Author

xx@gmail.com

Library Management System

Abstract

The system differs greatly from the traditional library management system, it covers all functions of the traditional system, and it appeals to students and teachers because of its new concept, living library, also known as human library, also, it is very useful due to its multiple features such as micro-message reminder, used book trade, recommendation, etc.  
You can get a detailed critical review of the project from this document.

Table of Contents

[1 Objectives 2](#_Toc398382861)

[2 Scope and Constraints 2](#_Toc398382862)

[3 Project Details 3](#_Toc398382863)

[3.1 Background 3](#_Toc398382867)

[3.2 Problem Statement 4](#_Toc398382868)

[3.3 Solution 5](#_Toc398382869)

[4 Architecture 6](#_Toc398382870)

[5 User Requirements 11](#_Toc398382871)

[5.1 Use Cases 11](#_Toc398382872)

[6 Detailed Design 19](#_Toc398382873)

[7 Development Process 25](#_Toc398382874)

[7.1 Version 1.0 – Basic Borrow 25](#_Toc398382875)

[8 Testing 39](#_Toc398382876)

[8.1 Functional Test 39](#_Toc398382877)

[8.2 Usability Test 39](#_Toc398382878)

[9 Conclusion 49](#_Toc398382879)

[9.1 Technical Complexity 49](#_Toc398382880)

[9.2 Innovation 49](#_Toc398382881)

[9.3 Future Enhancements 49](#_Toc398382882)

[10 Appendices 49](#_Toc398382883)

[10.1 Reference Guide 49](#_Toc398382884)

[10.2 Usability test questionnaire 49](#_Toc398382885)

# Objectives

The Library Management System aims to provide an online library management for the students and teachers of Jiangsu Second Normal University to search, borrow, return, comment and rate books, also, it aims to build an online community for them to share knowledge, recommend books, as well as trade used books.

The system introduces a new concept called Living Library, also known as Human Library, which started in Denmark that takes human as an education tool or knowledge carrier for people to borrow, you may talk with these borrowed people about their specific areas, skills or experience so that you can get firsthand materials of the theme you want to explore, or fresh impression about the things you’ve learned before, or brand new understanding about a certain kind of people, etc.

The system is designed to make the living library easier to apply, it accepts reservation and allows 2 types of talking with the borrowed people, face-to-face talk and online talk using Skype. It also supports searching, commenting and rating just as the books.

In addition, the system will make it more efficient and convenient for administrators to monitor books, by looking up the rate, comments and borrowed count of the book, they can track users’ needs and take some action to meet the needs. For example, they may collect more demanded books if a book is highly recommended but there are few to lend, or they will consider to buy more books that are the same kind as the hot books.

What’s more, the system will focus on user experience and will promote the usage of the library, for instance, the system will develop a module of notification that combines e-mail, micro-blog, and wechat (like twitter and MSN) for better user experience and higher usage of the library. It will bring a revolution to the library of Jiangsu Second Normal University.

# Scope and Constraints

The Library Management System provides services of borrowing, renewing, returning, commenting and rating books. It allows students and teachers to borrow living library via its online reservation system. It also builds a platform for trading used books.

The system is not redesigning the old library management system, it will select a few of books that are valuable and useful to suggest a high-quality reading for users. The survey shows that…

For better user experience and higher usage of the library, the system will have its own notification module, it will combine e-mail, micro-blog and wechat together to make a good reminder for user to manage books.

Due to budget constraint as well as the space constraint of the system, it will not provide user’s personal space in this version, but just a simple page for sample, and this could be a future enhancement of the system.

# 3 Project Details



## Background

Project Background:

Located in Nanjing, Jiangsu province, Jiangsu Second Normal University is a full-time undergraduate school. At present, it has 16 institutes and nearly 7,000 full-time students. In recent years, the university is developing rapidly with enlarging dimensions. However, the traditional library management system is out of date, it’s running slowly, and it’s offline that only provides service inside the library which is a great limit for students who expect to inquire books in and out of the library at any time, as a matter of fact, the old library management system leads to very low usage of the library. Compared with many online systems, the old system lacks interaction with users, and the functions seem to be too simple to meet the various needs of users.

After talking with our sponsor about our observation and ideas, we finally get a support to develop a new library management system for the university.

The system is a not redeveloped library management system, but a new system to attract more students to participate and learn from each other by sharing reading experience and giving guides to green hands. It also brings convenience to students to store personal data (as a future enhancement) and trade their used books. Moreover, they can book human books and get precious inspiration or tutorials because of online living library reservation. On the other hand, compared with the old fashion way, librarians, now play the role of system administrators, are able to interact with users, and manage books more positively and efficiently according to books’ comments and rates. With increasingly participation of students on the new system, more and more students will find their interests in reading and studying. We have the faith that one day, the new system will totally replace the traditional library management system.

Our ideas brain-storming figure:

Sponsor Background:

Located in Nanjing, Jiangsu Province, Jiangsu Second Normal University is a full-time undergraduate school. Covering an area of about 35 hectare, it currently has three campuses of CaoChangMen, PuKou and XiaoYing. Besides normal education, it also opens some majors for Jiangsu economic and social development. At present, it has 16 institutes and nearly 7,000 full-time students. The school has three libraries, collecting about half million books covering almost all subjects. The libraries grow from two locations within 20 staffs to its current three locations within 32 staffs. Each library provides private learning rooms and public learning areas.

Living Library Background:

Living Library, also known as Human Library, is started in Denmark. The core idea is taking human as an education tool or knowledge carrier to enhance people’s participation. These borrowed people may be male and female, old and young, and many of them have controversial social statuses like transgendered people, radicals, AIDS patients, homosexuals, Mohammedans, strippers and alcoholics. People may borrow these living books and talk with them, thus they can learn each other better and reduce the prejudice and discrimination in the society.

## Problem Statement

Although the library management system is still available in Jiangsu Second Normal University, it is out of date. A lot of books are in low rate of usage because of the isolation of system and lack of updated statistics. Users can only browse books inside the library, that is, the old system is offline. There are a lot latest books’ information also need to be updated into the system. The database of the system doesn’t have backup service and the processing speed is low when a lot inquires executed simultaneously.

From the survey (see appendix) which we aim to find out needs for future library management system and drawbacks for the current one, we know that the majority of students think they should be able to search books outside the library and keep up with the information of latest books. They want to share reading experience and rate the books they’ve read just like what they can do at Amazon book market. They also want to have their own space to store personal electric documents. Here is one figure drawn from the survey,

On the other hand, they complain that the current system is boring, too simple to meet the needs, and the UI design is bad, and there is no way to find out the latest or most popular books, that the library doesn’t provide activities or any kind of communication between readers. They expect to get some tutorial or some kind of guide as well. Here is another figure according to the survey,

Based on all the facts of current system and the result of the survey, we decide to develop an online library management system for the JSNU (Jiangsu Second Normal University) which is more suitable for current situation.

Before we start, we ask ourselves some critical questions as the table shows below:

|  |  |
| --- | --- |
| 1 | How to design a reasonable system architecture to make all subsystems work? |
| 2 | How to apply the concept “living library” to a real online platform? |
| 3 | How to design an attractive and user-friendly UI? |
| 4 | How to protect users’ privacy (e.g. Skype ID is only visible when a reservation is completed)? |
| 5 | How to make use of some external APIs (e.g. Java Email service or even SMS service)? |
| 6 | How to build up a reminder to notify users when books are about to overdue, or a reservation has been made or cancelled? |
| 7 | How to design used books trading platform with some suitable restrictions or rules to prevent spamming? |
| 8 | How to sort the books based on rates, popularity, or the number of comments? |
| 9 | How to use some web security strategies to protect the safety of the system? |

Figure 3. Problem Statement

## Solution

Living library is a total new concept for students and teachers in Jiangsu Second Normal University, my sponsors show great interest in this idea and is willing to provide me any useful resources for designing the library (e.g. Study rooms for communication, some candidate teachers may become living books).

As far as I am concerned, there are only two organizations providing service of living library, one is Douglas College, the other is Coquitlam Library. The form of these living libraries is simple. They offer some activities, inviting some typical and critical people (e.g. ex-offenders, HIV carriers and gays) and sort them into different titles of books for people to borrow. After a talk with the living book, readers can learn from their stories or experience, reduce prejudice and remove stereotype.

However, the current living library has some limits (see figure).

Limits

Figure Limits of Living Library

Firstly, there are very few of book categories, most of them are centralized in very critical and typical area. On the other hand, living books related to academic research areas are almost zero. Thus the mode of the traditional living library is not suitable for JSNU. It can hardly make contribution to the students’ academic study.

Secondly, the reservation of traditional living library only stays on massive paper work, it’s isolated and has limited communication form among library administrators, users and living books. It can hardly attract students to participate.

Thirdly, the traditional living library lacks availability. Users can only borrow living books during library’s activities. If there is no activity, there is no chance to borrow any living books. It is inconvenient for students to study constantly.

Fourthly, it is hard for librarians (system administrators) to track performance of each living book due to lack of users’ feedbacks.

Last but not least, the form of using living library is single, it only allows for talking face to face, there are no supports for people chatting via online chatting tools such as Skype or MSN. When the users or the living books encounter time or place problems, chatting online will be a good way to solve the problem.

Above all, the limits show that the current living library is not enough for JSNU, so we think about a solution to apply the living library to our system. That is,

# 4 User Requirements

## 4.1 Use Cases

First, we concluded a list of user requirements according to our survey, observation at the library, and talking with some librarians as well, here is what we concluded:

The system is required by these stick-holders: users, administrators, and living books

Users’ Requirements:

1. Users can register with valid student ID through the website.
2. Users can edit personal information.
3. Users can search a book according to its name, category, publish time, press, author, book ID, etc.
4. Users can sort books according to borrowing times, the number of comments or rates.
5. Users can borrow a book via the system; they will received a confirmation letter when they successfully complete a booking, then, they can pick up a book with a valid student ID.
6. Users can renew the borrowed books, however, renewing is only allowed once and duration is a month.
7. Users can rate and comment books when they successfully return books.
8. Users can post sale information of used books by simply clicking “I want to sell used books”.
9. Users can post demand information of used books by simply clicking “I want to buy used books”.
10. Users can search used books.
11. Users can upload pictures of used books.
12. Users can edit or delete their posted information.
13. Users can contact a buyer or seller by clicking “contact” button and they can communicate by emails. For protection of privacy, real email addresses of buyers and sellers should be hidden by using anonymous email addresses.
14. Users can search human books according to different subjects (e.g. math, computer, history, and physics).
15. Users can sort human books according to borrowing times, the number of comments or rates.
16. Users can borrow a human book via the system and choose its available schedule; two forms of meeting is optional for users: online meeting (via Skype) or offline meeting.
17. Users submit meeting form and meeting theme before booking living book.
18. Users can contact wanted human books by simply clicking “contact” button. They can communicate by email. For protection of privacy, real email addresses of buyers and sellers should be hidden by using anonymous email addresses.
19. Users can cancel a booking.
20. Users can rate and comment human books when they finished reading.

Administrators’ Requirements:

1. Administrators can add, delete and edit books
2. Administrators can delete over-offensive comments.
3. Administrators can confirm a booking when users pick up with a valid student ID, however, a booking is only hold up 48 hours before its cancelation.
4. Administrators can confirm a returning of books when they receive them.
5. Administrators can delete posts irrelevant to the used books.
6. Administrators can add human books, delete human books or edit information of human books.
7. Administrators can add users to blacklist based on following situations:
   1. Users don’t return books or don’t return books on time.
   2. Users often post junk information irrelevant to used books.
   3. Users often fail to keep the appointment with reserved human books.
   4. Users often cancel the meeting with human books.
   5. Users often post malicious comments.

Living Books’ Requirements:

1. Living books can edit personal information. (Such as the title, subject, special filed, interests, introduction, available schedule for booking, reading forms, contact information and etc.).
2. Living books can cancel the reservation made by users.
3. Living books can comment the users.
4. Living books can send and receive the letter inside station.

With the conclusion list of the requirements, we analyze them using UML and try to make it clearer for the system to be understood, in this phrase, we find that the notification of system should be put on the table to meet the needs. In UML, such a **notification of system** is also treated as a user. Here are what we analyzed in UML:



Figure Users' Use Case

As it shows in the figure, users mainly have 4 use cases, borrow books, borrow living books, trade used books, and manage personal information, these 4 cases have extended cases, or son cases, and some of them share one or more extended cases because of the same logic. Then we analyze each case, mainly the stimulation and respond sequence of the use case.

**Stimulation/Respond Sequence:**

1. User searches a book by the category, title, publishing time, author, ISBN etc.
   1. The system shows the result of searching in a list
2. User sorts books according to the reading count, rating score, and the adding time.
   1. The system gives out the sorting result
3. User borrows a book online
   1. The system checks if the user could borrow
   2. If so, sends a success message to the user
   3. If not, sends a failure notification to the user.
4. User renews a borrowed book
   1. The system checks if the user could renew the book
   2. If so, sends a success message to the user
   3. If not, sends a failure notification.
5. User collects a book
   1. The system checks if the book is already in the collection of the user
   2. If so, sends a notification
   3. If not, add the book to its collection list.
6. User comments a book
   1. The system checks if the user could comment the book (the user is not in the blacklist)
   2. If so, add the comments to the book
   3. If not, gives out a notification to the user.
7. User reserves the living book
   1. The system checks if the reservation is valid (all required information are filled and valid, and the user is allowed to make a reservation, that is, the user is not in the blacklist)
   2. If so, sends a success message to the user and the living book ( the message includes the meeting time, meeting form, the theme to be discussed, Skype ID if meeting form is online, room number if meeting form is offline )
   3. If not, sends a notification to the user.
8. User cancels the reservation
   1. The system checks if it is 6 hours ahead of the reserved time
   2. If so, cancel the reservation, sends a success message to the user and the living book
   3. If not, gives a notification to the user that he may not cancel it.
9. User posts used book for sale
   1. The system checks if the information is valid and completed (title, ISBN, author, quality, price, etc.)
   2. If so, post it to the page of used book trading
   3. If not, gives hints to the user
10. User posts used book for purchase
    1. The system checks if the information is valid and completed (title and author)
    2. If so, post it to the page of used book trading
    3. If not, gives hints to the user
11. User communicates with the other user on the used book trading platform
    1. The system pops a dialog to require the sending content

11.1 User fill in the blank and click send

* 1. The system will send a letter inside station and an e-mail as well to the other user and his default mailbox

1. User registers
   1. The system shows a page to require the basic information of the user ( name, birthday, gender, student number or faculty number, hobby, email address, Skype ID, password and the questions of finding password)

12.1 User fills in the blank

* 1. The system checks if the information is valid and completed
  2. If so, the system sends out a success message to the user
  3. If not, sends a notification to the user

1. User logs in
   1. The system checks if the name and password are correct
   2. If so, jumps to the page with user information
   3. If not, gives hints to user for incorrect name or password
2. User logs out
   1. The system save the data of the user and jumps to the default page without user information
3. User modify personal information
   1. The system checks the information and saves the modified information



Figure Administrators' Use Cases

So administrators mainly do the management work, they manage books, living books, posts, comments and users, often the management work deals with adding, deleting, modifying data, so is the work in our system, and there’s no need to illustrate any more, what we focus on is some special operations admins can do in our system, we analyze these operations with stimulation and respond sequence shown as below.

**Stimulation/Respond Sequence:**

1. Admin confirms a borrowing after checking the users’ status
   1. The system checks the book’s state then shows a message of success or notification
   2. If the book is borrowed, and the user doesn’t come to fetch, then the system will cancel the borrowing, that is, without a confirmation of admin within **2 days**, the borrowing will be canceled automatically by system
2. Admin confirms a returning after checking the book
   1. The system will modify the book’s state and shows a message
3. Admin adds users to blacklist
   1. The system shows a list of blacklist with add and remove buttons

3.1 Admin clicks add button

* 1. The system pops a dialog to require the user’s ID

3.2 Admin enters the user id

* 1. The system searches the user id, shows the information of the user

3.3 Admin clicks confirm button

* 1. The system adds the user to its blacklist, and the user is now shown on the list



Figure Living Books' Use Cases

The living books use the system in a simple and straight way, they only need do the interaction with the user and manage their personal information especially their time tables which are very important for the whole process of borrowing living books. The modify information case has one special feature, that’s the time table, how to design the time table to make it easy for living books to modify and for users to select is critical, but on this phrase, we don’t need to care about that. The comment user case is just like what it does in the Users’ Use Case, so we don’t analyze repeatedly. The cancel reservation case is the same as what a user does in canceling a reservation except the time limit, the stimulation and respond sequence is:

Living book cancels the reservation,

1. The system checks if it is 2 hours ahead of the reserved time
2. If so, cancel the reservation, sends a success message to the user and the living book
3. If not, gives a notification to the living book that he may not cancel it.

As for the send and receive letter inside station case, actually it is a feature of the system, that is, all the users, admins and living books can use the letter inside station, but we put it here to press the duty to interact with users of living books.



Figure Notification Use Cases

This is what we concluded from the stick-holders’ statements and the systems’ features, for better user experience, we put the notification system on table, we hope that we can notify users in time, so the letter inside station and email are not enough, we think about combining the wechat and the system, so that when there are books to be overdue or a reservation cancelled, the users and the living books can get a timely message with the notifications.

# 5 Architecture

The system makes use of the Java lightweight open source frameworks Struts2, Spring and Hibernate, MySql for database management, Tomcat for web server, and MyEclipse for developing. The architecture of the system is shown as the following figure:

Hibernate support

Struts2 support

Service Layer

Data Layer

Presentation Layer

Figure System Architecture

As shown in the figure, the service layer of the system uses Spring, the data layer uses Hibernate that can be integrated into Spring via supporting classes of Spring. The presentation layer uses Struts2 to pass data and control the web pages written in JSP, EL, and JSTL. The whole system uses the 3 layer architecture, which allows only the upper layer calls the lower layer, so as to achieve loose coupling among layers.

Struts2 framework:

Struts2 is a Web application framework based on MVC. The core meaning of MVC pattern is MVC pattern decoupling, dividing the entire application into three parts, model, view, and controller. It tightly controls the communication between the three parts, in order to obtain a clear-structured, function-distribution-reasonable, reusable, extensible, and maintainable applications. By using Struts2 to control the page jumping, you don’t have to write complex code written in Servlet, the robust value stack and OGNL expressions of Struts2 can be used for transmission and control of data.

For enterprise applications, server-side validation is necessary, strict check should be taken on the user provided data before any business logic codes are called, in tradition, data validation needs programmers writing code to achieve, and often the codes are mixed with business logic codes. However, Struts2 provides a framework for data checking, and you can easily tell which codes are for data validation and which are for business logic because it separates the data validation and the business logic. Also, Struts2 has powerful tag library and filters, all of the factors improve the efficiency for the enterprise application development.

The operation process of Struts2 is very simple, when the user request arrives at FilterDispatcher controller, the controller will execute the corresponding Action according to the submitted URL and configuration in the struts.xml. Struts2 Action realizes the decoupling with Servlet API, Action does not require any class inheritance or interface implementation. After finishing processing the user request, Action will jump to the page that is pre-configured in Result according the processed result, and display the data caught in Action to user.

