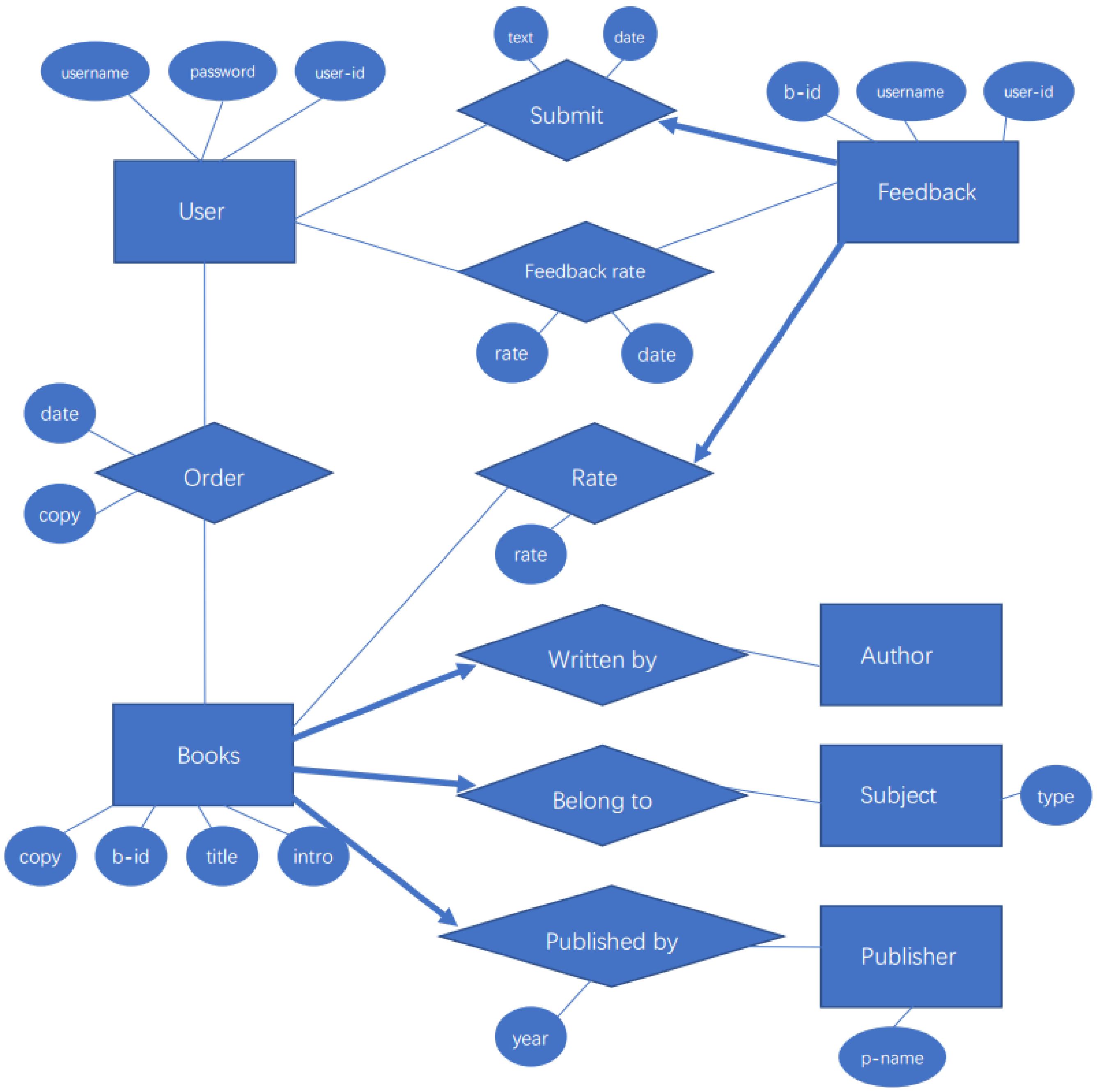
50.008 Database Project Report

Du Haoran 1000915

Xie Qiaoyi 1000923

1. ER Model

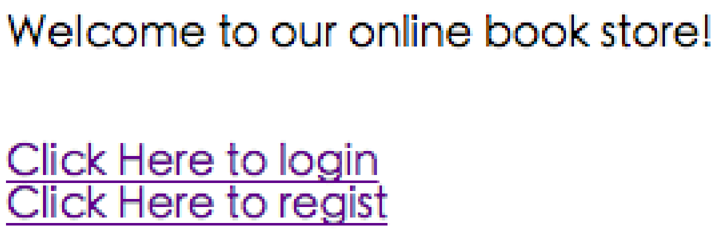


2. Code

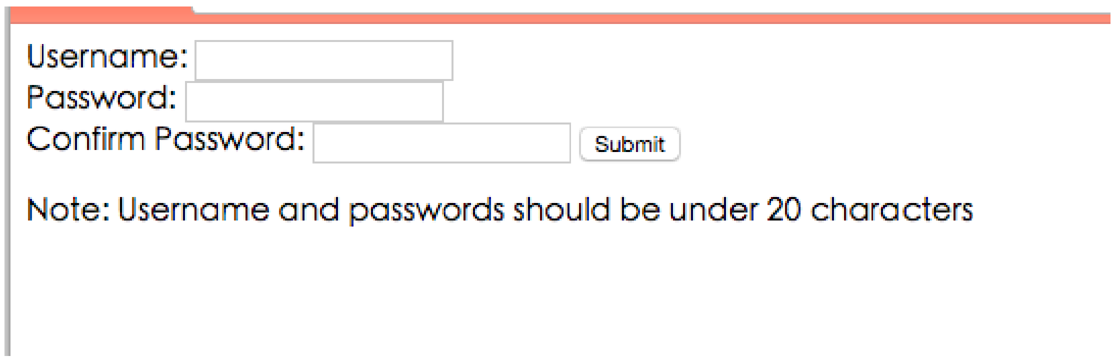
```
class TableCreation():
    with connection.cursor() as cursor:
        cursor.execute('''CREATE Table Authors(
        name CHAR(64) PRIMARY KEY
        )''')
        cursor.execute('''CREATE Table Publishers(
        name CHAR(64) PRIMARY KEY
        )''')
        cursor.execute('''CREATE Table Subjects(
        name CHAR(64) PRIMARY KEY
        )''')
        cursor.execute('''CREATE Table Books(
        id INT(8) PRIMARY KEY,
        title CHAR(128),
        intro CHAR(512),
        copy INT(4),
        author CHAR NOT NULL,
        publisher CHAR NOT NULL,
        publishyear INT(4),
        subject CHAR NOT NULL,
        FOREIGN KEY (author) REFERENCES Authors(name),
        FOREIGN KEY (publisher) REFERENCES Publishers(name),
        FOREIGN KEY (subject) REFERENCES Subjects(name)
        )''')
        cursor.execute('''CREATE TABLE Feedbacks(
        username CHAR NOT NULL,
        bookid INT NOT NULL,
        booktitle CHAR NOT NULL,
        text CHAR(512),
        rate INT(1),
        date DATE,
        PRIMARY KEY (username,bookid),
        FOREIGN KEY (uername) REFERENCES auth_user(username),
        FOREIGN KEY (bookid) REFERENCES Book(id),
        FOREIGN KEY (booktitle) REFERENCES Book(title).
        ON DELETE CASCADE
        )''')
        cursor.execute('''CREATE TABLE Feedbackrates(
        username CHAR NOT NULL,
        feedback_username CHAR NOT NULL,
        feedback_bookid INT NOT NULL,
        rate INT(1),
        date DATE
        PRIMARY KEY (username,feedback_username,feedback_bookid),
        FOREIGN KEY (uername) REFERENCES auth_user(username),
        FOREIGN KEY (feedback_username) REFERENCES auth_user(username),
        FOREIGN KEY (feedback_bookid) REFERENCES Book(id),
        ON DELETE CASCADE
        )''')
```

3. Requirements

1. **Registration: a new user has to provide necessary information; he/she can pick a login-name and a password. The login name should be checked for uniqueness. Use Django's auth mode and session DB module for this.**


Welcome to our online book store!

