', 144.00, 68, 68);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`) VALUES ('3456789087654', '小说', '百年孤独', '南海出版公司', 1982, '加西亚·马尔克斯', 62.00, 18, 18);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`) VALUES ('4321098765432', '教育学', '教育心理学', '北京师范大学出版社', 2006, '王铁崖', 68.00, 15, 14);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`) VALUES ('5678945612123', '历史', '中国古代史', '北京大学出版社', 2005, '王国维', 45.00, 15, 15);
```

```sql
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`,
`author`, `price`, `total`, `stock`) VALUES ('6543210987654', '医学', '人体解
剖学', '人民卫生出版社', 2015, '李时中', 92.00, 10, 10);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`,
`author`, `price`, `total`, `stock`) VALUES ('6789098765432', '法律', '刑法学
', '法律出版社', 2008, '宋茂荣', 72.00, 12, 12);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`,
`author`, `price`, `total`, `stock`) VALUES ('7654323456789', '哲学', '论人的
尊严', '商务印书馆', 1785, '康德', 75.00, 8, 7);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`,
`author`, `price`, `total`, `stock`) VALUES ('8765432109876', '政治学', '政治
学原理', '清华大学出版社', 1985, '哈伯马斯', 79.00, 6, 6);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`,
`author`, `price`, `total`, `stock`) VALUES ('8909876543210', '计算机科学', '
计算机网络', '清华大学出版社', 2007, '谢希仁', 78.00, 8, 8);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`,
`author`, `price`, `total`, `stock`) VALUES ('9087654321345', '科普', '宇宙奇
妙之旅', '科学出版社', 2011, '斯蒂芬·霍金', 68.00, 12, 10);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`,
`author`, `price`, `total`, `stock`) VALUES ('9787010009254', '马列主义毛邓思想
', '毛泽东选集', '人民出版社', 1991, '毛泽东', 81.00, 5, 5);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`,
`author`, `price`, `total`, `stock`) VALUES ('9876543212345', '心理学', '乌合
之众', '上海人民出版社', 1895, '勒庞', 59.00, 14, 13);
COMMIT;

-- ----------------------------
-- Table structure for web_borrow_list
-- ----------------------------
DROP TABLE IF EXISTS `web_borrow_list`;
CREATE TABLE `web_borrow_list` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `borrow_time` date NOT NULL,
  `return_time` date NOT NULL,
  `book_id` varchar(32) NOT NULL,
  `card_id` varchar(32) NOT NULL,
  `manager_id` varchar(32) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `web_borrow_list_book_id_6ec60c09_fk_web_book_bno` (`book_id`),
  KEY `web_borrow_list_manager_id_397054ab_fk_web_manager_id`
(`manager_id`),
  KEY `web_borrow_list_card_id_5de02fe0_fk` (`card_id`),
  CONSTRAINT `web_borrow_list_book_id_6ec60c09_fk_web_book_bno` FOREIGN KEY
(`book_id`) REFERENCES `web_book` (`bno`),
```

```
  CONSTRAINT `web_borrow_list_card_id_5de02fe0_fk` FOREIGN KEY (`card_id`)
REFERENCES `web_card` (`cno`),
  CONSTRAINT `web_borrow_list_manager_id_397054ab_fk_web_manager_id` FOREIGN
KEY (`manager_id`) REFERENCES `web_manager` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=40 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

-- ----------------------------
-- Records of web_borrow_list
-- ----------------------------
BEGIN;
INSERT INTO `web_borrow_list` (`id`, `borrow_time`, `return_time`,
`book_id`, `card_id`, `manager_id`) VALUES (24, '2024-11-27', '2025-01-27',
'1234534655322', '202066666666', NULL);
INSERT INTO `web_borrow_list` (`id`, `borrow_time`, `return_time`,
`book_id`, `card_id`, `manager_id`) VALUES (25, '2024-11-27', '2024-12-27',
'1098765432109', '202066666666', 'admin');
INSERT INTO `web_borrow_list` (`id`, `borrow_time`, `return_time`,
`book_id`, `card_id`, `manager_id`) VALUES (26, '2024-11-27', '2024-12-27',
'2345678909876', '202066666666', 'admin');
INSERT INTO `web_borrow_list` (`id`, `borrow_time`, `return_time`,
`book_id`, `card_id`, `manager_id`) VALUES (27, '2024-11-27', '2024-12-27',
'2145642675346', '202130600012', 'admin');
INSERT INTO `web_borrow_list` (`id`, `borrow_time`, `return_time`,
`book_id`, `card_id`, `manager_id`) VALUES (28, '2024-11-27', '2024-12-27',
'4321098765432', '202130600012', 'admin');
INSERT INTO `web_borrow_list` (`id`, `borrow_time`, `return_time`,
`book_id`, `card_id`, `manager_id`) VALUES (29, '2024-11-27', '2024-12-27',
'9087654321345', '202177777777', 'admin');
INSERT INTO `web_borrow_list` (`id`, `borrow_time`, `return_time`,
`book_id`, `card_id`, `manager_id`) VALUES (30, '2024-11-27', '2024-12-27',
'7654323456789', '202177777777', 'admin');
COMMIT;

-- ----------------------------
-- Table structure for web_card
-- ----------------------------
DROP TABLE IF EXISTS `web_card`;
CREATE TABLE `web_card` (
  `cno` varchar(32) NOT NULL,
  `name` varchar(32) NOT NULL,
  `department` varchar(32) NOT NULL,
  `type` smallint NOT NULL,
```

```
  `password` varchar(32) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci
NOT NULL,
  PRIMARY KEY (`cno`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

