1.4 Web Application

---BookShop manager

By Yang Cheng (474340)

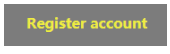
16/05/2021

Content

Product Introduction

This web application is called "Bookshop Manager". Its purpose is to help bookshop managers manage their book shops. Considering that one manager may have multiple bookshops, the bookshop supports CRUD (Creating, Reading, Updating and Deleting bookshop). What's more, the manager can also manage the books in each shop and support CRUD (Creating, Reading, Updating and Deleting books) .

These are the main functions of this APP. As required, I use HTML+css+JS to complete the front end, use fetchAPI to connect to the backend (transfer data), and use Springboot for the backend to connect to the mysql database through JPA Hibernate. One user can register for multiple bookshops, and each bookshop contains many books. Each bookShop has a label for the user, and each book has a label for the bookShop. Also, in order to make it more real, the bookshop changes will affect the book. For example,when you delete the bookshop, all the books in that bookshop will also be deleted at the same time.

Both bookshop and book have four APIs that implement CRUD. The update and add functions have separate pages for users to input add/update information. I recommend running the code directly to view it on the browser instead of using Postman, because using the api directly in postman will cause the user_email / shop_id to be null in the bookshop, because the login step/create bookshop step may be skipped by you. The login page(index page) of this web app is http://localhost:8080/index. You must register before logging in, register through this 

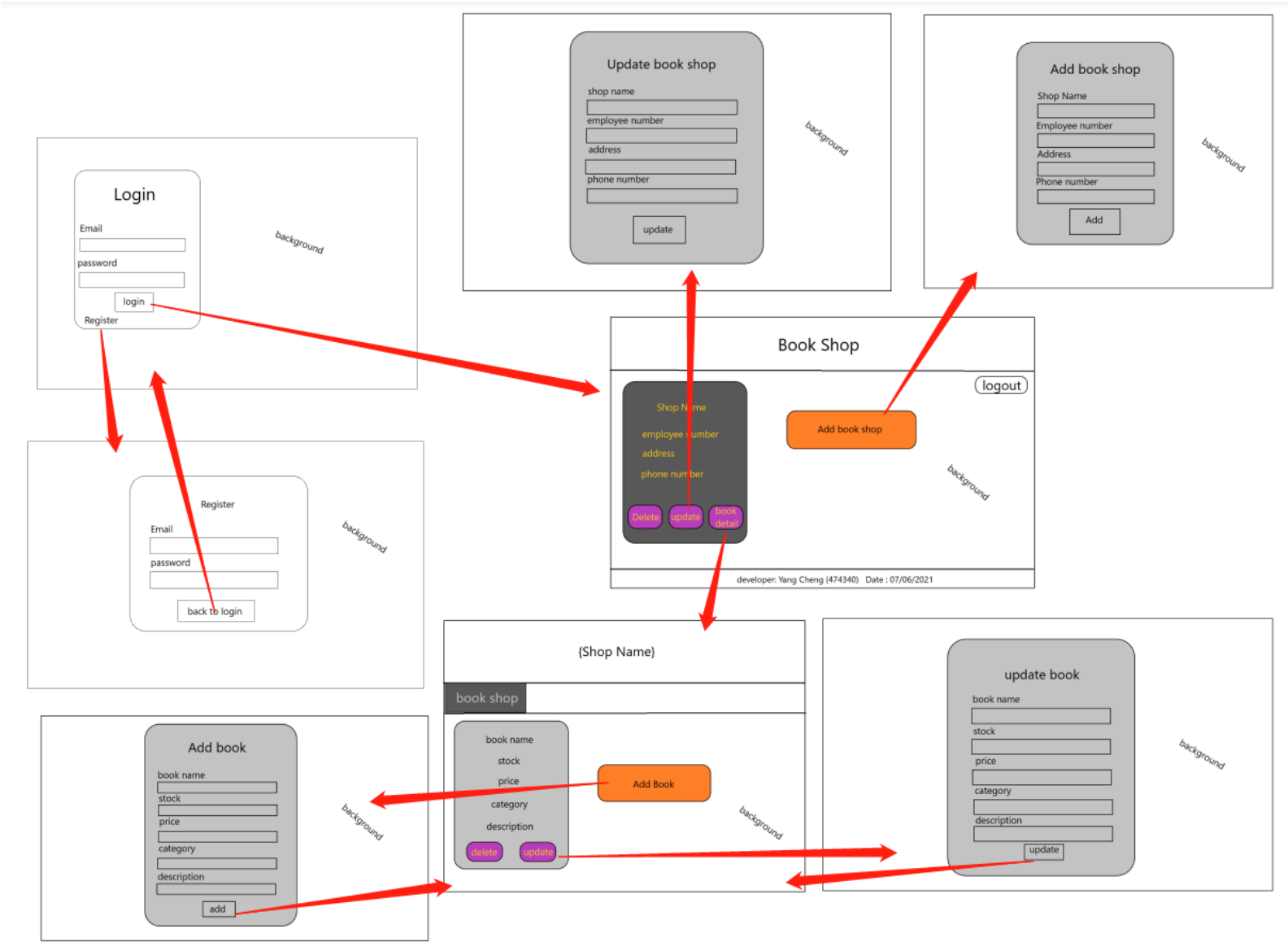
.

The detailed steps will be mentioned in the wireframe.

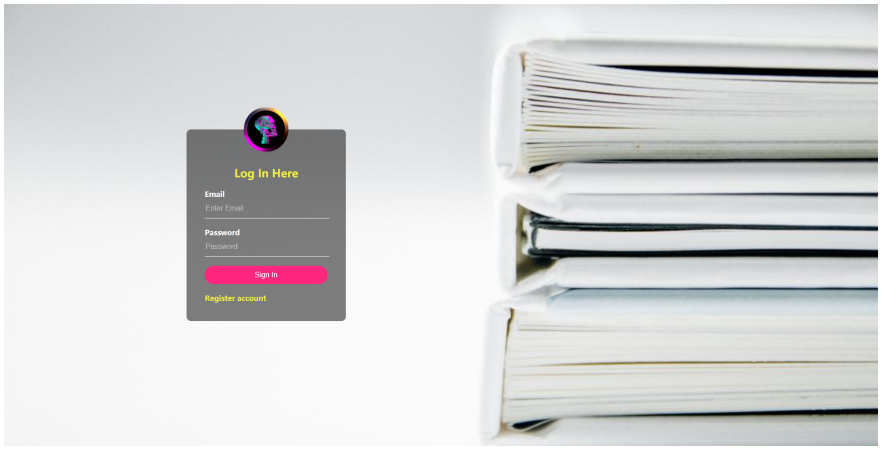
Functional design

Wireframes

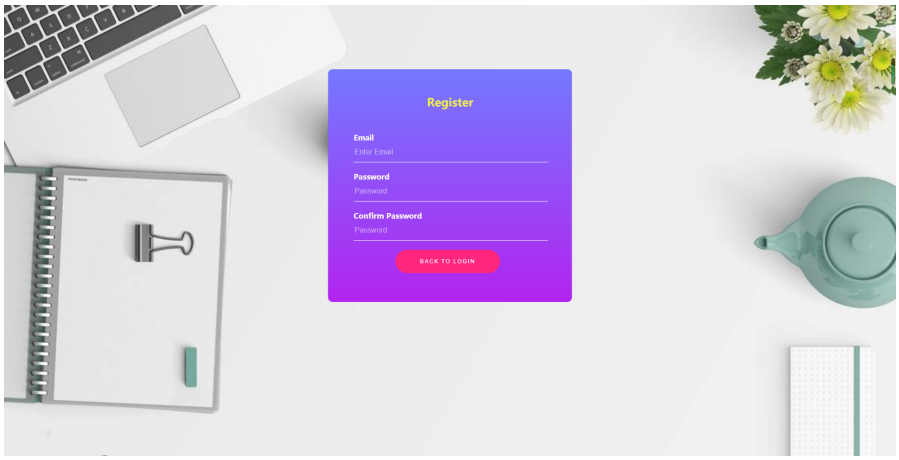
Overview:



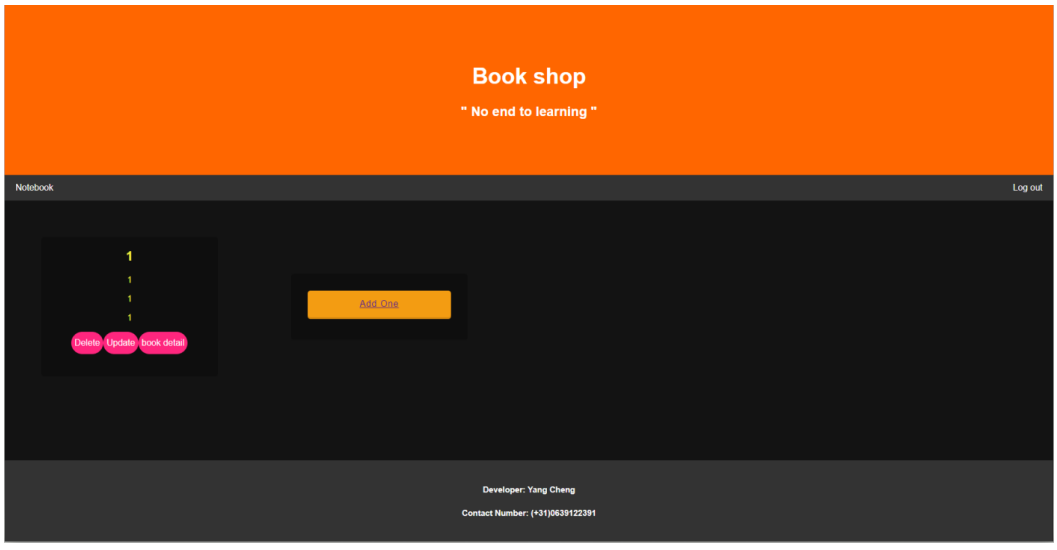
1. Login page: http://localhost:8080/index