Hibernate framework:

Hibernate is a persistent software based on Java open source, it encapsulates a lightweight package to provide JDBC, ORM (Object Relational Mapping) service. ORM automatically maps the objects of the program to tables of a relational database according the metadata which describes the mapping between objects and tables of database. If we use JDBC to connect to the database and do some operation on database, we need to write a lot of code, while using Hibernate we only need to configure the mapping between Java entity class and the relational database tables, and the method for Hibernate to query and acquire data, thus reduce the amount of code, and improve the efficiency of development.

Hibernate provides one to many, many to one, many to many relationships of objects association, as well as immediate loading and delayed loading support, which facilitates the manipulation of data. The object-oriented HQL query will generate the corresponding SQL statement according to the mapping relationship between objects and database tables, which simplifies the complex SQL statements writing. Hibernate also provides a 1 level cache and 2 level cache, the rational use of 2 level cache can reduce the number of database access effectively which will enhance the overall performance of the system.

Spring framework:

Spring is a lightweight open-source framework of Java SE/EE application, with IoC (Inverse of Control) and AOP (Aspect Oriented Programming) as the core. Inverse of control is also called dependency injection, it makes the object class a passive receiving dependent class rather than class that need to find service on their own. Dependency injection gives the control of dependencies among objects to Spring, so you don’t need to worry about when to instantiate an object, and just focus on the business logic. Spring also provides a powerful support to the Aspect Oriented Programming, by separating out the business logic from application services, it achieves cohesion development.

To integrate Spring and Hibernate, we can put the SessionFactory interface of Hibernate that manage the data access to the IoC container of Spring, so that we only need configure the file rather than manually create an instance of SessionFactory when Hibernate accesses the database. We can also use the transaction mechanism of Spring, so as to switch different data sources without modifying the source code.

To integrate Spring and Struts2, the instantiating the Struts2 Action no longer needs to be managed by Struts2, but managed by the IoC container of Spring instead, thus we can reduce the coupling procedures, and separate the controller and the business logic, which brings great convenience to future maintenance and expansion.

ExtJs:

ExtJs is an independent of back-end, JavaScript written Ajax framework. The power of ExtJs lies in its various components, you can build rich and colorful front pages use the components. ExtJs form controllers are perfect, they support functions as sorting, caching, draging, hiding and editing data, etc. Form controllers support not only the beautiful appearance but also support data check at the front end. Layout controllers can take charge of the entire page layout without writing a lot of code in JavaScript. In addition, ExtJs can reduce as much as possible of the count of jump pages, or even eliminate the need for page jumping.

MyEclipse:

MyEclipse is a very good J2EE tool for integrated development based on Eclipse, it has powerful collection of plugins supports than Eclipse, and better support the open source products. It is an enterprise level development platform, an extension of Eclipse. In MyEclipse, database and J2EE development can be integrated easily, as for the program compiling, running, testing, deploying and releasing, they can all be finished in MyEclipse. Besides, MyEclipse has a good support for the Java open-source framework like Struts2, Spring and Hibernate. Another commonly used integrated development tool is called Intellij IDE, but it lacks plugins, and occupies large memory. Therefore, we choose MyEclipse as the top tool for J2EE development to speed up the development of J2EE applications.

Optimizations for the system:

* **Apache Tomcat web server and load balancer**



The library management system should allow large number of queries processing parallel, therefore single server cannot satisfy the need of loading, what’s more, if the server fails to do normal operation, the whole system ducks. So we must find out the way to ensure that the system doesn’t crush easily. We choose to use the Tomcat cluster technology to solve the problems.

The Tomcat cluster can be achieved by two or more server software instances running on one or more server computers, the servers together make it transparent to clients, the clients see only a high available service. The reverse proxy cluster system use Apache to achieve load balancing, Apache will do the dispatching work, it receives the client’s request and then forwards the requests to different Tomcat servers, thus make the loading balanced. You can clearly see the working principle of the Tomcat cluster.

集群系统由一台或多台服务器计算机上运行的两个或更多服务器软件实例组成，这些服务器计算机彼此协同合作以透明地服务客户端的请求，从而从客户端角度看，整个集群租是一个高可用性服务。网站的集群系统使用Apache的反向代理来实现负载均衡，Apache会将用户的请求分别转发给不同的Tomcat服务器，以此来实现Tomcat的集群。集群部署图如图所示。

* **Cache of Hibernate**

The cache is located between applications and physical data source at the computer memory, its purpose is to reduce the number of applications accessing to physical data sources, so as to improve the performance of applications. When a program needs to query the database, it will first look up the data in cache, if hit, that is, if find the data, then it will not have to access the database anymore.

* **Partition and creating index for database**

A partition is a division of a logical [database](http://en.wikipedia.org/wiki/Database) or its constituent elements into distinct independent parts. Database partitioning is normally done for manageability, [performance](http://en.wikipedia.org/wiki/Optimization_(computer_science)) or [availability](http://en.wikipedia.org/wiki/Availability) reasons.

The partitioning can be done by either building separate smaller databases (each with its own [tables](http://en.wikipedia.org/wiki/Table_(database)), [indices](http://en.wikipedia.org/wiki/Index_(database)), and [transaction](http://en.wikipedia.org/wiki/Database_transaction) [logs](http://en.wikipedia.org/wiki/Database_log)), or by splitting selected elements, for example just one table. Horizontal partitioning (also see [*shard*](http://en.wikipedia.org/wiki/Shard_(database_architecture))) involves putting different rows into different tables. Perhaps customers with [ZIP codes](http://en.wikipedia.org/wiki/ZIP_code) less than 50000 are stored in CustomersEast, while customers with ZIP codes greater than or equal to 50000 are stored in CustomersWest. The two partition tables are then CustomersEast and CustomersWest, while a [view](http://en.wikipedia.org/wiki/View_(database)) with a union might be created over both of them to provide a complete view of all customers.

Vertical partitioning involves creating tables with fewer columns and using additional tables to store the remaining columns. [Normalization](http://en.wikipedia.org/wiki/Database_normalization) also involves this splitting of columns across tables, but vertical partitioning goes beyond that and partitions columns even when already normalized. Different physical storage might be used to realize vertical partitioning as well; storing infrequently used or very wide columns on a different device, for example, is a method of vertical partitioning. Done explicitly or implicitly, this type of partitioning is called "row splitting" (the row is split by its columns). A common form of vertical partitioning is to split dynamic data (slow to find) from static data (fast to find) in a table where the dynamic data is not used as often as the static. Creating a view across the two newly created tables restores the original table with a performance penalty, however performance will increase when accessing the static data e.g. for statistical analysis.

Creating index for database can improve the speed of data retrieval operations on a [database table](http://en.wikipedia.org/wiki/Table_(database)) at the cost of additional writes and storage space to maintain the index data structure. Indexes are used to quickly locate data without having to search every row in a database table every time a database table is accessed. Indexes can be created using one or more [columns of a database table](http://en.wikipedia.org/wiki/Column_(database)), providing the basis for both rapid random [lookups](http://en.wikipedia.org/wiki/Lookup) and efficient access of ordered records.

# 6 Detailed Design

## 6.1 Class Design

With the analysis of the user requirements and architecture as well, we now move on to the phrase of detailed designing, the system has 4 layers, namely are presentation layer, service layer, logic layer, and data layer, and we build these 4 layers with SSH (Struts2, Spring and Hibernate) framework.

### 6.1.1 Data Layer Design

The data layer includes models and DAO. Models are the data structure, or the single objects for the system to pass and process. Let’s take the model of book for example, its private properties like author, press, title, ISBN are the corresponding keys of the book table in the database, and its methods are getters and setters for these data properties like getAuthor, setAuthor, getPress, setPress, getTitle, setTitle, getISBN and setISBN, etc.

We use class diagram to illustrate the model design. Here’s the figure:



Figure class diagram for model

DAO is for data access object, which is responsible for accessing data of the database, in the system, we make use of Hibernate framework to support DAO.



Figure Class Diagram For DAO

### 6.1.2 Logic Layer Design

Logic Layer is for the business logic, it’s the core of the system, and it uses the resources of the system to make the system work like the real world. In the Library Management System, we name the layer Action out of habit. Action does the logical operations on the data by the support of Spring and Struts2, here’s what we designed:



Figure Class Diagram for Logic Layer

### 6.1.3 Service Layer Design

Service layer is responsible for the interaction between logic layer and data layer, supported by Spring. It decouple the layers and make the system clear and maintainable. Here’s what we designed on the service layer:



Figure Class Diagram for Service Layer

### 6.1.4 Presentation Layer Design

Finally, the presentation layer deals with UI and calls the logic layer, we use Struts2 for support. Since we use Dynamic Web Project in MyEclipse to develop the system, we need to add tags to the *web.xml* file under the WEB-INF folder:

|  |
| --- |
| <filter>  <filter-name>struts2</filter-name>  <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>  </filter>  <filter-mapping>  <filter-name>struts2</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping> |

Figure Add tags to support Struts2

## **6.2 Database Design**

Database is the cornerstone of the entire system, database design directly affects the merits of the success or failure of the entire system design, database design is described in this section.   
Database design is to convert solid models and needs of the real world into the process model database, which is the establishment of a database application system core issues. Database and application performance are based upon good database design, database data is the foundation of all operations, if the database design is not good, then all the other methods used to improve database performance is limited effective. The key database design is how to make the design of the database can reasonably store user data that is easy for user data processing.   
Database design must follow certain rules, in a relational database, this rule is the paradigm, and the paradigm is in line with the set of a certain level of relationship patterns. It follows the general design of the database third paradigm. Namely: database table does not contain non-primary key information that is is included in the other tables. Adopted paradigm reduces data redundancy, saving storage space, while speeding up the adding, deleting, and modifying data.

### 6.2.1 Book Table

Book table is mainly used for storing library books in the possession of the relevant information, including relevant information at the time of book storage adding by the administrator, this table is mainly used for readers and librarians to check books in the library system according to user- queried property of books, readers and librarians will be aware of other relevant information, like ISBN, which is to help readers to find books on the shelves to borrow, the book price is used for the compensation basis when the reader inadvertently lost books. Concrete table shown in Table 6.1.

Table 6.1 Book Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Field Length | Whether allowed to be empty | Description |
| ID | text | 20 | No | Primary Key |
| ISBN | text | 20 | No | International Standard Book Number |
| name | text | 50 | No | Book Name |
| series | text | 20 | Yes | / |
| authors | text | 200 | Yes | / |
| press | text | 50 | Yes | / |
| size | text | 50 | Yes | Folio size |
| pages | num | int | Yes | / |
| price | num | float | No | / |
| introduction | text | 255 | Yes | Introduction of content |
| imgURL | text | 100 | Yes | Cover image URL |
| Publish date | text | 30 | Yes | / |
| Category | Text | 30 | Yes | Foreign Key |
| State | Num | Int | No | 0 for AVAILABLE, 1 for BORROWED, 2 for RENEWED, 3 for OVERDUE |

### 6.2.2 Category Table

The table is primarily designed to facilitate the management of library management , and query of books, librarians can set up different borrow period according to different types of books depending on the circumstances of the maintenance and management of books, the specific design of the structure of the table is as shown in Table 6.2.

Table 6.2 Category Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Field Length | Whether allowed to be empty | Description |
| BookId | Text | 20 | No | Primary Key |
| type | Num | Int | Yes | / |
| period | Num | Int | Yes | Time period to borrow |
| Renew period | Num | Int | Yes | Time period to renew |
| dailyfine | Num | Float | Yes | Fined money per day when overdue |

### 6.2.3 User Table

User table is designed to manage the readers, readers need to enter information of the identity when borrowing books, the telephone and other information is for contacting with readers, reader type information determines the number of books one can borrow, and registration time can be used to check the identity of the reader to calculate the effective period. The specific structure of the table design shown in Table 6.3.

Table 6.3 User Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | | Field Length | Whether allowed to be empty | Description |
| id | text | 20 | | No | Primary Key | |
| Password | Text | 50 | | No | / | |
| name | Text | 20 | | Yes | User name | |
| gender | Num | Int | | Yes | 0 for male, 1 for female | |
| addresse | Text | 50 | | Yes | / | |
| Tel | Text | 20 | | Yes | Telephone | |
| startdate | Text | 50 | | Yes | Registration date | |
| enddate | Text | 50 | | Yes | / | |
| type | Num | Int | | Yes | 1 for undergraduates  2 for graduates  3 for teachers and PHDs | |

### 6.2.4 Dictionary Table

This table is designed in order to classify different users for the convenience of management, from the table we can tell readers from administrators. The table design is shown in Table 6.4.

Table 6.4 Dictionary Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Field Length | Whether allowed to be empty | Description |
| userid | Text | 20 | No | Primary key |
| bookp | Num | Int | yes | Whether has the privilege of managing book (add, delete, modify) 0 for no, 1 for yes |
| readerp | Num | INT | yes | Whether has the privilege of managing readers(add, delete, modify) 0 for no, 1 for yes |
| parameterp | Num | Int | yes | Whether has the privilege of setting the parameters(fined money, borrow periods, borrow amount,etc.) 0 for no, 1 for yes |

### **6.2.5 Borrow Book Table**

The design of the table is used to manage the readers to borrow books, barcode property is a unique identifier ID, ​​the ID number of records corresponding readers to borrow, borrow time records the corresponding borrow time, and due time is the return time according to the borrowing rule, the return time is the actual time user return the book, the table design is shown in Figure 6.5.

Table 6.5 Borrow Book Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Field Length | Whether allowed to be empty | Description | |
| id | Text | 20 | No | Id of borrow | |
| readerID | Text | 20 | No | Id of borrower |
| bookcode | Text | 20 | No | Id of book |
| borrowdate | Date | 20 | yes | / |
| duedate | Date | 20 | yes | / |
| returndate | Date | 20 | yes | / |
| renew | Num | Int | yes | 0 for no, 1 for renewed |

# 7 Development Process

## 7.1 Version 1.0 – Groundwork

**Goal**

The Goal for this iteration is creating a foundation for further development. It includes technology choosing, setting up framework and integrating with each other for whole system, and then basic functions of login service to make sure system work stable and well-connected.

**Details**

**Chosen Development technology and tools**

SSH framework



Figure Division of tasks of SSH framework

**Integration of Spring framework and Hibernate framework**

It is easy to integrate Spring framework with Hibernate because of its extensibility and openness. Spring framework provides unified management of data source. Instead of configuring configuration file - **hibernate.cfg.xml** in the Hibernate, it only needs to configure data source and control attributes for the Hibernate in the **applicationContext.xml** file in the Spring. Meanwhile, in order to easy to use, Spring framework provides Hibernate Template which can easily control database without tedious work. In Spring, database connection and transaction management all begin with setting up SessionFactory. SessionFactory only requires one instance in the application, so the instance can be created by Spring and injected into related dependent objects. The Code for configuration file is as follows:

|  |
| --- |
| <!—configure Hibernate database source -->  <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close"  <!-- get database connection information from configuration file -->  p:driverClassName="${jdbc.driverClassName}"  p:url="${jdbc.url}"  p:username="${jdbc.username}"  p:password="${jdbc.password}" />  <!-- instantiate SessionFactory-->  <bean id="sessionFactory" class="org.springframework.orm.hibernate3.LocalSessionFactoryBean"  <!-- reference data source -->  p:dataSource-ref="dataSource"  <!-- specify mapping files of Hibernate -->  p:mappingDirectoryLocations="classpath:/com/books/domain">  <property name="hibernateProperties"><props>  <!-- set the dialect of Hibernate -->  <prop key="hibernate.dialect">  org.hibernate.dialect.MysqlDialect  </prop>  <!-- background output SQL statements operated by Hibernate and format -->  prop key="hibernate.show\_sql">true</prop>  <prop key="hibernate.format\_sql">true</prop>  </props></property></bean>  <!-- configure HibernateTemplate -->  <bean id="hibernateTemplate"  class="org.springframework.orm.hibernate3.HibernateTemplate"  p:sessionFactory-ref="sessionFactory" /> |

Figure Configure Hibernate into Spring

**Integration of Spring framework and Strus2 framework**

Spring not only offers an outstanding open source MVC framework referred as SpringMVC, but also supports well of integration with other web frameworks. During the integration of Spring and Struts2, the main step is making Spring’s IOC container manage Struts2’s Action, then the Action classes of Struts2 are able to require their instances through Spring. Before the integration, in Struts2’s configuration file - **struts.xml,** we need to convert Struts2’s request processor to Spring’s request processor, and upload ApplicationContext when starting Web. In addition, using comment @Controller can realize class injections for these Action classes need to be injected. Configuration code of web.xml is as follows:

|  |
| --- |
| <context-param>  <!—configure applicationContext.xml for files upload paths-->  <param-name>contextConfigLocation</param-name>  <param-value>/WEB-INF/applicationContext\*.xml</param-value>  </context-param>  <!—use ContextLoaderListener(from Spring) to create ApplicationContext-->  <listener>  <listener-class>  org.springframework.web.context.ContextLoaderListener  </listener-class></listener> |

Figure Configure Spring into Struts2

**Tomcat server**

Tomcat server is a lightweight application server, and it is widely used to handle the normal amount of concurrent accesses in small or medium system. It is also the first choose for debugging JSP program. Moreover, Tomcat server has good compatibility and supported by many well-known software companies. It runs stable and has a good development prospect. Deployment diagram of Tomcat server for this system is as follows:



Figure Deployment of Tomcat server

The following steps are the installation of Tomcat server in MyEclipse:

1. In MyEclipse, open display window from window – show view – servers, as shown in figure:



Figure open window from the menu

2. Right click New – server in Servers window, it will let you to select version of Tomcat:



Figure select version of Tomcat server

3. After done select version, click Next to access the place where to add directory for Tomcat server, then choose the proper directory and change JRE to JRE6, as shown in figure:



Figure add directory for Tomcat server

4. After adding directory, click Next to the place where to upload project to Tomcat, choose the proper project and click Fish, as shown in figure:



Figure upload the project to the server

**MySQL database**

MySQL is an open source relational database management system that runs as a server providing multi-user access to a number of databases. The SQL phrase stands for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack. Free-software-open source projects that require a full-featured database management system often use MySQL.

The following steps shows the installation of MySQL plugin in Eclipse:

1. open the Data Source Explorer window from window – show view – data Source Explorer:



Figure Data source explorer window

2. Right click in Data Source Explorer windows, and open New Connection Profile window:



Figure new connection profile window

3. In New Connection Profile window, select desire type of database for adding, and access to New Driver Definition window by double clicking. Then, configure all the values according to chosen database:



Figure New driver definition window

4. Click finish to complete.

**Basic functions of login**

After setting up system frameworks and platforms, login in module will be the first priority to concern on, since realizing basic login module is very necessary for testing system frameworks configuration and integration, and I will explain the concept of SSH clearer by this practical instance. Login module services have two actors – normal user and administrator. Normal users can login and perform all the functions provided by the system to meet their demands, conversely, administrators perform their duties to manage system, they login the backstage of system to manage administration module.