-- ----------------------------
-- Records of web_card
-- ----------------------------
BEGIN;
INSERT INTO `web_card` (`cno`, `name`, `department`, `type`, `password`)
VALUES ('202066666666', '梅欢', '土木工程学院', 1, '111111');
INSERT INTO `web_card` (`cno`, `name`, `department`, `type`, `password`)
VALUES ('202130600012', '张三', '计算机学院', 1, '111111');
INSERT INTO `web_card` (`cno`, `name`, `department`, `type`, `password`)
VALUES ('20214434552', '雪雪', '艺术学院', 2, '111111');
INSERT INTO `web_card` (`cno`, `name`, `department`, `type`, `password`)
VALUES ('202177777777', '戚戚', '设计学院', 1, '111111');
COMMIT;

-- ----------------------------
-- Table structure for web_manager
-- ----------------------------
DROP TABLE IF EXISTS `web_manager`;
CREATE TABLE `web_manager` (
  `id` varchar(32) NOT NULL,
  `password` varchar(32) NOT NULL,
  `name` varchar(32) NOT NULL,
  `contact` varchar(20) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

-- ----------------------------
-- Records of web_manager
-- ----------------------------
BEGIN;
INSERT INTO `web_manager` (`id`, `password`, `name`, `contact`) VALUES
('admin', '123456', 'admin', '18732847831');
COMMIT;