Click Here to login
Click Here to regist

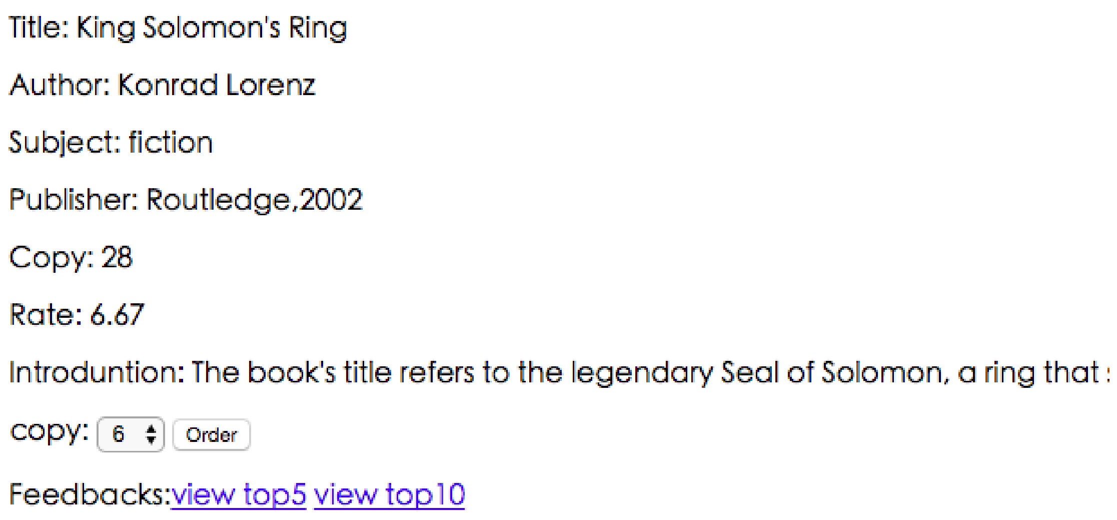

Username:
Password:
Confirm Password:

Note: Username and passwords should be under 20 characters

CODE:

```
user = User.objects.create_user(username,"",password)
## ROW SQL ##
#with connection.cursor() as cursor:
#        cursor.execute("INSERT        INTO        auth_user        (password,username)
Values('%s','%s')"%(password,username))
```

2. **Ordering: After registration, a user can order one or more books. A user may order multiple copies of a book, one or more times. (The charging of the credit card and the shipment of the books are outside the scope of this project).**



Title: King Solomon's Ring

Author: Konrad Lorenz

Subject: fiction

Publisher: Routledge,2002

Copy: 28

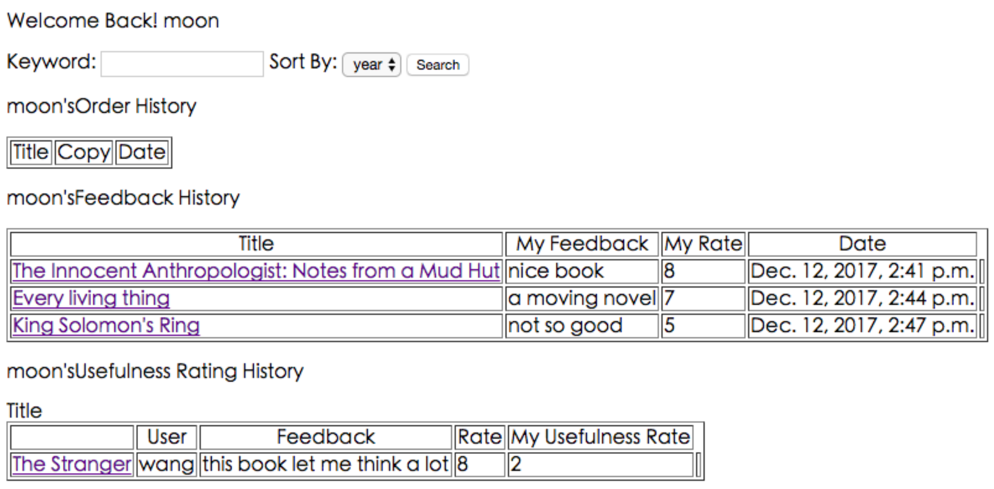
Rate: 6.67

Introduntion: The book's title refers to the legendary Seal of Solomon, a ring that