Activity diagram of Login Account is illustrated by the following figure:



Figure Activity diagram of Login process

Description

Actors: Normal User and Administrator

Actors access to login page, and input valid user name and password. System will verify User name and password, if is correct, login successfully, otherwise, login failed.

**Database design**

The following figure shows design of user table in database. All Users fell mainly into two classes: normal users and administrators, presented by user state 1 and user state 2 respectively. I also add certain attributes (Skype\_ID, email, etc.) related to further design of book-borrowing and living library system into the table.

User table of database shows as below:



Figure User table of database

**Code Structure**

The system code structure belongs to three layers: Dao layer, Service layer and Action layer. Dao layer uses Hibernate to operate the underlying database; Action layer’s main job is controlling front web pages, passing parameters and calling Service layer to process business logic.

System code structure is showed as below:



Figure Code structure

Since Dao layer uses Hibernate, it is easy to generate entity classes through the database table structure. It is unnessary to require mapping files from Hibernate becase of supportive comments of Java. Similarly, Struts2 and Spring also use comments, in this way, programmer can simplify work of configurate files and easily manage configurations.

The following code shows user name and password will be introduced to Action of Structs after their submits.

|  |
| --- |
| @Action(value = "login", results = {@Result(name = "success",type = "redirect", location = "/admin/main.jsp"),  @Result(name = "failure", location = "/admin/login.jsp")})  **public** String doLogin() {  List<User> users = userService.login(user);  **int** ret = users.size();  **if**(ret > 0) {  **return** "success";  } **else** {  **return** "failure";  }  } |

Action layer calls service layer to process user login logic, then Action decides actions for login successful or failed. This is how Struts2 plays the role here. Service layer calls Dao layer to operate database. After verification of information passed, administrators will be able to access to management interface of system backstage.

After successful login, administrators management interface is showed as below:



Figure Administrator's home page

## 7.2 Version 1.1 – Basic Borrow

### 7.2.1 Search a book

To search a book by its title, ISBN, the press or the author, we implement this function in the BookService class, which uses BookDao to query book table in the database.

Take searching a book by its title for example, the code in the BookDao shows as below:

|  |
| --- |
| **final** String hql = "from Book b where b.name like :keyword";  List<Book> books = hibernateTemplate.executeFind(  **new** HibernateCallback<Object>() {  **public** Object doInHibernate(Session session) {  Query query = session.createQuery(hql);  query.setParameter("keyword","%"+keyword+"%");  query.setFirstResult(0);  query.setMaxResults(10);  List<Book> list = query.list();  **return** list;  }  }); |

Figure Code Implementation for Searching a Book

The UI of searching a book is shown as below:

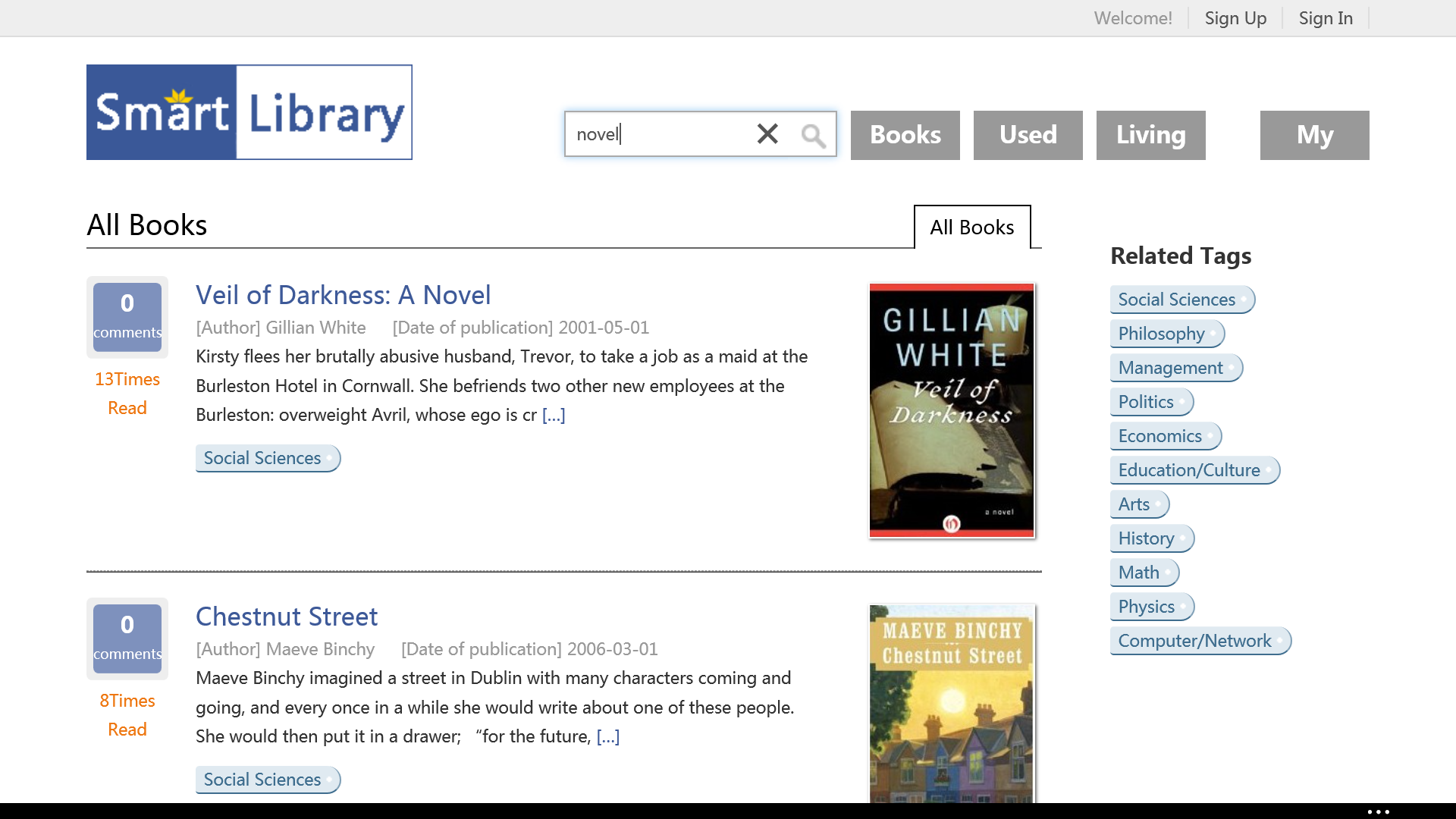


Figure Search a book

### 7.2.2 User information

Users may modify personal information on the personal page, like password, telephone, address, etc. This is implemented in the UserService, which calls UserDao to access data, again, we illustrate this by the code in UserDao:

|  |
| --- |
| **public** **int** updateUser(User user) {  User u = hibernateTemplate.get(User.**class**, user.getId());  List list = hibernateTemplate.find("from TimeTable tt where user.id="+u.getId());  **if**(list.size()>0){  hibernateTemplate.deleteAll(list);  }  List<TimeTable> timeTables = user.getTimeTables();  u.setUsername(user.getUsername());  u.setRealname(user.getRealname());  u.setDept(user.getDept());  u.setSpecialty(user.getSpecialty());  u.setStudentId(user.getStudentId());  u.setGender(user.getGender());  u.setAge(user.getAge());  u.setPhone(user.getPhone());  u.setSkype(user.getSkype());  u.setEmail(user.getEmail());  u.setRemark(user.getRemark());  hibernateTemplate.update(u);  **for** (**int** i = 0; i < timeTables.size() ; i++) {  TimeTable timeTable = timeTables.get(i);  timeTable.setUser(u);  timeTable.setState(1);  hibernateTemplate.save(timeTable);  }  **return** 1;  } |

Figure Code Implementation for User Modifying Information

The UI of this page shows as:

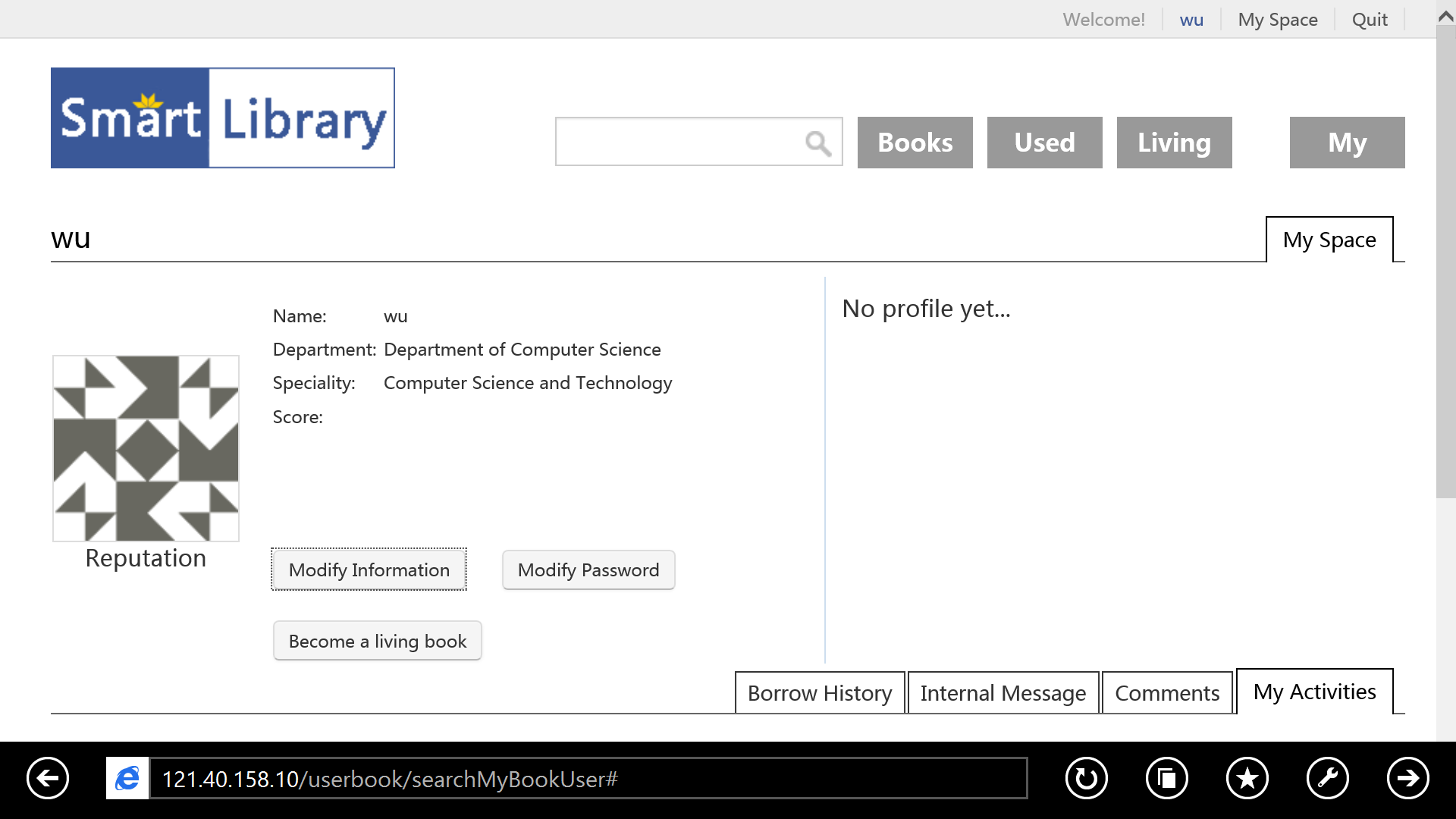


Figure User Information Page

### 7.2.3 Search a living book

Mainly we support searching a living book by its name at this phrase, this is implemented in the class UserBookService and UserBookDao. The code that works goes like this:

|  |
| --- |
| **public** List<User> getUserBookList(User user,String keyword, **final** Integer start) {  StringBuffer sql = **new** StringBuffer()  .append("from User u ")  .append("where u.state > 0 and u.flag=1");  **if**(user!=**null** && user.getSpecialty()!=**null** && !user.getSpecialty().equals("")) {  sql.append("and u.id != " + user.getId());  // sql.append(" and u.specialty.id = " + user.getSpecialty().getId());  };  **if** (keyword!=**null** && !keyword.equals("")) {  sql.append(" and u.specialty.name like '%" + keyword + "%' ");  sql.append(" or u.username like '%" + keyword + "%' ");  }  **final** String hql = sql.toString();  List users = hibernateTemplate.executeFind(**new** HibernateCallback<Object>() {  **public** Object doInHibernate(Session session) {  Query query = session.createQuery(hql);  query.setFirstResult(start);  query.setMaxResults(12);  List list = query.list();  **return** list;  }  });  **return** users;  } |

Figure Code Implementation for Searching a living book

The UI of searching a living book is in some way integrated with searching a book for better UI experience, this is how it looks like:

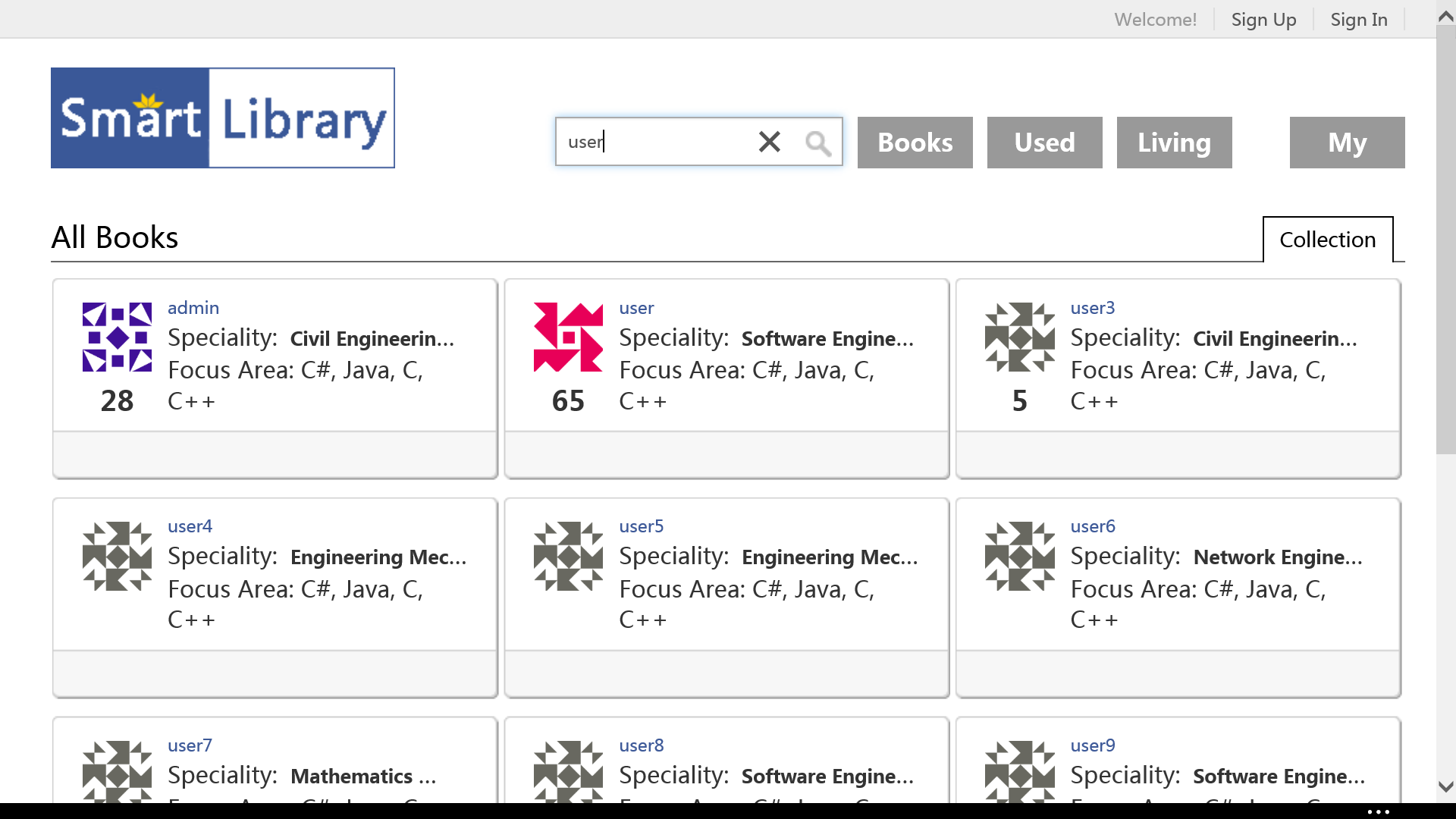


Figure Search a living book

### 7.2.4 Borrow, return and renew

The system feature of borrowing a book provides readers to borrow a book with their identity number and barcode of the book from librarians, and a borrowing needs the librarians to confirm, once borrowed successfully, the book state is set to BORROWED. So is the returning of a book, when confirmed by librarians, the state of the book in the database changes back to AVAILBLE, waiting to be borrowed by other readers.