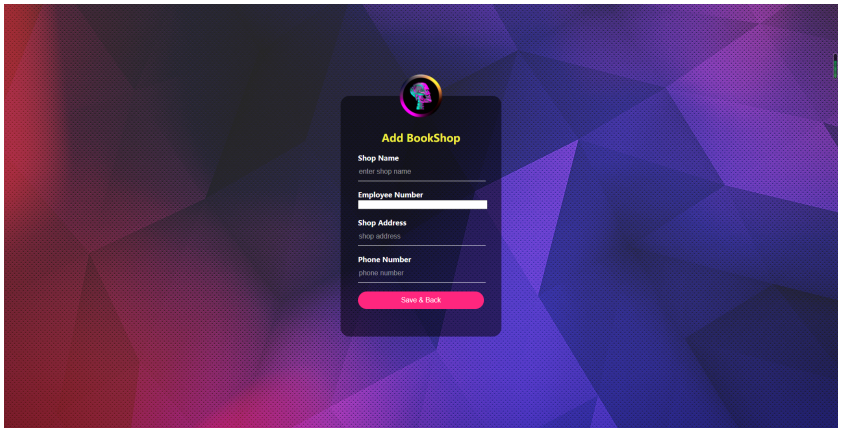
2. Register page: http://localhost:8080/register



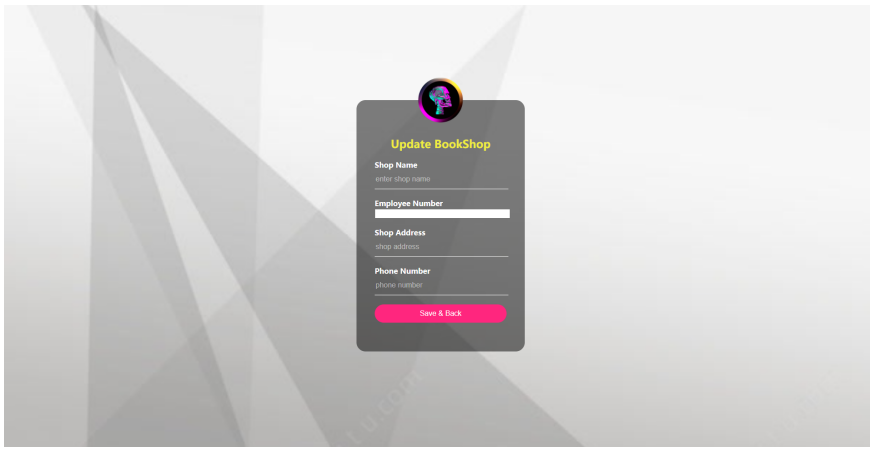
3. BookShop page: http://localhost:8080/bookShops



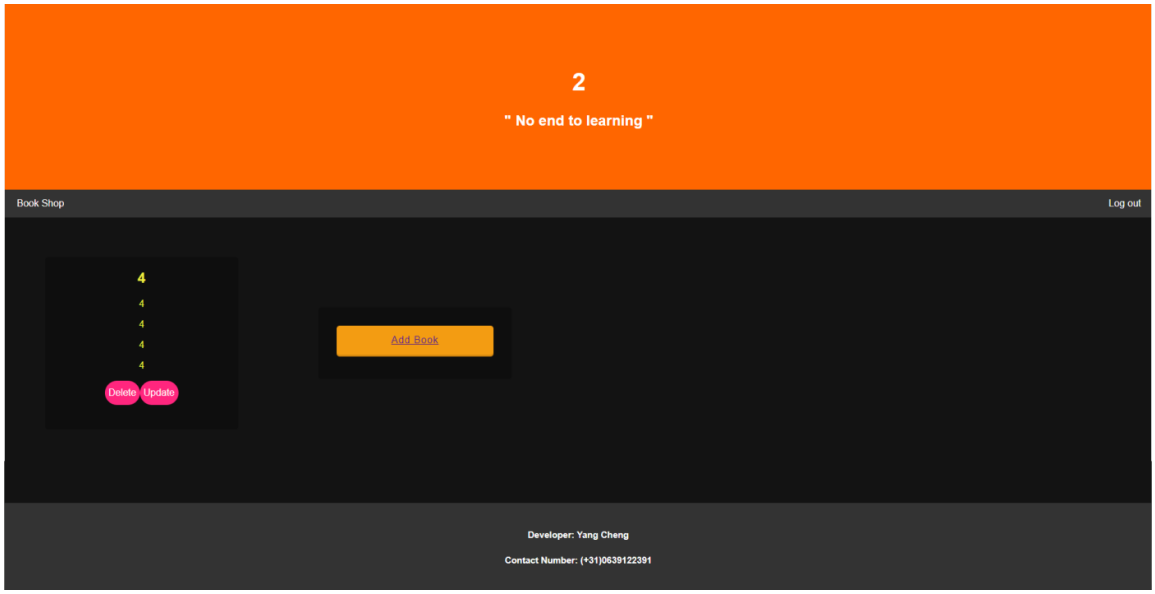
4. Add BookShop page: http://localhost:8080/addBookShop



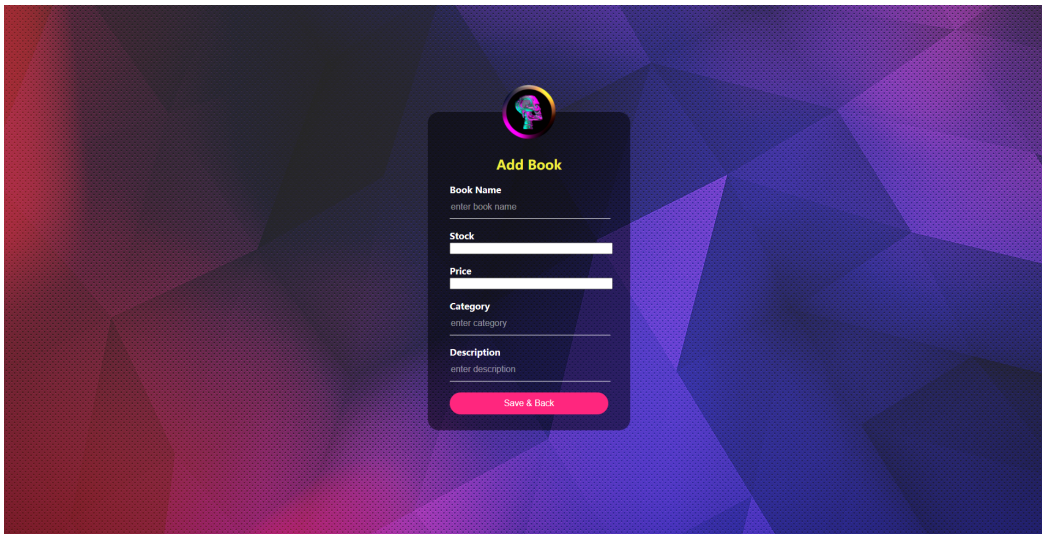
5. Update BookShop page: http://localhost:8080/bookShops/update/{id}



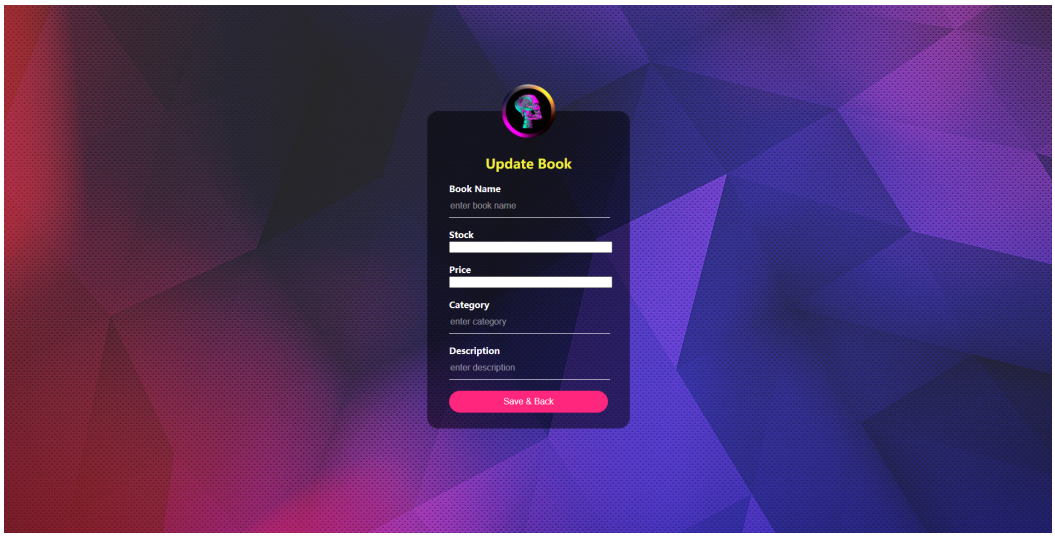
6. Book page: http://localhost:8080/{bookShop_id}/books