CODE:

```
with connection.cursor() as cursor:

    cursor.execute("UPDATE BookStore_Books SET copy = copy - %s WHERE id =
    '%s'"%(copy,b_id))

b = Books.objects.raw("SELECT * from BookStore_Books WHERE id = '%s'"%(b_id))[0]

Orders.objects.create(user=request.user,book=b,copy=copy)

## ROW SQL ##
#with connection.cursor() as cursor:

#cursor.execute("INSERT INTO BookStore_Orders (user_id,book_id,copy) VALUES
(%s,%s,%s)"%(request.user.id,b_id,copy))
```

3. **User record: upon user demand, you should print the full record of a user: • his/her account information • his/her full history of orders (book name, number of copies, date etc.) • his/her full history of feedbacks • the list of all the feedbacks he/she ranked with respect to usefulness**



Welcome Back! moon

Keyword: Sort By: year Search

moon'sOrder History

Title	Copy	Date

moon'sFeedback History

Title	My Feedback	My Rate	Date
The Innocent Anthropologist: Notes from a Mud Hut	nice book	8	Dec. 12, 2017, 2:41 p.m.
Every living thing	a moving novel	7	Dec. 12, 2017, 2:44 p.m.
King Solomon's Ring	not so good	5	Dec. 12, 2017, 2:47 p.m.

moon'sUsefulness Rating History

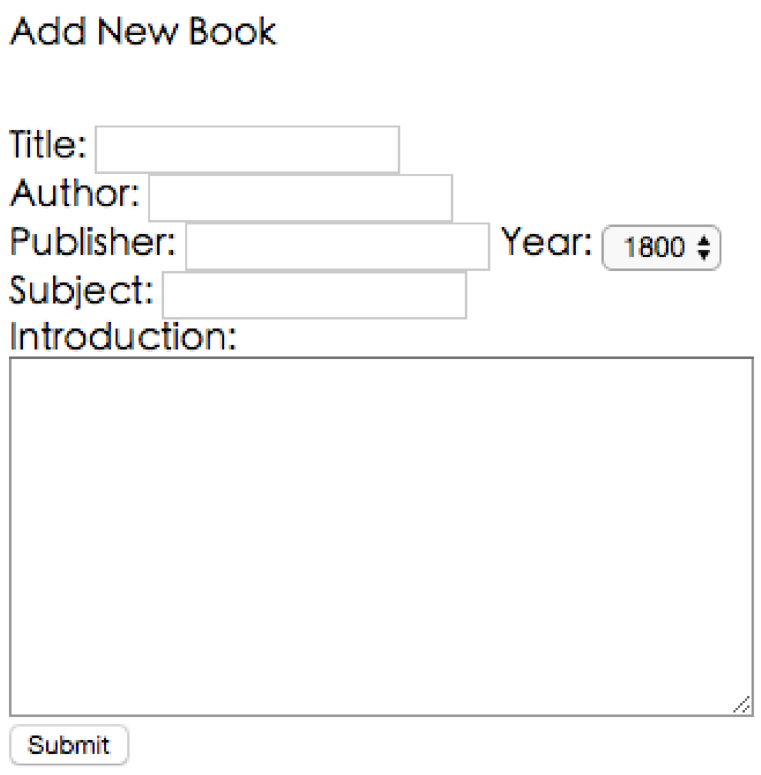
Title

	User	Feedback	Rate	My Usefulness Rate
The Stranger	wang	this book let me think a lot	8	2

CODE:

```
        orderlist = Orders.objects.raw("SELECT * from BookStore_Orders WHERE
user_id = '%s'"%(request.user.id))

        feedbacklist = Feedbacks.objects.raw("SELECT * from BookStore_Feedbacks
WHERE user_id = '%s'"%(request.user.id))

        feedbackratelist = Feedbackrates.objects.raw("SELECT * from
BookStore_Feedbackrates WHERE user_id = '%s'"%(request.user.id))
```

4. New book: The store manager records the details of a new book, along with the number of new books that have arrived in the warehouse.