SET FOREIGN_KEY_CHECKS = 1;
```

3.3 登录界面与功能介绍

（一）图形化页面介绍

图 2 登录界面

输入账户、密码，点击管理员登陆或读者登陆就会去数据库（读者表和管理员表）验证账户和密码是否正确，如若正确就进入图书管理系统。若没有账户，可以点击“去注册”，注册一个读者账户再返回登陆，如图 3 所示:

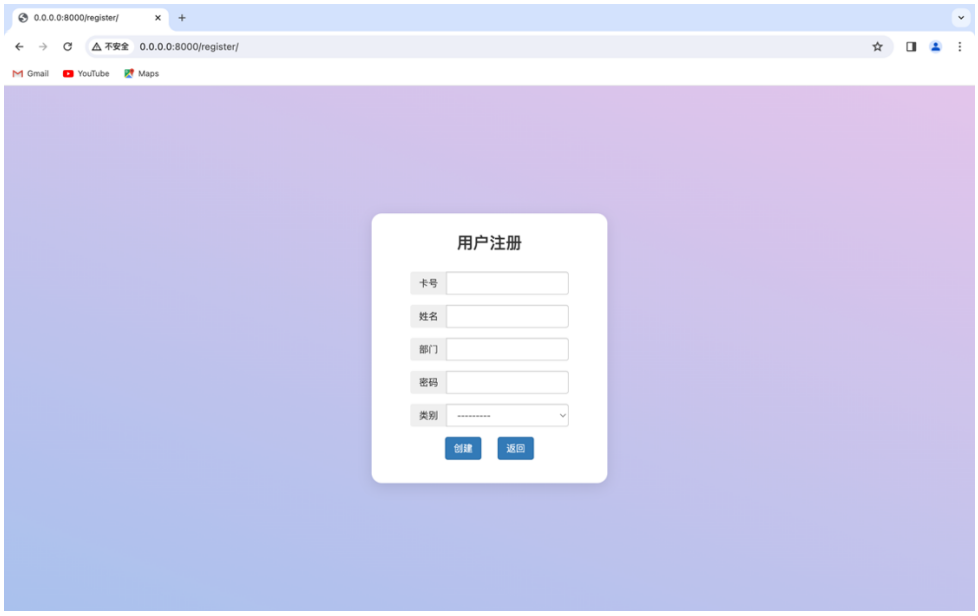

图 3 注册界面

以读者身份登陆成功后，就会来到图 4 所示界面，包括图书查询和借还图书两个功能；在图书查询中，可以输入书号、书名、作者、出版社，点击“搜索”就可以搜索书籍，如图 5 所示：

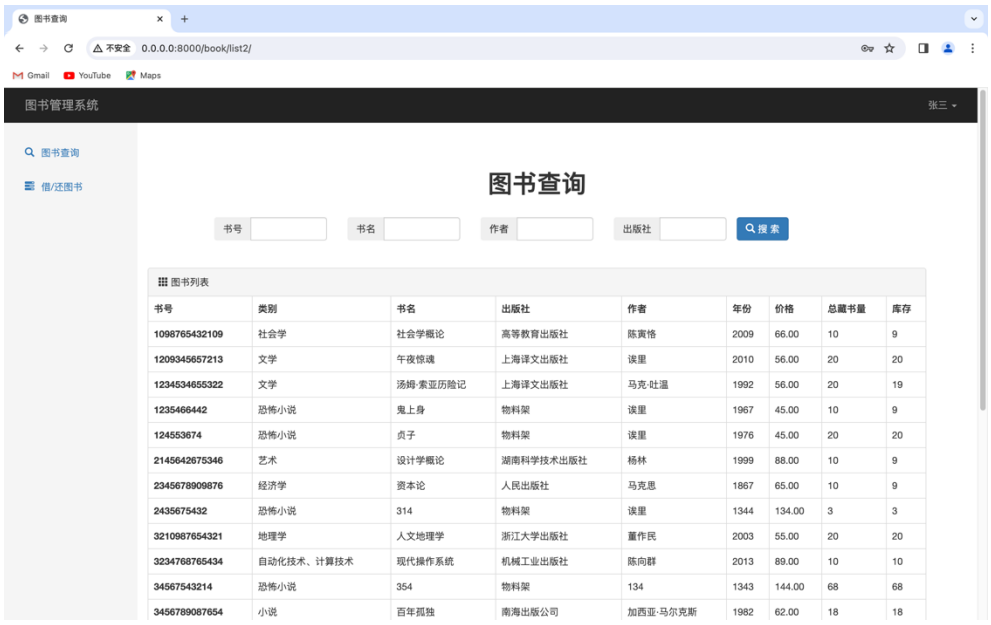

图 4 读者登陆后界面

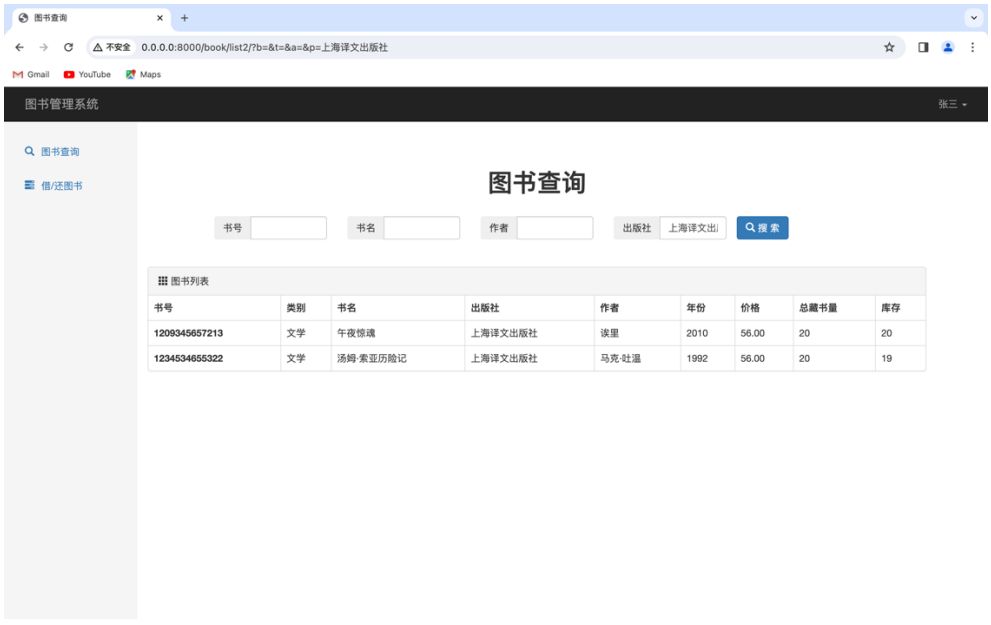

图 5 图书查询功能