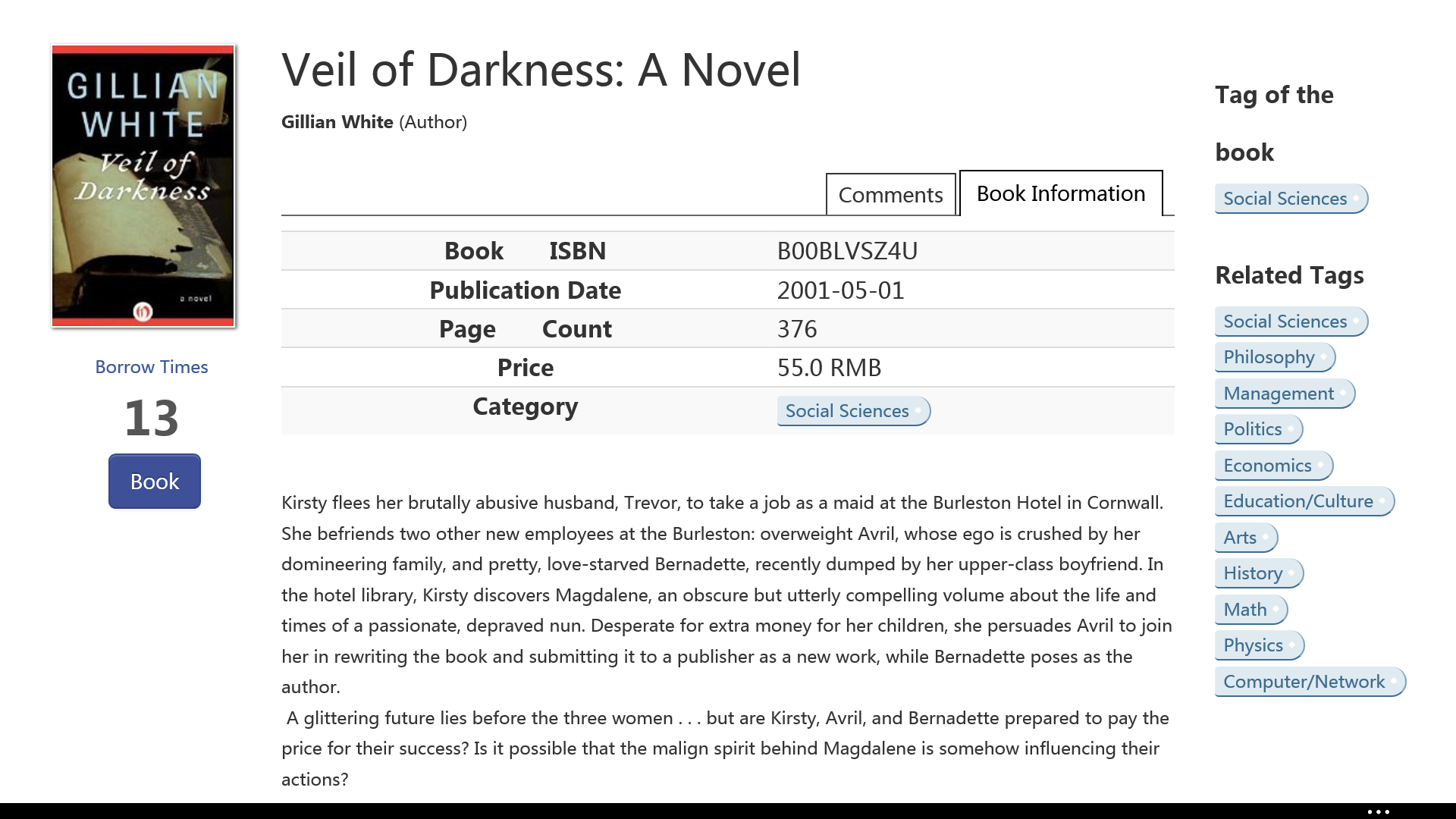


Figure To borrow a book, click the Book button

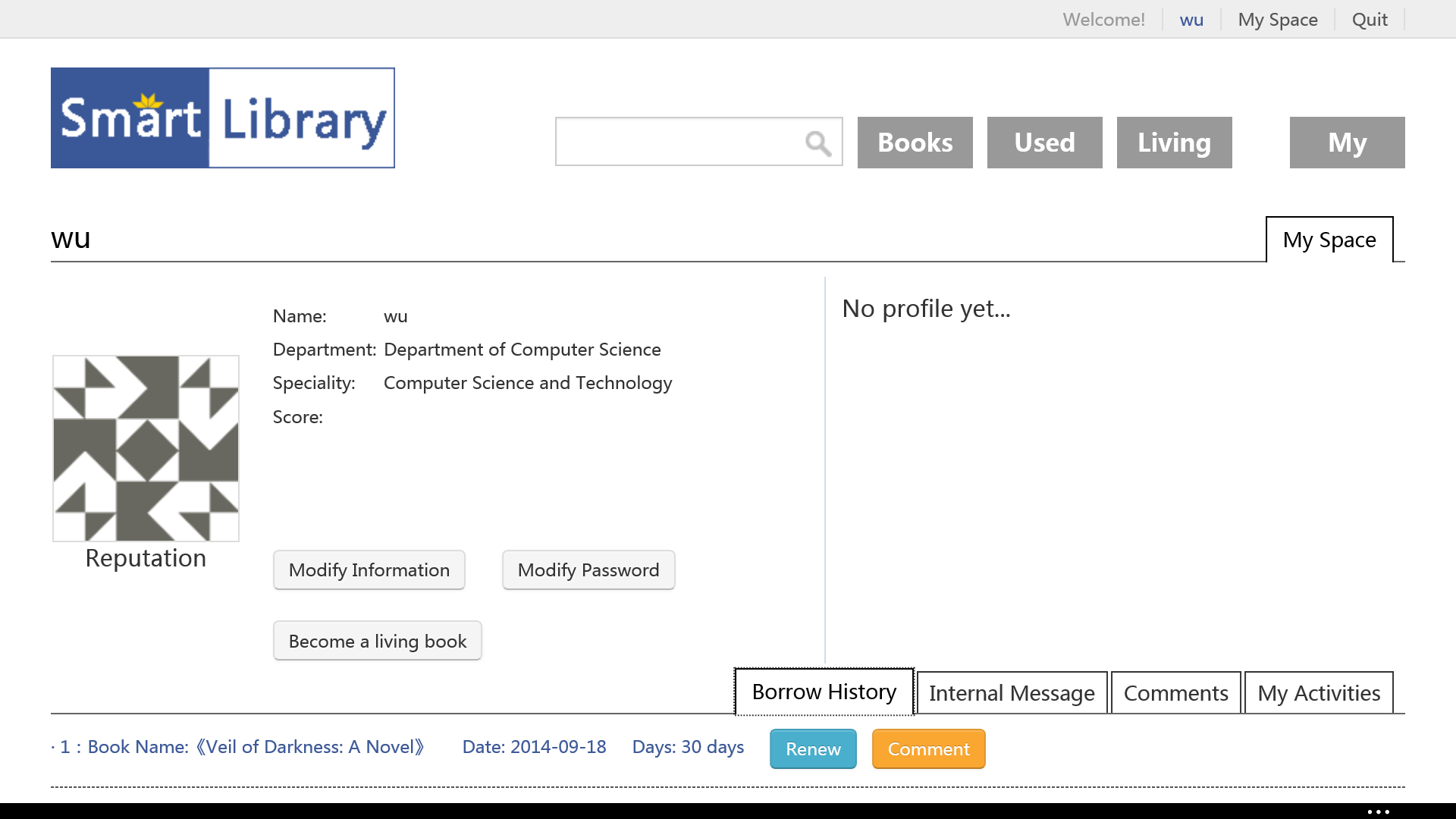


Figure One record appears in Borrow History when complete borrowing a book

The code of borrowing is shown as below:

|  |
| --- |
| if (judge.judgeBookBorrow(book) && judge.judgeReaderBorrow(book)  && judge.judgeBookYesOrNot(book) && judge.endDate(book) == true) {  if (judge.readerType(readerForm) == 1  && judge.bookCount(readerForm) >=judge.bookMaxNumber(readerForm)) {  JOptionPane.showMessageDialog(null, "Undergraduates can borrow at most "+judge.bookMaxNumber(readerForm)+"!", "Warning",JOptionPane.WARNING\_MESSAGE);  }  if (i == JOptionPane.YES\_OPTION) {  LibDataAccessor b = new LibDataAccessor();  b.borrowBookInfo(book,judge.dayMaxNumber(readerForm));//  int n = b.readerBorrowBook(book,judge.dayMaxNumber(readerForm));  if (n > 0) {  int j = JOptionPane.showConfirmDialog(null,"Borrow successfully!",  JOptionPane.WARNING\_MESSAGE);  if (j == JOptionPane.YES\_OPTION) {  bookFieldText.setText("");  } else {  dispose();  } |

Figure Code Implementation for Borrowing a book

The code of returning is shown as below, firstly we need to check if the book is existed, if the book is borrowed, if the book is renewed, and if the book is overdue.

|  |
| --- |
| if (b.judgeBookBorrow(book) && b.judgeBookYesOrNot(book) == false) {  int overduedays = b.overDueDays(book);// get the overdue days  int renew = b.renew(book);// check if renewed  float money1 = b.moneyUpdate(book,moneyNumber);// if overdue, get the fined money  String money = b.numberFormatter(money1);  int i = JOptionPane.showConfirmDialog(null, "Confirm returning?", "Yes",  JOptionPane.YES\_NO\_OPTION);  if (i == JOptionPane.YES\_OPTION) {  if (overduedays < 0) {  b.returnBookInfo(book); b.returnBookLendInfo(book); int j = JOptionPane.showConfirmDialog(null, "Return successfully! Continue?","Return successfully", JOptionPane.WARNING\_MESSAGE);  if (j == JOptionPane.YES\_OPTION) {  bookFieldText.setText("");} else {  dispose();}}  else if (overduedays == 0) {  int a = JOptionPane.showConfirmDialog(null, "The book is going to be overdue. Renew?", JOptionPane.WARNING\_MESSAGE);  if (a == JOptionPane.YES\_OPTION) {  boolean flag = b.renewBook(book); // renew the book  b.returnBookInfo(book);  b.returnBookLendInfo(book);  if (m == JOptionPane.YES\_OPTION) {  bookFieldText.setText("");  } else {dispose();} |

Figure Code Implementation of Returning and Renewing

### 7.2.5 Used book trade

Used book trade platform allows users to post information about their used book for sell or purchase. The page looks like:

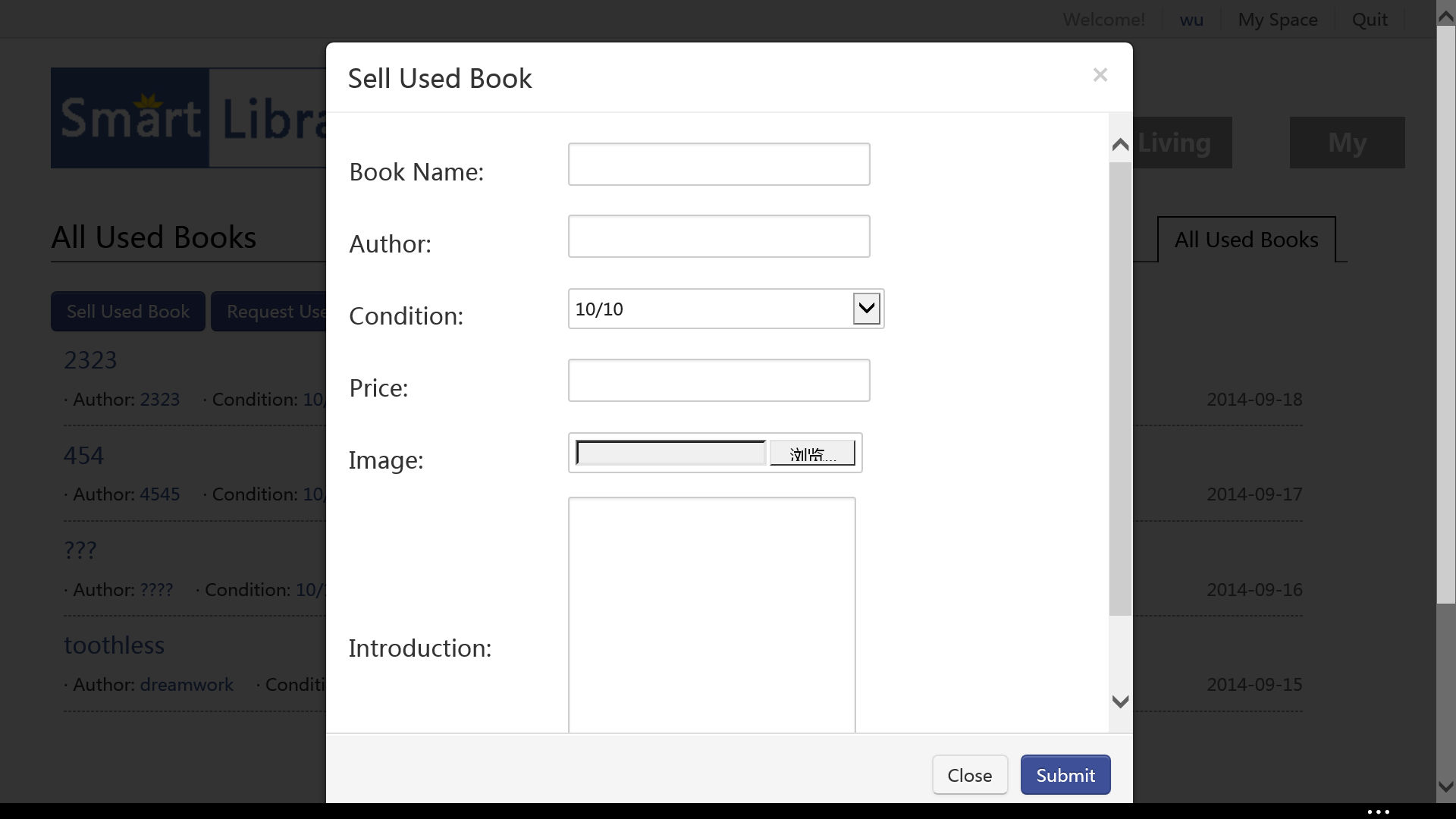


Figure Used book trade

The code is rather straight and simple, take a glimpse at the UsedBookService, it’s what we provide in this module.

|  |
| --- |
| **public** **class** UsedBookService {  @Autowired  **private** UsedBookDao usedBookDao;  **public** **int** saveUsedBook(UsedBook usedBook) {  **return** usedBookDao.saveUsedBook(usedBook);  }  **public** List<UsedBook> queryUsedBooks(String keyword) {  **return** usedBookDao.queryUsedBook(keyword);  }  **public** UsedBook acquireUsedBook(UsedBook usedbook) {  **return** usedBookDao.acquireUsedBook(usedbook);  }  **public** **void** saveComment(UsedBookComment usedBookComment) {  usedBookDao.saveComment(usedBookComment);  }  **public** List<UsedBookComment> queryUsedBookComment(UsedBook ub) {  **return** usedBookDao.queryUsedBookComment(ub);  }  **public** List<UsedBook> queryUsedBookList(**int** start, **int** limit) {  **return** usedBookDao.queryUsedBookList(start,limit);  }  **public** **int** getTotalCount() {  **return** usedBookDao.getTotalCount();  }  **public** **int** deleteUsedBook(Long usedBookId) {  **return** usedBookDao.deleteUsedBook(usedBookId);  }  } |

Figure Code Implementation of Used Book Service

## 7.3 Version 1.2 – Ranking and Recommendation

As for the ranking, we use HQL sentence to sort data, which is based on Hibernate. For example, we need to rank the top borrowed books, well the Book has ID and the borrow count property, we can sort the book by this property in a descending order.

|  |
| --- |
| Query q=session.createQuery("from Book order by borrowedCount desc");  List<Book> li=(List<Book>)q.list();  **for** (**int** i = 0; i < li.size(); i++) {  System.out.println(li.get(i).getName());  }  **return** li; |

Figure Ranking Top Borrowed Books

The ranking results shows on the home page like:



Figure Ranking Result

As we can see from the figure, there’s recommend reading on the left, this recommend list is formed by the users’ rating score, besides this, we have another recommendation shown as below:

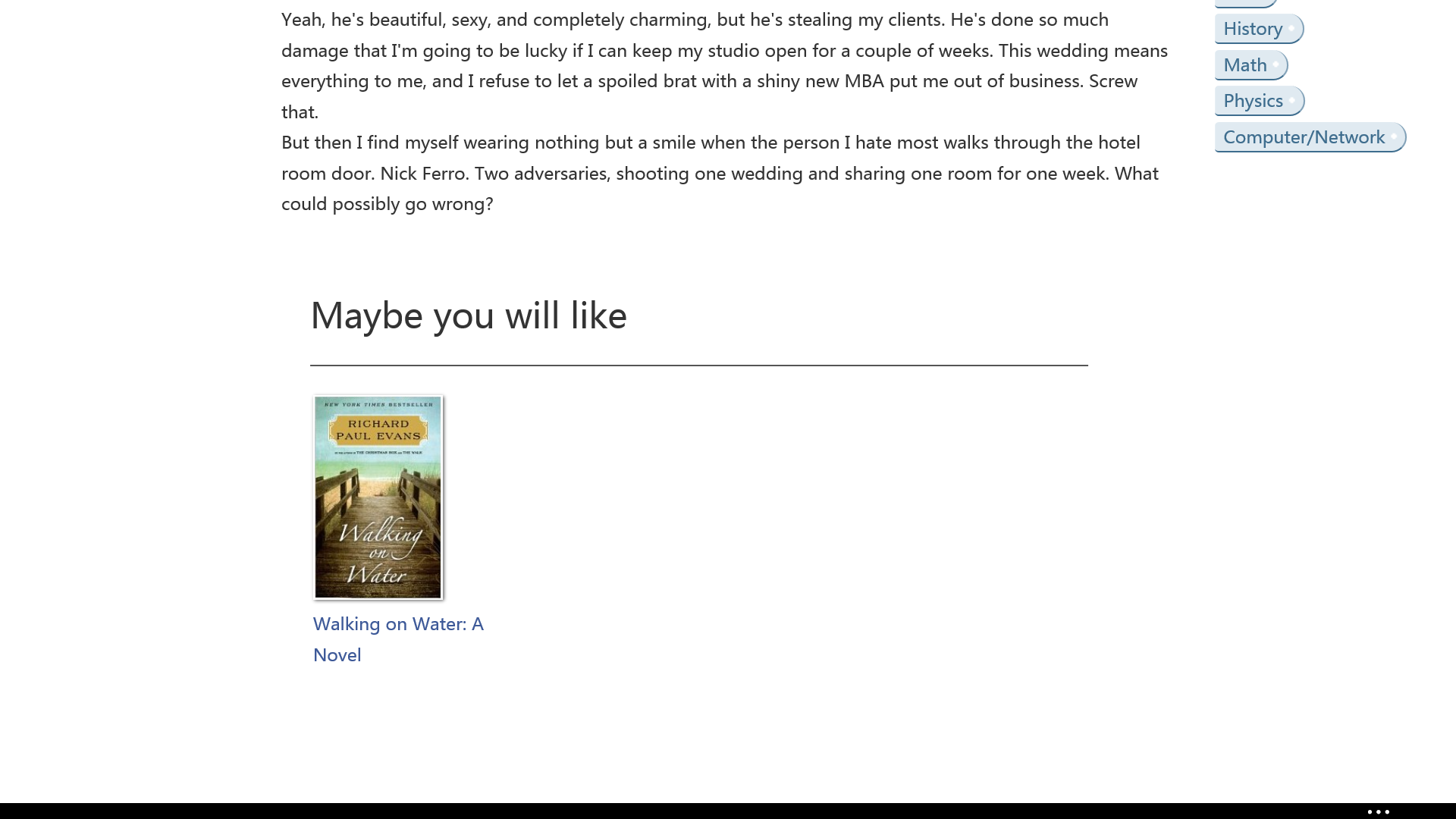


Figure Maybe you will like

Well, this kind of recommendation is a bit complicated, ‘cause we need to find relationships between books, one kind of relationships could be the same tags, another is based on the user actions, most of the users who borrowed this book also borrow that book, so we recommend that book. We implement this algorithm in a simple and straight way, RecommendAction will call the BorrowBookService to get recommending list to show on the page, the BorrowBookService will call the algorithm code in the BorrowBookDao, the code is shown as below:

|  |
| --- |
| @Override  **public** List<Book> getSupportRatingBooks() {  List<String> bookSet = **this**.makeBookSet();  **int** index = 0;    Book[] books = **new** Book[bookSet.size()];  List<Book> resultBookThree = **new** ArrayList<Book>();  Session session = **null**;  **try**{  Configuration config = **new** Configuration().configure();  @SuppressWarnings("deprecation")  SessionFactory sessionFactory = config.buildSessionFactory();  session = sessionFactory.openSession();    **for**(Iterator<String> iter = bookSet.iterator(); iter.hasNext();){  Books[index] = **new** Book();  books[index].setBookIds(iter.next());    **long** support = (Long)session.createQuery("select count(\*) from com.books.domain.BorrowBook as b where b.bids like '%"+books[index].getBookIds()+"%'").list().get(0);  // .setParameter("bookId", "%"+books[index].getBookIds()+"%")    books[index].setSupportRating((**int**)support);  index++;  }    **for**(**int** i = 0; i<books.length; i++){  System.out.println(books[i].getBookIds());  System.out.println(books[i].getSupportRating());  }  //  **int** num\_one = 0;  **for**(**int** i = 0; i < books.length; i++){  **if**(books[i].getSupportRating() > 1){  books[num\_one++] = books[i];  }  }  //  **int** length = (num\_one\*(num\_one-1))/2;  **int** tempIndex = 0;    Book[] bookTwo = **new** Book[length];    **for**(**int** i = 0; i < num\_one; i++){  **for**(**int** k = i+1; k < num\_one; k++){  StringBuffer sb = **new** StringBuffer();  sb.append(books[i].getBookIds()).append(',').append(books[k].getBookIds());    bookTwo[tempIndex] = **new** Book();  bookTwo[tempIndex].setBookIds(sb.toString());  bookTwo[tempIndex].setSupportRating(0);    tempIndex++;  }  }  //  **for**(**int** i = 0; i < bookTwo.length; i++){  String[] bookIds = bookTwo[i].getBookIds().split(",",2);    **long** support = (Long)session.createQuery("select count(\*) from com.books.domain.BorrowBook as b where b.bids like '%"+bookIds[0]+"%' and b.bids like '%"+bookIds[1]+"%'").list().get(0);  // .setParameter("bookId1", "%" + bookIds[0] +"%")  // .setParameter("bookId2", "%"+ bookIds[1] + "%")  // .uniqueResult();  bookTwo[i].setSupportRating((**int**)support);  }    **for**(**int** i = 0; i<bookTwo.length; i++){  System.out.println(bookTwo[i].getBookIds());  System.out.println(bookTwo[i].getSupportRating());  }  //  **int** index\_two = 0;  **for**(**int** i = 0; i < bookTwo.length; i++){  **if**(bookTwo[i].getSupportRating() > 1){  bookTwo[index\_two++] = bookTwo[i];  }  }    List<Book> tempBookThree = **new** ArrayList<Book>();  **for**(**int** i = 0; i < index\_two; i++){  **for**(**int** k = i+1; k < index\_two; k++){  String first = bookTwo[i].getBookIds().substring(0, bookTwo[i].getBookIds().indexOf(','));  String end = bookTwo[k].getBookIds().substring(0,bookTwo[k].getBookIds().indexOf(','));  **if**(first.equals(end)){  Book temp = **new** Book();  StringBuffer sb = **new** StringBuffer();  sb.append(bookTwo[i].getBookIds()).append(',').append(bookTwo[k].getBookIds().substring(bookTwo[k].getBookIds().indexOf(',')+1));    temp.setBookIds(sb.toString());  temp.setSupportRating(0);  tempBookThree.add(temp);  }    }  }    **for**(**int** i = 0; i<tempBookThree.size(); i++){  System.out.println(tempBookThree.get(i).getBookIds());  System.out.println(tempBookThree.get(i).getSupportRating());  }    Book maxSupportRatingBook = **null**;  **int** maxSupportRating = -1;  **for**(Book book: tempBookThree){  String[] bookIds = book.getBookIds().split(",",3);  **long** support = (Long)session.createQuery("select count(\*) from com.books.domain.BorrowBook b where b.bids like '%"+bookIds[0]+"%' and b.bids like '%"+bookIds[1]+"%' and b.bids like '%"+bookIds[2]+"%'").list().get(0);  // .setParameter("bookId1", "%"+bookIds[0]+"%")  // .setParameter("bookId2", "%"+bookIds[1]+"%")  // .setParameter("bookId3", "%"+bookIds[2]+"%")  // .uniqueResult();    book.setSupportRating((**int**)support);  // if(support > maxSupportRating){  // maxSupportRating = (int)support;  // maxSupportRatingBook = book;  // }  }      **for**(**int** i = 0; i < tempBookThree.size(); i++){  **if**(tempBookThree.get(i).getSupportRating() > 1){  resultBookThree.add(tempBookThree.get(i));  System.out.println("Recommend Result: "+tempBookThree.get(i).getBookIds()+"Support Rating: "+tempBookThree.get(i).getSupportRating());  }  }  // System.out.println("Set: "+maxSupportRatingBook.getBookIds()+"," +" SupportRating: " + maxSupportRatingBook.getSupportRating());    }**catch**(Exception e){  e.printStackTrace();    }**finally**{  **if**(session.isOpen()){  session.close();  }  }  **return** resultBookThree;  } |

Figure Code Implementation for Recommendation Algorithm

## 7.4 Version 1.3 – Notification

**1 定时器框架方案的选择**

    定时任务在现在的信息管理系统中已经大量的使用，在Spring框架中就可以选用3种不同的定时机制。

**1.1 基于TimerTask的机制**

    Java中的TimerTask类可以用来执行定时任务，由Spring框架进行封装调用。其中ScheduledTimerTask类定义任务的执行周期，timerTask属性指定执行的工作类；TimerFactoryBean类负责启动定时任务，scheduledTimerTasks属性显示一个需要启动的定时器任务的列表。TimerTask适用于时间间隔相对较短的简单任务。

**1.2 基于Quartz的机制**

    Quartz是一个开源任务调度框架，提供了一个比TimerTask更强大的企业级任务调度执行功能，Spring继承并简化了它。Quartz具有很大的灵活性而又不失简单性，能够执行复杂的任务调度。它允许开发人员灵活地定义触发器的调度规则，并可以对触发器和任务进行关联映射。Quartz为基于Spring应用程序提供了便利的类，如MethodlnvokingJobDetailFactoryBean，SimpleTriggerBean，CronTriggerBean和SchedulerFactoryBean，以便能够在Spring框架中实现依赖注入，降低了Quartz的使用难度。

**1.3 基于Executor的机制**

    Executor框架通过线程池来处理异步任务，也可以执行任务调度服务。其中ScheduledExecutorTask类来指定调度的规则，ScheduledExecutorFactoryBean类完成任务的调度，并且可以通过它来控制线程池中的线程数量。

    基于以上分析，TimerTask可以满足简单的定时企业需求；Quartz提供的任务调度服务最为完善，在企业应用非常广泛；Executor提供的线程池服务最为完善。下面将结合Spring框架，使用Quartz来实现定时器的设计。

**2 定时器框架的设计**

    基于Quartz的定时器框架首先必须导入所需的jar包，然后根据它所提供的一些类来依次实现相应的配置。详细步骤如下所示：

    (1)声明工作类，该类必须实现相应的业务逻辑，同时将该类中的dao对象也一并包含在内；

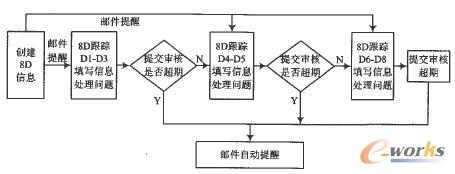
    (2)由MethodInvokingJobDetailFactoryBean类创建调度的任务，并在该任务中需要指明工作类和实现业务逻辑的方法；

    (3)由Spring提供的2个类来实现任务的调度规则，它们具体是：SimpleTriggerBean类配置简单的触发(调度)规则，指明任务执行的时机和频率；CronTriggerBean类配置定制模式的触发(调度)规则，它的功能比SimpleTriggerBean实现的功能要强大，能够控制任务触发(调度)的精确时间，最后还需指明调度触发器与任务之间的映射关系；

   (4)最后由SchedulerFactoryBean来启动触发器。

**3 定时器框架的应用**

    如图1所示，以无锡烨隆ERP系统8D管理模块为例，讲述如何基于Quartz框架开发邮件提醒的定时任务。

[[http://articles.e-works.net.cn/articles/images/zoom.gif](http://articles.e-works.net.cn/articles/articleimage/20126/129846471152968750_new.jpg)](http://articles.e-works.net.cn/articles/articleimage/20126/129846471152968750_new.jpg)  
图1 8D质量管理流程图

    8D管理模块构建使整个团队能够共享信息并努力达成产品质量目标的质量管理体系，能够针对出现的质量问题，及时找出问题产生的根本原因，提出短期、中期和长期对策，并采取相应行动措施。

    8D管理模块中由质量部管理系统中的创建8D，8D查询、8D审核、8D跟踪4个子模块以及其他部门管理系统中的8D跟踪子模块共同构成。

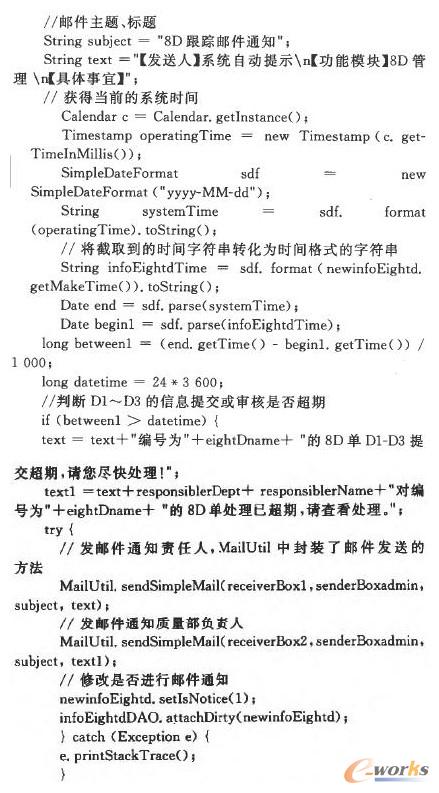
    首先由质量部的员工填写8D基本信息(发生时间、发生地点、发生事件)、选择责任人并允许上传附件(照片或文档描述)，生成一条新的8D信息。创建人(审核人)在完成8D创建(审核)操作后可以给责任人发邮件，提醒责任人及时填写并提交或审核D1-D3／D4-D5／D6-D8信息，相关责任人在规定期限内依据不同的权限在8D跟踪模块中按照步骤解决问题。如果D1-D3信息的提交超期或审核超期时，系统将给出超期提醒，并向责任人自动发送提醒邮件(D1-D3填写限期1天，审核限期为1天)，同样D4-D5／D6-D8信息如果提交超期或审核超期，系统都将向责任人自动发送提醒邮件。以便立即做出反应，解决问题，提高用户的满意度，降低成本和提高生产率。

    定时器框架的开发步骤可以归结为2点：编写任务类和编写配置文件。

**3.1 编写任务类**

    在com．cssrc．quartz．service路径下新建一个JobService类，通过该类中的方法来实现任务，如voidautoSendEmail()，该方法用来执行发送邮件的计划任务。

    方法中首先需要声明发送邮件所使用的dao，如userDAO，roleDAO，infoEightdDAO，infoOneThreedDAO和infoFourFivedDAO等，并生成相应的getter和setter方法。

    下面以D1～D3信息审核为例来说明发送邮件的实现，具体描述如下：  
[[http://articles.e-works.net.cn/articles/images/zoom.gif](http://articles.e-works.net.cn/articles/articleimage/20126/129846471615000000_new.jpg)](http://articles.e-works.net.cn/articles/articleimage/20126/129846471615000000_new.jpg)

    autoSendEmail方法中的这段代码首先判断当前系统的时间与8D信息表创建的时间之问是否超过l天，如果超过，则通过MailUtil类中的sendSim—pleMail方法向相关责任人发送普通邮件，该方法需要提供4个参数的数据，它们分别是接收邮箱、发送邮箱、邮件主题和内容。

**3.2 编写配置文件**

    在WEB-INF下新建applicationquartz．xml配置文件，此配置文件必须在项目的web．xml配置文件中声明，application—quartz．xml文件的具体配置代码如下：  
[[http://articles.e-works.net.cn/articles/images/zoom.gif](http://articles.e-works.net.cn/articles/articleimage/20126/129846472156093750_new.jpg)](http://articles.e-works.net.cn/articles/articleimage/20126/129846472156093750_new.jpg)

    该配置文件详细描述了quartz框架在系统启动24 h后触发sendEmail任务，该任务具体由quartz标签中指定的JobService类的autoSendEmail方法实现8D质量管理中邮件发送任务，而且以后每隔24h再触发，这样就可以实现ERP系统中邮件提醒定时器框架的设计。D4～D5和D6～D8的信息提交或审核是否超期的代码封装在autoSendEmail方法中，具体实现同D1～D3相似。D1～D3信息提交超期发送的邮件提醒见图2。

[[http://articles.e-works.net.cn/articles/images/zoom.gif](http://articles.e-works.net.cn/articles/articleimage/20126/129846472528437500_new.jpg)](http://articles.e-works.net.cn/articles/articleimage/20126/129846472528437500_new.jpg)  
图2 8D质量管理D1～D3信息提交超期

**4 结语**

    Spring是一个开源框架，通过依赖注入，能够有效地组织管理Java应用对象，降低了模块间的耦合的度，因此结合Spring框架，基于Quartz机制实现了邮件提醒定时器框架的设计，并运用到无锡烨隆ERP系统8D管理模块上，减少了人工干预，提高了工作效率。这个框架将应用到该ERP系统的其他模块中，用来实现ERP系统中客户信用额度的变化、产品价格变更通知、库存出入库盘点月底的报表和工作流程的审批等定时任务。

最近忙着用Redis实现一个消息通知系统，今天大概总结了一下技术细节，其中演示代码如果没有特殊说明，使用的都是[PhpRedis](https://github.com/nicolasff/phpredis)扩展来实现的。

## 内存

比如要推送一条全局消息，如果真的给所有用户都推送一遍的话，那么会占用很大的内存，实际上不管粘性有多高的产品，活跃用户同全部用户比起来，都会小很多，所以如果只处理登录用户的话，那么至少在内存消耗上是相当划算的，至于未登录用户，可以推迟到用户下次登录时再处理，如果用户一直不登录，就一了百了了。

## 队列

当大量用户同时登录的时候，如果全部都即时处理，那么很容易就崩溃了，此时可以使用一个队列来保存待处理的登录用户，如此一来顶多是反应慢点，但不会崩溃。

Redis的[LIST](http://www.redis.io/commands/#list)数据类型可以很自然的创建一个队列，代码如下：

<?php

$redis = new Redis;

$redis->connect('/tmp/redis.sock');

$redis->lPush('usr', <USRID>);

while ($usr = $redis->rPop('usr')) {

var\_dump($usr);

}

?>

出于类似的原因，我们还需要一个队列来保存待处理的消息。当然也可以使用LIST来实现，但LIST只能按照插入的先后顺序实现类似FIFO或LIFO形式的队列，然而消息实际上是有优先级的：比如说个人消息优先级高，全局消息优先级低。此时可以使用[ZSET](http://www.redis.io/commands/#sorted_set)来实现，它里面分数的概念很自然的实现了优先级。

不过ZSET没有原生的POP操作，所以我们需要模拟实现，代码如下：

<?php

class RedisClient extends Redis

{

const POSITION\_FIRST = 0;

const POSITION\_LAST = -1;

public function zPop($zset)

{

return $this->zsetPop($zset, self::POSITION\_FIRST);

}

public function zRevPop($zset)

{

return $this->zsetPop($zset, self::POSITION\_LAST);

}

private function zsetPop($zset, $position)

{

$this->watch($zset);

$element = $this->zRange($zset, $position, $position);

if (!isset($element[0])) {

return false;

}

if ($this->multi()->zRem($zset, $element[0])->exec()) {

return $element[0];

}

return $this->zsetPop($zset, $position);

}

}

?>

模拟实现了POP操作后，我们就可以使用ZSET实现队列了，代码如下：

<?php

$redis = new RedisClient;

$redis->connect('/tmp/redis.sock');

$redis->zAdd('msg', <PRIORITY>, <MSGID>);

while ($msg = $redis->zRevPop('msg')) {

var\_dump($msg);

}

?>

## 推拉

以前微博架构中推拉选择的问题已经被大家讨论过很多次了。实际上消息通知系统和微博差不多，也存在推拉选择的问题，同样答案也是类似的，那就是应该推拉结合。具体点说：在登陆用户获取消息的时候，就是一个拉消息的过程；在把消息发送给登陆用户的时候，就是一个推消息的过程。

## 速度

假设要推送一百万条消息的话，那么最直白的实现就是不断的插入，代码如下：

<?php

for ($msgid = 1; $msgid <= 1000000; $msgid++) {

$redis->sAdd('usr:<USRID>:msg', $msgid);

}

?>

Redis的速度是很快的，但是借助[PIPELINE](http://redis.io/topics/pipelining)，会更快，代码如下：

<?php

for ($i = 1; $i <= 100; $i++) {

$redis->multi(Redis::PIPELINE);

for ($j = 1; $j <= 10000; $j++) {

$msgid = ($i - 1) \* 10000 + $j;

$redis->sAdd('usr:<USRID>:msg', $msgid);

}

$redis->exec();

}

?>

说明：所谓PIPELINE，就是省略了无谓的折返跑，把命令打包给服务端统一处理。

前后两段代码在我的测试里，使用PIPELINE的速度大概是不使用PIPELINE的十倍。

## 查询

我们用Redis命令行来演示一下用户是如何查询消息的。

先插入三条消息，其<MSGID>分别是1，2，3：

redis> HMSET msg:1 title title1 content content1

redis> HMSET msg:2 title title2 content content2

redis> HMSET msg:3 title title3 content content3

再把这三条消息发送给某个用户，其<USRID>是123：

redis> SADD usr:123:msg 1

redis> SADD usr:123:msg 2

redis> SADD usr:123:msg 3

此时如果简单查询用户有哪些消息的话，无疑只能查到一些<MSGID>：

redis> SMEMBERS usr:123:msg

1) "1"

2) "2"

3) "3"

如果还需要用程序根据<MSGID>再来一次查询无疑有点低效，好在Redis内置的[SORT](http://www.redis.io/commands/sort)命令可以达到事半功倍的效果，实际上它类似于SQL中的JOIN：

redis> SORT usr:123:msg GET msg:\*->title

1) "title1"

2) "title2"

3) "title3"

redis> SORT usr:123:msg GET msg:\*->content

1) "content1"

2) "content2"

3) "content3"

SORT的缺点是它只能GET出字符串类型的数据，如果你想要多个数据，就要多次GET：

redis> SORT usr:123:msg GET msg:\*->title GET msg:\*->content

1) "title1"

2) "content1"

3) "title2"

4) "content2"

5) "title3"

6) "content3"

很多情况下这显得不够灵活，好在我们可以采用其他一些方法平衡一下利弊，比如说新加一个字段，冗余保存完整消息的序列化，接着只GET这个字段就OK了。

实际暴露查询接口的时候，不会使用PHP等程序来封装，因为那会成倍降低RPS，推荐使用[Webdis](http://webd.is/)，它是一个Redis的Web代理，效率没得说。

…

最近[Tumblr](https://www.tumblr.com/)发表了一篇类似的文章：[Staircar: Redis-powered notifications](http://engineering.tumblr.com/post/7819252942/staircar-redis-powered-notifications)，介绍