7. Add Book page: http://localhost:8080/addBook



8. Update Book page: http://localhost:8080/update/{id}/updateBooks



Technical design

API design（Fetch API, more detail API design is in the API document）

1. login

Functionality	User Login
URL	http://localhost:8080/index/confirm
Method	POST
Param info	• *user_id (Auto-growth in database）* • email (User's login email) • password (User's login password）

Return Msg (Json)	//successful login, jump to bookshop page
	return "redirect:/bookShops";
	
	// login failed , stay at login page
	return "redirect:/index";

2. Register page

Functionality	User Register
URL	http://localhost:8080/register/add
Method	POST
Param info	• *user_id (Auto-growth in database）* • email (User's login email) • password (User's login password）

Return Msg (Json)	//successful register, jump to login page
	return "index";
	
	// register failed , stay at register page
	return "Register";

3. get Bookshop

Functionality	get bookshop
URL	http://localhost:8080/api/getBooksShops
Method	GET
Param info	-

Return Msg (Json)	//success, return arraylist
	return showUserShop (Arraylist);
	
	//failed , return status code
	return "404";

4. add Bookshop

Functionality	add bookshop
URL	http://localhost:8080/api/addBookshop
Method	POST
Param info	-

Return Msg (Json)	//success,return status code
	return "200";
	
	//failed , return status code
	return "500";

5. update Bookshop

Functionality	update bookshop
URL	http://localhost:8080/api/updateBookshop
Method	PUT
Param info	id (bookshop id)

Return Msg (Json)	//success,return status code
	return "200";
	
	//failed , return status code
	return "406";

6. delete Bookshop

Functionality	delete bookshop
URL	http://localhost:8080/api/deleteBookshop/{id}
Method	DELETE
Param info	id (bookshop id)

Return Msg (Json)	//success,return status code
	return "200";
	
	//failed , return status code
	return "406";

7. get Book

Functionality	get book
URL	http://localhost:8080/api/getBooks
Method	GET
Param info	-

Return Msg (Json)	//success, return Arraylist<Book>.
	return showBooks (Arraylist);
	
	//failed , return status code
	return "404";

8. get bookshop name

Functionality	get bookshop name & put name on book page
URL	http://localhost:8080/books/getShopName
Method	GET
Param info	-

Return Msg (Json)	//success, return shop name & shop id.
	return shopName + shopID;
	
	//failed , return status code
	return "404";

9. add Book

Functionality	add book
URL	http://localhost:8080/api/addBook
Method	POST
Param info	-

Return Msg (Json)	//success,return shop id
	return shop_id;
	
	//failed , return status code
	return "500";

10. update Book

Functionality	update book
URL	http://localhost:8080/api/updateBook
Method	PUT
Param info	book (Book)

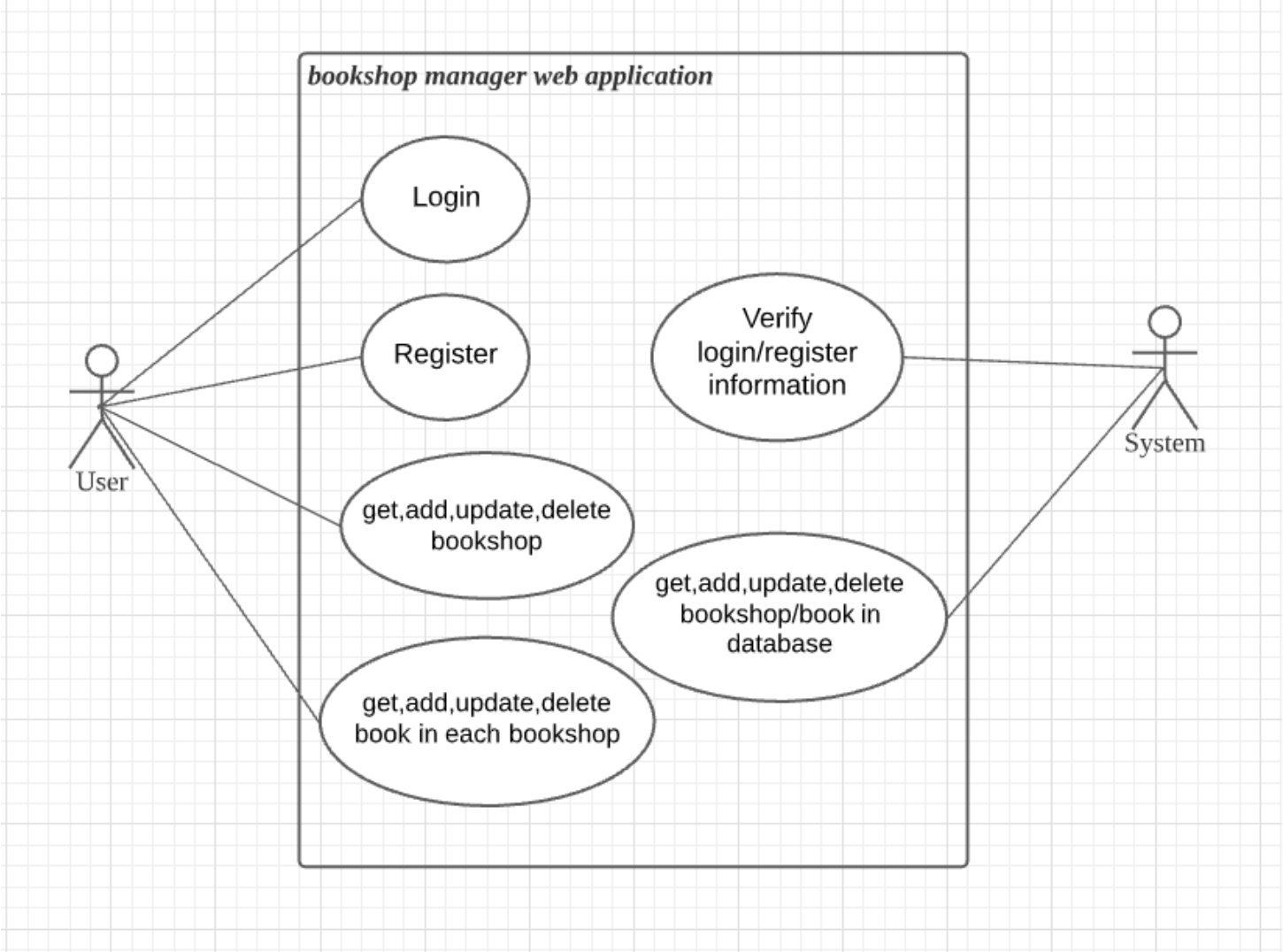
Return Msg (Json)	//success,return shop id
	return shop_id;
	
	//failed , return status code
	return "406";

11. delete Book

Functionality	delete book
URL	http://localhost:8080/api/deleteBook/{id}
Method	DELETE
Param info	id (book id)

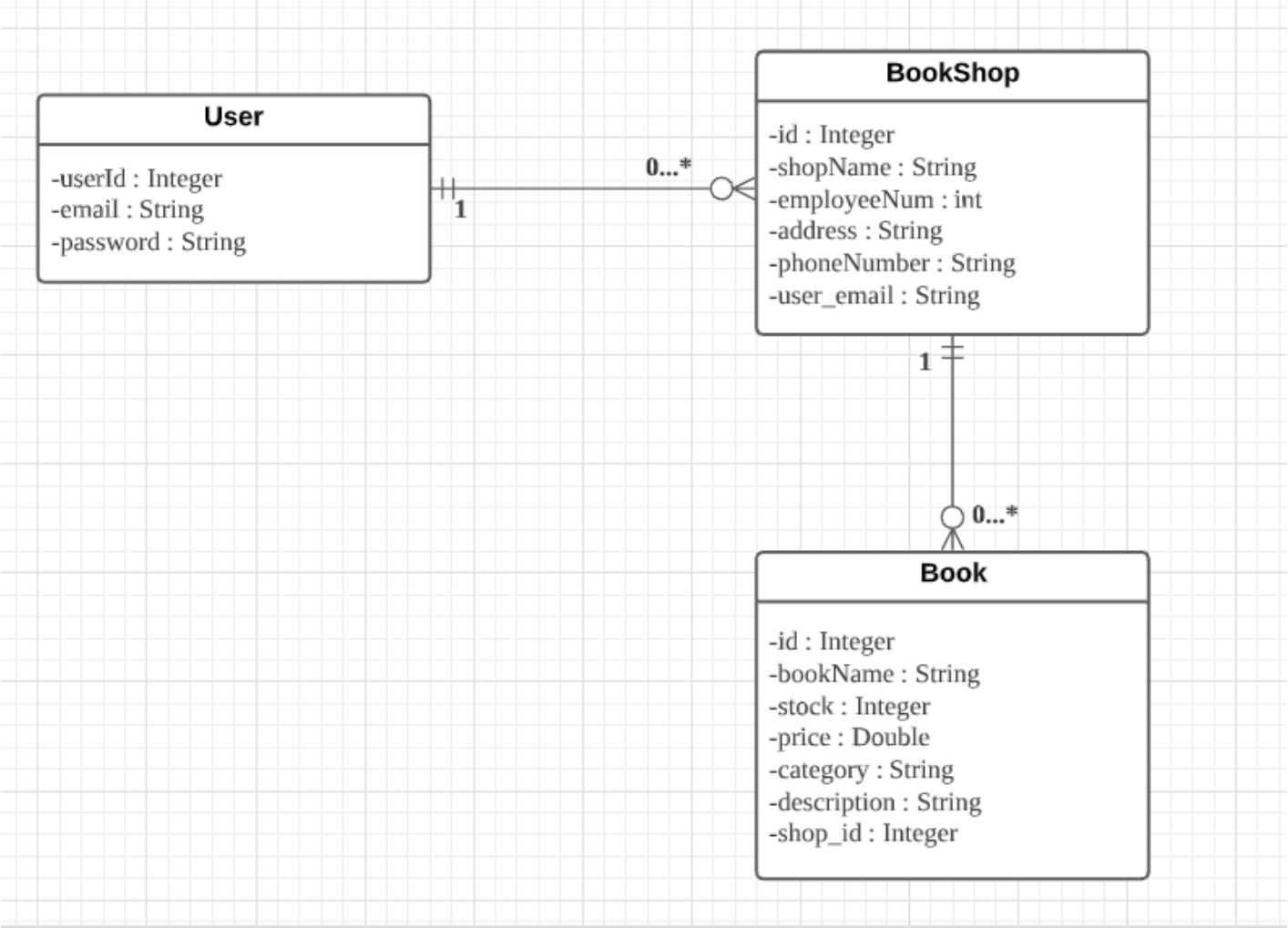
Return Msg (Json)	//success,return status code
	return "200";
	