在借还图书中，读者可以借阅图书，归还图书，续借图书，如图 6 所示；借阅图书会在借书记录表中添加一条记录；归还图书会在借书记录表中删除该条记录；续借图书会在借书记录表中修改该记录归还时间的值：

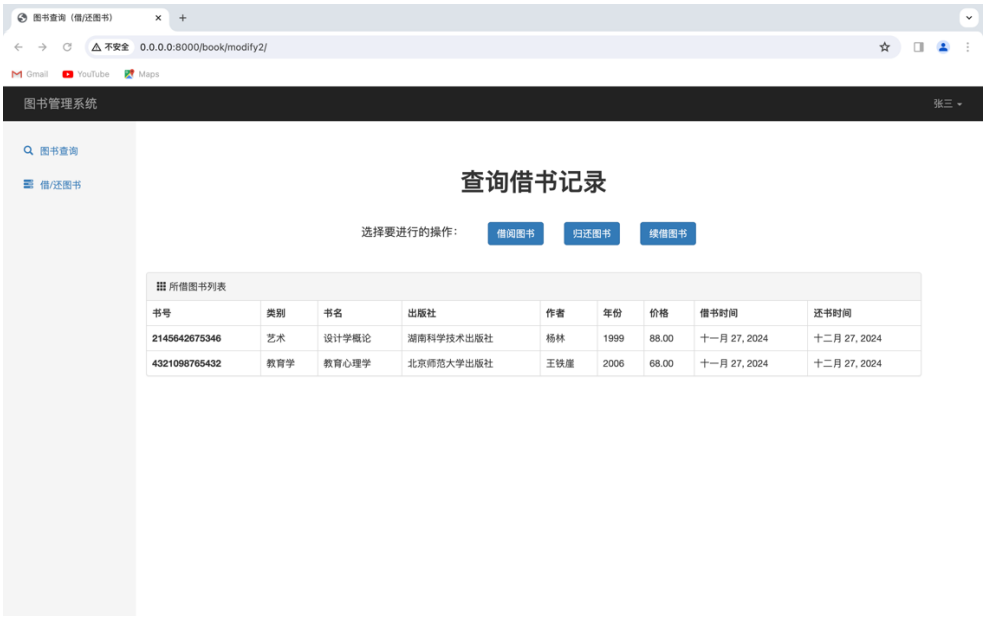

图 6 借/还图书功能

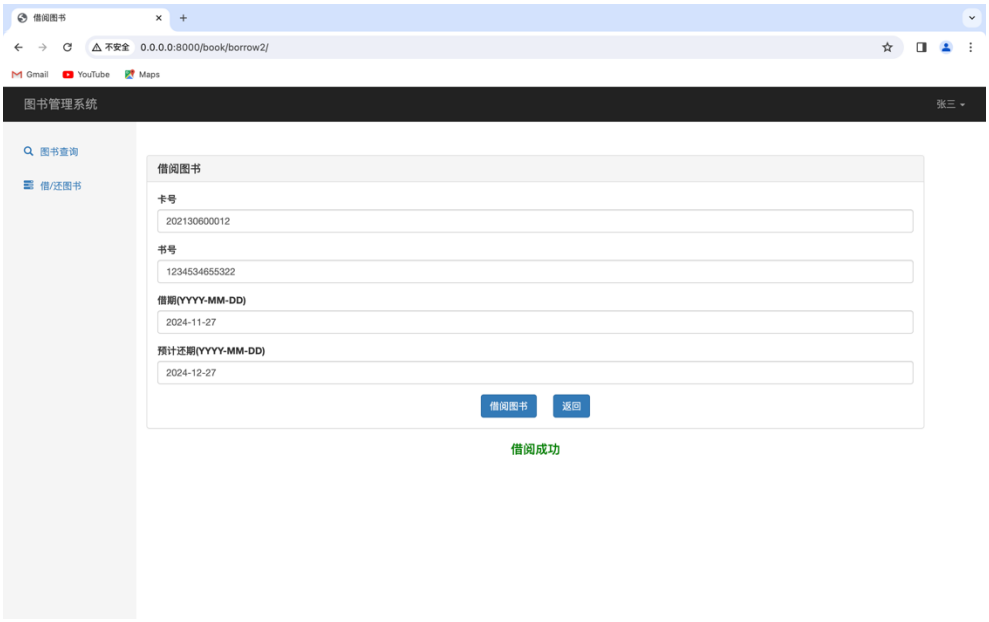

图 7 借阅图书

以管理员身份登陆成功后，就会来到图 8 所示界面，管理员身份多出两个功能，图书入库和借书证管理；在图书入库中，管理员可以输入图书的相关信息，点击图书入库就可以入库书籍，如图 9 所示；在借书证管理中，管理员可以对读者证进行增删改差，如图 10 所示：

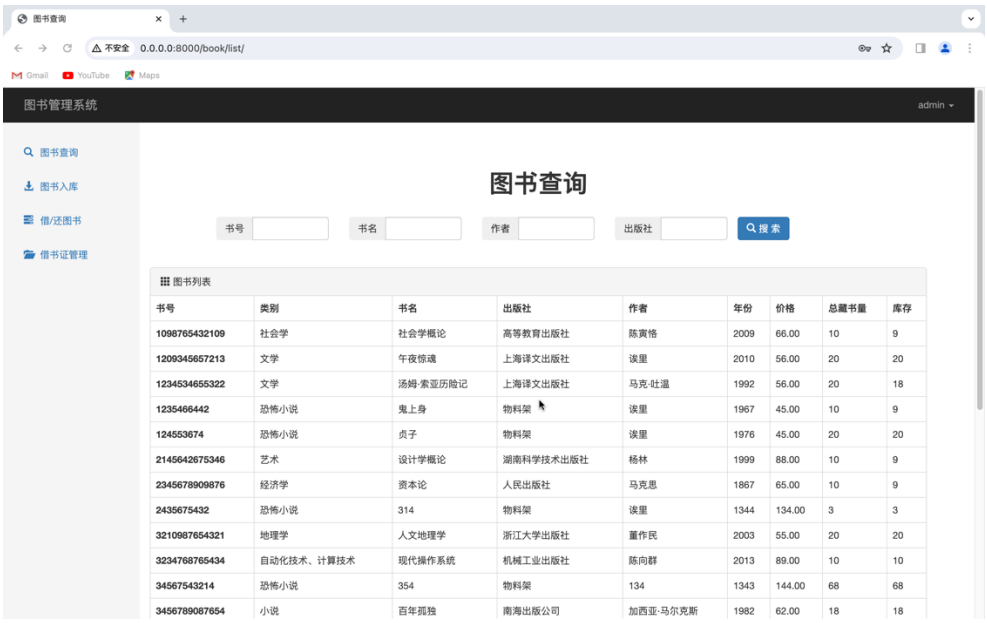

图 8 管理员登陆后界面

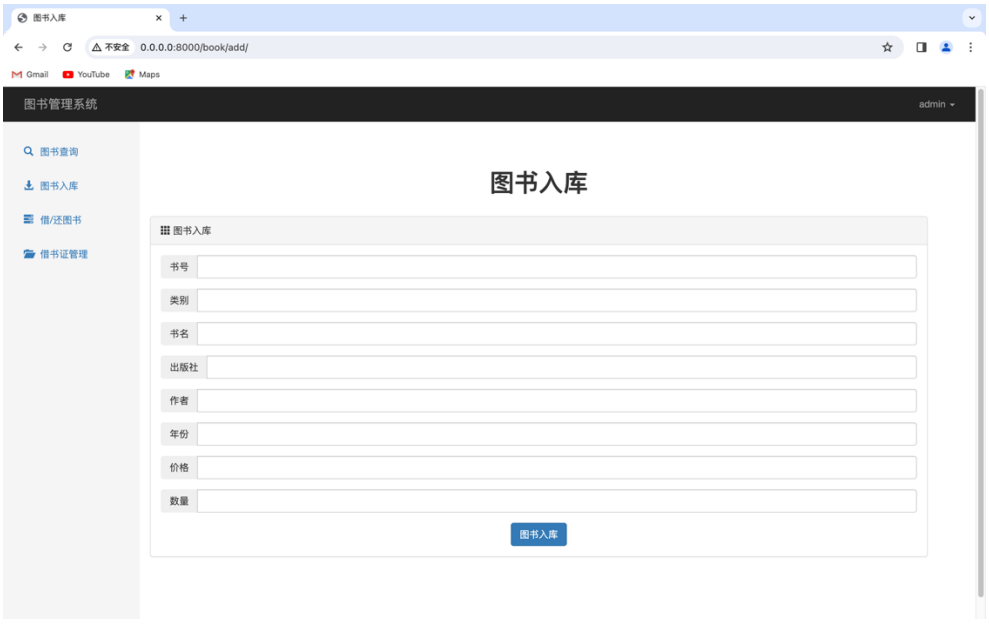

图 9 图书入库界面

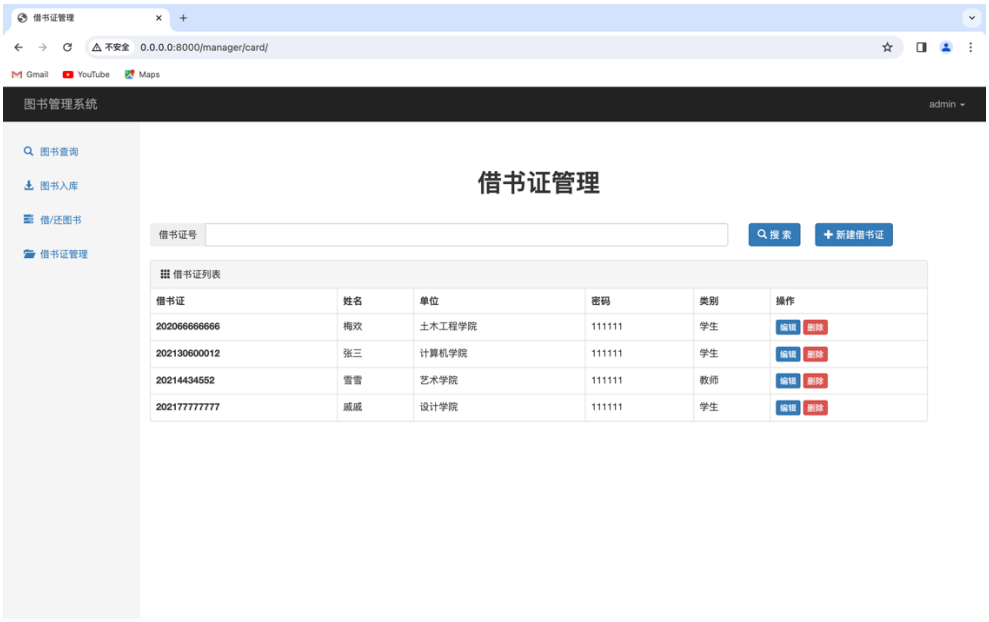

图 10 借书证管理界面

（二）代码解释