**一、通知系统定义**

　　通知系统，顾名思义即通知信息的传达处理系统。目的是为了让用户获得需要得到的消息及提醒并进行处理。

　　这里的“需要得到”有两层意思： 1、用户彼此互动触发的信息流(留言、评论或者回复、私信等) 2、网站希望用户了解关注的信息(系统公告等)



　　通知系统设计的原则可简单的归纳为： 1、消息传播效率最高(获取、处理、信息传达、用户反馈等效率) 2、避免产生骚扰(噪音、频繁提示)

**二、通知分类**

　　不用的平台和产品本身由于对业务的需求不一样，种类也是有区别的。

　　大致可分为以下几种：



**三、通知逻辑实现机制**

　　通知的逻辑精简后如下：

　　￼



　　现对这几个环节分开说明：

**(一)通知合并**

　　通知在推送之前需要进行汇总合并，目的在于提高消息传播处理效率;减少骚扰，降低噪音;平衡服务器压力。

　　1)合并周期：

　　固定时间内的消息全部汇总(24小时内/30天等);

　　无固定时间(只要未处理/未读即汇总)

　　当然一般都组合着用：合并24小时内未处理消息

　　2)分类合并

　　同种类进行合并(如n条留言合并为1条)

　　同一发起人合并(如张三给你发来的n条私信)

　　同一时间周期合并(如24小时共收到n条评论)

**(二)通知分发**

　　通知按照规则汇总完成后，系统将其通过通知管道推送到用户，以便用户处理。

　　1)分发方式

　　分发方式与Feed系统类似，多采用Push方式，即在指定时间内主动推送给用户。部分特定类型需要用户请求(Pull)拉取未读消息。

　　目前大部分通知优先推送未处理通知合并后的总数，已提醒用户已有新消息需要处理。用户点击数字后再去服务端请求具体的消息内容。此种方式综合考虑了成本、压力和体验。当然，某些极端情况下需要进行优化处理：如未读消息超过1000，用户请求时先推送前50条或者放入cache中等。技术童鞋会有各种手段，这里不做详述。

　　2)分发频率(时间)

　　分发时间主要根据消息的优先级来做区隔：



　　3)分发管道

　　分发管道即消息通知的具体推送渠道，根据业务类型可以分为：Web、App、短信、邮件等。

**(三)用户处理**

　　根据前文提到的分发方式，对于通知的处理在逻辑上可以分为两层：通知状态的处理和通知内容的处理。

　　1)状态的处理狭义的理解即为是否已读(已处理)。

　　通常初始数字即为系统推送过来的未读总量，用户点击数字进入相关功能列表查阅后，读取的动作完成，未读数字相应减少。

Web网站通知系统设计,互联网的一些事

　　有几种情况需要变通处理：

　　若用户未读信息较多(m=100)，但第一页列表只能显示(n=10)条的话，那未读数字即为m-n=90;

　　某些产品会将点击等同于已读。即用户只要点击无论是否打开列表查看均认为已读。

　　这样的处理一般用于重要级别较低的消息。点击即已读可有效降低骚扰。

　　某些重要级别较高的消息已处理状态可以定义为用户进行相关操作后才为已处理，而非查阅。

　　如用户进行评论、回复、点击忽略或点击删除等动作时才认为已处理。

　　2)内容的处理狭义的理解即为用户是否操作。

　　根据不同消息的种类和业务的需要，操作可分为：

　　处理：用户必须点击功能链接进行处理。如：你的密码过于简单，点此进行修改;

　　回复：如回复私信，对评论进行回复;

　　确认：对消息做出确认的反馈，如某些系统提示可设置”我已知道，不再提示”的选项;

　　忽略：用户进行忽略操作或不进行任何操作;

　　删除：用户删除本消息。

　　3)消息处理后的状态需要统一。

　　消息需要标记是否已处理的状态，且状态在不同的终端是打通的。

　　如：用户在客户端对消息进行了查看，在web站点本消息应自动标记为已读状态。

**(四)通知回收**

　　回收主要针对用户已处理消息的操作。

　　用户之间触发的消息一般需要留档保存。

　　如评论/回复/留言/私信等。产品可提供选项询问用户是否超过一定周期自动清理。

　　在部分产品中，还需要考虑功能的优先级。

　　如解除好友关系或加入黑名单后自动将删除双方的私信记录。

　　系统触发的消息一般设置一定的回收删除时间。

　　如系统提醒、通知、公告等。过期后自动在产品里删除。物理上可以设置是否备份。

　　过期但用户未处理消息(用户长时间未登录但收到他人的回复)可以根据业务需求来处理。

　　如未读的私信/评论/回复永久保留等。重要未读消息可尝试二次推送或使用其他途径(邮箱、APP、短信等)通知。

**四、通知的交互方式(已读未读)**

　　注：具体的交互需要考虑本身业务特点和目标需求。特定业务可能需要强调，某些业务又需要考虑骚扰，故抛开具体情境本身谈交互是无耻的。

　　这里只针对一般的社区网站，描述一下个人所喜欢的交互方式。

**1、新消息到达时提醒交互**

　　当新消息到达时，可以使用以下提醒方式

　　标题闪动

Web网站通知系统设计,互联网的一些事

　　声音提醒 新消息到达后自动触发声音



　　气泡+数字

Web网站通知系统设计,互联网的一些事

　　新消息浮层



　　标示提示

　　弹窗 ￼

**2、消息处理**

　　目前消息多采用当前触发、即时处理类似“所见即所得”的交互方式。 ￼ 采用此方式的原有主要有： 1、消息通知位于全局导航，访问任何频道时都可保证及时收到新消息; 2、消息在浮层中处理完毕后，用户可继续进行之前的操作，不至于造成打扰; 3、因导航面积有限，需对消息种类进行统一整理和规划;(Facebook的分类为好友请求、私信、通知。) 4、提供历史记录(更多、全部消息)的入口(二级页面) 5、标记已读未读状态，处理好消息提醒数字的关系 ￼ 五、防骚扰(打扰) 因消息本身业务性质，过多无用通知势必会造成噪音，打扰到用户。因此合理设置消息的通知频率和渠道，以防早上体验和效率上的损失。 1、提供通知频率和渠道的管理功能 如常见的邮件退订管理，消息通知类型管理。 ￼

　　Facebook通知设置 编号 通知渠道 通知类型 1 在facebook(web) 你参与的动态 2 电子邮件(email) 挚友的动态 3 推送通知(app) 标签 4 短信通知(message) 群组动态 5 应用请求和动态 备注：通知渠道和通知类型可以结合在一起综合使用。 2、增加屏蔽功能 消息屏蔽功能在业务上应该属于第一条中通知类型管理，当业务模块较多且之前关联分散时，或者开放平台功能接入的第三方应用通知时，可使用屏蔽功能。 ￼ ￼

　　3、结合权限体系 1、功能隐私设置 使用隐私设置界定具体的接收权限、范围等 ￼ 2、结合黑名单功能 使用黑名单可屏蔽指定用户或关键词的具体消息通知。 ￼

**六、其他**

　　1、消息拉回： 当用户长时间不登陆或对消息不处理时，可使用其他渠道推送通知，已达到拉回的目的。 标号 1 触发条件 1)用户长时间不登陆 2 2)长时间不处理消息 3 3)主要通知方式失效(被屏蔽或堵塞) 3 4)存在次要的通知方式 4 通知渠道 1)web 5 2)Email 6 3)短信 7 4)APP 8 备注 1)同步已读未读状态 9 2)拉回进行相关引导 10 3)控制频率和方式，防止造成骚扰

**2、私信与webim**

## 7.5 Version 2.0 – Login with QR code

**Picture the scene:**It’s Monday morning, you’ve arrived to the office, brand new MacBook Air is waiting for you on your desk - the amazing product you’ve been working on flat-out for the last 4 months has just been approved by the client and your boss, being generous enough decided to reward you for all your overtime and working weekends during this period by buying you one of those shiny, sleek machines…

…but, that sounds rather unlikely, let’s go back to reality; It’s Monday morning, your head is quite painful after last night’s party that finished at 5am, and your desktop computer died on Friday afternoon, while you were trying to do as much as possible, before the deadline that has been already moved twice. Bummer.

The broken computer has been replaced by your manager’s private laptop, that by all chance has thousands of suspicious apps, keyloggers and it’s running Vista. Ugh… **Now, how do you log into your mailbox or your company’s web app without risking that all your passwords will be stolen and your accounts compromised?**

**Here comes the QR-logging-in idea:**

The said web app is displaying QR code on the login screen, right next to regular username/password  login form. You’re taking out your phone, snapping a photo with any QR reader app for your iPhone/Android device and in less than 5 seconds you’re logged into the web app **on the computer.** No passwords, no hassle. And you didn’t even have to touch the keyboard! After that you can log into another app. And another one. And another…

Do you like this idea? Yeah, me too.

**So, how to achieve that?**

([you can see a quick proof of concept here](http://sandbox.self.li/qr-login))

The main goal was to avoid typing in anything and let users log in from any machine, even the one they haven’t used before, having only their phone within easy reach.

For the purposes of this post I’m assuming that the phone has built-in camera, installed QR reader app, a browser capable of saving cookies and some internet connection (wifi/3g/gprs/whatever). That fits most iPhone/Android/Windows Phone/Blackberry users out there.

As previously mentioned, the web app is showing QR code. The code contains address to the central server + unique identifier of the token that has been assigned by server to the session on the desktop computer. After scanning the code, the phone is opening a page that checks if it has been used with this service before by looking for a cookie containing encrypted information about the user’s credentials (hash of username/user id). The hash is being checked against server’s database and if it’s valid - token in database is being updated with information that the access is granted to user X. Phone shows information that the user has logged to site XYZ. Browser on desktop is constantly checking status of the token and once it says that the user has “logged in” -  it’s redirecting to secure part of website. Job done!

**What happens if you want to use your phone for the first time?**

If the phone is not paired with any user account - the desktop computer will show regular log in form, so it would be good idea to log in first time from machine that you trust. After filling out the log-in form - user can log in automatically, as described above. Want to make it safer (but less convenient) ? Log in form for first time users could be displayed on the phone instead.

This idea could probably work much better than OpenID/BrowserID. Multiple websites could use one centralised system, that would allow user to pair his/her phone once and use it everywhere, without typing anything anywhere. No more typos, no more forgotten username/passwords.

All that said, I believe that QR-logging-in should be an additional way to log into the system, not the only available, because not every user has capable enough phone, also the phone could be lost, discharged etc.

## 7.6 Version 2.1 – Publish to web

## 工具/原料

* 空间
* 域名
* FTP软件
* wordpress源程序

## 方法/步骤

1. 1

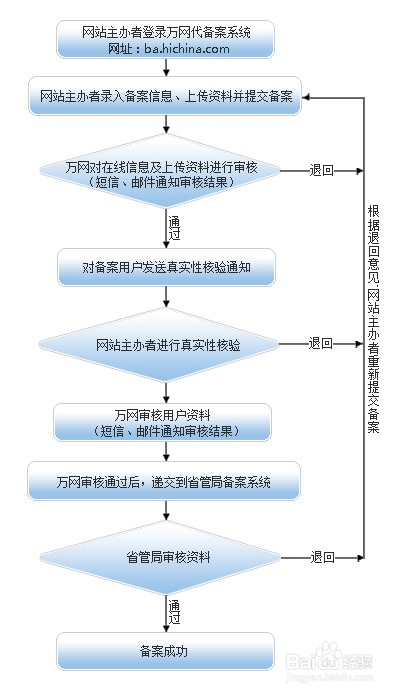
首先购买空间、域名

需要提醒一下的是，根据自己使用的编程语言来选择合适的操作系统。如网站使用ASP、ASP.net编写的，请选用Windows系列虚拟主机。PHP的，选用UNIX系列虚拟主机。如果您只想做几个静态页面发布到网站上，则可以选择全静态HTML的虚拟主机。如果您的网站需要使用数据库，也要注意选择合适的操作系统：使用Microsoft SQL Server数据库，需选择Windows主机；使用MySQL数据库，需选择UNIX主机。有些同学也都明白，但是直到要用的时候，才发觉自己买错了，所以在这里提醒一下。

1. 2

申请ICP备案

根据国家信息产业部要求，国内开通网站必须先办理ICP网站备案，所以您在主机购买成功后，首先要备案。备案时间大概在20天左右。各地的备案过程稍有不同，详见注册商给的备案说明。下图是以万网为例的备案流程。如果您的网站受众主要在国外，可以考虑国外主机，免备案。

[[](http://jingyan.baidu.com/album/d713063514091013fdf47591.html?picindex=1)步骤阅读](http://jingyan.baidu.com/album/d713063514091013fdf47591.html?picindex=1)

1. 3

上传网站

网站在备案的过程中，您的域名一般是不能被解析的，或者解析后是不生效的。一般注册商会给你一个临时的二级域名提供访问。所以我们在备案的同时，可以先调试网站程序。

上传网页常用的工具有CuteFTP、LeapFTP、FlashFXP，在这里我们再以LeapFTP 为例详细说一下上传的方法。

打开LeapFTP软件。

[[](http://jingyan.baidu.com/album/d713063514091013fdf47591.html?picindex=2)步骤阅读](http://jingyan.baidu.com/album/d713063514091013fdf47591.html?picindex=2)

1. 4

按F4 打开“站点管理器”。选择”站点”>”添加”>”站点”。输入站点名称。

[[](http://jingyan.baidu.com/album/d713063514091013fdf47591.html?picindex=3)步骤阅读](http://jingyan.baidu.com/album/d713063514091013fdf47591.html?picindex=3)

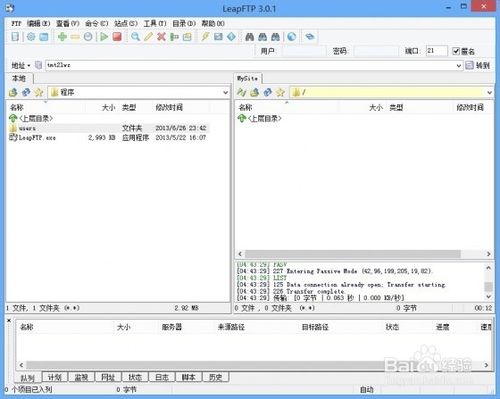
1. 5

站点名称随意，本文为“MySite”， “FTP主机地址” 中填入您的IP地址或您的域名都可，如：123.123.123.123 。去掉“匿名登录”的选择，输入“用户名” “密码”（ftp帐号密码由注册商提供）。端口为默认“21”，登录类型请选择“标准”。然后点击应用按钮。

[[](http://jingyan.baidu.com/album/d713063514091013fdf47591.html?picindex=4)步骤阅读](http://jingyan.baidu.com/album/d713063514091013fdf47591.html?picindex=4)

1. 6

如果设置正确，点“连接”就可以成功连接虚拟主机了。

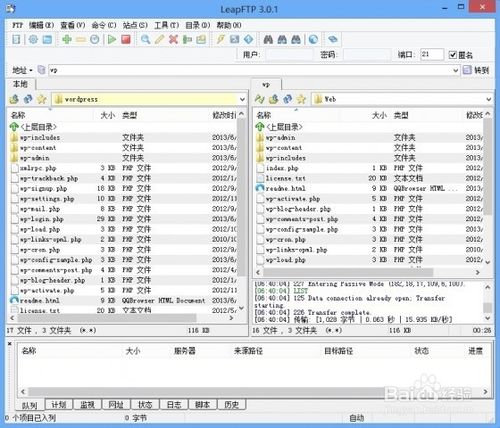
[[](http://jingyan.baidu.com/album/d713063514091013fdf47591.html?picindex=5)步骤阅读](http://jingyan.baidu.com/album/d713063514091013fdf47591.html?picindex=5)

1. 7

连接后大致可以分为左右两大部分和下边部分。左边区域是本地磁盘，可以访问本地目录文件。右侧是远程服务器，可以和管理本地文件一样管理远程文件。右键可以新建目录，双击可以进入目录。下方区域显示的是传送文件时的进度。

解压后，把wordpress源程序解压后上传到网站根目录。

[[](http://jingyan.baidu.com/album/d713063514091013fdf47591.html?picindex=6)步骤阅读](http://jingyan.baidu.com/album/d713063514091013fdf47591.html?picindex=6)

[[](http://jingyan.baidu.com/album/d713063514091013fdf47591.html?picindex=7)步骤阅读](http://jingyan.baidu.com/album/d713063514091013fdf47591.html?picindex=7)

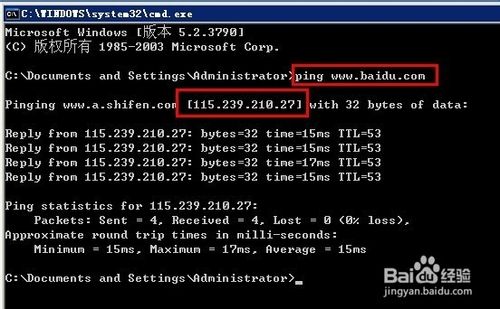
1. 8

域名解析

域名的解析和绑定可以在备案成功后进行，因为在备案过程中，域名应该是不能访问状态。

首先登录域名管理后台。根据域名注册商不同，解析操作上会有些细微的差别。总体来说，域名解析的时候都只是要添加一个子域名为”www”的A纪录，填上你主机的IP，点击添加。域名解析生效的时间一般在2小时以内。判断域名有没有生效的方法如下：开始>运行>然后输入”cmd”，最后输入”ping www.XXX.com“命令，ping与域名中间有一个空格，如果发现上面的IP和你主机的IP一样，就说明已经生效了。

[[如何发布一个网站](http://jingyan.baidu.com/album/d713063514091013fdf47591.html?picindex=8)步骤阅读](http://jingyan.baidu.com/album/d713063514091013fdf47591.html?picindex=8)

[[](http://jingyan.baidu.com/album/d713063514091013fdf47591.html?picindex=9)步骤阅读](http://jingyan.baidu.com/album/d713063514091013fdf47591.html?picindex=9)

1. 9

域名绑定到空间

在注册商提供的虚拟主机控制面板，大都会有域名绑定的设置。只有在这里绑定了你的域名且你的域名解析到了这个主机上，域名才能访问这个空间里的内容。

经过 域名空间购买、申请ICP备案、网站源程序上传、域名解析、域名绑定这几个步骤。外界就可以访问我们的网站了。当然这里，我们输入域名，看到的还是wp网站的设置页面。在后面的文章中，将向大家分享wp的安装与调试。完成之后你就会看到你成型的网站了。

# 8 Testing

## 8.1 Functional Test

## 软件测试的方法与步骤

在开发软件系统的过程中，需要面对错综复杂的问题，因此，在软件生存周期的每个阶段都不可避免地会产生错误。测试目的在于：发现错误而执行一个程序的过程,测试重要发现一个发现其中尚未发现的错误[12]。

为了设计出有效地测试方案按照下面准则进行测试:所有测试都应追溯到用户需求；在完成了需求模型就要着手制定测试计划，在编码之前最所有测试工作进行计划和设计；运用Pareto原理着重对占出现错误80%的容易出错的20%的模块进行测试,从小规模开始逐步进行大规模测试,通常先重点测试单个程序模块再转向集成的模块簇;精心设计测试方案,尽可能充分覆盖程序逻辑使之达到要求的可靠性。

