**Book Q**

**Final Report**

Date : 2016/06/10

20101011 Kang Hyeong Min

20121399 Lee Seung Hwan

20131169 Kim Hyeon Yeong

**Index**

**Functional spec**

1. Description of the application

2. Top-level information flow diagram: High level overall workflow (List of tasks)

3. Task GUI : user interfaces for major tasks/features

**Design specification**

1. E-R diagram

2. Explanations of the non-obvious entities and relationships

3. SQL DDL statements to create the relational schema (including constraints)

4. Algorithms

5. Platforms

**Functional spec**

1. **A description of the application**

‘Book.Q’ is Book curation service based on analyzing personal taste. (Like ‘Watcha’ <https://watcha.net/>) Let users grade books which they have read, and collect that data(user’s data, and book’s data). Therefore this service provides book recommendation.

Users can search books with keyword and user can be provided recommended book lists with themes.

1. **Top-level information flow diagram: High level overall workflow (List of tasks)**

Figure Top-level information flow diagram

**User**

**Book**

**Read**

**Input user data**

**Grading**

**Recommend**

**Books**

**Find correlation**

**Find correlation**

1. **Task GUI : user interfaces for major tasks/features**

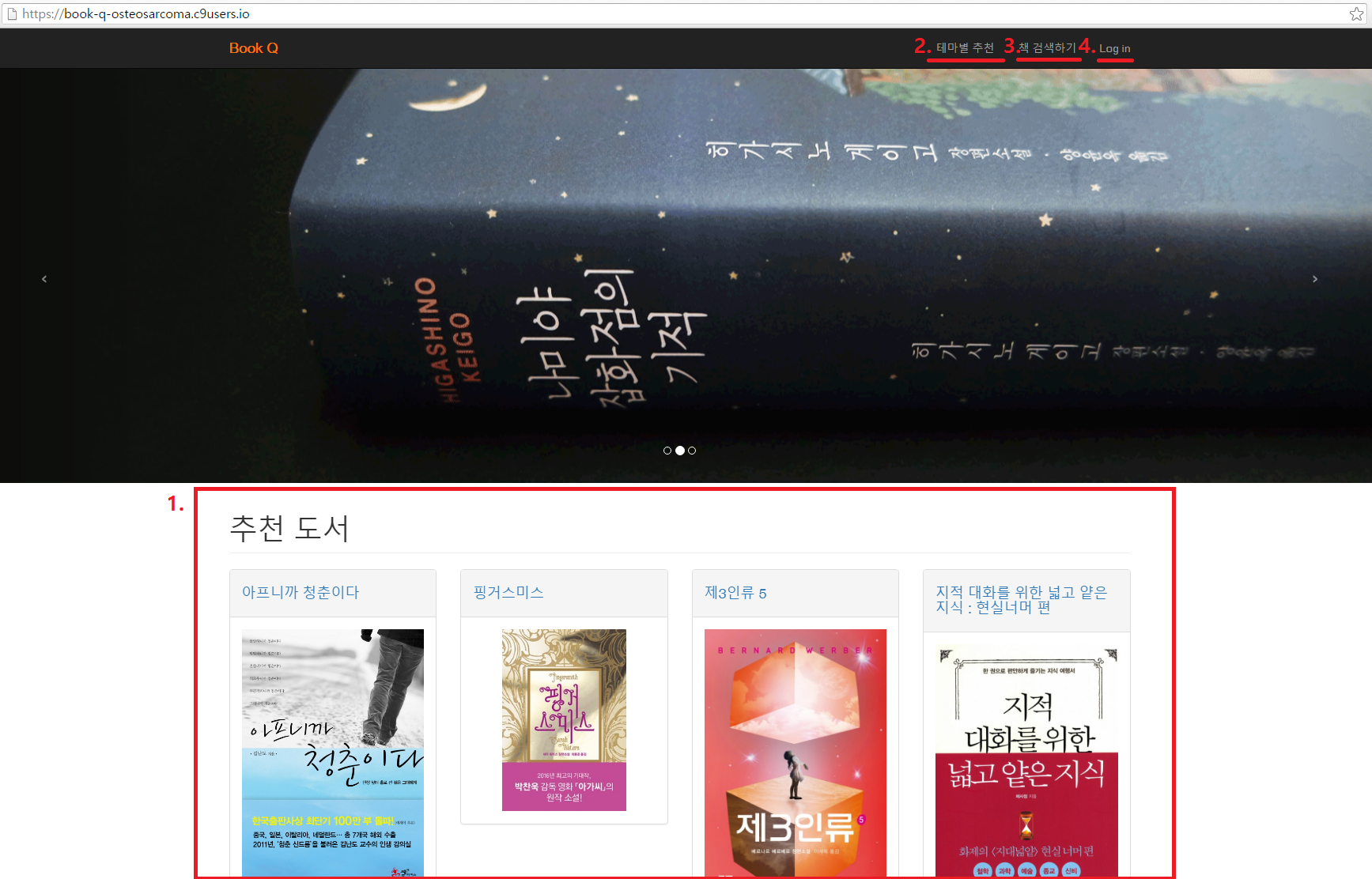


Figure UI - Before login, index page

1. List of recommended books
2. Find recommended books with theme
3. Searching book
4. Log in

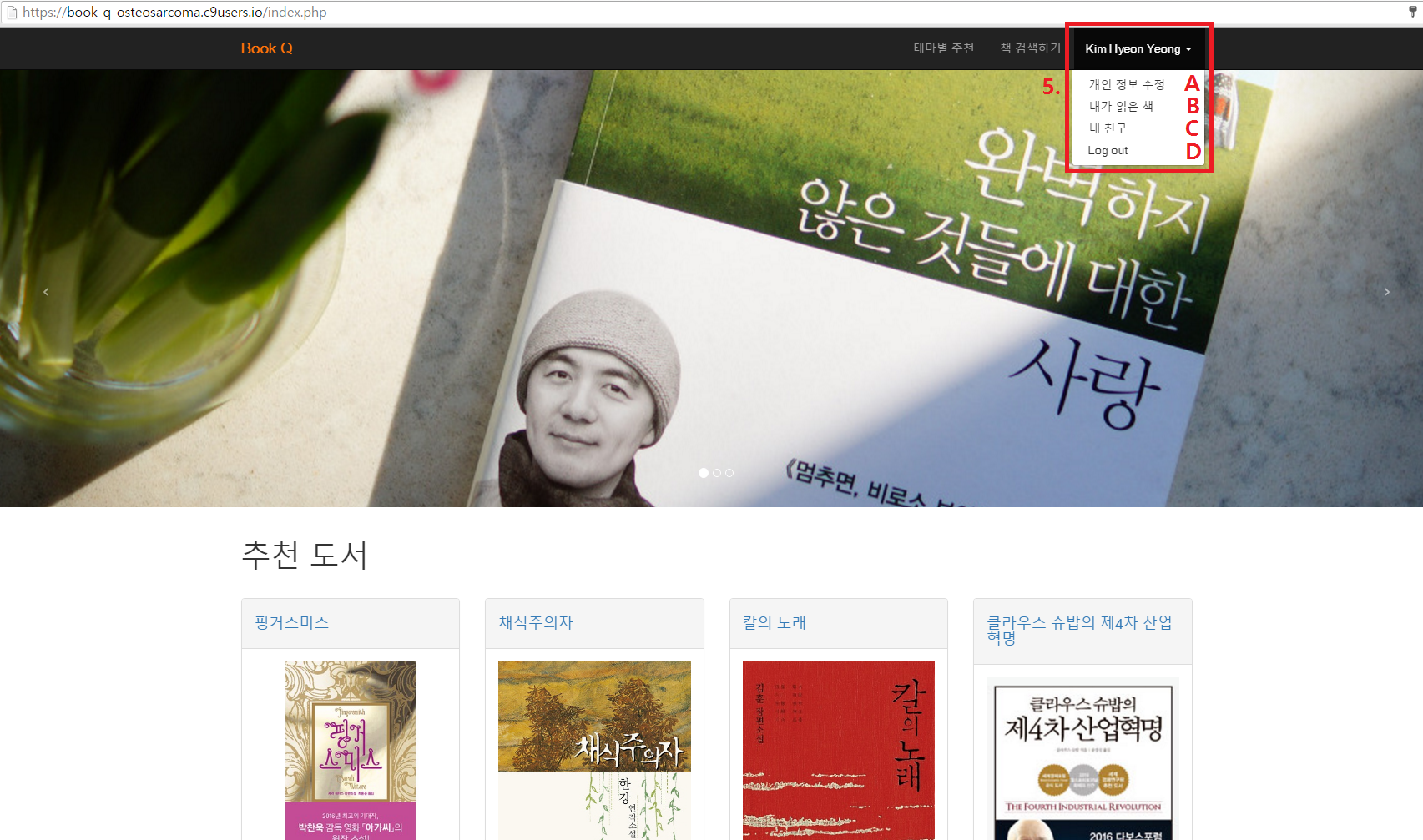


Figure UI - After login, index page

1. My page : 4 sub-menu
   * + - 1. Edit personal information (unimplemented)
         2. My read book list
         3. My friends (unimplemented)
         4. Log out

