华南理工大学

South China University of Technology

Library management system Development Specification

surname Zejun Huang

Faculty School of Design

Professional Product Design

Student number 202130670058

Completion time 2023.1128

Table of Contents

1. Project planning 1

1.1 Project Background 1

1.2 Positioning and function of the system 2

1.3 Development schedule 2

2. Database design 2

2.1 Functional requirements design 3

(1) Book query function 3

(2) Book warehousing function 3

(3) Borrow/return books 3

(4) Library card management features 3

2.2 Conceptual Structural Design (ER Diagram) 3

2.3 Logical structure design 4

(1) Library table 4

(2) Book warehousing table 4

(3) Reader's Table 5

(4) Administrator Table 5

2.4 Data Dictionary 5

3. System development and implementation 9

3.1 Development Tools and Environment 9

(1) Development tools 9

(2) Environment configuration 9

3.2 Create a table and insert initial data.....9

3.3 Introduction to the login interface and functions.....30

(1) Graphical page introduction.....30

(ii) Code Interpretation.....35

3.4 Database Connections.....56

IV. Summary.....57

1. The functions implemented by the library information management system.....58

(1) Book query function.....58

(2) Book storage function.....58

(3) Borrow/return the function of books.....58

(4) Library card management function.....58

2. Deficiencies in the library information management system.....58

(1) This system and many other systems lack geographic information systems.....59

(2) There is a serious phenomenon of information islands.....59

(3) The evaluation module is not taken into account, so that the reader loses his voice in the process of borrowing" 59

3. Experience.....60

5. References 60

1、Project planning

1.1 Project Background

The library management system is the use of information technology (Information Technology, abbreviated as IT) is an information management system built by electronically managing the library[1]。 With the rapid development of informatization, more and more libraries have transformed the traditional book management mode and built an information-based management system to improve work efficiency.

Compared with the traditional manual management mode, the library management system has the advantages of convenient and fast retrieval, fast retrieval, accurate retrieval results, large amount of stored data, low cost, resource saving, and good human-computer interaction interface[2]。 The library of South China University of Technology was completed and put into use in September 2006, with a construction area of 42,300 square meters, and has five administrative departments, including office, literature resources construction department, literature resources service department, reference consulting service department, and information technology service department. This project consulted the staff of the library of the University Town Campus of South China University of Technology, and combined with the characteristics of the library, designed a library management system with the functions of library information management, reader information management, query and modification.

1.2 Positioning and function of the system

A simple library management database system was designed and developed. The main databases of the system include library book information, school teacher and student information, and teacher and student borrowed book information. The system is aimed at librarians and patrons.

The main implemented functions are as follows:

Librarians may complete the registration, modification, and cancellation of books based on the flow of books, and complete the registration, modification, and cancellation management of readers based on their registration.

Readers can borrow, renew, and return books on their borrowing status.

1.3 Development schedule

On September 10, 2023, it was determined to develop a library management system and draw up the functional structure design of the management system.

On September 20, 2023, the entity relationship diagram (E-R diagram) and data table were created, and the database was created in the Mysql database.

On October 10, 2023, the user interface was initially established, the user interface was optimized, and the user terminal functions were improved.

On October 30, 2023, the front-end application will connect to the database.

On November 20, 2023, the final commissioning was completed.

2、Database design

2.1 Functional requirements design

(1) Book query function

(2) Book warehousing function

Add and modify book information.

(3) Borrow/return books

There are two prerequisites to consider when lending books:

A. whether the book is in the library;

B. whether the reader has borrowed the full limit;

If none of the above is true, it can be loaned.

Readers can renew the book when they return the book, and the renewal process is mainly to modify the return date in the borrowing record.

(4) Library card management features

Add, modify, and delete the reader's login account and password.

2.2 Conceptual Structural Design (ER Diagram)

The data information in the database includes the following:

(1) Book information

(2) Borrowing record information

(3) Reader Information

(4) Administrator Information

The relationship between these data items can be illustrated in Figure 1:

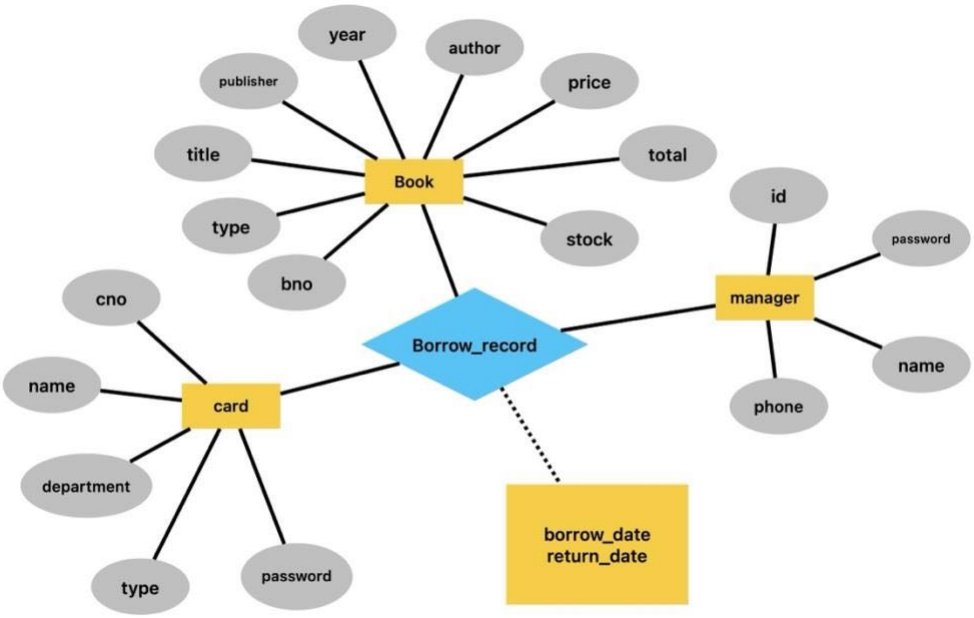

Figure 1 E-R diagram

2.3 Logical structure design

A total of 4 tables are designed in this system.

According to the needs of the library management information system, through the analysis of the content and data flow of the library management process, four data tables are designed as follows:

(1) Library table

Attributes: Book Number, Category, Book Title, Publisher, Year, Author, Price, Total, Inventory

Primary key: Book number

(2) Borrowing record form

Attributes: Borrowing record number, borrowing date, return date, book number, library card ID, handling person number

Primary key: The borrowing record number

(3) Reader's Table

Attributes: Reader ID, Reader Name, Department (College), Identity, Password

Primary key: Reader ID

(4) Administrator Table

Attributes: Admin ID, Password, Name, Contact Information

Primary key: Administrator ID

2.4 Data Dictionary

(1) Library table

Library table: web_book

The name of the data item	type	length	precision	scale	Value range	Default value	restraint
ISBN: BNO	VARCHAR	32					PRIMARY KEY
Category: type	VARCHAR	30					NOT NULL
Title: Title	VARCHAR	32					NOT NULL
Publisher: publisher	VARCHAR	32					NOT NULL

Year: year	INT						NOT NULL
Author: author	VARCHAR	32					NOT NULL
Price: price	DECIMAL	8		2			NOT NULL
Total: total	INT						NOT NULL
Stock: Stock	INT						NOT NULL

(2) Borrowing record form

Borrowing Record: Web_borrow_list

The name of the data item	type	length	precision	scale	Value range	Default value	restraint
Borrowing Record Number: ID	BIGINT						PRIMARY KEY
Loan date: borrow_time	DATE						NOT NULL

Return date: return_time	DATE						NOT NULL
ISBN: book_id	VARCHAR	32					FOREIGN KEY
Reader ID: Card_id	VARCHAR	32					FOREIGN KEY
Handler Number: Manage_ ID	VARCHAR	32					FOREIGN KEY

(3) Reader's Table

Reader's Table: web_card

The name of the data item	type	length	precision	scale	Value range	Default value	restraint
Reader ID: CNo	VARCHAR	32					PRIMARY KEY
Reader's Name: name	VARCHAR	32					NOT NULL

Department (college) :department	VARCHAR	32					NOT NULL
Identity: type	SMALLINT				1 or 2		NOT NULL
Password: password	VARCHAR	32					NOT NULL

(4) Administrator Table

Administrator table: web_manager

The name of the data item	type	length	precision	scale	Value range	Default value	restraint
Admin ID: id	VARCHAR	32					PRIMARY KEY
Password: password	VARCHAR	32					NOT NULL
Name: name	VARCHAR	32					NOT NULL
Contact: contact	VARCHAR	20					NOT NULL

3、System development and implementation

3.1 Development Tools and Environment

(1) Development tools

The development environment is Visual Studio Code, the programming language is python3.9, the database is Mysql, and the database design and modification use Navicat Premium.

When creating pages, use Bootstrap (which relies on jQuery) as the front-end framework.

Django's ORM feature enables you to manipulate data in a database using operands.

The library management system on the web side is generated on the local port (localhost), and the administrator users on the same LAN can access the web side through the address to manage the book.

(2) Environment configuration

Computer　MacBook AIR

Chip Apple M1

OS version　macOS Sonoma Version 14.1.1

3.2 Create a table and insert initial data

```
SET NAMES utf8mb4;

SET FOREIGN_KEY_CHECKS = 0;



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

```sql
-- 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`;
```

```sql
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;
```

```sql
-- ----------------------------

-- 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
-- ----------------------------
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;
```

```sql
-- ----------------------------

-- 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
-- --------------------------
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_au2TkHR1
tFz-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;

INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`)

VALUES ( '1098765432109', 'Sociology', 'Introduction to Sociology', 'Higher Education Press', 2009,