	//failed , return status code
	return "406";

User diagram



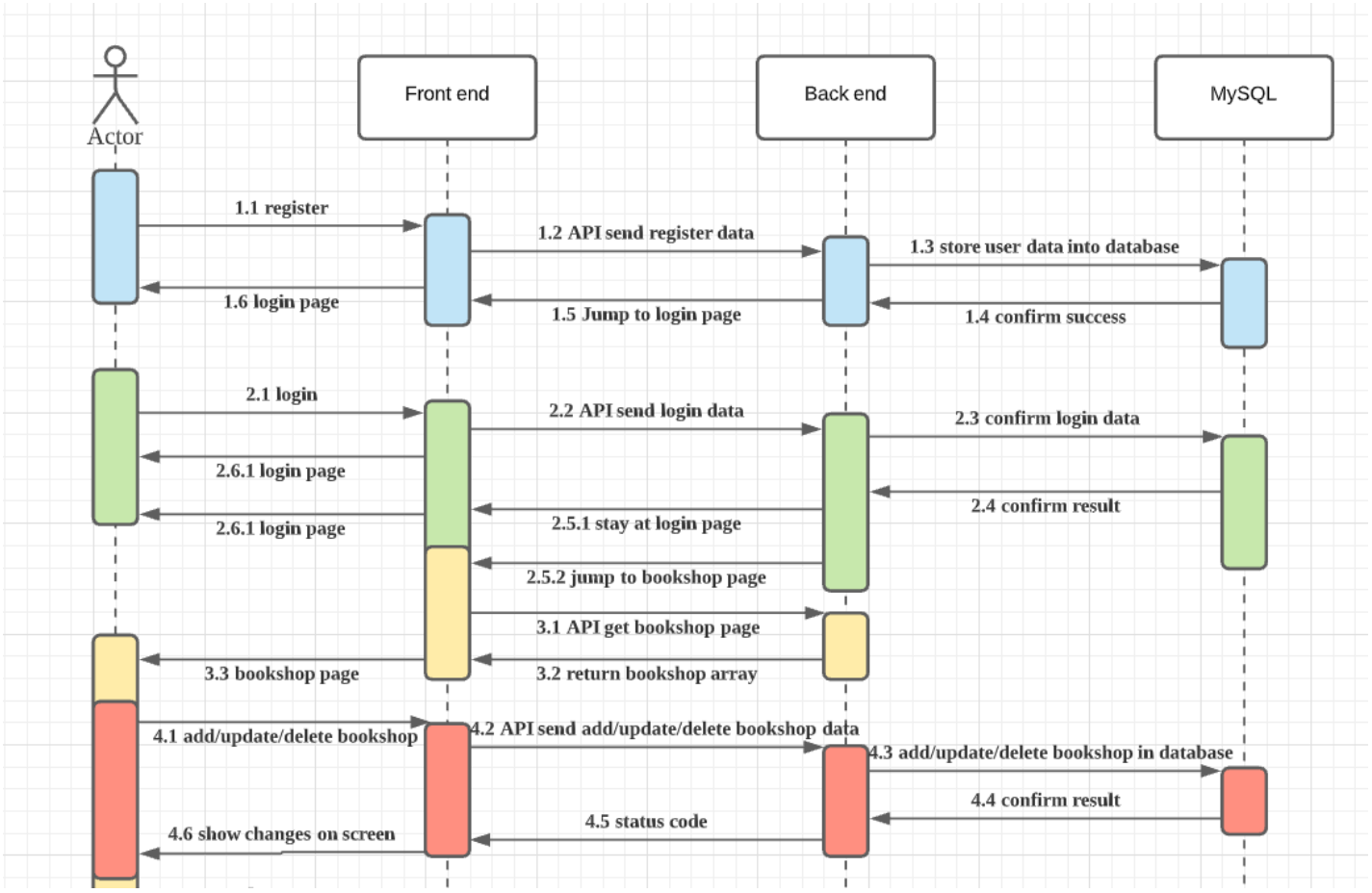
The main functions of users are login, registration, operation of bookshop(CRUD) and operation of book(CRUD) in each bookshop.

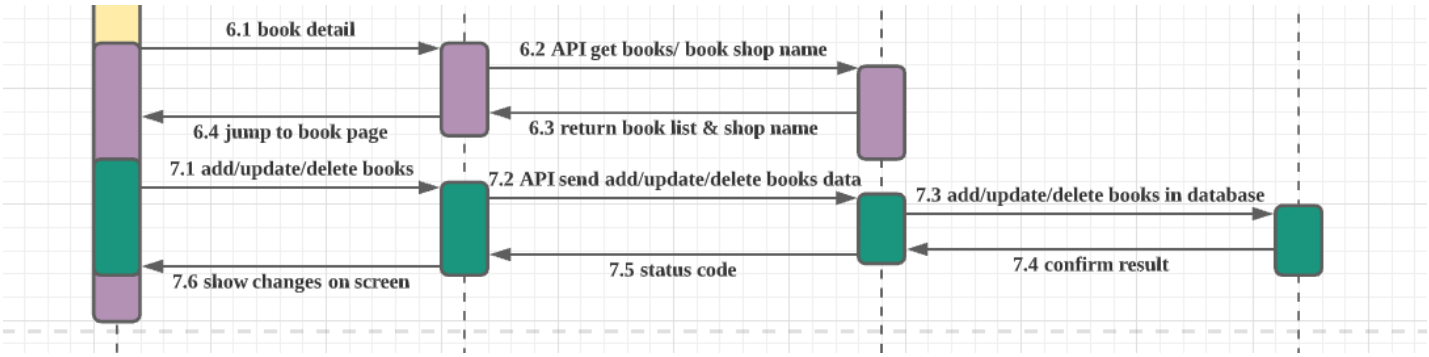
class diagram



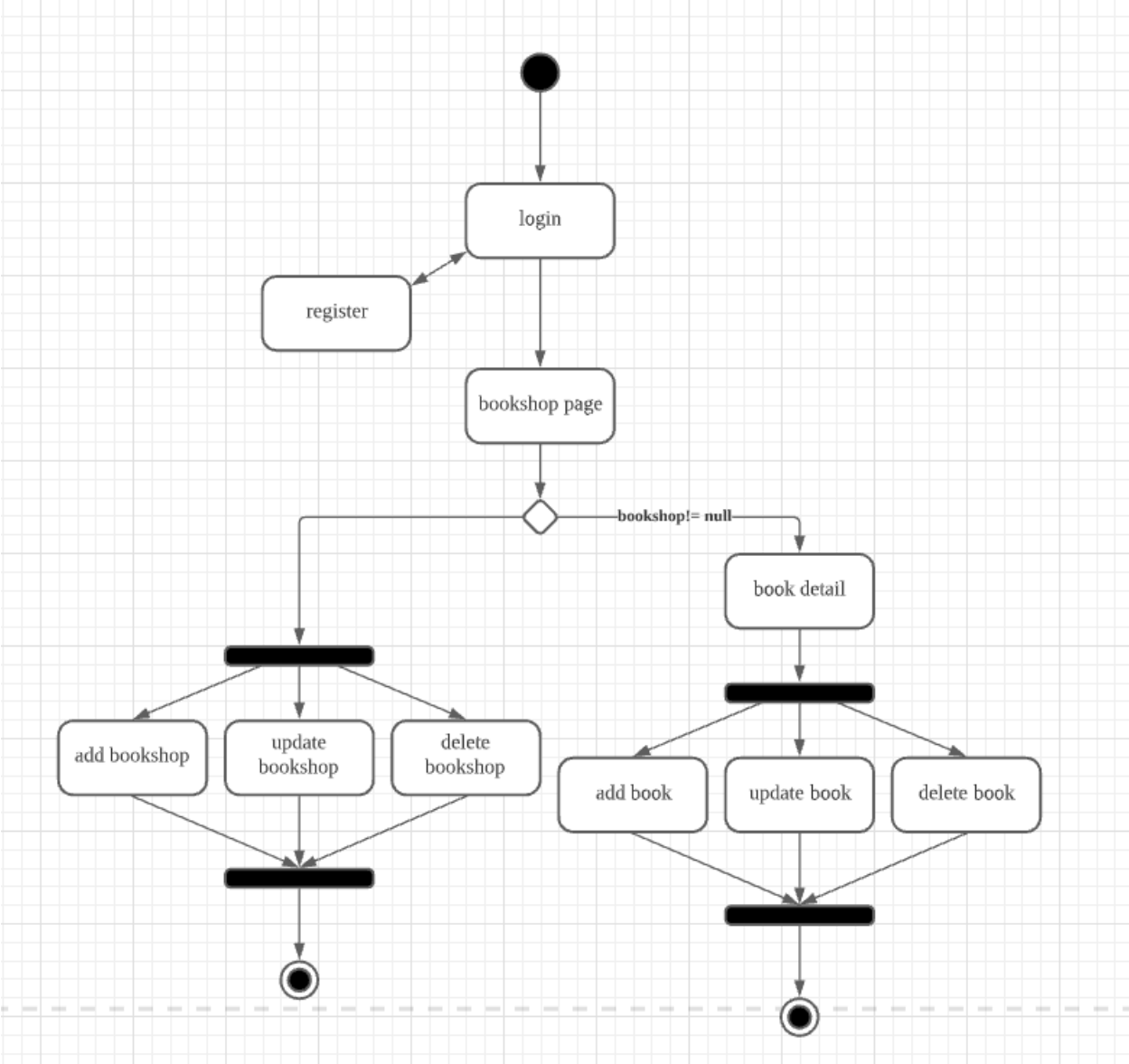
One user can register for multiple bookshops, and each bookshop contains many books. Bookstores and books are related. When a bookshop is deleted, all books in this bookshop will also be deleted at the same time. If the name of the bookshop is updated, the name of the bookshop in the book page will also be updated.