```
from dataclasses import field
from django import forms
from logging import PlaceHolder
from django.http import HttpResponse
from django.shortcuts import redirect, render
from web import models
import re


def manager(request):
    name = request.session["info"]["name"]
    id = request.session["info"]["id"]
    return render(request, 'manager.html', {"name": name})

def reader(request):
    name = request.session["info"]["name"]
    id = request.session["info"]["cno"]
    return render(request, 'reader.html', {"name": name})

def manager_card(request):
    name = request.session["info"]["name"]
    id = request.session["info"]["id"]
    nid = request.POST.get("nid")
    if (not nid):
```

```
        nid = ""
        queryset = models.card.objects.all()
        return render(request, 'manager_card.html', {"queryset": queryset,
"name": name, "nid": nid})

    queryset = models.card.objects.filter(cno=nid)
    if queryset:

        request.session["info"]["nid"] = nid
        request.session.set_expiry(60 * 60 * 24 * 7)
        print(request.session["info"])
        return render(request, 'manager_card.html', {"queryset": queryset,
"name": name, "nid": nid})
    else:
        return render(request, 'manager_card.html', {"error_msg": "无该借书证,
请检查", "name": name, "nid": nid})

def manager_card_delete(request):
    name = request.session["info"]["name"]
    id = request.session["info"]["id"]
    nid = request.GET.get('nid')
    print(nid)
    models.card.objects.filter(cno=nid).delete()
    return redirect('/manager/card/', {"name": name})

class CardModelform(forms.ModelForm):
    class Meta:
        model = models.card
        fields = '__all__'

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

        for name, field in self.fields.items():
            field.widget.attrs = {"class": "form-control"}

def manager_card_add(request):
    name = request.session["info"]["name"]
    id = request.session["info"]["id"]
    if request.method == "GET":
        form = CardModelform()
        return render(request, 'manager_card_add.html', {"form": form, "name":
name})
```

```
    form = CardModelform(data=request.POST)
    if form.is_valid():
        form.save()
        return redirect('/manager/card/')

    return render(request, 'manager_card_add.html', {"form": form, "name":
name})

class BookModelform(forms.ModelForm):
    num = forms.IntegerField(label='数量')
    book_id = forms.CharField(label='书号')

    class Meta:
        model = models.book
        fields = ['book_id', 'type', 'title',
                'publisher', 'author', 'year', 'price', 'num']

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

        for name, field in self.fields.items():
            field.widget.attrs = {"class": "form-control"}


def book_add(request):
    name = request.session["info"]["name"]
    id = request.session["info"]["id"]
    form = BookModelform()
    if request.method == "GET":
        return render(request, 'book_add.html', {"form": form, "name": name})
    data = request.POST
    form = BookModelform(data=request.POST)
    bno = data['book_id']
    if form.is_valid():
        obj = models.book.objects.filter(bno=bno)
        if obj:
            print(data['num'])
            row_object = obj[0]
            row_object.stock = row_object.stock+int(data['num'])
            row_object.total = row_object.total+int(data['num'])
            row_object.save()
        else:
            models.book.objects.create(bno=bno, type=data['type'],
title=data['title'], publisher=data['publisher'],
```

