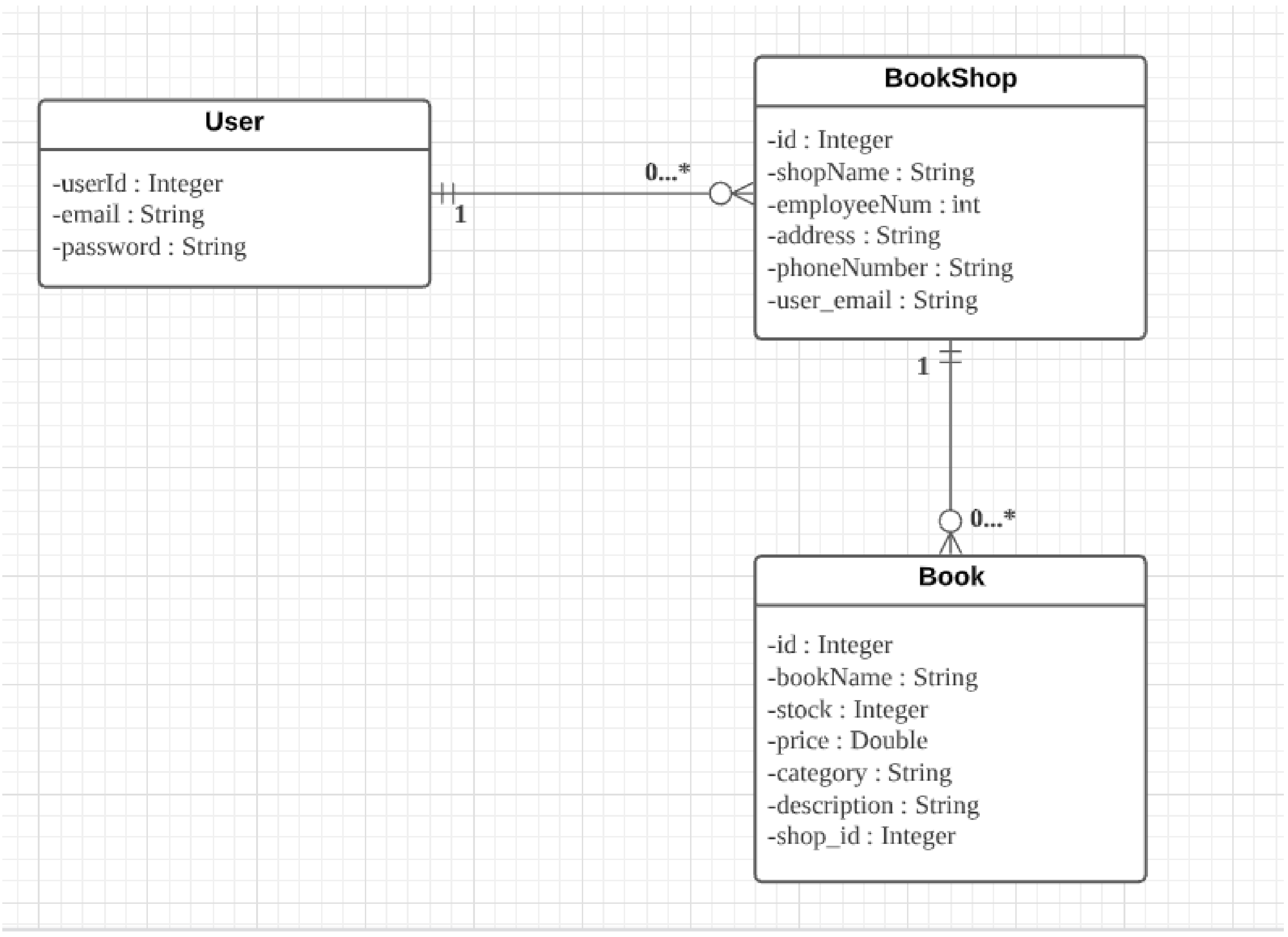
API description

Please see the template tables (for each verb) that you can use to create the API specification that fits the given API (posted on Blackboard). To see an example for each Verb, see the result for the Server-side homework week 1.

Instead of this document and templates, you can use other tools to create the API specification as well. An example of such a tool is the Swagger Editor (https://editor.swagger.io/).