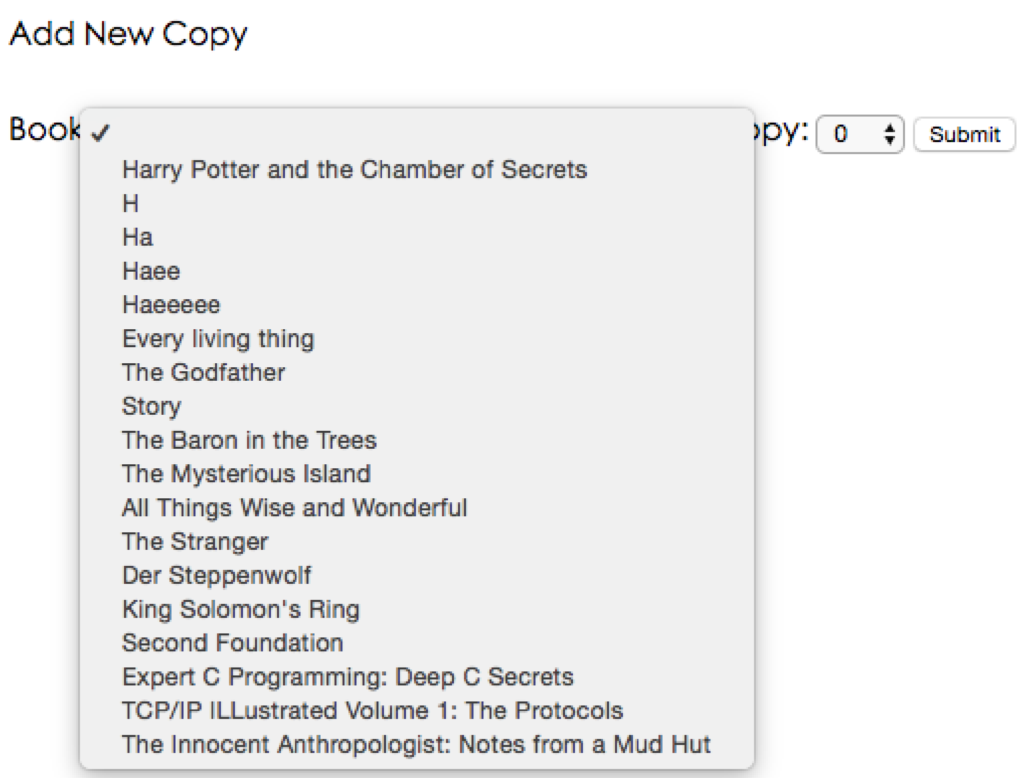

Add New Book

Title:
Author:
Publisher: Year: 1800
Subject:
Introduction:

CODE:

```
Books.objects.create(title=title,author=author,intro=intro,publisher=publisher,year=int(year),subject=subject)

## ROW SQL ##
#with connection.cursor() as cursor:

#	cursor.execute("INSERT INTO BookStore_Books VALUES ('%s','%s','%s',0,'%s',%s,'%s')",[title,author,intro,publisher,year,subject])
```

5. **Arrival of more copies: The store manager increases the number of copies in inventory.**