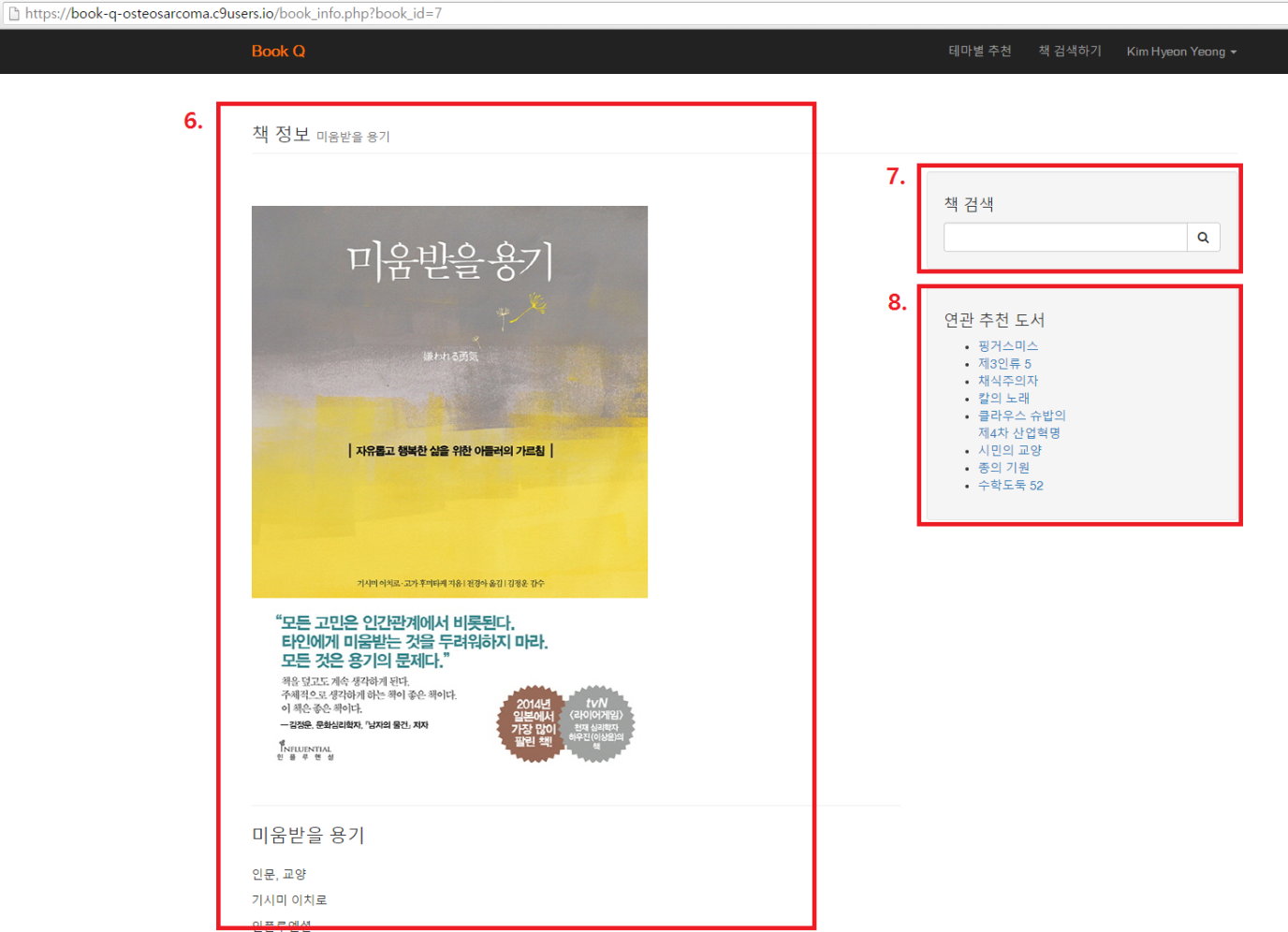


Figure UI - book\_info page

1. Book information
2. Searching book
3. Related recommended books

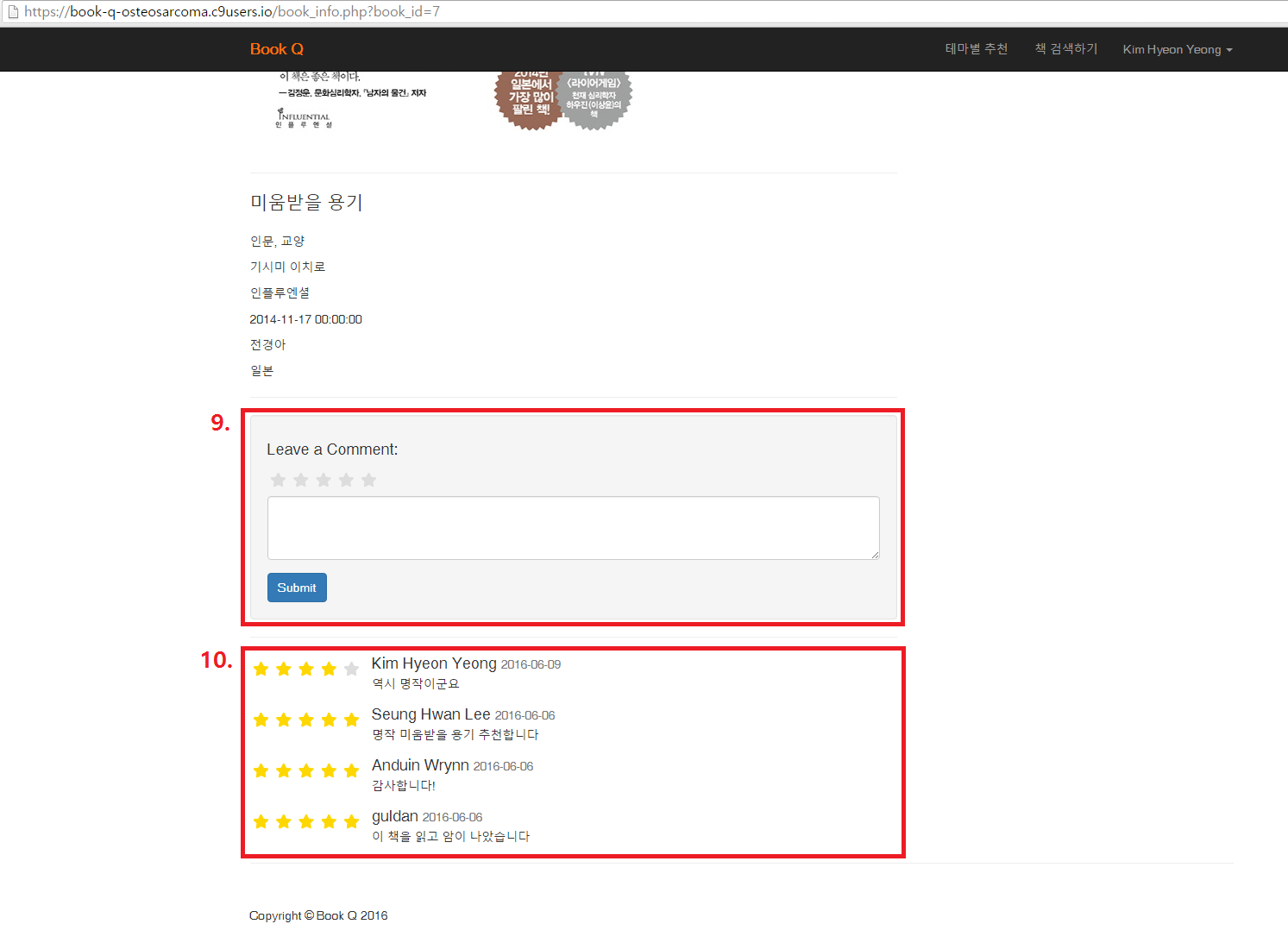


Figure UI - Comments in book\_info page

1. Leave a comment with rating

10. Comments list

**Design specification**

1. **E-R diagram**

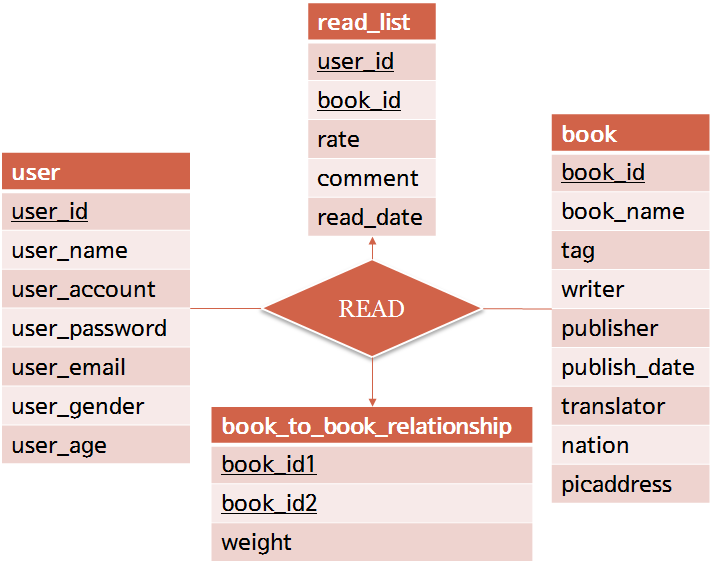


Figure 6 E-R diagram

1. **Explanations of the non-obvious entities and relationships**

user : Store the information about the user.

book : Store the information about the book.

read\_list : The data which are user made commentary and grade about book.

book\_to\_book\_relationship : Store the weight about a relationship between two books.

1. **SQL DDL statements to create the relational schema (including constraints)**

create table user(

user\_id varchar(10) primary key,

user\_name varchar(20) not null,

user\_account varchar(20) not null,

user\_password varchar(20) not null,

user\_email varchar(20),

user\_gender varchar(6),

user\_age int

);

create table book(

book\_id varchar(10) primary key,

book\_name varchar(40) not null,

tag varchar(100),

writer varchar(10),