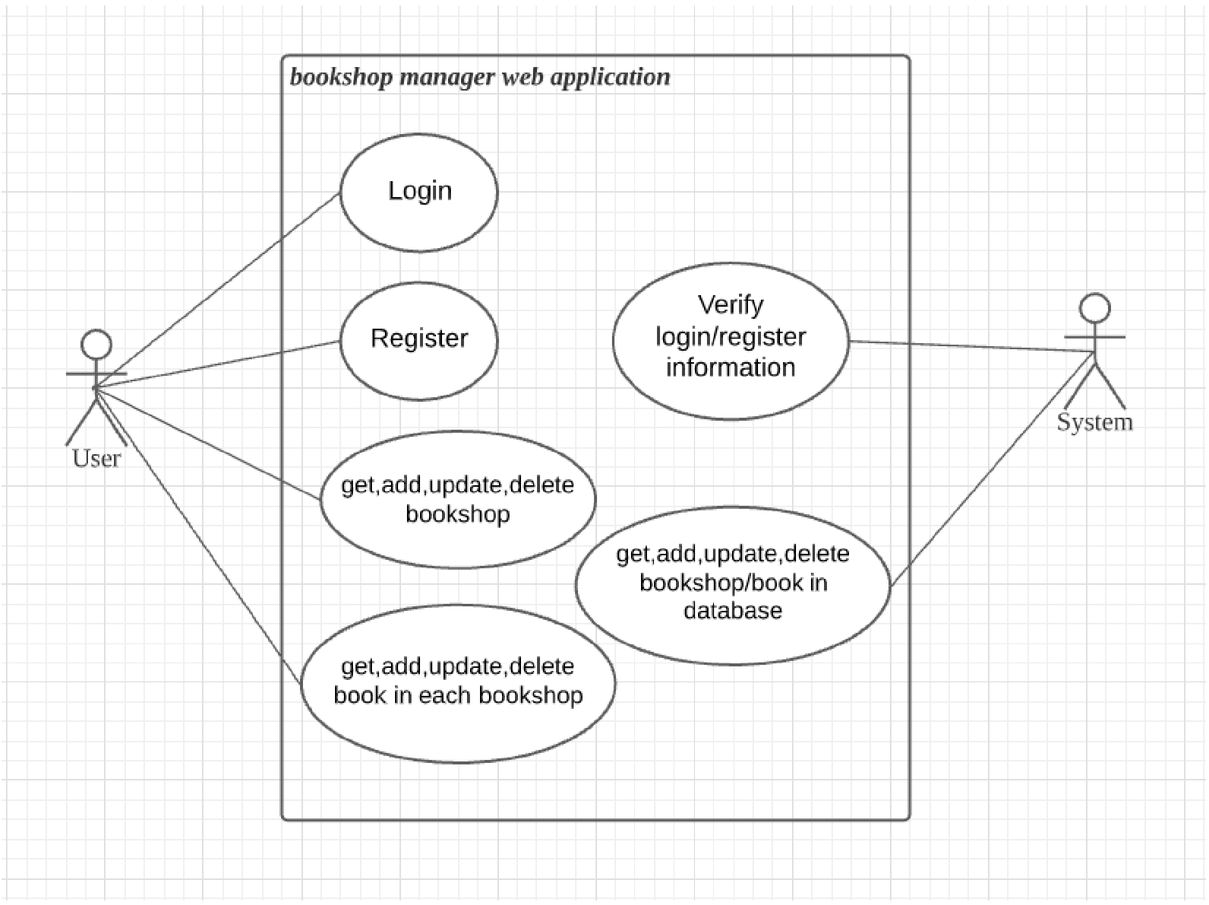
Table of Contents

1. Class diagram



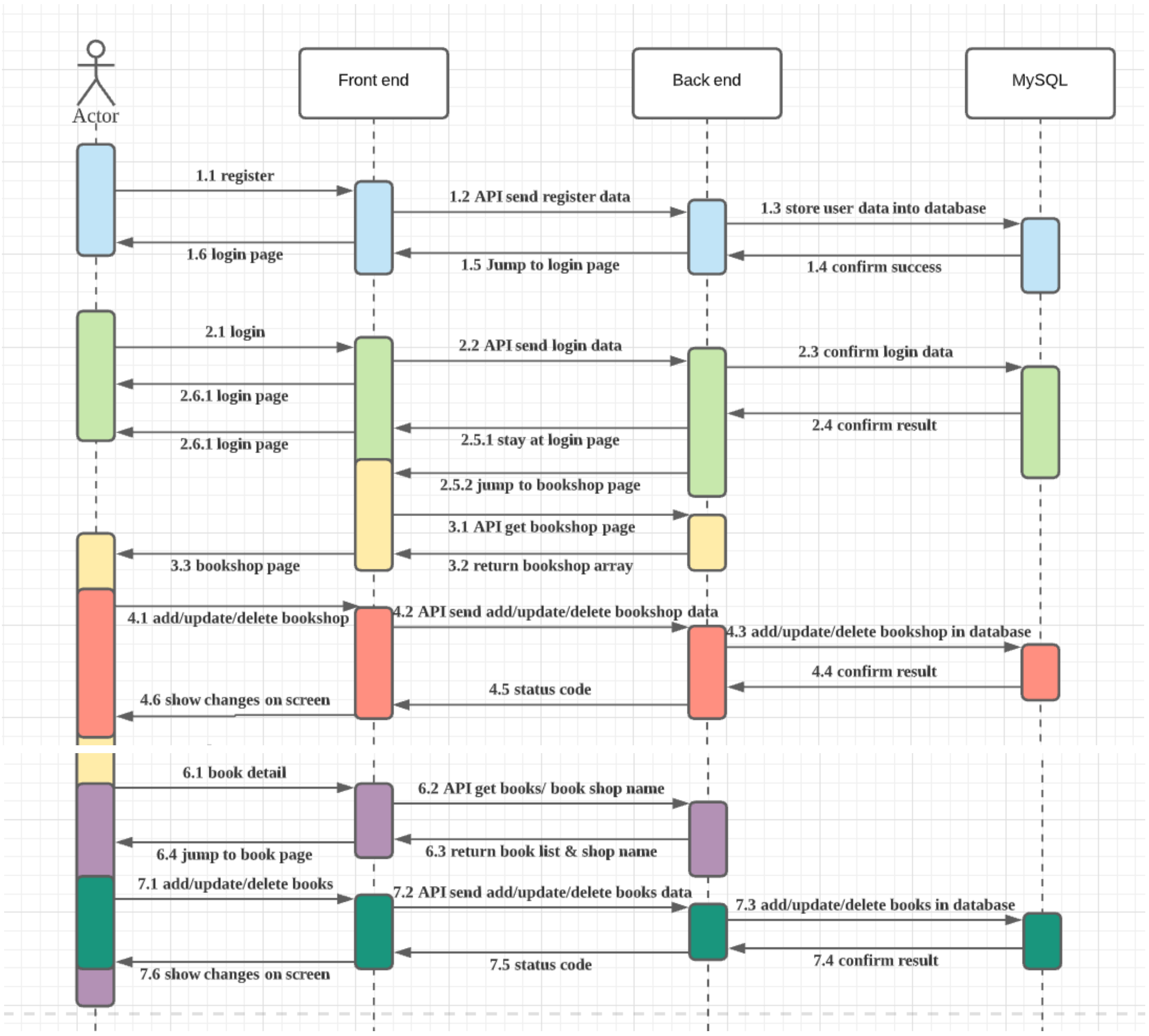
This web application is called "Bookshop Manager". Its purpose is to help bookshop managers manage their book shops. Considering that one manager may have multiple bookshops, the bookshop supports CRUD (Creating, Reading, Updating and Deleting bookshop). What's more, the manager can also manage the books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