Add New Copy

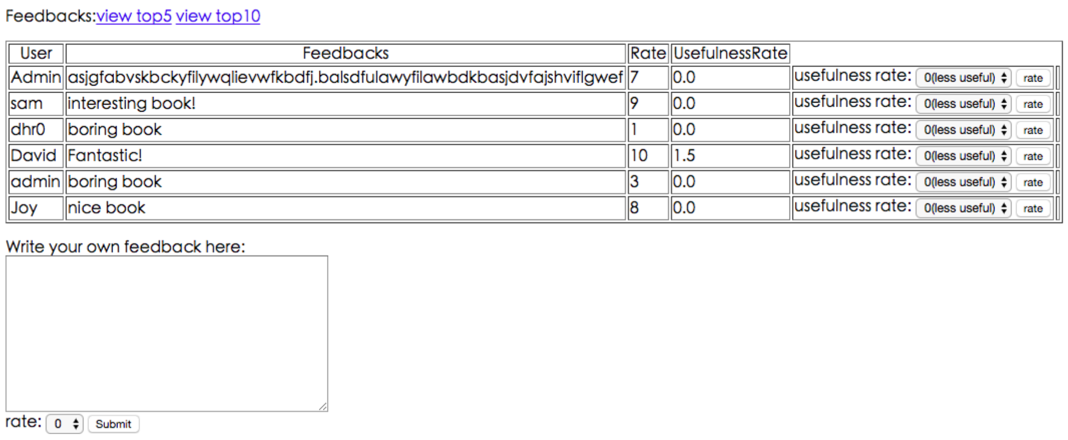
Book ✓ ...py: 0 Submit

Harry Potter and the Chamber of Secrets
H
Ha
Haee
Haeeeee
Every living thing
The Godfather
Story
The Baron in the Trees
The Mysterious Island
All Things Wise and Wonderful
The Stranger
Der Steppenwolf
King Solomon's Ring
Second Foundation
Expert C Programming: Deep C Secrets
TCP/IP ILLustrated Volume 1: The Protocols
The Innocent Anthropologist: Notes from a Mud Hut

CODE:

```
with connection.cursor() as cursor:

        cursor.execute("UPDATE BookStore_Books SET copy = copy + %s WHERE id = '%s'"%(copy,b_id))
```

6. **Feedback recordings:**



Feedbacks:view top5 view top10

User	Feedbacks	Rate	UsefulnessRate	
Admin	asjgfabvskbckyfilywqlievwfkbdfj.balsdfulawyfilawbdkbasjdvfajshviflgwef	7	0.0	usefulness rate: 0(less useful) rate
sam	interesting book!	9	0.0	usefulness rate: 0(less useful) rate
dhr0	boring book	1	0.0	usefulness rate: 0(less useful) rate
David	Fantastic!	10	1.5	usefulness rate: 0(less useful) rate
admin	boring book	3	0.0	usefulness rate: 0(less useful) rate
Joy	nice book	8	0.0	usefulness rate: 0(less useful) rate

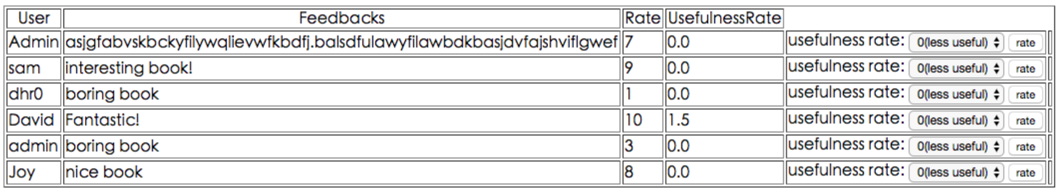
Write your own feedback here:

rate: 0 Submit

CODE:

```
    count = 0

    for feedback in Feedbacks.objects.raw("SELECT * from BookStore_Feedbacks
WHERE book_id=%s AND user_id=%s"%(b_id,request.user.id)):

        count += 1

    if count:

        return HttpResponse("You have already rated for this book")

    b = Books.objects.raw("SELECT * from BookStore_Books WHERE id =
'%s'"%(b_id))[0]

    Feedbacks.objects.create(user=request.user,book=b,text=text,rate=rate)

   ## ROW SQL ##
    #with connection.cursor() as cursor:

    #cursor.execute("INSERT INTO BookStore_Feedbacks
(user_id,book_id,text,rate) VALUES
(%s,%s,'%s',%s)"%(request.user.id,b_id,text,rate))
```

7. Usefulness ratings: Users can assess other uses feedback, give a numerical score 0, 1, or 2 (useless, useful, very useful respectively). A user is not allowed to rate his/her own feedback.