Sequence diagram





Activity diagram

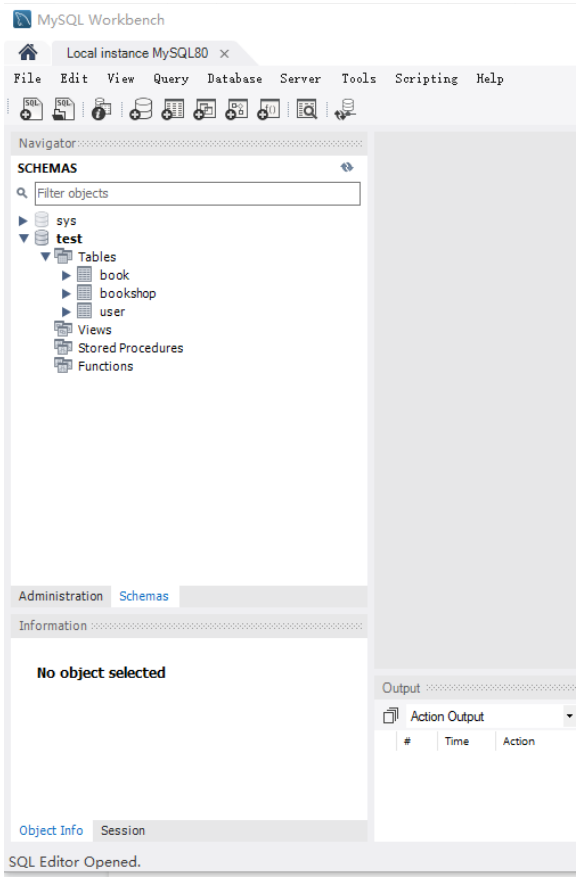


Technical overview

As required, I use HTML+css+JS to complete the front end, use fetchAPI to connect to the backend (transfer data), and use Springboot for the backend to connect to the mysql database through JPA Hibernate. About the database, I chose MySQL. In using MySQL, I encountered a lot of errors, and it tooks me a lot of time to solve them,for example, mysql does not support camelcase writing , the use of update actually shares a method with add in mySQL, after deleting a row, the id will still be retained...etcs.

In the book page, after operating the book, there may be a bug that the page does not display the book. This is caused by the asynchronous processing of fetch API. Because I use two fetch API at the same time in the book page, one is to get the name of the bookshop where the book is located, and the other is to get all the books in the bookshop. When the program gets all the books in the bookshop first, and then gets the name of the bookshop, it will cause this bug. The solution is to Insert the fetch api of get books into the response of get bookshop name fetch api.

Database setting (MySQL)



According to the class diagram, I created a MySQL database, which contains three tables: user, bookshop and book.

Table: book

Columns:

book_id	int AI PK
book_name	varchar(45)
stock	int
price	double
category	varchar(45)
description	varchar(45)
shop_id	int

Table: bookshop

Columns:

shop_id	int AI PK
shop_name	varchar(45)
employee_number	int
address	varchar(45)
phone_number	varchar(45)
user_email	varchar(45)

Table: user

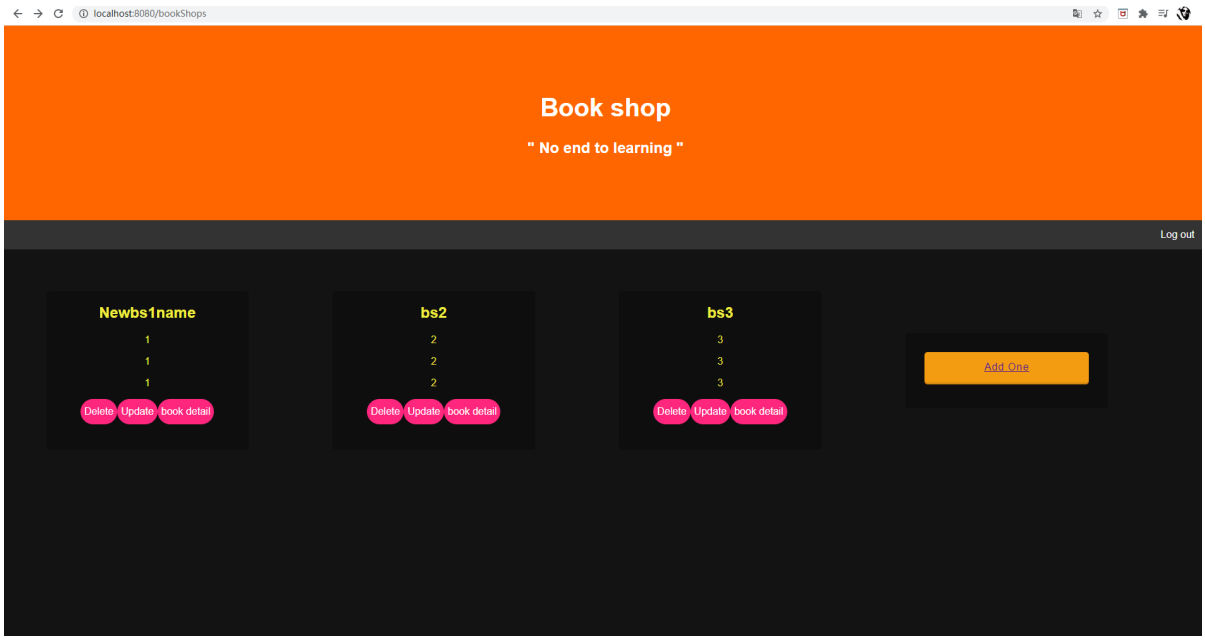
Columns:

user_id	int AI PK
email	varchar(45)
password	varchar(45)

Test & Result

After many tests, the app can run normally.

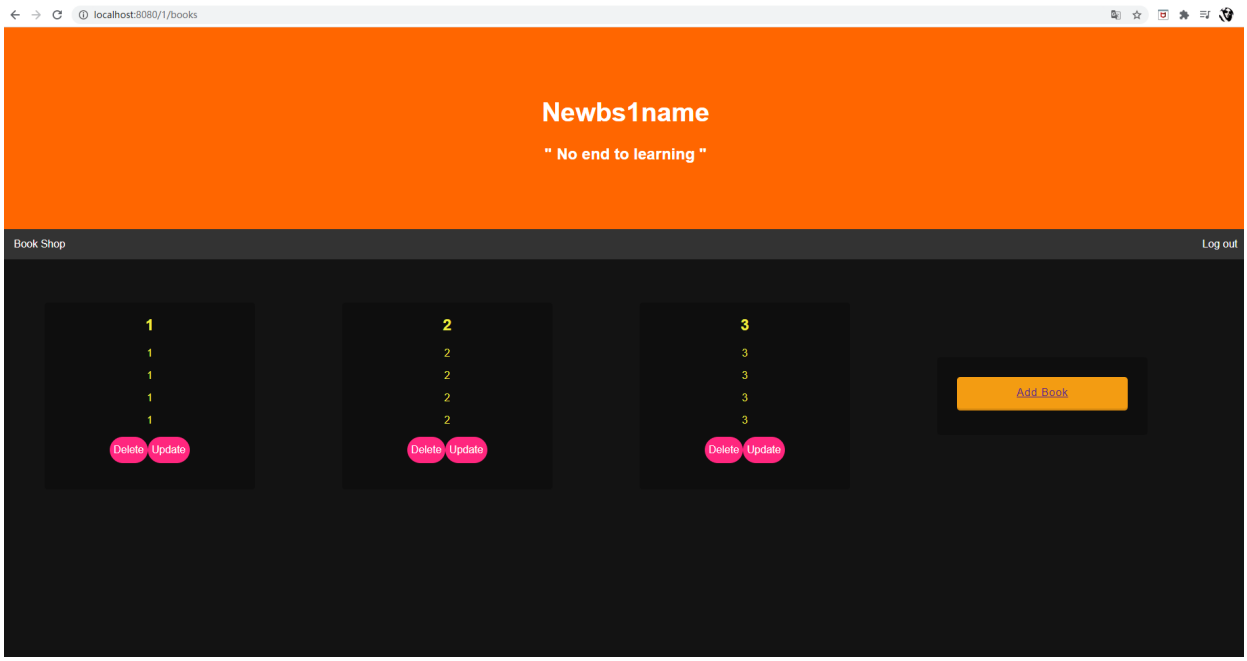
1. ***Deleting a bookshop will delete all the books in the shop at the same time. The bookstore and book pages meet CRUD.***