```
                                year=data['year'], author=data['author'],
price=data['price'], total=data['num'], stock=data['num'])
        return redirect('/book/add/suc/', {"name": name})
    return render(request, 'book_add.html', {"form": form, "name": name})


def book_add_suc(request):
    name = request.session["info"]["name"]
    id = request.session["info"]["id"]
    if request.method == "GET":
        return render(request, 'book_add_suc.html')
    return redirect('/book/add/', {"name": name})


def book_list(request):
    name = request.session["info"]["name"]
    id = request.session["info"]["id"]
    search_b = request.GET.get('b', "")
    search_t = request.GET.get('t', "")
    search_a = request.GET.get('a', "")
    search_p = request.GET.get('p', "")
    search_pl = request.GET.get('pl', "")
    search_pr = request.GET.get('pr', "")
    search_yl = request.GET.get('yl', "")
    search_yr = request.GET.get('yr', "")
    order = request.GET.get('order', "")

    res = models.book.objects.all().order_by('bno')

    if search_b:
        res = res.filter(bno__contains=search_b)
    if search_t:
        res = res.filter(title__contains=search_t)
    if search_a:
        res = res.filter(author__contains=search_a)
    if search_p:
        res = res.filter(publisher__contains=search_p)
    res = res.all()[:50]

    return render(request, 'book_list.html', {"name": name, "queryset":
res,"search_b":search_b, "search_t": search_t, "search_a": search_a,
"search_p": search_p, "search_pl": search_pl, "search_pr": search_pr,
"search_yl": search_yl, "search_yr": search_yr, "order": order})

def book_list2(request):
```

```
    name = request.session["info"]["name"]
    id = request.session["info"]["id"]
    search_b = request.GET.get('b', "")
    search_t = request.GET.get('t', "")
    search_a = request.GET.get('a', "")
    search_p = request.GET.get('p', "")
    search_pl = request.GET.get('pl', "")
    search_pr = request.GET.get('pr', "")
    search_yl = request.GET.get('yl', "")
    search_yr = request.GET.get('yr', "")
    order = request.GET.get('order', "")

    res = models.book.objects.all().order_by('bno')

    if search_b:
        res = res.filter(bno__contains=search_b)
    if search_t:
        res = res.filter(title__contains=search_t)
    if search_a:
        res = res.filter(author__contains=search_a)
    if search_p:
        res = res.filter(publisher__contains=search_p)
    res = res.all()[:50]

    return render(request, 'book_list2.html', {"name": name, "queryset":
res,"search_b":search_b, "search_t": search_t, "search_a": search_a,
"search_p": search_p, "search_pl": search_pl, "search_pr": search_pr,
"search_yl": search_yl, "search_yr": search_yr, "order": order})
class Borrowform(forms.Form):
    nid = forms.CharField(
        label="卡号",
        widget=forms.TextInput(attrs={"class": "form-control"})
    )
    bno = forms.CharField(
        label="书号",
        widget=forms.TextInput(attrs={"class": "form-control"})
    )

    borrow_date = forms.DateField(
        label="借期(YYYY-MM-DD)",
        widget=forms.TextInput(attrs={"class": "form-control"})
    )

    return_date = forms.DateField(
```

```
        label="预计还期(YYYY-MM-DD)",
        widget=forms.TextInput(attrs={"class": "form-control"})
    )


def book_borrow(request):
    name = request.session["info"]["name"]
    id = request.session["info"]["id"]
    nid = request.session["info"]["nid"]
    if request.method == "GET":
        form = Borrowform()
        return render(request, 'book_borrow.html', {"form": form, "name":
name, "nid": nid})

    form = Borrowform(data=request.POST)

    if form.is_valid():
        data = form.cleaned_data
        nid = data.get('nid')
        borrow_book = models.book.objects.filter(bno=data['bno'])
        if borrow_book:
            the_book = models.book.objects.filter(bno=data['bno'])[0]

        if (not borrow_book):
            return render(request, 'book_borrow.html', {"form": form,
"error_msg": "该图书不存在，请检查", "name": name, "nid": nid})
        elif the_book.stock <= 0:
            earliest_books = models.borrow_list.objects.filter(
                book_id=data['bno']).order_by("return_time")
            if earliest_books:
                earliest_book = earliest_books[0]
                return render(request, 'book_borrow.html', {"form": form,
"name": name, "nid": nid, "error_msg": "库存不足，借阅失败,预计最快归还时间: ",
"date": earliest_book.return_time})
            else:
                return render(request, 'book_borrow.html', {"form": form,
"error_msg": "该图书无库存，请检查", "name": name, "nid": nid})
        else:
            the_book.stock -= 1
            the_book.save()
            models.borrow_list.objects.create(
                book_id=data['bno'], card_id=nid, manager_id=id,
borrow_time=data['borrow_date'], return_time=data['return_date'])
            return render(request, 'book_borrow.html', {"form": form,
"suc_msg": "借阅成功", "name": name, "nid": nid})
```

```
    return render(request, 'book_borrow.html', {"form": form, "name": name,
"nid": nid})



def book_borrow2(request):
    name = request.session["info"]["name"]
    nid = request.session["info"]["id"]
    if request.method == "GET":
        form = Borrowform()
        return render(request, 'book_borrow2.html', {"form": form, "name":
name, "nid": nid})

    form = Borrowform(data=request.POST)
    if form.is_valid():
        data = form.cleaned_data
        nid = data.get('nid')
        borrow_book = models.book.objects.filter(bno=data['bno'])
        if borrow_book:
            the_book = models.book.objects.filter(bno=data['bno'])[0]

        if (not borrow_book):
            return render(request, 'book_borrow2.html', {"form": form,
"error_msg": "该图书不存在，请检查", "name": name, "nid": nid})
        elif the_book.stock <= 0:
            earliest_books = models.borrow_list.objects.filter(
                book_id=data['bno']).order_by("return_time")
            if earliest_books:
                earliest_book = earliest_books[0]
                return render(request, 'book_borrow2.html', {"form": form,
"name": name, "nid": nid, "error_msg": "库存不足，借阅失败,预计最快归还时间: ",
"date": earliest_book.return_time})
            else:
                return render(request, 'book_borrow2.html', {"form": form,
"error_msg": "该图书无库存，请检查", "name": name, "nid": nid})
        else:
            the_book.stock -= 1
            the_book.save()
            models.borrow_list.objects.create(
                book_id=data['bno'], card_id=nid,
borrow_time=data['borrow_date'], return_time=data['return_date'])
            return render(request, 'book_borrow2.html', {"form": form,
"suc_msg": "借阅成功", "name": name, "nid": nid})
    return render(request, 'book_borrow2.html', {"form": form, "name": name,
"nid": nid})
```