'Chen Yinke', 6600, 10, 9);
```

```
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`)
VALUES ( '1209345657213', 'Literature', 'Midnight Terror', 'Shanghai Translation Publishing House',
2010, ' Eh, 5600, 20, 20);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`)
VALUES ( '1234534655322', 'Literature', 'The Adventures of Tom Sawyer', 'Shanghai Translation
Publishing House', 1992, 'Mark Twain', 5600, 20, 18);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`)
VALUES ( '1235466442', 'Horror Fiction', 'Ghost Body', 'Material Shelf', 1967, 'Eh' , 45.00, 10, 9);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`)
VALUES ( '124553674', 'Horror Novel', 'Sadako', 'Material Shelf', 1976, 'E-ri', 45.00, 20, 20);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`)
VALUES ( '2145642675346', 'Art', 'Introduction to Design', 'Hunan Science and Technology Press', 1999
'Yang Lin', 8800, 10, 9);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`)
VALUES ( '2345678909876', 'Economics', 'Capital', 'People's Publishing House', 1867, ' Marx', 6500, 10,
9);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`)
VALUES ( '2435675432', 'Horror Novel', '314', 'Material Shelf', 1344, 'E-Li' , 134.00, 3, 3);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`)
VALUES ( '3210987654321', 'Geography', 'Human Geography', 'Zhejiang University Press', 2003 'Dong
Zuomin', 5500, 20, 20);
```

```
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`)
VALUES ( '3234768765434', 'Automation Technology, Computing Technology', 'Modern Operating
System', 'China Machine Press', 2013, 'Chen Xiangqun', 8900, 10, 10);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`)
VALUES ( '34567543214', 'Horror Fiction', '354', 'Material Shelf', 1343, '134' , 144.00, 68, 68);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`)
VALUES ( '3456789087654', 'Novel', 'One Hundred Years of Solitude', 'Nanhai Publishing Company',
1982, ' García Márquez', 6200, 18, 18);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`)
VALUES ( '4321098765432', 'Pedagogy', 'Educational Psychology', 'Beijing Normal University Press',
2006 'Wang Tieya', 6800, 15, 14);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`)
VALUES ( '5678945612123', 'History', 'Ancient Chinese History', 'Peking University Press', 2005, ' Wang
Guowei', 4500, 15, 15);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`)
VALUES ( '6543210987654', 'Medicine', 'Human Anatomy', 'People's Medical Publishing House', 2015, '
LI Shizhong', 9200, 10, 10);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`)
VALUES ( '6789098765432', 'Law', 'Criminal Law', 'Law Press', 2008, 'Song Maorong' , 72.00, 12, 12);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`)
VALUES ( '7654323456789', 'Philosophy', 'On Human Dignity', 'The Commercial Press', 1785, ' Kant',
7500, 8, 7);
```

```
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`)
VALUES ( '8765432109876', 'Political Science', 'Principles of Political Science', 'Tsinghua University
Press', 1985, 'Habermas', 7900, 6, 6);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`)
VALUES ( '8909876543210', 'Computer Science', 'Computer Networks', 'Tsinghua University Press',
2007 'Xie Xiren', 7800, 8, 8);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`)
VALUES ( '9087654321345', 'Popular Science', 'A Wonderful Journey to the Universe', 'Science Press',
2011, ' Stephen Hawking', 6800, 12, 10);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`)
VALUES ( '9787010009254', 'Marxist-Leninist Mao Deng Thought', 'Selected Works of Mao Zedong',
'People's Publishing House', 1991 'Mao Zedong', 8100, 5, 5);
INSERT INTO `web_book` (`bno`, `type`, `title`, `publisher`, `year`, `author`, `price`, `total`, `stock`)
VALUES ( '9876543212345', 'Psychology', 'The Rabble', 'Shanghai People's Publishing House', 1895, '
Le Pen', 5900, 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
-- ----------------------------
```

```sql
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',
'Mei Huan', 'College of Civil Engineering', 1, '111111');
INSERT INTO `web_card` (`cno`, `name`, `department`, `type`, `password') VALUES ('202130600012',
'ZHANG San', 'Computer Science', 1, '111111');
INSERT INTO `web_card` (`cno`, `name`, `department`, `type`, `password') VALUES ('20214434552',
'Xuexue', 'Art Academy', 2, '111111');
INSERT INTO `web_card` (`cno`, `name`, `department`, `type`, `password') VALUES ('202177777777',
'Kin', 'College of Design', 1, '111111');
COMMIT;
```

```sql
-- ----------------------------

-- 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 Introduction to the login interface and functions