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap

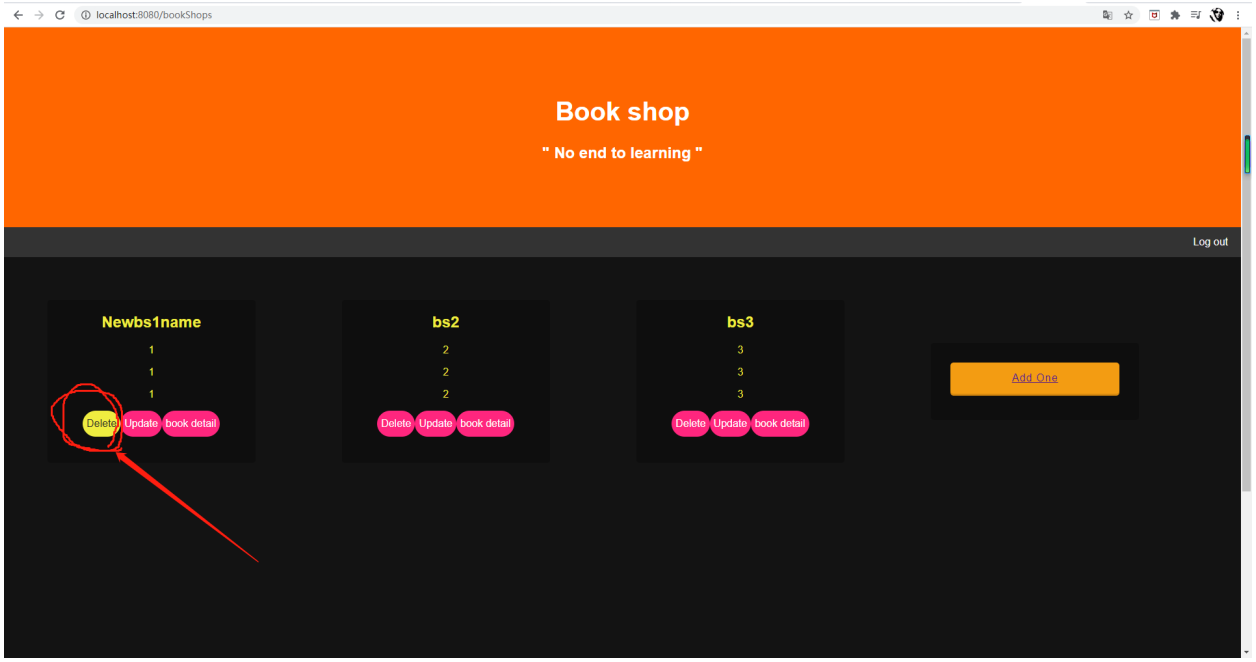
shop_id	shop_name	employee_number	address	phone_number	user_email
1	Newbs1name	1	1	1	474340@student.saxion.nl
2	bs2	2	2	2	474340@student.saxion.nl
3	bs3	3	3	3	474340@student.saxion.nl

- Create 3 bookshops

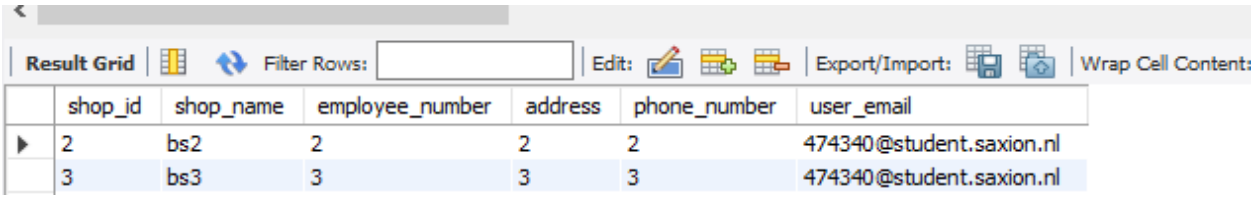


	book_id	book_name	stock	price	category	description	shop_id
▶	1	1	1	1	1	1	1
	2	2	2	2	2	2	1
	3	3	3	3	3	3	1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

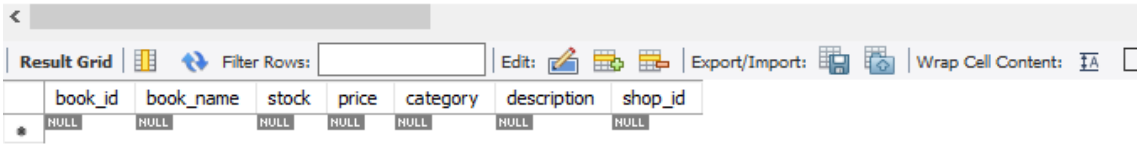
- In the first bookshop, create 3 books.



- Delete first bookshop.



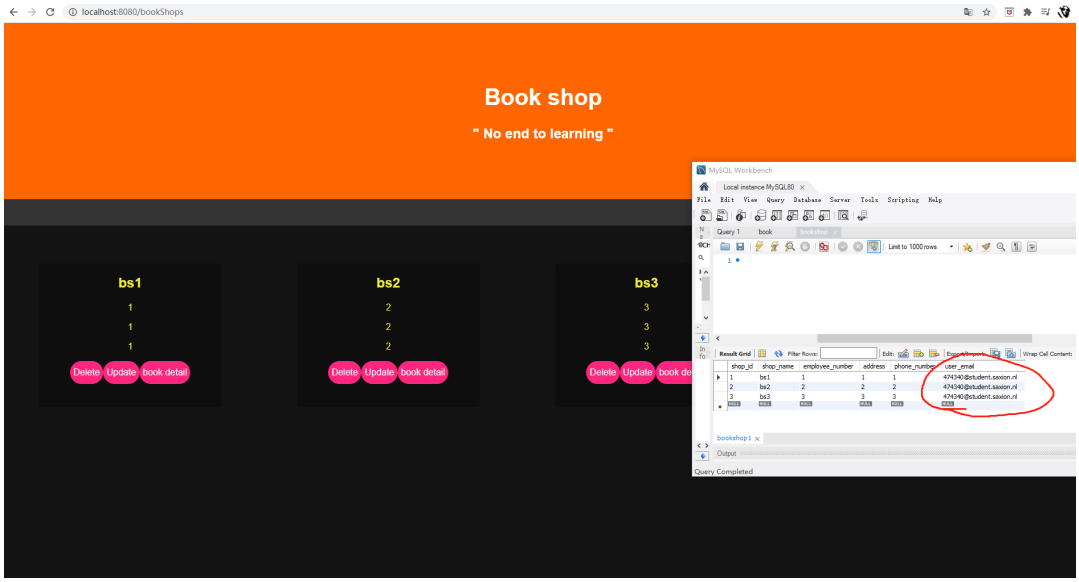
shop_id	shop_name	employee_number	address	phone_number	user_email
2	bs2	2	2	2	474340@student.saxion.nl
3	bs3	3	3	3	474340@student.saxion.nl



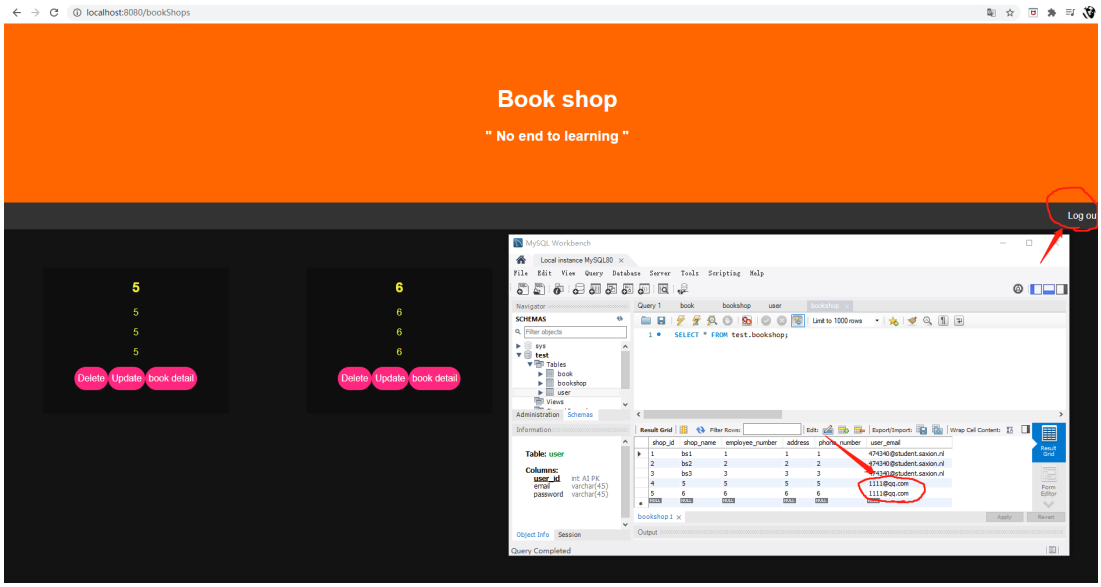
book_id	book_name	stock	price	category	description	shop_id
NULL	NULL	NULL	NULL	NULL	NULL	NULL

- books in the first bookshop are also deleted at the same time.

