****

**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](#_Toc9211)

[1.1 Project Background 1](#_Toc17087)

[1.2 Positioning and function of the system 1](#_Toc32047)

[1.3 Development schedule 1](#_Toc28007)

[**2. Database design** 2](#_Toc26958)

[2.1 Functional requirements design 2](#_Toc30787)

[(1) Book query function 2](#_Toc18134)

[(2) Book warehousing function 2](#_Toc7801)

[(3) Borrow/return books 2](#_Toc27759)

[(4) Library card management features 2](#_Toc26398)

[2.2 Conceptual Structural Design (ER Diagram) 3](#_Toc24523)

[2.3 Logical structure design 3](#_Toc14413)

[(1) Library table 4](#_Toc12570)

[(2) Book warehousing table 4](#_Toc8683)

[(3) Reader's Table 4](#_Toc22336)

[(4) Administrator Table 4](#_Toc5012)

[2.4 Data Dictionary 4](#_Toc21017)

[**3. System development and implementation** 7](#_Toc15805)

[3.1 Development Tools and Environment 7](#_Toc22926)

[(1) Development tools 7](#_Toc21712)

[(2) Environment configuration 7](#_Toc30231)

[3.2 Create a table and insert initial data 8](#_Toc7049)

[3.3 Introduction to the login interface and functions 18](#_Toc2065)

[(1) Graphical page introduction 19](#_Toc468)

[(ii) Code Interpretation 23](#_Toc5293)

[3.4 Database Connections 34](#_Toc24873)

[**IV. Summary** 35](#_Toc26585)

[1. The functions implemented by the library information management system 35](#_Toc10399)

[(1) Book query function 35](#_Toc21399)

[(2) Book storage function 36](#_Toc31045)

[(3) Borrow/return the function of books 36](#_Toc2348)

[(4) Library card management function 36](#_Toc14488)

[2. Deficiencies in the library information management system 36](#_Toc10017)

[(1) This system and many other systems lack geographic information systems 36](#_Toc22157)

[(2) There is a serious phenomenon of information islands 36](#_Toc17122)

[(3) The evaluation module is not taken into account, so that the reader loses his voice in the process of borrowing” 37](#_Toc2820)

[3. Experience 37](#_Toc21612)

[**5. References** 37](#_Toc30520)

1. **Project planning**
   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. **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. **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.

1. **Database design**

**2.1 Functional requirements design**

1. **Book query function**
2. **Book warehousing function**

Add and modify book information.

1. **Borrow/return books**

There are two prerequisites to consider when lending books:

1. whether the book is in the library;
2. 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.

1. **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:

Icon

The description is automatically generated

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) :d epartment | 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 |

1. **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;

-- ----------------------------

-- 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

-- ----------------------------

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

-- ----------------------------

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\_au2TkHR1tFz-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

-- ----------------------------

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;