publisher varchar(20),

publish\_date TIMESTAMP not null,,

translator varchar(10),

nation varchar(10),

picaddress TEXT

);

create table book\_to\_book\_relationship{

Book\_id1 varchar(10) not null,

Book\_id2 varchar(10) not null,

Weight float,

}

create table read\_list(

user\_id varchar(10),

book\_id varchar(10),

rate float(2,1),

comment TEXT,

read\_date DATE not null

);

1. **Algorithms**

Recommendation

Recommendation algorithm is constructed by two sub-algorithm. First is updating the weights; relation of book to book and second is multiplying the weights with users book rating and sorting.

Every books have each relationship and it shows in weight attribute of ‘book\_to\_book\_relationship’ table. The weights are updated when some user read the books. For example, user\_1 read the book with id number 42 with 4.5 rating, our program work as below.

j = 42;  
given\_rate = 4.5;  
threshold = 2.5;

for every book\_(i) were read by user\_1  
 book\_to\_book\_relationship( i , j ). weight +=  
 ( user\_1.rate.book\_( i ) - threshold )\*( given\_rate - threshold );

After weight updating, We make one temporary array that store individual recommending weights. For the same example as shown above, our program work as below.

Array recommand\_weight;  
threshold = 2.5;

for every book\_(i) were not read by user\_1  
 for every book\_( j ) were read by user\_1  
 recommand\_weight[ i ] +=   
 book\_to\_book\_relationship.weight( i , j ) \*  
 ( user\_1.rate.book\_( j ) - threshold);

After then, our program sort the array by decreasing order and display with certain form on our website.

These algorithms are implemented in index.php[184:267] of project source code.

1. **platforms**

Front-end : Html/CSS, using framework, bootstrap.

Back-end : php, Mysql

IDE : C9

Version management tool : Github

**you can access in** [**https://ide.c9.io/osteosarcoma/book-q**](https://ide.c9.io/osteosarcoma/book-q) **with an account  
Temp ID & Password :** [**vit424@unist.ac.kr**](mailto:vit424@unist.ac.kr) **/ asdfasdf**

**Click ‘Run Project’ button in** [**https://ide.c9.io/osteosarcoma/book-q**](https://ide.c9.io/osteosarcoma/book-q)

**And then, you can check working project in** [**https://book-q-osteosarcoma.c9users.io/**](https://book-q-osteosarcoma.c9users.io/)