2. User diagram

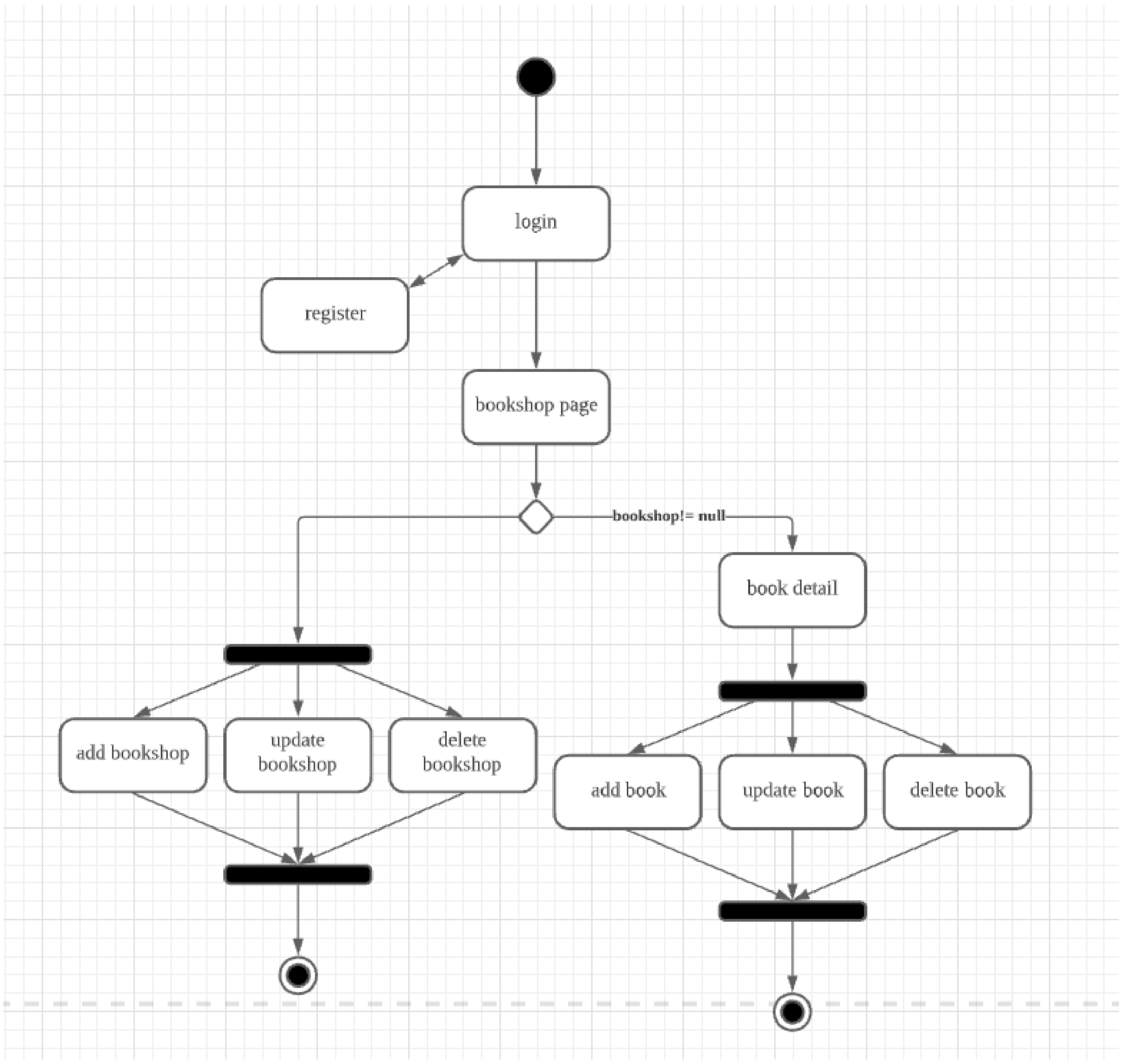

bookshop manager web application
Login
Register
Verify login/register information
get,add,update,delete bookshop
get,add,update,delete bookshop/book in database
get,add,update,delete book in each bookshop
User
System

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

3. Sequence diagram


Actor
Front end
Back end
MySQL
1.1 register
1.2 API send register data
1.3 store user data into database
1.4 confirm success
1.5 Jump to login page
1.6 login page
2.1 login
2.2 API send login data
2.3 confirm login data
2.4 confirm result
2.5.1 stay at login page
2.5.2 jump to bookshop page
2.6.1 login page
2.6.1 login page
3.1 API get bookshop page
3.2 return bookshop array
3.3 bookshop page
4.1 add/update/delete bookshop
4.2 API send add/update/delete bookshop data
4.3 add/update/delete bookshop in database
4.4 confirm result
4.5 status code
4.6 show changes on screen
6.1 book detail
6.2 API get books/ book shop name
6.3 return book list & shop name
6.4 jump to book page
7.1 add/update/delete books
7.2 API send add/update/delete books data
7.3 add/update/delete books in database
7.4 confirm result
7.5 status code
7.6 show changes on screen

4. Activity diagram



5. GET requests

Add your requests here. Copy-paste the template for each different request.

GET	**/index**		
To the login page			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*			
Responses:	**Code**	**Description / example if successful**	
	return "index";	Jump to login page.	

GET	**/register**		
To the register page			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*			
Responses:	**Code**	**Description / example if successful**	
	return "Register";	Jump to register page	

GET	**/api/getBooksShops**		
Get bookshops list from backend.			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*			

Responses:	**Code**	**Description / example if successful**
	return showUserShop;	If successful, return a bookshops list.

GET	**/bookShops**		
Jump bookshops page.			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*			
Responses:	**Code**	**Description / example if successful**	
	return “BookShop”;	Jump to bookshop page	

GET	**/bookShops/update/{id}**		
Jump update bookshops page.			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	id*	path	Which bookshop need to be update
Responses:	**Code**	**Description / example if successful**	
	return "UpdateBookShop";	Jump to update book shop page.	