```python
class Reborrowform(forms.Form):
    nid = forms.CharField(
        label="卡号",
        widget=forms.TextInput(attrs={"class": "form-control"})
    )
    bno = forms.CharField(
        label="书号",
        widget=forms.TextInput(attrs={"class": "form-control"})
    )
    return_date = forms.DateField(
        label="预计还期(YYYY-MM-DD)",
        widget=forms.TextInput(attrs={"class": "form-control"})
    )


def book_reborrow(request):
    name = request.session["info"]["name"]
    id = request.session["info"]["id"]
    nid = request.session["info"]["nid"]
    if request.method == "GET":
        form = Reborrowform()
        return render(request, 'book_reborrow.html', {"form": form, "name":
name, "nid": nid})

    form = Reborrowform(data=request.POST)

    if form.is_valid():
        data = form.cleaned_data
        nid = data.get('nid')
        borrow_book = models.book.objects.filter(bno=data['bno'])
        new_return_time = data.get('return_date')

        try:
            borrow_list = models.borrow_list.objects.get(card_id=nid,
book_id=data['bno'])
        except models.borrow_list.DoesNotExist:
            return render(request, 'book_reborrow.html', {"form": form,
"error_msg": "该借阅记录不存在，请检查", "name": name, "nid": nid})

        borrow_list.return_time = new_return_time
        borrow_list.save()

        return render(request, 'book_reborrow.html', {"form": form, "suc_msg":
"续借成功", "name": name, "nid": nid})
```

```
    return render(request, 'book_reborrow.html', {"form": form, "name": name,
"nid": nid})

def book_reborrow2(request):
    name = request.session["info"]["name"]
    nid = request.session["info"]["id"]
    if request.method == "GET":
        form = Reborrowform()
        return render(request, 'book_reborrow2.html', {"form": form, "name":
name, "nid": nid})

    form = Reborrowform(data=request.POST)
    if form.is_valid():
        data = form.cleaned_data
        nid = data.get('nid')
        borrow_book = models.book.objects.filter(bno=data['bno'])
        new_return_time = data.get('return_date')

        try:
            borrow_list = models.borrow_list.objects.get(card_id=nid,
book_id=data['bno'])
        except models.borrow_list.DoesNotExist:
            return render(request, 'book_reborrow2.html', {"form": form,
"error_msg": "该借阅记录不存在，请检查", "name": name, "nid": nid})

        borrow_list.return_time = new_return_time
        borrow_list.save()

        return render(request, 'book_reborrow2.html', {"form": form,
"suc_msg": "续借成功", "name": name, "nid": nid})

    return render(request, 'book_reborrow2.html', {"form": form, "name":
name, "nid": nid})
class Returnform(forms.Form):
    bno = forms.CharField(
        label="书号",
        widget=forms.TextInput(attrs={"class": "form-control"})
    )

def book_return(request):
    name = request.session["info"]["name"]
    id = request.session["info"]["id"]
    nid = request.session["info"]["nid"]
```

```
    if request.method == "GET":
        form = Returnform()
        return render(request, 'book_return.html', {"form": form, "name":
name, "nid": nid})
    form = Returnform(data=request.POST)
    if form.is_valid():
        data = form.cleaned_data
        bno = data['bno']
        cno = nid
        info = models.borrow_list.objects.filter(book_id=bno, card_id=cno)
        if info:
            obj = info[0]
            obj.delete()
            the_book = models.book.objects.filter(bno=data['bno'])[0]
            the_book.stock += 1
            the_book.save()
            return render(request, 'book_return.html', {"form": form,
"suc_msg": "归还成功", "name": name, "nid": nid})
        else:
            return render(request, 'book_return.html', {"form": form,
"error_msg": "归还失败，该书不存在该借书证借阅列表中", "name": name, "nid": nid})
    return render(request, 'book_return.html', {"form": form, "name": name,
"nid": nid})

def book_return2(request):
    name = request.session["info"]["name"]
    id = request.session["info"]["id"]
    nid = request.session["info"]["nid"]

    if request.method == "GET":
        form = Returnform()
        return render(request, 'book_return2.html', {"form": form, "name":
name, "nid": nid})
    form = Returnform(data=request.POST)
    if form.is_valid():
        data = form.cleaned_data
        bno = data['bno']
        cno = nid
        info = models.borrow_list.objects.filter(book_id=bno, card_id=cno)
        if info:
            obj = info[0]
            obj.delete()
            the_book = models.book.objects.filter(bno=data['bno'])[0]
```

```
            the_book.stock += 1
            the_book.save()
            return render(request, 'book_return2.html', {"form": form,
"suc_msg": "归还成功", "name": name, "nid": nid})
        else:
            return render(request, 'book_return2.html', {"form": form,
"error_msg": "归还失败，该书不存在该借书证借阅列表中", "name": name, "nid": nid})
    return render(request, 'book_return2.html', {"form": form, "name": name,
"nid": nid})

def book_modify(request):
    name = request.session["info"]["name"]
    id = request.session["info"]["id"]
    if request.method == "GET":
        request.session["info"]["nid"] = ""
        request.session.set_expiry(60 * 60 * 24 * 7)
        return render(request, 'book_modify.html', {'name': name})

    nid = request.POST.get("nid")
    card = models.card.objects.filter(cno=nid)

    if card:

        books =
models.borrow_list.objects.filter(card_id=nid).order_by('book_id')
        request.session["info"]["nid"] = nid
        request.session.set_expiry(60 * 60 * 24 * 7)
        print(request.session["info"])