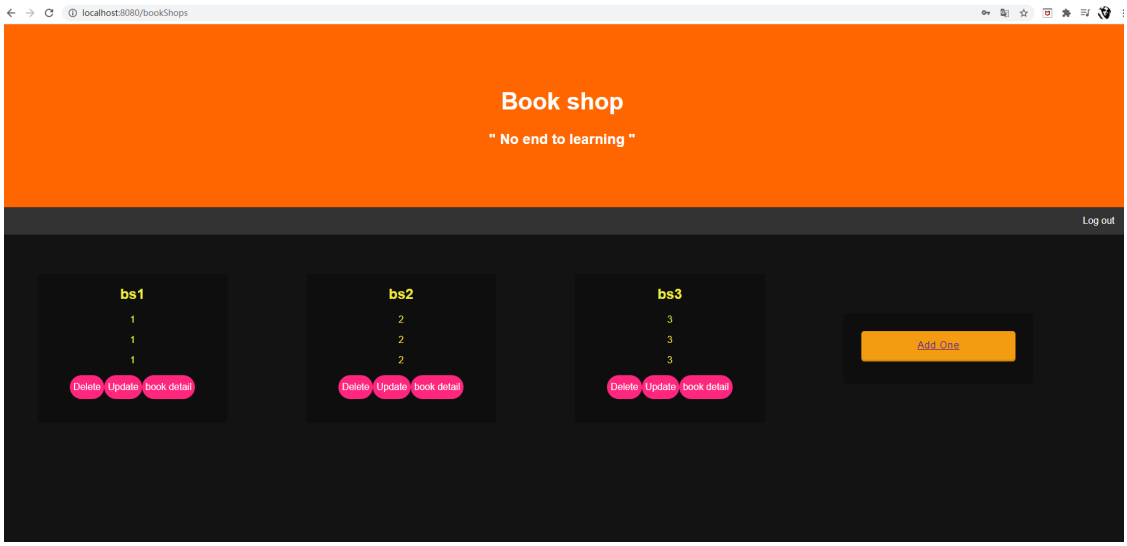
2. When users switch accounts, the app can display the bookshop they created before.



- create account A, then create some bookshops.

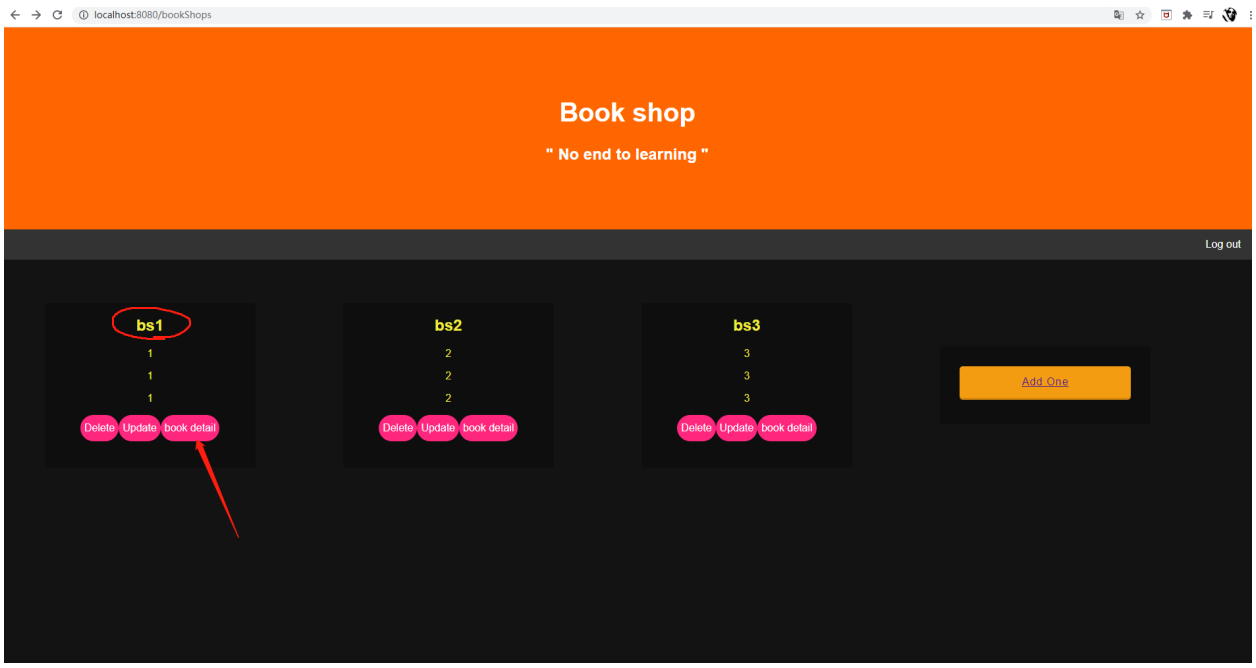


- logout A, create account B, then create some bookshops.

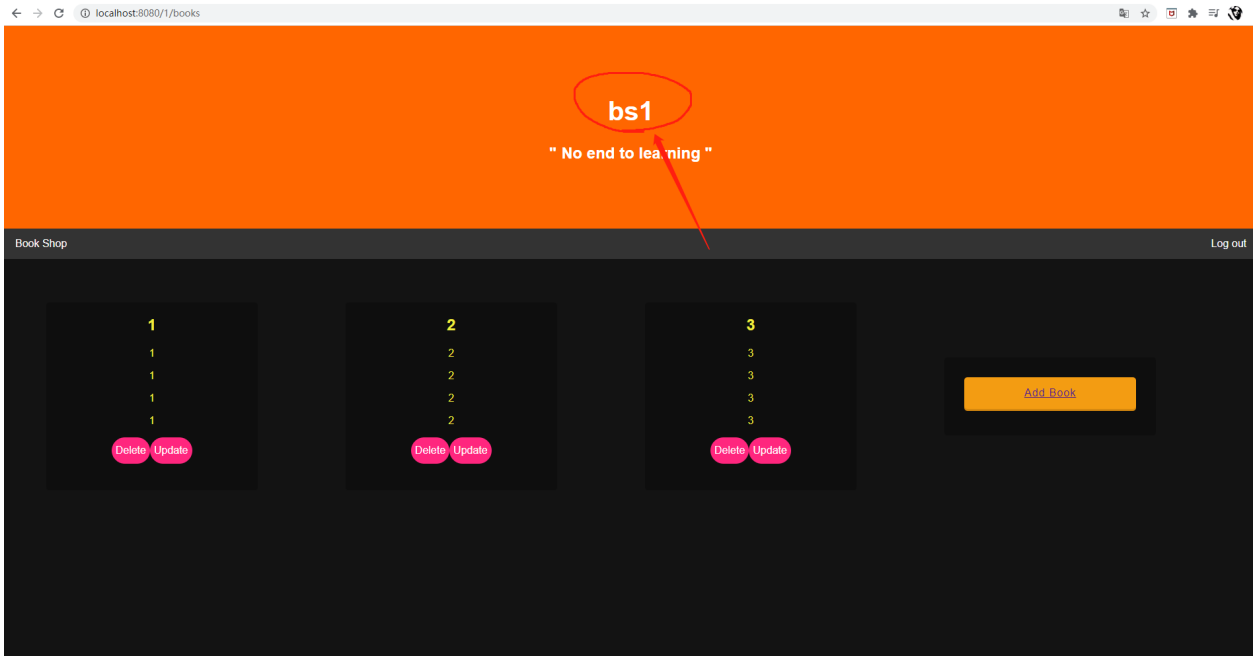


- Switch account to A, the original bookstore is still there.

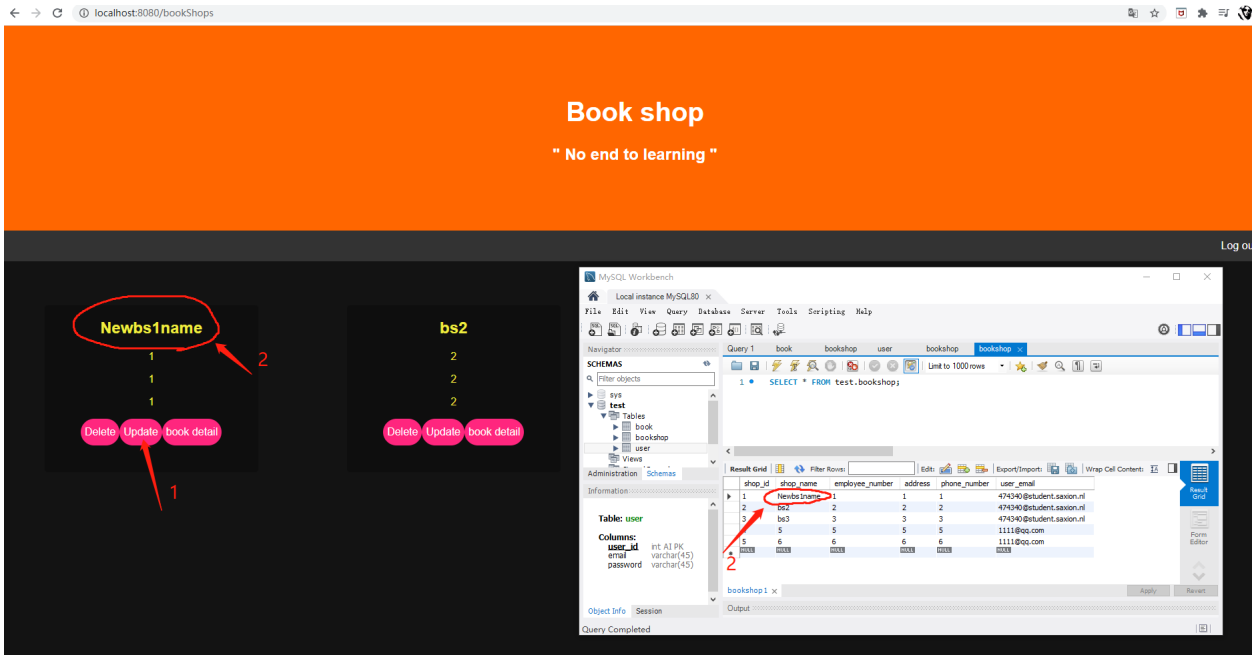
3. When the bookstore name is updated, the bookshop name on the book page will also be updated.