按照软件工程的观点，软件测试（主要是指多模块程序的测试）共包括4个层次。

1)单元测试。单元测试的用例从单元详细设计中导出。在单元测试中可以采用功能性测试和结构性测试两种。

2)集成测试和确认测试。这一阶段的任务，是通过了单元测试的模块逐步组装起来，通过测试与纠错，最终得到一个满足需求的目标软件。

3)验证测试。在这个测试步骤中所发现的往往是需求规格说明的错误。一般来说，系统测试是功能性测试，不是结构性测试。

## 6.2 测试用例设计与测试用例的运行过程及测试结果分析

### **6.2.1 模块测试**

选取了借阅模块进行测试

表6.1 借阅模块用例测试

|  |  |  |
| --- | --- | --- |
| 用例 | 测试数据 | 预期结果 |
| UC1:输入正确的图书证号码和图书条形码 | 图书证号:0001  条形码:00023 | 系统跳转到借阅成功页面 |
| UC2:输入错误的图书证号 | 用户名:1 | 对不起，没有此读者，请注册 |
| UC3:输入错误的图书条形码 | 条形码:1111 | 对不起没有此图书 |

当输入的图书证号和图书条形码是错误的时，系统会跳转到出错页面,用户点击确定,系统会重新返回登陆页面,其中错误信息提示页面如图6.1所示:



图6.1 系统出错提示页面

### **6.2.2 集成测试**

把经过单元测试的模块放在一起形成一个子系统并对它进行测试用于检验模块相互间的协调和通信,这里主要对图书管理员功能系统进行了测试,经过检验其功能均得以顺利实现,下图为管理员操作的读者借阅情况的成功运行界面。



图6.2 图书借阅信息查询功能测试图

把经过测试的子系统装配成一个完整的系统进行测试，经过黑盒测试于白盒测试相结合的方式，对整个系统的各个功能模块进行了测试，并调试改正其中的设计和编码错误，经过这个环节的操作整个系统的功能基本实现成功运行。

### **6.2.3 验收测试**

在进行了以上的测试工作后，将整个软件系统作为单一的实体进行测试，测试内容与系统测试基本类似，但它是在用户积极参与下进行的，此测试过程主要使用实际数据,进行测试,验证测试的目的是验证系统确实能够满足用户的需求,经过这个环节的实际数据测试，系统的各个功能实现都达到了系统需求设计的要求[7]。

## 功能测试

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 模块名 | | 说说 | | | | | | | | |
| 开发人员 | |  | | | 版本号 | | |  | | |
| 用例作者 | | 窦逸飞 | | | 设计日期 | | | 2012-6-7 | | |
| 测试类型 | | 功能测试 | | | | 测试工具 | | 无 | | |
| **用例ID** | **用例名称** | | **测试目的** | **输入描述** | | | **预期结果** | | **实际结果** | **测试数据** |
| SS0001 | 发表说说 | | 测试说说能否正确发布，内容是否能够被系统审核 | 1. 选择说说类别  2.点击写说说，输入说说内容  3.  4. 提交  6.查看前台的相关页面内容 | | | 1.可以正常的输入要素  2.提示“日志已经生成，是否发布或者保存为草稿”  3.显示管理页面  4.进入具体类别页面  5.系统审核状态为已审  6.前台能看到后台添加的相关内容 | | 日志成功发布 |  |
| SS0002 | 说说管理 | | 说说以及好友回复的删除与恢复 | 1.添加说说  2.插入图片和视频  3.删除说说  4.恢复说说 | | | 1.可以正常发布说说  2.提示“图片插入成功”  3.提示“删除成功”  4.提示“恢复成功” | | 达到与其目的 |  |
| SS0003 | 退出系统 | | 测试能否正确退出系统 | 点“返回”按钮 | | | 退出后台进入前台界面 | |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 模块名 | | 应用中心 | | | | | | | | |
| 开发人员 | |  | | | 版本号 | | |  | | |
| 用例作者 | | 窦逸飞 | | | 设计日期 | | | 2012-6-7 | | |
| 测试类型 | | 手工测试 | | | | 测试工具 | | 无 | | |
| **用例ID** | **用例名称** | | **测试目的** | **操作步骤** | | | **预期结果** | | **实际结果** | **测试数据** |
| YYZX0001 | 应用添加 | | 测试应用中心能否正确打开，应用能否正确添加 | 1.进入应用中心  2.点击所要添加的应用  3.点“进入应用”按钮下载应用  4.开始应用 | | | 1.可以正常的进入应用中心  2.显示推荐“热门应用”  3.应用成功加载 | | 应用添加成功 |  |
| YYZX0002 | 关闭应用 | | 测试应用能否正常关闭 | 1.选择要删除的内容  2.点“删除” | | | 1.进入帖子内容页面  2.提示“您确定要执行该操作吗？” | |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 模块名 | | 登录界面 | | | | | | | | |
| 开发人员 | |  | | | 版本号 | | |  | | |
| 用例作者 | | 窦逸飞 | | | 设计日期 | | | 2012-6-7 | | |
| 测试类型 | | 手工测试 | | | | 测试工具 | | 无 | | |
| **用例ID** | **用例名称** | | **测试目的** | **操作步骤** | | | **预期结果** | | **实际结果** | **测试数据** |
| LOGIN0001 | 用户登录 | | 测试输入不同信息时返回的登录结果 | 1.打开浏览器进入QZONE.QQ.COM  2.输入用户名和密码  3.点“登录”按钮进入 | | | 1.密码或用户名错误  2.登陆成功  3.进入系统 | | 实现了测试预期目标 |  |
| LOGIN0002 | 注册用户 | | 测试用户注册功能 | 1.以不同字段形式分别填写注册信息  2.点“注册按钮”  3.注册成功 | | | 1.填写注册信息不符合要求，重新填写  2.注册成功 | |  |  |

## 身份验证测试

考虑3种情况：

* + 用户名或密码有空
  + 用户名或密码有误
  + 用户名和密码都正确

### 3.1.1控制

输入：半自动

操作顺序：先输入用户名和密码，再登录

结果记录方法：记录输入信息、提示信息、进入界面

### 3.1.2输入、输出、过程

|  |  |  |  |
| --- | --- | --- | --- |
| 输入数据 | 选择策略 | 命令 | 输出数据 |
| 用户名‘ ’密码‘111’ | 测试当用户名为空时的情况 | 登录按钮 | “用户名或密码不能为空”的提示框 |
| 用户名‘070505201’密码‘ ’ | 测试当密码为空时的情况 | 登录按钮 | “用户名或密码不能为空”的提示框 |
| 用户名‘070505201’密码‘112’ | 测试当密码错误时的情况 | 登录按钮 | “用户名或密码”的提示框 |
| 用户名‘070505211’密码‘111’ | 测试当用户名错误或不存在时的情况 | 登录按钮 | “用户名或密码”的提示框 |
| 用户名‘070505201’密码‘111’ | 测试用户名和密码都正确的情况 | 登录按钮 | 进入用户或管理界面 |

## 3.2借书测试

考虑7种情况：

* + 借阅证号或书号为空
  + 借阅证号不存在
  + 书号不存在
  + 该读者有欠费记录
  + 该读者已借书中有逾期的书籍
  + 该读者已借书的数量已达到8本
  + 借阅证号和书号都正确，且该读者有借书的资格

### 3.2.1控制

输入：半自动

操作顺序：先按上面7种情况的顺序输入相应的借阅证号和书号，再提交。

结果记录方法：记录输入信息、提示信息、数据库中每个表的更改信息。

### 3.2.2输入、输出、过程

|  |  |  |  |
| --- | --- | --- | --- |
| 输入数据 | 选择策略 | 命令 | 输出数据 |
| 借阅证号为空，正确书号 | 测试借阅证号为空的情况 | 借书按钮 | “借阅证号不能为空”的提示框 |
| 正确借阅证号，书号为空 | 测试书号为空的情况 | 借书按钮 | “书号不能为空”的提示框 |
| 错误的借阅证号，正确书号 | 测试当不存在改借阅证时的情况 | 借书按钮 | “该读者不存在”的提示框 |
| 正确的借阅证号，错误的书号 | 测试当书号错误的情况 | 借书按钮 | “该书不存在”的提示框 |
| 修改数据库，使得Punish\_Info表中有某读者的欠费记录，输入这个读者的借阅证号，输入正确的书号 | 测试读者有欠费记录时的情况 | 借书按钮 | “该读者已欠费…”的提示框 |
| 修改数据库，使某读者的已借书中有逾期的，输入该读者借阅证号和该逾期书籍号 | 测试读者已借书中有逾期的书籍的情况 | 借书按钮 | “该读者已欠费…”的提示框 |
| 使某读者已借阅8本书，该读者继续借书 | 测试当某读者已借阅书籍已达到8本书的情况 | 借书按钮 | “该读者已借书8本”的提示框 |
| 以上情况都没有且借阅证号和书号都正确 | 测试当读者符合借书条件的情况 | 借书按钮 | “借书成功”的提示框 |

## 3.3还书测试

考虑5种情况：

* + 借阅证号或书号为空
  + 借阅证号不存在
  + 书号不存在
  + 该书逾期
  + 借阅证号和书号都正确

### 3.3.1控制

输入：半自动

操作顺序：先按上面5种情况的顺序输入相应的借阅证号和书号，再提交。

结果记录方法：记录输入信息、提示信息、数据库中每个表的更改信息。

### 3.3.2输入、输出、过程

|  |  |  |  |
| --- | --- | --- | --- |
| 输入数据 | 选择策略 | 命令 | 输出数据 |
| 借阅证号为空，正确书号 | 测试借阅证号为空的情况 | 还书按钮 | “借阅证号不能为空”的提示框 |
| 正确借阅证号，书号为空 | 测试书号为空的情况 | 还书按钮 | “书号不能为空”的提示框 |
| 错误的借阅证号，正确书号 | 测试当不存在改借阅证时的情况 | 还书按钮 | “该读者不存在”的提示框 |
| 正确的借阅证号，错误的书号 | 测试当书号错误的情况 | 还书按钮 | “该书不存在”的提示框 |
| 修改数据库，使某读者已借的该书已逾期，输入该读者借阅证号和该书籍号 | 测试读者该书已逾期的情况 | 还书按钮 | “该书已欠费…”和“还书成功”的提示框 |
| 以上情况都没有且借阅证号和书号都正确 | 测试当读者符合还书条件的情况 | 还书按钮 | “还书成功”的提示框 |

## 3.4图书库管理测试

考虑3种情况：

* + 书号、书名、作者、出版社、单价、总量有空项
  + 该书图书库中已存在
  + 该书图书库中不存在

### 3.4.1控制

输入：半自动

操作顺序：先按上面3种情况的顺序输入相应的借阅证号和书号，再提交。

结果记录方法：记录输入信息、提示信息、数据库中每个表的更改信息。

### 3.4.2输入、输出、过程

|  |  |  |  |
| --- | --- | --- | --- |
| 输入数据 | 选择策略 | 命令 | 输出数据 |
| 书号、书名、作者、出版社、单价、总量有空项 | 测试书籍信息填写不完整的情况 | 入库按钮 | “请将信息填写完整”的提示框 |
| 书号在图书库中已存在 | 测试书号为空的情况 | 入库按钮 | 数据库中该书的总量和现存量各增加 |
| 书号在图书库中不存在 | 测试当不存在改借阅证时的情况 | 入库按钮 | Book\_Info表中增加一条记录 |

## 3.5图书查询、借阅证信息查询、借阅信息查询测试

考虑9种情况：

* + 图书查询：图书号或图书名不存在
  + 图书查询：图书号或图书名有空
  + 图书查询：图书存在
  + 借阅证信息查询：借阅证号不存在
  + 借阅证信息查询：借阅证号为空
  + 借阅证信息查询：借阅证号存在
  + 借阅信息查询：借阅证号不存在
  + 借阅信息查询：借阅证号为空
  + 借阅信息查询：借阅证号存在

### 3.5.1控制

输入：半自动

操作顺序：先按上面9种情况的顺序输入相应的借阅证号和书号，再提交。

结果记录方法：记录输入信息、提示信息、数据库中每个表的更改信息。

### 3.5.2输入、输出、过程

|  |  |  |  |
| --- | --- | --- | --- |
| 输入数据 | 选择策略 | 命令 | 输出数据 |
| 错误的图书号或书名 | 测试书号或书名有错误的情况 | 图书信息查询按钮 | “该图书不存在”的提示框 |
| 书号或书名有空 | 测试书号或书名为空的情况 | “请将信息填写完整”的提示框 |
| 正确的书号或书名 | 测试书号或书名存在的情况 | 书籍基本信息 |
| 错误的借阅证号 | 测试借阅证号错误的情况 | 借阅证信息查询 | “该读者不存在”的提示框 |
| 借阅证号未填 | 测试借阅证号为空的情况 | “借阅证号不能为空”的提示框 |
| 正确的借阅证号 | 测试借阅证号正确的情况 | 读者的基本信息 |
| 错误的借阅证号 | 测试借阅证号错误的情况 | 借阅信息查询 | “该读者不存在”的提示框 |
| 借阅证号未填 | 测试借阅证号为空的情况 | “借阅证号不能为空”的提示框 |
| 正确的借阅证号 | 测试借阅证号正确的情况 | 读者的借阅信息 |

## 3.6缴费测试

考虑2种情况：

* + 该读者没有欠费记录
  + 该读者有欠费记录

### 3.6.1控制

输入：半自动

操作顺序：先按上面2种情况的顺序输入相应的借阅证号和书号，再提交。

结果记录方法：记录输入信息、提示信息、数据库中每个表的更改信息。

### 3.6.2输入、输出、过程

|  |  |  |  |
| --- | --- | --- | --- |
| 输入数据 | 选择策略 | 命令 | 输出数据 |
| 没有欠费记录的读者的借阅证号 | 测试该读者没有欠费记录的情况 | 缴费按钮 | “该读者没有欠费记录”的提示框 |
| 有欠费记录的读者的借阅证号 | 测试该读者有欠费记录的情况 | 缴费按钮 | “缴费成功”的提示框  Punish\_Info表中删除一条记录 |

## 3.7书籍挂失

考虑2种情况：

* + 该读者的这本书逾期
  + 该读者的这本书在借书期限内

### 3.7.1控制

输入：半自动

操作顺序：先按上面2种情况的顺序输入相应的借阅证号和书号，再提交。

结果记录方法：记录输入信息、提示信息、数据库中每个表的更改信息。

### 3.7.2输入、输出、过程

|  |  |  |  |
| --- | --- | --- | --- |
| 输入数据 | 选择策略 | 命令 | 输出数据 |
| 该读者的这本书已逾期 | 测试该书已逾期的情况 | 挂失按钮 | “该书已欠费…”的提示框，Punish\_Info表中的欠费金额是书本身的罚款再加上逾期的罚款 |
| 该读者的这本书在借书期限内 | 测试该书没有逾期的情况 | 挂失按钮 | “该书已欠费…”的提示框，Punish\_Info表中的欠费金额是书本身的罚款 |

## 3.8办理借阅证测试

考虑4种情况：

* + 借阅证号已存在
  + 该身份证号已存在一张借阅证
  + 读者基本信息有空
  + 符合办理借阅证的条件

### 3.8.1控制

输入：半自动

操作顺序：先按上面4种情况的顺序输入相应的借阅证号和书号，再提交。

结果记录方法：记录输入信息、提示信息、数据库中每个表的更改信息。

### 3.8.2输入、输出、过程

|  |  |  |  |
| --- | --- | --- | --- |
| 输入数据 | 选择策略 | 命令 | 输出数据 |
| 读者的基本信息有空 | 测试读者信息有空的情况 | 办理借阅证按钮 | “请将信息填写完整”的提示框 |
| 已存在的借阅证号 | 测试该借阅证号已存在的情况 | 办理借阅证按钮 | “该借阅证号已存在”的提示框 |
| 该身份证号已存在一张借阅证 | 测试一位读者限办一张借阅证的情况 | 办理借阅证按钮 | “该读者已办理一张借阅证，按照规定不能再办理借阅证”的提示框 |
| 以上情况都没有且符合办理借阅证的条件 | 测试符合办理借阅证条件的情况 | 办理借阅证按钮 | “办理借阅证成功”的提示框，Proof\_Info表中增加一条记录 |

## 3.9修改密码测试

考虑4种情况：

* + 用户不存在
  + 用户名、密码、修改密码、确认密码有空项
  + 确认密码和修改密码不一样
  + 以上情况都没有，可以修改

### 3.7.1控制

输入：半自动

操作顺序：先按上面4种情况的顺序输入相应的借阅证号和书号，再提交。

结果记录方法：记录输入信息、提示信息、数据库中每个表的更改信息。

### 3.7.2输入、输出、过程

|  |  |  |  |
| --- | --- | --- | --- |
| 输入数据 | 选择策略 | 命令 | 输出数据 |
| 用户名或密码错误 | 测试用户不存在的情况 | 修改密码按钮 | “该用户不存在”的提示信息 |
| 用户名、密码、修改密码、确认密码有空项 | 测试填写信息由空项的情况 | 修改密码按钮 | “请认真填写信息”的提示信息 |
| 确认密码和修改密码不一样 | 测试确认密码和修改密码不一样的情况 | 修改密码按钮 | “确认密码和修改密码不一致”的提示信息 |
| 以上情况都没有，符合修改密码条件 | 测试可以修改密码的情况 | 修改密码按钮 | “修改密码成功”的提示信息，Admini\_Info表中相应的用户密码被修改 |

## 3.10注册管理员测试

考虑5种情况：

* + 用户名已存在，用户名和密码的组合不存在
  + 密码已存在，用户名和密码的组合不存在
  + 用户名和密码都已存在，但该用户名和密码的组合不存在
  + 用户名和密码都不存在
  + 该用户名和密码的组合已存在

### 3.7.1控制

输入：半自动

操作顺序：先按上面5种情况的顺序输入相应的借阅证号和书号，再提交。

结果记录方法：记录输入信息、提示信息、数据库中每个表的更改信息。

### 3.7.2输入、输出、过程

|  |  |  |  |
| --- | --- | --- | --- |
| 输入数据 | 选择策略 | 命令 | 输出数据 |
| 已存在的用户名，但用户名和密码的组合不存在 | 测试注册的条件 | 注册  按钮 | “注册成功”的提示信息，Admini\_Info表中增加一条记录 |
| 已存在的密码，但用户名和密码的组合不存在 | 测试注册的条件 | 注册  按钮 | “注册成功”的提示信息，Admini\_Info表中增加一条记录 |
| 已存在的用户名和密码，但该用户名和密码的组合不存在 | 测试注册的条件 | 注册  按钮 | “注册成功”的提示信息，Admini\_Info表中增加一条记录 |
| 不存在的用户名和密码 | 测试注册的条件 | 注册  按钮 | “注册成功”的提示信息，Admini\_Info表中增加一条记录 |
| 已存在的用户名和密码且该用户名和密码的组合已存在 | 测试注册的条件 | 注册  按钮 | “该用户已存在”的提示信息 |

# 4评价准则

## 4.1范围

此测试计划说明书中的测试用例能基本上包括所有的情况，基本上能反映此软件是否存在错误。其局限性是用例比较多，比较花时间。

## 4.2数据整理

由于系统规模不大，输入输出的数据类型简单，所以，我们采用手工方式将测试数据加工成便于评价的适当的形式。

## 4.3尺度

* + 以能发现错误为准则
  + 能正确完成功能要求。
  + 测试用例正确执行，要求输出与预期的输出结果完全一致。

## 8.2 Usability Test

**Library Management System test description**

We use an approach of usability testing that involves testing a relatively small, representative group of users in order to produce reliable results, reduce testing costs, and reduce the amount of time spent conducting and analyzing test results. Industry studies confirm that carefully designed tests with a small number of test participants (4-6, typically) discover at least 80% of the usability problems revealed in formal tests with much larger numbers of test participants.