        queryset = []
        for obj in books:
            book = models.book.objects.get(bno=obj.book_id)
            borrow_info = models.borrow_list.objects.get(book=obj.book_id)
            book.borrow_time = borrow_info.borrow_time
            book.return_time = borrow_info.return_time
            queryset.append(book)
        return render(request, 'book_modify.html', {"queryset": queryset,
"name": name, "nid": nid})
    else:
        return render(request, 'book_modify.html', {"error_msg": "无该借书证，请
检查", "name": name, "nid": nid})


def book_modify2(request):
    name = request.session["info"]["name"]
```

```
    uno = request.session["info"]["id"]

    if request.method == "GET":
        request.session["info"]["nid"] = ""
        request.session.set_expiry(60 * 60 * 24 * 7)
        books =
models.borrow_list.objects.filter(card_id=uno).order_by('book_id')  # 使用读者
身份号查询借阅的图书信息
        request.session["info"]["nid"] = uno
        request.session.set_expiry(60 * 60 * 24 * 7)
        print(request.session["info"])

        queryset = []
        for obj in books:
            book = models.book.objects.get(bno=obj.book_id)
            borrow_info = models.borrow_list.objects.get(book=obj.book_id)
            book.borrow_time = borrow_info.borrow_time
            book.return_time = borrow_info.return_time
            queryset.append(book)
        return render(request, 'book_modify2.html', {"queryset": queryset,
"name": name, "nid": uno})
```

3.4 数据库的连接

（1）安装依赖:

django==3.2.16

mysqlclient==2.1.1

（2）创建数据库：命名为 booksystem

（3）配置 Mysql 接口：数据库库的配置文件在./library/settings.py，在 settings 文件中修改用户名和密码

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',  # 默认
        'NAME': 'booksystem',  # 连接的数据库
        'HOST': '127.0.0.1',  # mysql 的 ip 地址
        'PORT': 3306,  # mysql 的端口
        'USER': 'root',  # mysql 的用户名
        'PASSWORD': '123456'  # mysql 的密码
```

```
    }
}
```

（4）启动数据库迁移：

python manage.py makemigrations

python manage.py migrate

（5）运行输入：

python manage.py runserver 0.0.0.0:8000

（6）显示创建成功后，浏览器转入：

http://127.0.0.1:8000/login/

即可进入登陆界面

四、总结

21 世纪是信息化时代，作为信息搜集、存储、加工、传播中心的图书馆更要适应时代变迁采用信息化的管理方式[3]。数据库管理系统实现管理的自动化和科学化，将其引入图书馆的建设必将改变图书馆原有的面貌，便携图书馆的管理者以及借阅者。

项目对图书管理系统进行了合理、全面的分析，该系统对图书馆简化流程和数据准确等方面达到预期目标，提高了图书馆管理工作的效率和速度[4]。运行中该系统操作方便、运行稳定，可以满足中小型图书馆管理的需要。当然，我们也看到现有系统存在的一些不足，进行反思为下一步的研发打下基础。

1. 图书信息管理系统实现的功能

（1）图书查询功能

（2）图书入库功能

对图书信息添加、修改。

（3）借/还图书功能

图书出借时考虑两个个前提：

A. 该书是否在库中；

B. 读者是否已经借满其限额；

如果不存在以上情况，则可以出借。

读者还书的时候可以续借该图书，续借的过程主要是修改借书记录里的还书日期。

（4）读者证管理功能

对读者的登录账号、密码进行添加、修改、删除。

2. 图书信息管理系统存在的不足

（1）本系统与众多其他系统均缺乏地理信息系统

A.不能准确直观地指明图书所在的空间位置

B.不能清楚表达各图书相关要件的准确位置和它们之间的相对关系

C.不能回答“某本书位于某本书的那个方位，距离多远，某两本书之间是否相邻等问题”

（2）存在严重的信息孤岛现象

信息孤岛现象指的是指图书馆会持续不断增加新的独立系统，然而这些管理系统却不能包含图书馆所有的业务，使得每个系统之间孤立而无联系。现在许多图书馆都要大力开展数字化的业务，也会建立许多独立的系统，这无疑会增加图书馆整体运营成本，同时也会给读者带来诸多不便，更对图书馆整体资源的有机整合造成不利影响。

（3）更是没有考虑到评价模块，以至于读者在读书借阅过程中的“失声”

图书馆的所有功能应当以读者为核心，图书馆和内部工作人员就要建立以人为本的思想理念。然而目前图书馆还没有建立切实可行的服务评价运行模式，图书馆也就不能及时有效地了解到读者的意见和建议，更无法熟悉不同读者的不同需求。从而使得图书馆的服务方式落后陈旧，显然不能更好地发挥图书馆的价值。

3. 心得体会

在制作图书馆管理系统的过程中，我明显觉察到自身所学的知识仍需继续填充。就像烘培，在精准加料、稳妥实施中，一个管理系统在我们手下从无到有诞生了，正如谚语——成功的经验是自信的源泉，我们也在这个过程中对数据库翔实的步骤、思想、方法和技术成功复现，尤其是对基本表、视图、索引、存储过程的运用熟练度也更上一层了于是，我们或多或少增加了对数据库系统操作的信心，。

当然，实践中，除却再次运用课堂所学，更是自我探索了许多有趣的新知识，比如 Bootstrap 等等，培养了相当客观的自学能力。

五、参考文献

[1] 赵满华,高洁．图书馆白动化管理系统建设与发展口．北京：情报科学，2009，20

[2] 杜洋.图书馆图书管理系统的设计与实现[D].电子科技大学，2013.

[3] 宫昌利.图书管理系统的设计与实现[D].山东大学，2009.

[4] 顾俐.图书馆图书管理系统的设计[J].中国科技信息，2007 (11)：175-177.