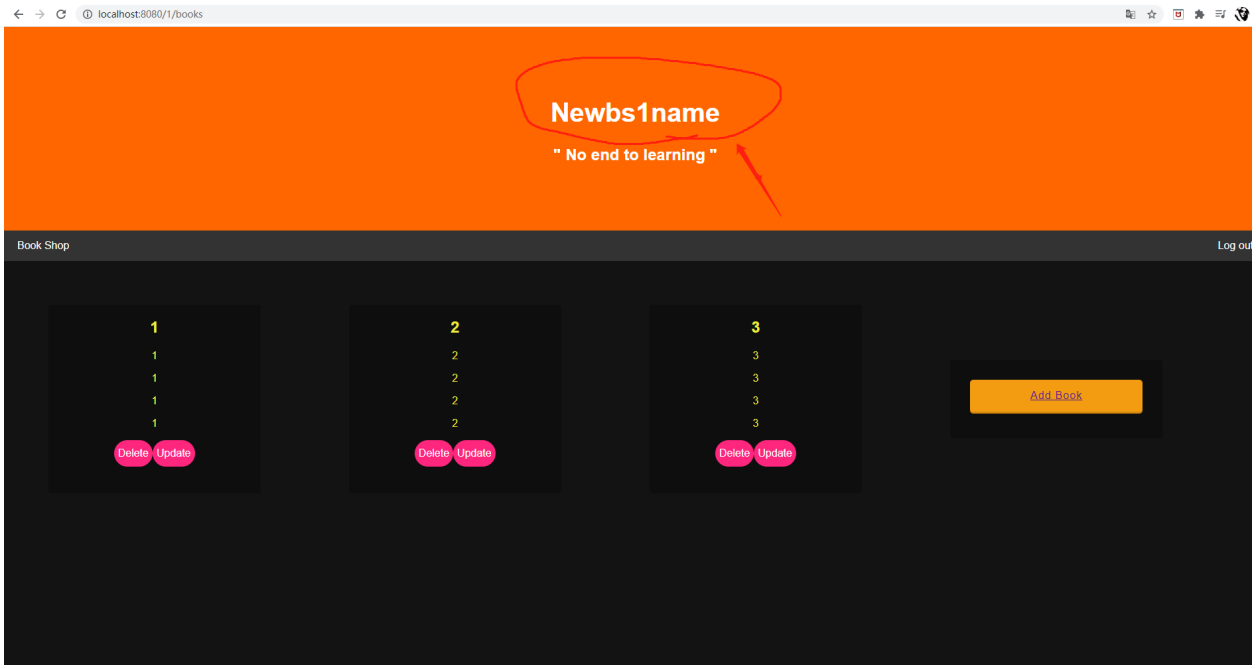
- Choose a bookshop and enter its book page.



- The title of the book page is the same as the name of the bookshop.

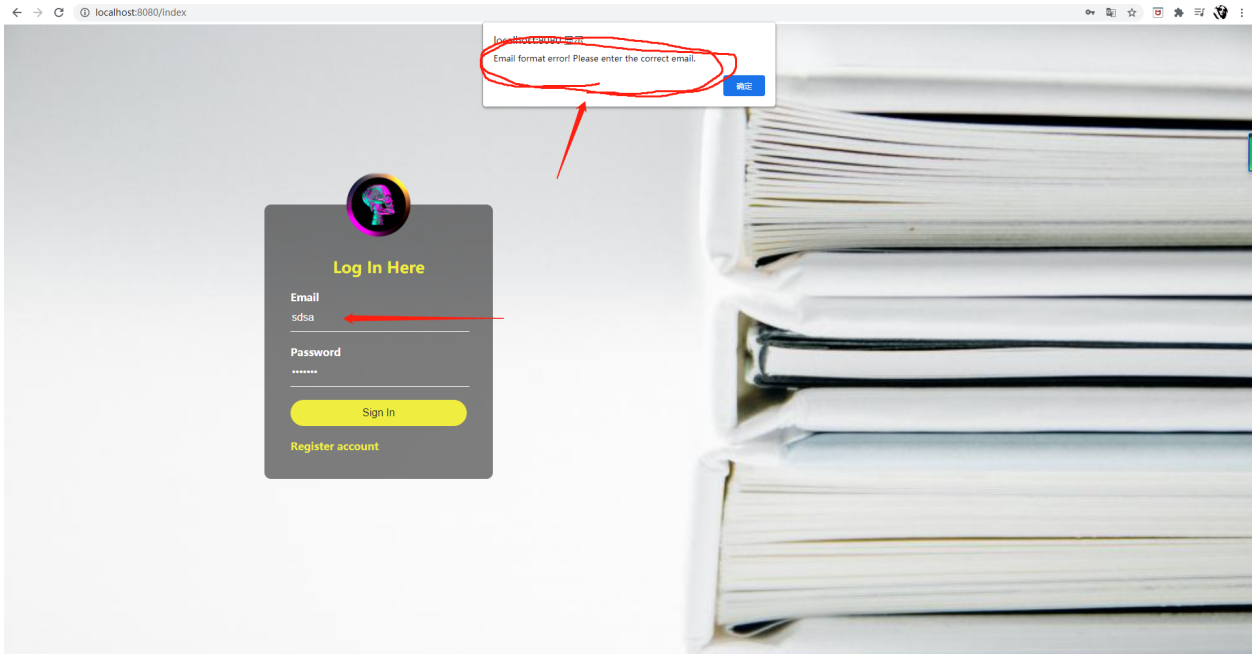


- Go back to the book shop page, update the name of this bookshop into “Newbs1name”.

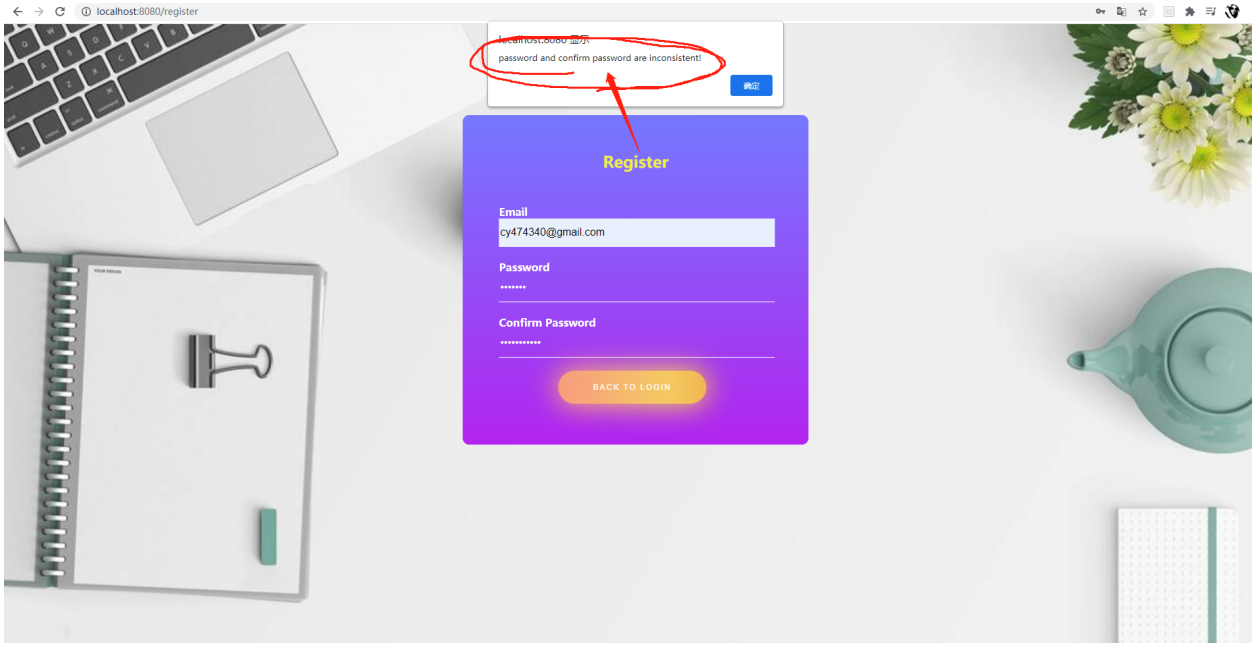


- The bookshop name in the book page has also been updated

4. In the login and registration pages, the built-in form validation and JS validation are used.



- email format is incorrect



- The passwords entered two times are inconsistent.