**Test format**

The complete test for each test participant involved the following activities:

Pre-test activities consisted of phone calls to qualify the test participants and completion of a pre-test questionnaire designed to confirm that the test participant met our user profile requirements.

Formal testing in a usability lab involved testing individual participants in a lab environment set up much like a dorm, classroom or library. Users completed real-life scenarios using Library Management System while we recorded their activities on video and observed them through a one-way mirror. The usability test room (where test participants worked) was furnished with three video cameras set up to tape test participants from various angles.

Walkthrough—To test our scenarios and the general “do-ability” of the test, we invited a sample test participant to complete the test scenarios in the lab while we logged the results. We used the results from our walkthrough test participant to confirm that the test scenarios were readily understandable and to review our timing assumptions. The walkthrough experience gave us a chance to revisit some awkward wording and revise our test time limits.

Pilot test—After refining the test scenarios and questionnaires, we conducted a pilot test under authentic test conditions to make sure our revisions had corrected all critical problems with the test.

Post-task and post-test activities consisted of short questionnaires administered between scenarios and a longer questionnaire and interview administered after the final scenario. The test team used the final interview as an opportunity to ask test participants about specific issues that arose during their individual tests.

Summary sessions conducted after each test allowed the evaluation team to quickly examine test results and categorize usability issues discovered in the test.

**Test times**

We tested Library Management System both on weekday and on weekends. We tested two users per 1-hour session.

**Test participant selections**

We decided to test two groups of users, novice users and advanced users. By testing novices, we were more likely to collect information about the intuitiveness of the application. By targeting advanced users, we could be confident of assembling a group of people capable of performing the more difficult tasks we were asked to test.

Novice Profile: We selected novice test participants who met our requirements of having basic computer experience, including keyboarding skills, Windows experience, and the ability to access and navigate a Web browser.

Advanced Profile: We selected advanced users who met the minimal requirements of the novice users, but who in addition had previous experience using other library management programs (except for this Library Management System). We limited this study to participants who had extensive Web experience (6 months +).

**Test problems**

Test participant selection problems

We originally recruited and qualified three novice test participants and three advanced participants, each group containing a mixture of ages and genders. However, during one evening testing session, Library Management System was unexpectedly taken down for maintenance. We had to excuse our sole male novice participant test participant and recruit another on short notice. The replacement participant subject was a 45+-year-old female with advanced user skills.

Because of this substitution, our final test sample was composed of four advanced participant users and only two novice users.

Wide variations in Library Management System response

We noticed a considerable difference in system responsiveness depending on the time of day that the tests took place. Library Management System was very slow on weekday evenings, but quite responsive on weekend mornings. This is important to note because it may help explain why there is such a wide divergence of opinion among the test participants concerning the "speediness" of Library Management System.

**Scenario and questionnaire design**

We created a set of scenarios designed to test the issues discussed previously. Each scenario was designed to require between 15-25 minutes to complete. We constructed the scenarios to be as real-world as possible, containing multiple related tasks (e.g. searching a book and then borrowing it).

The questionnaires administered after each scenario and after the complete tests were designed to collect some of the "qualitative" data omitted from the formal testing. The questionnaires quizzed test participants about their perceptions about Library Management System and how well they understood and liked the product.

**Scenario design**

We revised our test scenarios slightly because of some problems that occurred during testing. The final scenarios and tasks were as follows:

|  |
| --- |
| Scenario 1: Registering for Library Management System and Logging On (time limit: 10 minutes) |
| Scenario 2: Searching, sorting and borrowing books (time limit: 15 minutes)  Task A: Search a book by the title  Task B: Sort the results by the author  Task C: Click one book and borrow it |
| Advanced user scenario  Scenario 3: Search a book by the title, ISBN, author, press, published date (time limit: 15 minutes)  Task A (Discontinued because application wouldn't support a search for non-existing Library Management System books).  Task B: Sort the results by the title, author, press, published date  Task C: Borrow a book  Task D: Check the books’ comments |
| Scenario 4: Cancel borrowing and Leaving Library Management System (time limit: 10 minutes)  Task A: cancel the previous borrowing  Task B: Sign off from Library Management System. |

**Time limits**

We allotted 50 minutes for the test participants to complete all four scenarios.

When we originally designed the scenarios, we believed that because of their limited abilities and experience, novice test participants might not have enough time to attempt Scenario 3, which tests the advanced Library Management System tasks.

However, because our novice users were able to complete scenarios 1, 2, and 4 reasonably quickly (setting aside system-imposed delays), we decided to allow them to attempt the more advanced tasks. As a result, the test scenarios and time allotments are the same for the two groups of test participants.

**Test questionnaires and interview design**

We designed a series of questionnaires to qualify potential test participants and to collect subjective feedback from the test participants during and after testing. The Appendix contains the actual questionnaires.

Pre-test questionnaire

To qualify test participants, we asked a series of questions designed to collect demographic information and to assess their level of computer and Internet experience. We accepted for testing those test participants who met our minimum requirements for novice or advanced users.

Post-task questionnaires

After the test participants completed each scenario, we administered a post-task questionnaire. The questionnaire was designed to capture feedback about Library Management System tasks while the test participant's memory was fresh. The questionnaires asked test participants to rate the ease or difficulty of the tasks, describe whether Library Management System's terminology relative to the task was clear, and provide free-form comments concerning the tasks.

Post-test questionnaire

After the test participants completed the final scenario, we administered a questionnaire designed to capture their general opinions about Library Management System. The questionnaire asked test participants to rate their interest in having online library management, their impressions about Library Management System's speed and responsiveness, and their overall satisfaction with the product.

Post-test interview

After each of the test participants completed the post-test questionnaire, the test briefer (the evaluation team's liaison with the test participant) scanned the questionnaire and asked the test participant to explain some of the responses. As the need arose, the briefer occasionally presented the test participant with additional questions that the evaluation team compiled while observing the test.

**Usability criteria**

To categorize the test observations, we applied a usability test analysis model that categorizes product usability problems according to whether they fail to meet any of the following 102 criteria:

1. Concept: Does the product use effective metaphors? Is it intuitive?

2. Consistency: Does the product look and perform similarly through all parts of the application?

3. Content: Is the content accurate, appropriately complex, and provided in the right amount? 4. Feedback: Does the product provide appropriate feedback to the user?

5. Interaction Model: Are user responses and other system interactions handled according to established models?

6. Navigation: Is it easy to get where you want to go in the product?

7. Terminology: Is the interface's language easy to understand for the audience(s)?

8. User Assistance: Does the product supply an appropriate amount of user help (e.g., online help, how to get customer service, other instructions)?

9. User Preference: Does use of the product cause difficulty for individual users in other ways not covered in these categories (e.g., does it favor "mouse users" over "keystroke users"; is it perceived to be too slow)?

10. Quality Assurance: Is the product robust? Does it fail in ways that prevent users from performing tasks?

Our test results produced findings in most of these areas, which we rated using the rating scales discussed in the following section.

**Scope and severity ratings**

After identifying usability problems, we rated their scope and severity. Scope refers to how widespread the problem was throughout the product, and severity codes rate the seriousness of the problem. Scope Local problems consisted of problems that occurred only in a particular part restricted to a particular part of the application, while global problems indicated far-reaching design flaws that occurred with consistency throughout the product. In general, global problems tend to be critical to correct, because they affect usability of the entire application, have far-ranging effects, and tend to be critical to correct. However, some local problems are critical enough to hamper severely the users' ability to perform key tasks.

Severity

We used the following severity codes to classify the seriousness of the problems we discovered:

1 - Prevents completion of a task

2 - Causes significant delays in completing a task

3 - Causes minor usability problems, but users can complete the task

4 - Minor annoyance - does not significantly impact usability, but should be corrected if time allows

# 9 Conclusion

## 9.1 Technical Complexity

One of the technical complexities we encountered in the system is how to do the recommendation, there are lots of algorithms for recommendation, typically producing a list of recommendations in one of two ways - through collaborative or content-based filtering. [Collaborative filtering](http://en.wikipedia.org/wiki/Collaborative_filtering) approaches build a model from a user's past behavior (items previously purchased or selected and/or numerical ratings given to those items) as well as similar decisions made by other users; then use that model to predict items (or ratings for items) that the user may have an interest in. [Content-based filtering](http://en.wikipedia.org/w/index.php?title=Content-based_filtering&action=edit&redlink=1) approaches utilize a series of discrete characteristics of an item in order to recommend additional items with similar properties.

We did a study for the algorithms and finally decided to make a simple and straight algorithm out of the principle which indicates that if the book is read by these users, and these users mostly read that book, then we recommend that book to the current user. What’s more, we use tags to find the books that are of the same kind, so if the book has the tags, we also look up other books that has the same tags and are rating high scores to recommend to the user.

## 9.2 Innovation

Living library is a new concept to apply to the school’s library, we have illustrated the concept in the previous chapter, however it’s easy to develop and hard to apply, the key solution is to provide a set of reasonable and applicable rules to make the system really work. After several talks with the leaders and administrators of the library of JSNU, we put forward such a set of rules to apply:

**Select living books**

1. Form a committee for living library to be responsible for the selecting living books
2. The committee is made of 8 people that are professors from several departments and advanced administrators of the library
3. There are 2 steps for contestants to be selected for living book. First, submit the application form online, second, take the interview of the committee members if pass the trial.
4. To pass the trial, there should be at least 4 members out of 8 to admit, to pass the review, the contestant need more than 5 members’ admissions.

**Basic requirements and steps to apply for living book**

1. Must be student or faculty member of the school
2. Must have some contributions at a certain area, e.g. published paper or medals
3. Must have good moral and strong psychological quality
4. In order for the usability and vitality of the living library, the applicant must provide at least 2 periods of available time in a week, and each period must last 30 minutes.
5. The living library is a nonprofit public service, and all the living books are volunteers to make contribute to living library, the school will only give the living books compensatory payment
6. The applicants can download the application form from the library management system’s home page and submit the form to the committee’s mailbox
7. The committee will notify the applicants the result of the trial by email or text message in 14 working days. Those who pass the trial will be told the time and place of the interview.
8. After the interview, the committee will notify the interviewees the result by email or text message in 3 working days. Those who pass the interview will be admitted to living library.

**Steps of quitting the living library**

1. The applicants can download the form on the library system’s home page, fill in the form and then submit it to the committee’s mailbox.
2. The committee member will verify the form and notify the applicant the result by email or text message.
3. Once quit the living library, the applicant cannot apply again in 3 months, so think carefully before quitting.

**Management for living library**

1. If the living book have more than 5 times of breaking the promise with the user, then administrator will give warning, if it is more than 10 times, then administrator will disqualify the living book.
2. If the user have more than 5 times of breaking the promise with the living book, then administrator will give warning, if it is more than 10 times, then administrator will add the user to blacklist.
3. If users or the living books say or do something related with discrimination, attack, trick, or sexual offend, once verified by administrator, the user or the living book will be added into blacklist or disqualified.
4. If the comments have content related with discrimination, attack, trick, or sexual offend, the user or the living book who write the comment will be warned by administrator the first time, if it is the second time, the comment maker will be added to blacklist at once.

个性化的借阅排行和新书推荐

QR login

## 9.3 Future Enhancements

E-books management

2、 与书评网、网上书店的互联互通  
BOOK+为您提供豆瓣网的互通服务，可以看到豆瓣网上读者对这本书的评价哦。如果该书没有馆藏或者不可借，还可以直接连接网上书店多家比价，提供快速的购书通道。  
3、 简单、实用的期刊目次推送  
想知道本专业的期刊篇名目次？BOOK+让您自主定制，并定期把最新的目次推送给您。  
4、 荐购的绿色通道  
在豆瓣网看到了好书，可是图书馆没有馆藏，怎么办？可以立即向图书馆申请荐购，以后就可以在图书馆借阅啦。

1. 高质量的学术搜索引擎技术

2. 体验最佳的检索结果相关性排序

3. 强大的全文检索能力

4. 复杂的高级检索功能

**Mobi+移动图书馆简介**

Mobi+移动图书馆产品是为了满足读者希望在任何地点、任何时间都可以实现快速查询和浏览图书馆资源的一款基于wap、Android、ios平台开发的高性能手机软件。

Mobi+移动图书馆依托成熟的无线移动网络、互联网以及多媒体技术，使读者不受时间、地点和空间的限制，通过各种便携移动设备（手机、PDA、手持阅读器、平板电脑、MP4等）方便灵活的进行图书馆的信息查询、浏览；一站式查找并获取图书馆纸本图书及电子资源，并可以帮助读者通过该软件享受图书馆提供的一系列个性化服务。Mobi+移动图书馆能从各个方面帮助读者实现信息化时代对信息和资源随时随地获取和使用的需求。

# 10 Appendices

## 10.1 Reference Guide

## 10.2 Usability test questionnaire

Library Management System Usability Test Final Report

Pre-Test Questionnaire

|  |
| --- |
| **Thank you for considering being a volunteer for our Usability Test**. We will work to make sure the test environment is pleasant and fun for you, with a casual dinner being served.  The results from our Usability Test will be used to help improve a computer software product’s ease of use.  **Please answer the following questions.** We will use your answers to determine if you will be a participant in our Usability Test.  **The testing will take place on July 22 and July 29 from 6-8 p.m.** The usability test will require 1 1/2 hours of your time.  Please place a check next to the dates on which you are available, if any.  July 22 \_\_\_\_\_\_ July 29 \_\_\_\_\_\_ Either Date \_\_\_\_\_\_ Neither Date \_\_\_\_\_  **Please disregard the rest of the questionnaire if you are not able to attend one of the dates**. **Thank you again for your consideration.** |

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Home Phone: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Work Phone: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Age: [ ] 15-20 [ ] 21-30 [ ] 31-40 [ ] 41-50 [ ] 51 or above

Sex: [ ] Male [ ] Female

[ ] Right handed [ ] Left handed

Please answer the following questions about your computer experience:

1. Do you use a personal computer?

[ ] Yes [ ] No

If you answered “no,” please disregard the remaining parts of the questionnaire.

2. What kind(s) of programs have you worked with? Check all that apply.

[ ] Word Processing [ ] Spreadsheets [ ] Graphics [ ] Other(s) specify\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. How long have you been using personal computers?

[ ] 0-3 months [ ] 4-6 months [ ] 7-9 months [ ] 10-12 months [ ] More than 12 months

4. Have you ever used a Web browser? [ ] Yes [ ] No

If you answered “no,” please proceed to question 7.

5. Which Web browser have you used? Check all that apply.

[ ] Microsoft Internet Explorer [ ] Netscape Navigator [ ] Other(s) specify\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. How long have you been browsing the Web?

[ ] 0-3 months [ ] 4-6 months [ ] 7-9 months [ ] 10-12 months [ ] More than 12 months

7. Which Internet Service Provider(s) do you use? Check all that apply.

[ ] AOL [ ] Prodigy [ ] CompuServe [ ] Mindspring [ ] other (specify)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[ ] I do not use/have an Internet Service Provider

8. How long have you been using your work/school location’s network?

[ ] 0-3 months [ ] 4-6 months [ ] 7-9 months [ ] 10-12 months [ ] More than 12 months

9. Do you use the library system for borrowing a book?

[ ] Yes [ ] No

If answered “no,” please proceed to question 12.

10. Which function of the system do you use? Check all that apply.

[ ] Search a book [ ] Borrow a book

[ ] browse new books [ ] other (specify)

11. How long have you been using your school’s library system?

[ ] 0-3 months [ ] 4-6 months [ ] 7-9 months [ ] 10-12 months [ ] More than 12 months

12. What do you borrow books for? Check all that apply.

[ ] Study [ ] Kill time [ ] Other (specify)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

13. Do you know how to search a book on the internet? [ ] Yes [ ] No

17. Do you know how to make the keywords out of a piece of information? [ ] Yes [ ] No

18. Do you know how to create and send a simple email message? [ ] Yes [ ] No

Thank you for completing our questionnaire. We greatly appreciate your consideration and time. We will be in contact with you if you are selected to participate in our Usability Test. Thank you!

**Post-test questionnaire**

We reviewed the results with the test participants in post-test interviews.

|  |
| --- |
| **Thanks for completing the usability test.**  Please answer the following questions about your experience with Library Management System. We will use your answers to provide important feedback to Library Management System's marketing and development staff. |

1. On the following scale, rate your need for / interest in having an online library management system:

No interest/need [ ] Low interest/need [ ] Don't feel strongly either way [ ]

Moderate interest/need [ ] High interest/need [ ]

2. On the following scale, rate your impression of Library Management System's speed and responsiveness:

Very slow [ ] Moderately slow [ ] Neither fast nor slow [ ]

Moderately fast [ ] Very fast [ ]

3. Will you use Library Management System in the future?

Never [ ] Rarely [ ] Sometimes [ ] Fairly frequently [ ] Very frequently [ ]

4. On the following scale, rate how highly you would recommend Library Management System to your friends and associates:

Would NOT [ ] Would recommend [ ] Don't feel strongly either way [ ] Would probably recommend [ ] Would strongly recommend [ ]

5. If you plan to use Library Management System at all in the future, please indicate how you might use it (Check all that apply):

Weekends for killing time [ ] Weekends for studying [ ] Weekdays for homework [ ] At dorm [ ] At classroom [ ] Not applicable--will not use [ ]

6. What did you like MOST about Library Management System?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

8. What did you like LEAST about Library Management System?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

9. What would you change about Library Management System?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

10. Optional: Please add any other comments about Library Management System that might be useful in helping Library Management System staff improve this product:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 10.3 Application form of living book

**Apply to Join:**

|  |  |  |  |
| --- | --- | --- | --- |
| Name |  | Age |  |
| Major |  | Student/Faculty Number |  |
| Phone |  | Email Address |  |
| Skype Account |  | Address |  |
| Provide at least 2 periods of available time (at least 30 minutes per period) |  | | |
| Profile |  | | |
| Specific Area and the relating expertise |  | | |
| Achievements |  | | |

**Apply to Quit:**

|  |  |  |  |
| --- | --- | --- | --- |
| Name |  | Age |  |
| Major |  | Student/Faculty Number |  |
| Phone |  | Email Address |  |
| Skype Account |  | Address |  |
| The reasons for quitting living library |  | | |