（1）Graphical page introduction

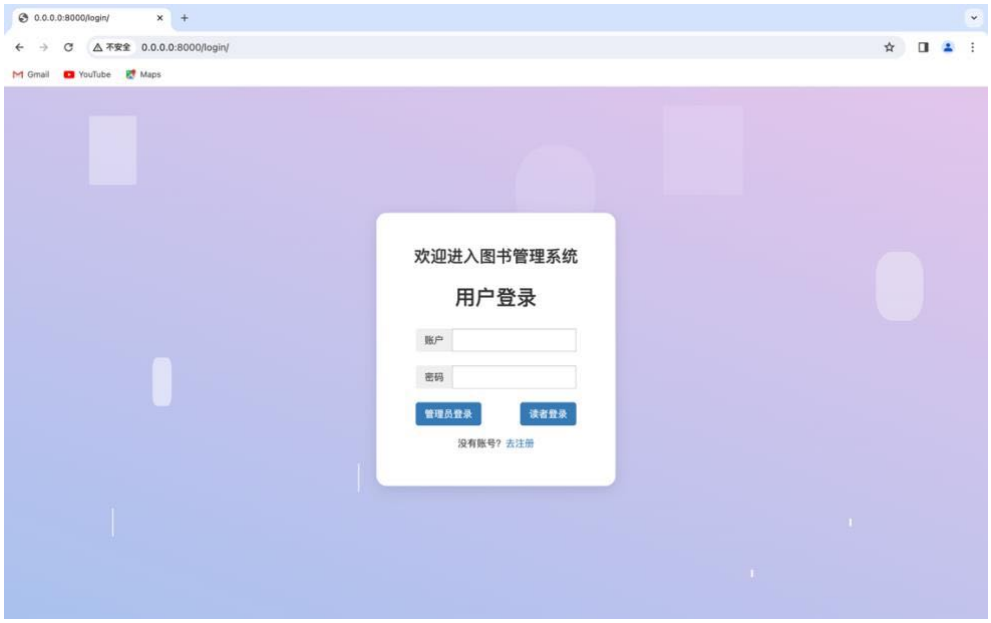

Figure 2 Login page

Enter the account and password, click Administrator Login or Reader Login to go to the database (Reader Table and Administrator Table) to verify whether the account and password are correct, if so, enter the library management system. If you don't have an account, you can click "Go to Register" to register a reader account and then log in, as shown in Figure 3:

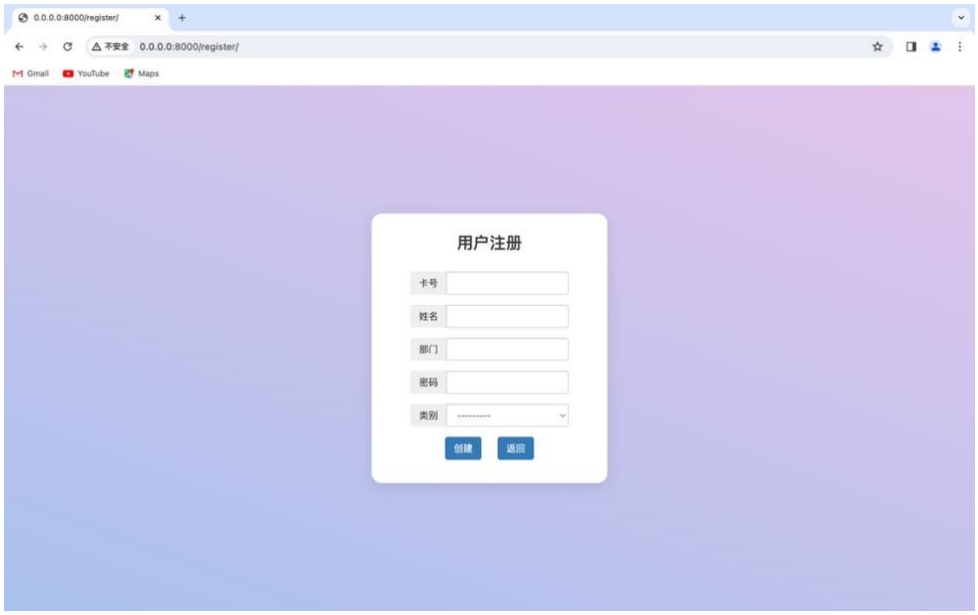

Figure 3 Registration page

After logging in as a reader, you will come to the interface shown in Figure 4, including two functions of book query and borrowing and returning books; in the book query, you can enter the book number, book title, author, publisher, and click "Search" to search for books, as shown in Figure 5:

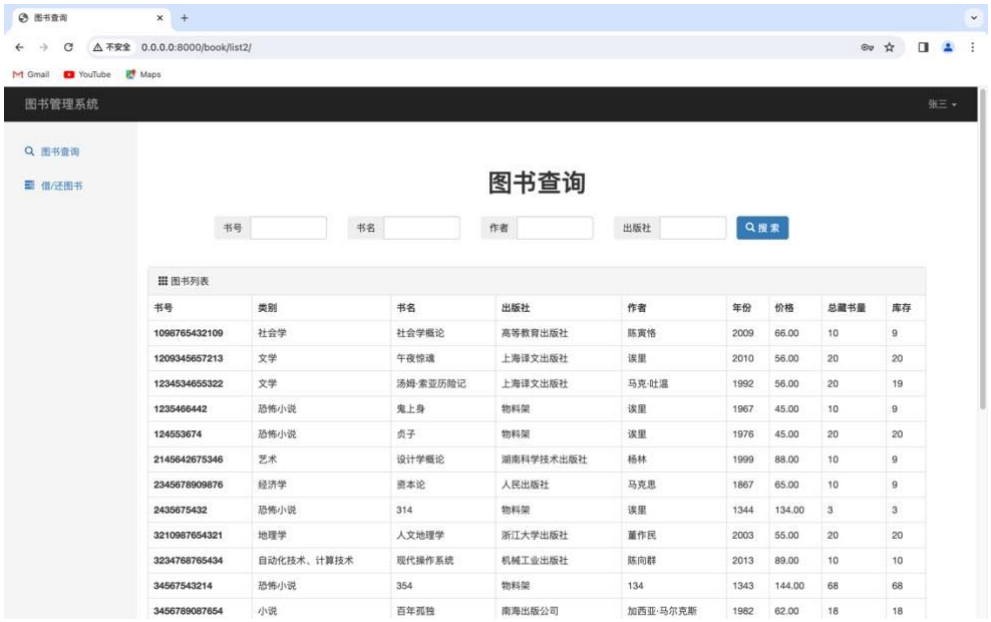

Figure 4 The screen after the reader logs in

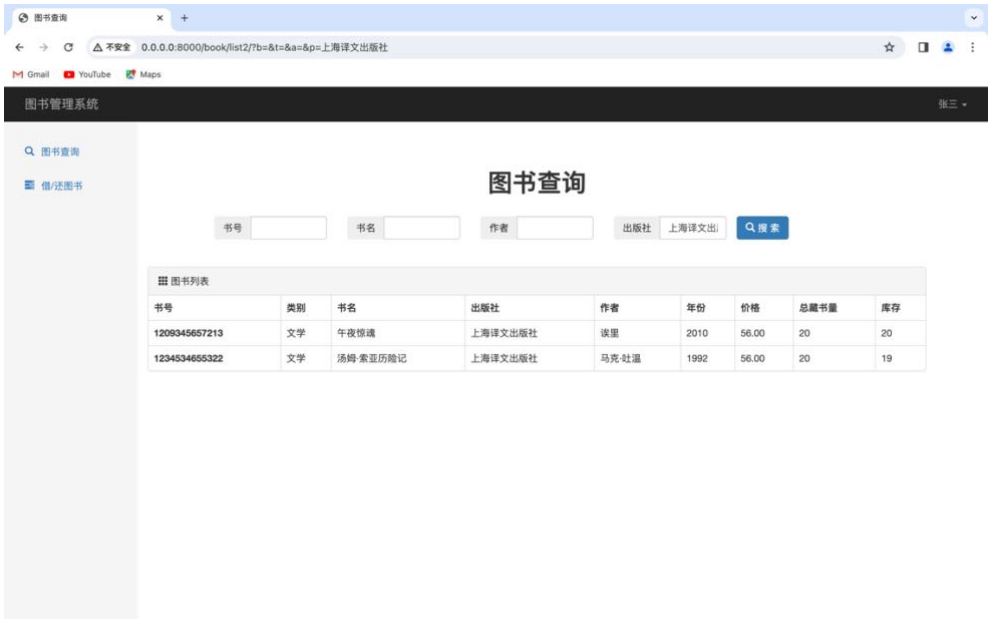

Figure 5 Book query function

In borrowing and returning a book, the reader can borrow the book, return the book, and renew the book, as shown in Figure 6; the borrowed book will add a record to the borrowing record table; the return of the book will delete the record in the borrowing record table; and the value of the return time of the record will be modified in the borrowing record table when renewing the book:

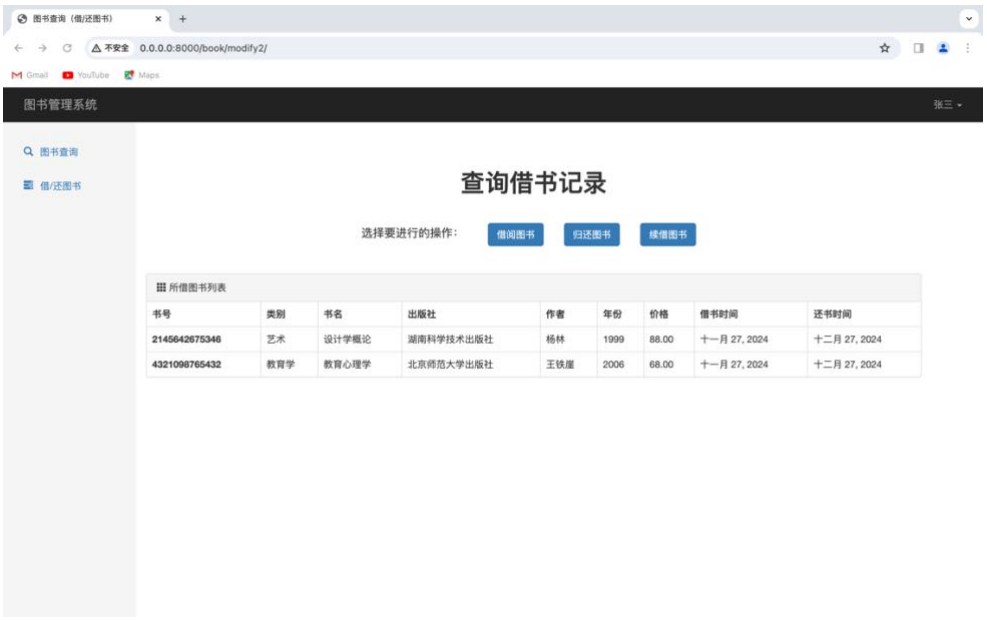

Figure 6 Borrowing/returning books

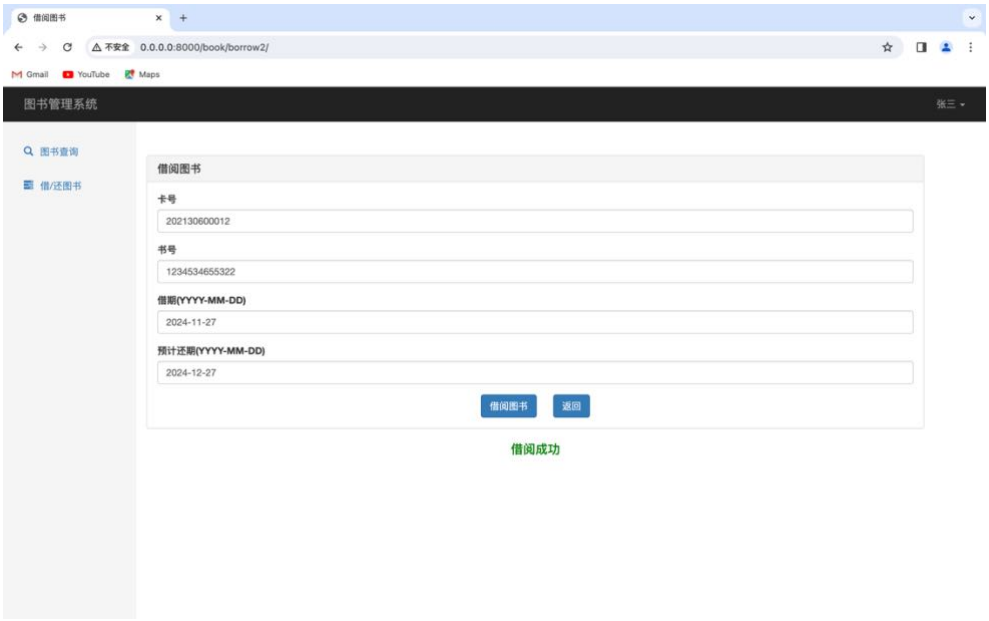

Figure 7 Borrowing books

After logging in as an administrator successfully, you will come to the interface shown in Figure 8, and there are two more functions as an administrator, book storage and library card management; in the book storage, the administrator can enter the relevant information of the book, click the book

to enter the library, as shown in Figure 9; in the library card management, the administrator can add, delete and modify the library card, as shown in Figure 10:

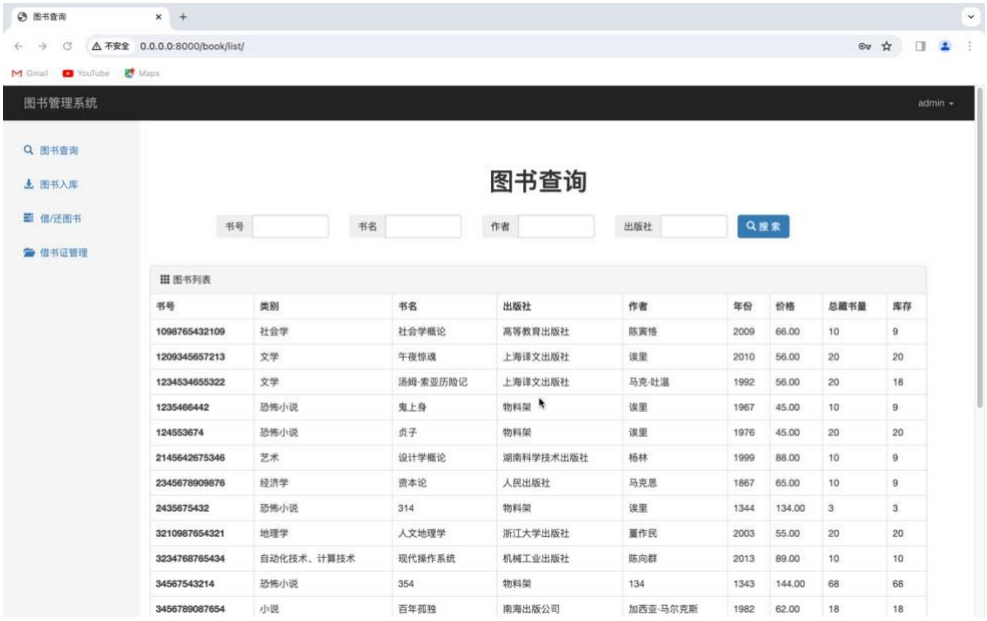

Figure 8 After the administrator logs in

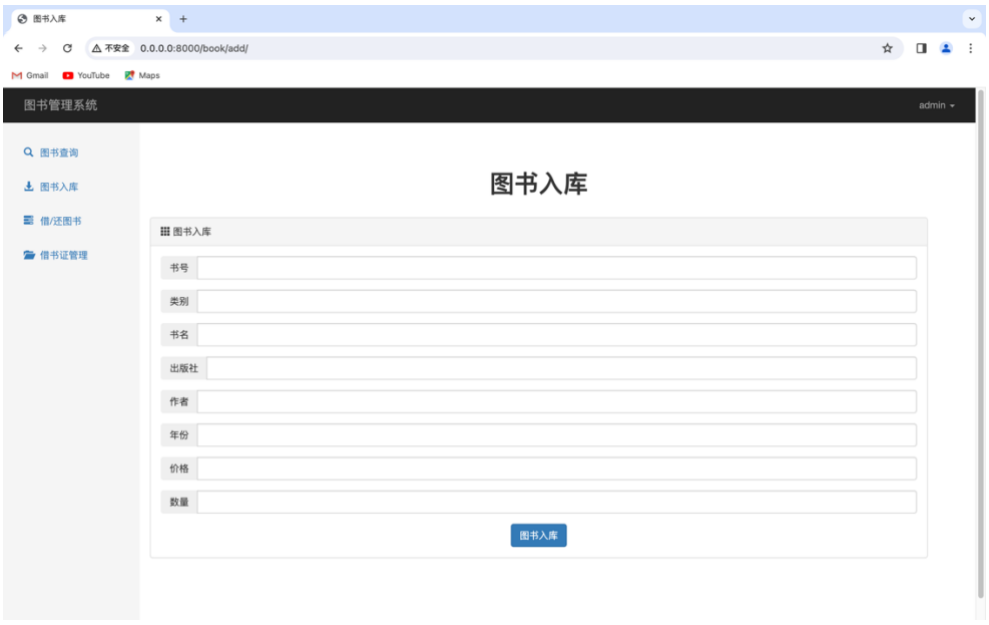

Figure 9 The book storage page

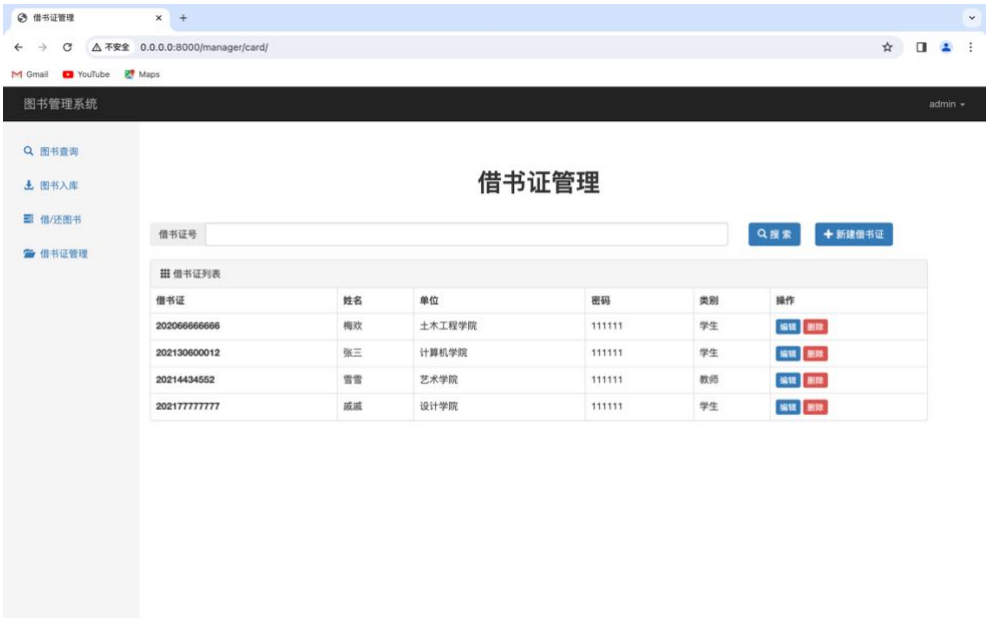

Figure 10 Library card management page

（2）Code Explanation

```
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": "No library card, please check",
"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='Quantity').
    book_id = forms.CharField(label='ISBN').


    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="Card Number",
        widget=forms.TextInput(attrs={"class": "form-control"})
    )
    bno = forms.CharField(
        label="ISBN",
        widget=forms.TextInput(attrs={"class": "form-control"})
    )

    borrow_date = forms.DateField(
```

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

        return_date = forms.DateField(
            label="Estimated Repayment (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": "The book does not
exist, please check", "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": "Out of stock, failed to borrow, estimated fastest return time: "., "date":
earliest_book.return_time})
                else:
                    return render(request, 'book_borrow.html', {"form": form, "error_ msg": "This book is
out of stock, please check", "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": "borrowing
successful", "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": "The book does
not exist, please check", "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": "Out of stock, failed to borrow, estimated fastest return time: "., "date":
earliest_book.return_time})
                else:
                    return render(request, 'book_borrow2.html', {"form": form, "error_ msg": "This book is
out of stock, please check", "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": "borrowing
successful", "name": name, "nid": nid}).
```

```
    return render(request, 'book_borrow2.html', {"form": form, "name" : name, "nid": nid})



class Reborrowform(forms.Form):
    nid = forms.CharField(
        label="Card Number",
        widget=forms.TextInput(attrs={"class": "form-control"})
    )
    bno = forms.CharField(
        label="ISBN",
        widget=forms.TextInput(attrs={"class": "form-control"})
    )
    return_date = forms.DateField(
        label="Estimated Repayment (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": "This borrowing
record does not exist, please check", "name": name, "nid": nid})



            borrow_list.return_time = new_return_time

            borrow_list.save()



            return render(request, 'book_reborrow.html', {"form": form, "suc_ msg": "Renewal successful",
"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": "This borrowing

record does not exist, please check", "name": name, "nid": nid})
```

```
            borrow_list.return_time = new_return_time

            borrow_list.save()



            return render(request, 'book_reborrow2.html', {"form": form, "suc_ msg": "Renewal successful",
"name": name, "nid": nid}).