-- ----------------------------

-- 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**

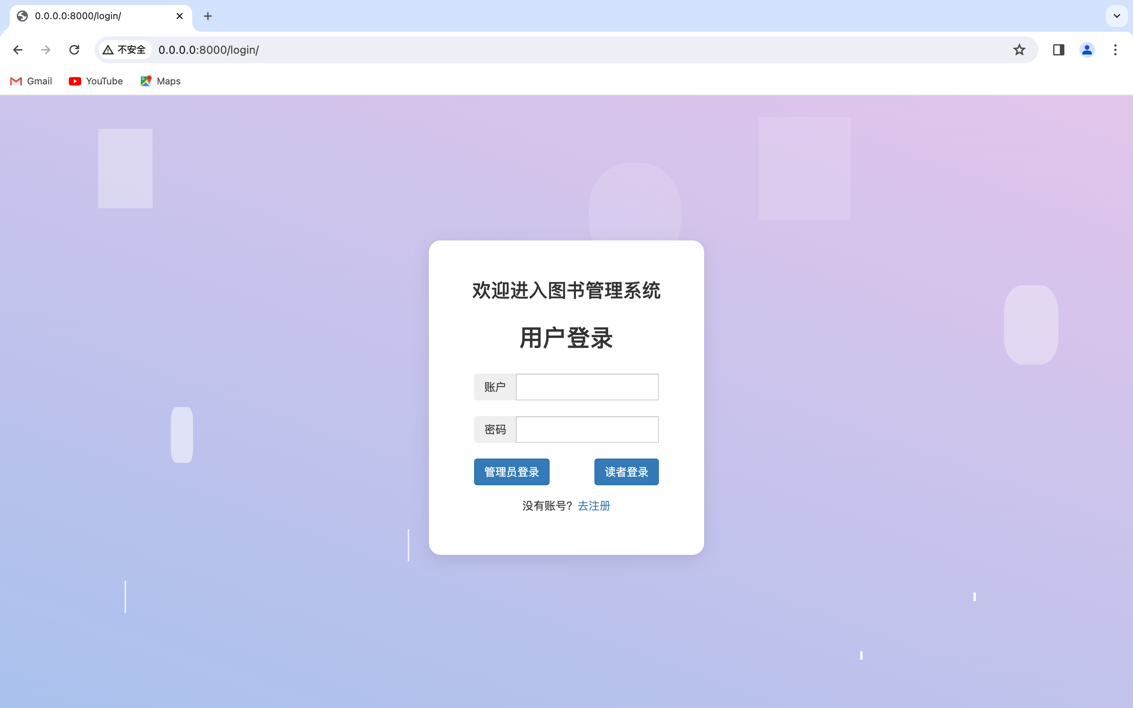
****

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:

Graphical user interface, application, website

The description is automatically generated

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:

**Graphical user interface, application, table

The description is automatically generated**

Figure 4 The screen after the reader logs in

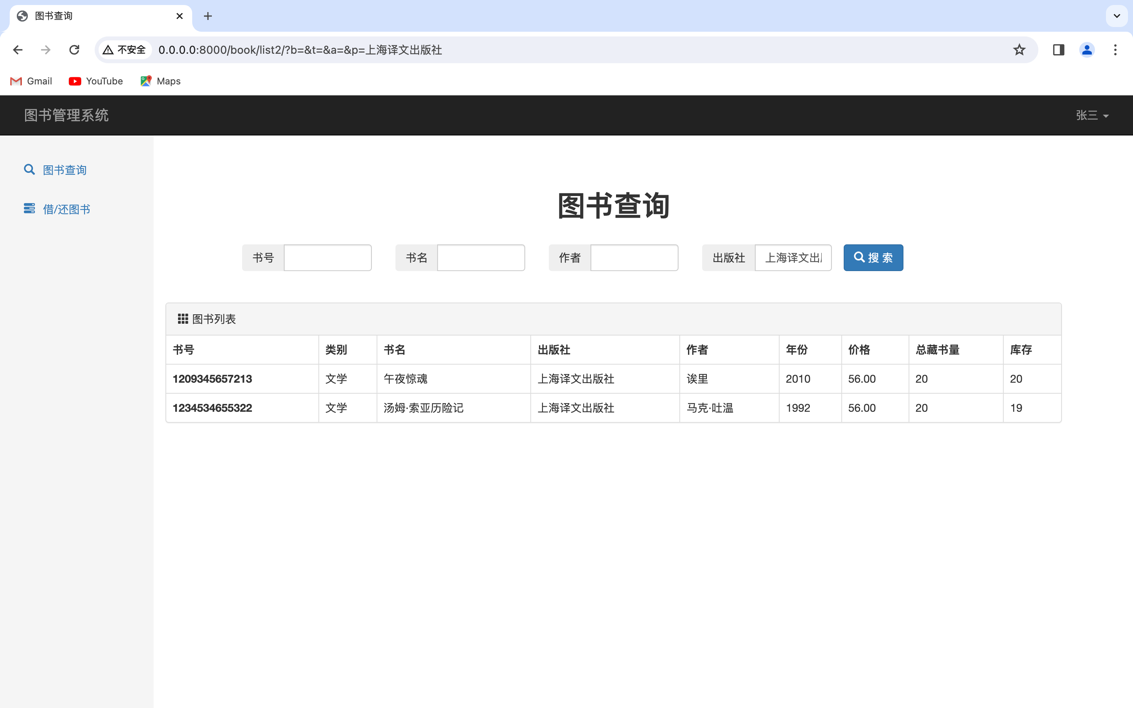


Figure 5 Book query function

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:

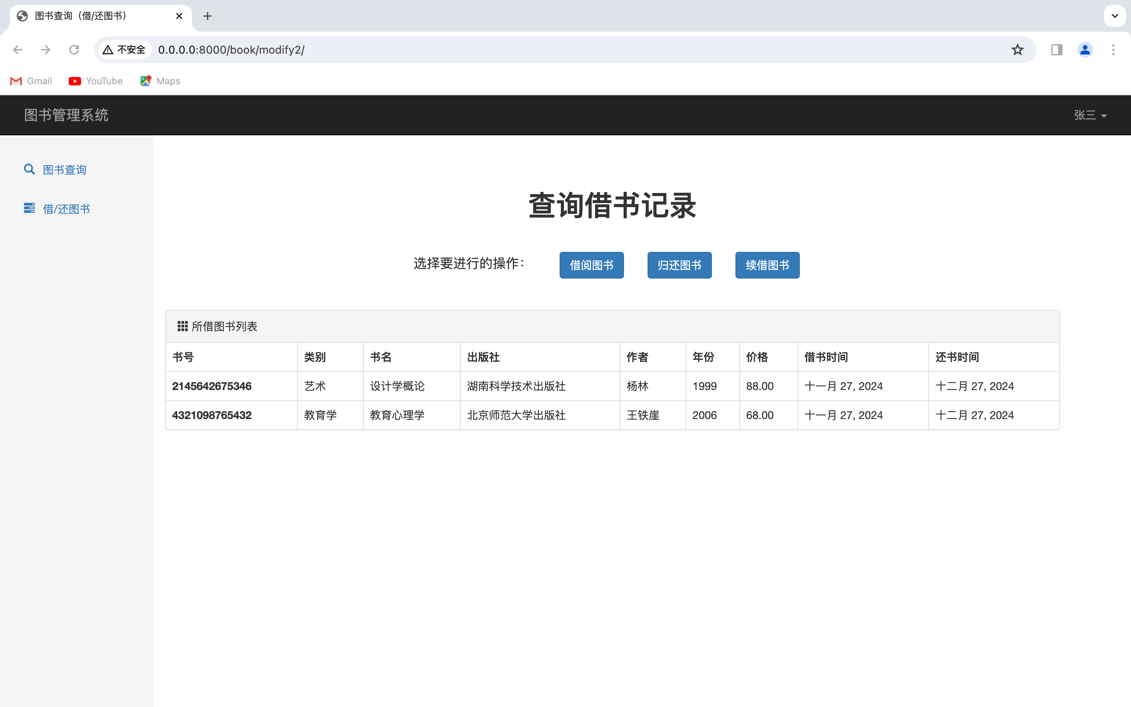
****

Figure 6 Borrowing/returning books

Graphical user interface, application, email

The description is automatically generated

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:

Graphical user interface, tables

The description is automatically generated

Figure 8 After the administrator logs in

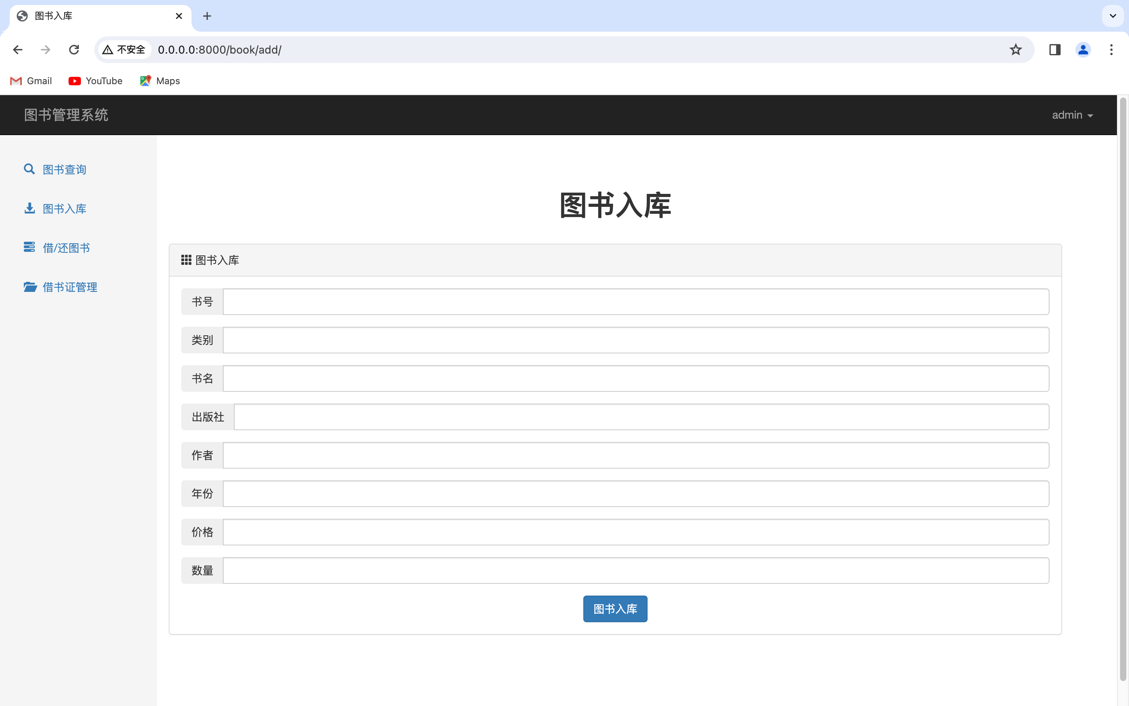
****

Figure 9 The book storage page

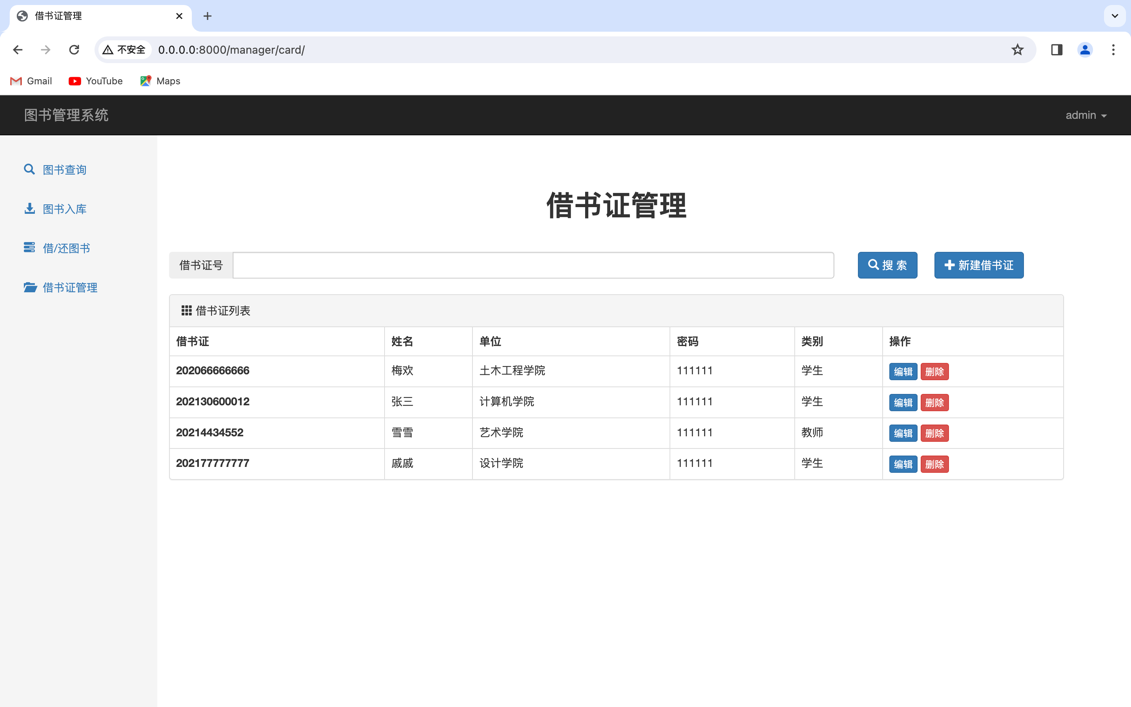
****

Figure 10 Library card management page

1. **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

1. **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.

1. **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.

1. **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.