GET	**/addBookShop**		
Jump add bookshops page.			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	id*	path	Which bookshop need to be update
Responses:	**Code**	**Description / example if successful**	
	return "AddBookShop";	Jump to add book shop page.	

GET	**{id}/books**		
Jump to book page			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	id*	path	Which bookshop's book page, this id is bookshop id.
Responses:	**Code**	**Description / example if successful**	
	return "Book";	Jump to that bookshop's book page.	

GET	**/books/getShopName**		
Get bookshop name ,show it on book page title.			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*			
Responses:	**Code**	**Description / example if successful**	
	return shopName+";"+shopId;	Return the bookshop name +id, which corresponding to this book page.	

GET	/addBook		
Jump to add book page			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*			
Responses:	**Code**	**Description / example if successful**	
	return "AddBook";	Jump to add book page	

GET	/api/getBooks		
Get all the books in one bookshop			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*			
Responses:	**Code**	**Description / example if successful**	
	return showBooks	Return a list of book	

GET	/update/{id}/updateBooks		
Jump update book page.			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	id*	path	Which book need to be update
Responses:	**Code**	**Description / example if successful**	
	return "UpdateBook";	Jump to update book page.	

6. POST requests

Add your requests here. Copy-paste the template for each different request.

Template POST table:

POST	**/index/confirm**		
After clicking the login button, confirm the user login data with data in database.			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	Email*	body	User login email data
	Password*	body	User login password data
Responses:	**Code**	**Description / example if successful**	
	return “redirect:/bookShops”;	If login success, the login data is consistent with that in the database, jump to bookshop page.	
	Return “redirect:/index”;	If login failed, stay at login page.	

POST	**/register/add**		
After clicking the register button, get the user’s registration data			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	Email*	body	User register email data
	password*	body	User register password data
Responses:	**Code**	**Description / example if successful**	
	return “index”;	If register success ,return login page.	
	Return “Register”;	If register failed, stay at register page.	

POST	/api/addBookshop		
Add a bookshop			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	shopName*	body	BookShop name
	employeeNum*	body	BookShop employee number
	address*	body	Bookshop address
	phoneNumber*	body	PhoneNumber
Responses:	**Code**	**Description / example if successful**	
	return "200";	If add success, return status code: 200	
	return "500";	If add failed, return status code: 500	

POST	/api/addBook		
Add a book			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	bookName*	body	Book name
	stock*	body	Book stock, location
	price*	body	Book price
	category*	body	Book category
	description*	body	Book description
Responses:	**Code**	**Description / example if successful**	
	Return shop_id	If success , return the id of the bookshop where this book is located	
	Return 500	If add failed, return status code: 500	

7. PUT requests

Add your requests here. Copy-paste the template for each different request.

Template PUT table:

PUT	/api/updateBookshop		
Update the bookshop information.			
Parameters:	**Name**	**Type**	**Description**
Add a * to the name of required parameters.	shopName*	body	new bookShop name
	employeeNum*	body	new bookShop employee number
	address*	body	new bookshop address
	phoneNumber*	body	new phoneNumber
Responses:	**Code**	**Description / example if successful**	
	return "200";	If update success, return status code: 200	
	return "406";	If update failed, return status code: 406	

PUT	/api/updateBook		
Update the book information.			
Parameters:	**Name**	**Type**	**Description**
Add a * to the name of required parameters.	bookName*	body	New Book name
	stock*	body	New Book stock, location
	price*	body	New Book price
	category*	body	New Book category
	description*	body	New Book description
Responses:	**Code**	**Description / example if successful**	
	Return shop_id	If success , return the id of the bookshop where this book is located	
	Return 406	If update failed, return status code: 406	

8. DELETE requests

Add your requests here. Copy-paste the template for each different request.

Template PUT table:

DELETE	/api/deleteBookshop/{id}		
Delete one bookshop			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	id*	path	The bookshop id, which need to be deleted.
Responses:	**Code**	**Description / example if successful**	
	return "200";	If delete success, return status code: 200	
	return "406";	If delete failed, return status code: 406	

DELETE	/api/deleteBook/{id}		
Delete one book			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	id*	path	The book id, which need to be deleted.
Responses:	**Code**	**Description / example if successful**	
	return "200";	If delete success, return status code: 200	
	return "406";	If delete failed, return status code: 406	