        return render(request, 'book_reborrow2.html', {"form": form, " name": name, "nid": nid})
class Returnform(forms.Form):
        bno = forms.CharField(

            label="ISBN",

            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": "Return Successful",
"name": name, "nid": nid}).
        else:
            return render(request, 'book_return.html', {"form": form, "error_ msg": "Failed to return, the
book does not exist in the library card list", "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": "Return Successful",
"name": name, "nid": nid}).
        else:
            return render(request, 'book_return2.html', {"form": form, "error_ msg": "Failed to return,
the book does not exist in the library card list", "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": "No library card, please check",
"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')   # Use the reader
ID number to query the borrowed book information

        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 Database Connections

(1) Installation dependencies:

django==3.2.16

mysqlclient==2.1.1

(2) Create a database: name it booksystem

(3) Configure the Mysql interface: The configuration file of the database library is in/library/settings.py and modify the username and password in the settings file

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',   # default
        'NAME': 'booksystem',   # connected database
        'HOST': '127.0.0.1',   # the ip address of mysql
        'PORT': 3306,   # mysql port
```

```
        'USER': 'root',    # mysql username
        'PASSWORD': '123456'    # mysql password
    }
}
```

(4) Start database migration

python manage.py makemigrations

python manage.py migrate

(5) Running input:

python manage.py runserver 0.0.0.0:8000

(6) After the display is successfully created, the browser will be transferred to:

http://127.0.0.1:8000/login/

to enter the login screen

4、summary

The 21st century is the information age, and the library, as the center of information collection, storage, processing and dissemination, must adapt to the changes of the times and adopt information management methods[3]。 The database management system realizes the automation and scientificization of management, and the introduction of its construction into the library will surely change the original appearance of the library, and the managers and borrowers of the portable library.

The project carried out a reasonable and comprehensive analysis of the library management system, which achieved the expected goals in terms of

simplifying the library process and data accuracy, and improved the efficiency and speed of library management[4]。 In operation, the system is easy to operate and stable, which can meet the needs of small and medium-sized library management. Of course, we also saw some shortcomings in the existing system, and reflected on it to lay the foundation for the next step of research and development.

1. The functions implemented by the library information management system

(1) Book query function

(2) Book storage function

Add and modify book information.

(3) Borrow/return the function of books

There are two prerequisites to consider when lending books:

A. Whether the book is in the library;

B. whether the reader has borrowed the full limit;

If none of the above is true, it can be loaned.

Readers can renew the book when they return the book, and the renewal process is mainly to modify the return date in the borrowing record.

(4) Library card management function

Add, modify, and delete the reader's login account and password.

2. Deficiencies in the library information management system

(1) This system and many other systems lack geographic information systems

A. Unable to accurately and intuitively indicate the spatial location of the book

B. Failure to clearly express the exact location of the relevant elements of each book and the relative relationship between them

C. Cannot answer questions such as "where is a book located in a certain book, how far is it, whether two books are adjacent to each other?"

(2) There is a serious phenomenon of information islands

The phenomenon of information silos refers to the fact that libraries are constantly adding new independent systems, but these management systems do not cover all of the library's business, leaving each system isolated and disconnected. Nowadays, many libraries have to vigorously carry out digital business and establish many independent systems, which will undoubtedly increase the overall operating costs of the library, and at the same time, it will also bring a lot of inconvenience to readers, and adversely affect the organic integration of the overall resources of the library.

(3) The evaluation module is not taken into account, so that readers lose their voices in the process of reading and borrowing

All functions of the library should be user-centered, and the library and internal staff should establish a people-oriented mindset. However, at present, the library has not established a feasible service evaluation operation model, and

the library cannot understand the opinions and suggestions of readers in a timely and effective manner, let alone be familiar with the different needs of different readers. As a result, the service mode of the library is backward and outdated, and it is obvious that the value of the library cannot be better played.

3. Experience

In the process of working on the library management system, it became clear to me that I still needed to fill in what I had learned. Just like baking, in the precise feeding, steady implementation, a management system was born from scratch under our hands, just as the proverb - successful experience is the source of self-confidence, we also in this process of the database detailed steps, ideas, methods and technology successfully reproduced, especially the use of basic tables, views, indexes, storage procedures proficiency has also reached a higher level, so we have more or less increased confidence in the operation of the database system.

Of course, in practice, in addition to applying what I learned in class again, I also explored a lot of interesting new knowledge, such as Bootstrap and so on, and cultivated quite objective self-learning ability.

5. References

[1] Zhao Manhua, Gao Jie.Construction and Development of Library Whitening Management System, Beijing: Information Science, 2009, 20

[2] Design and implementation of library library management system[D].University of Electronic Science and Technology of China,2013.

[3] Design and implementation of library management system[D].Shandong University,2009.

[4] Design of library book management system[J].China Science and Technology Information,2007 (11):175-177.