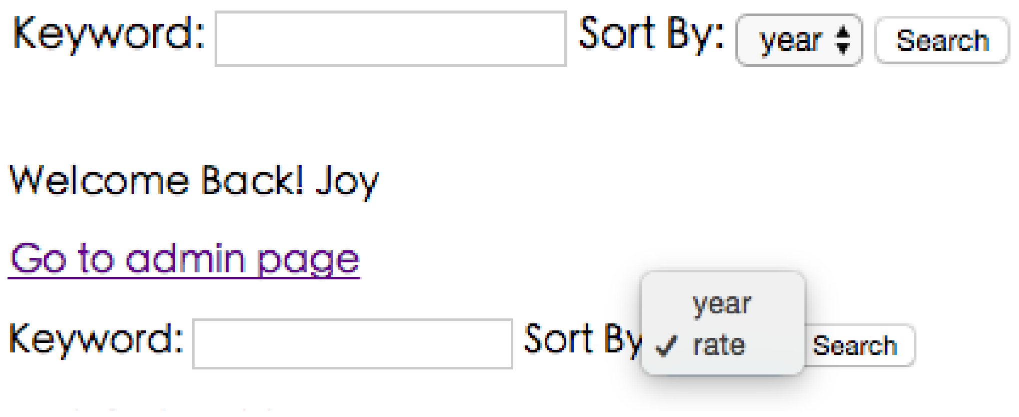


User	Feedbacks	Rate	UsefulnessRate	
Admin	asjgfabvskbckyfilywqlievwfkbdfj.balsdfulawyfilawbdkbasjdvfajshviflgwef	7	0.0	usefulness rate: 0(less useful) rate
sam	interesting book!	9	0.0	usefulness rate: 0(less useful) rate
dhr0	boring book	1	0.0	usefulness rate: 0(less useful) rate
David	Fantastic!	10	1.5	usefulness rate: 0(less useful) rate
admin	boring book	3	0.0	usefulness rate: 0(less useful) rate
Joy	nice book	8	0.0	usefulness rate: 0(less useful) rate

CODE:

```
    f = Feedbacks.objects.raw("SELECT * from BookStore_Feedbacks WHERE id =
'%s'"%(f_id))[0]

    if f.user.id==request.user.id:

        return HttpResponse("You cannot rate your own feedback")

    count = 0

    for feedbackrate in Feedbackrates.objects.raw("SELECT * from
BookStore_Feedbackrates WHERE user_id=%s AND
feedback_id=%s"%(request.user.id,f_id)):

        count += 1

    if count:

         return HttpResponse("You have already rated for this feedback")

    Feedbackrates.objects.create(user=request.user,rate=rate,feedback=f)

    #with connection.cursor() as cursor:

    #cursor.execute("INSERT INTO BookStore_Feedbackrates
(user_id,rate,feedback_id) VALUES (%s,%s,%s)"%(request.user.id,rate,f.id))
```

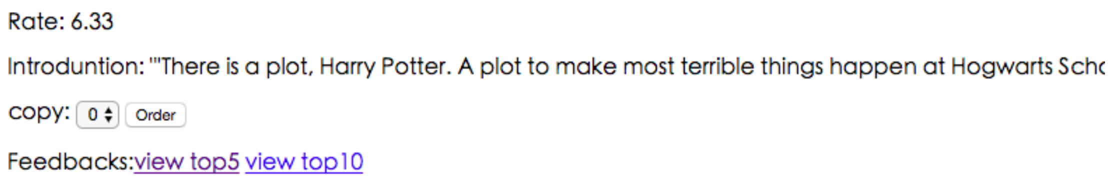
8. Book Browsing: Users may search for books, by asking conjunctive queries on the authors, and/or publisher, and/or title, and/or subject. Your system should allow the user to specify that the results are to be sorted a) by year, or b) by the average score of the feedbacks.



CODE:

```
booklist = Books.objects.raw("SELECT * FROM BookStore_Books WHERE title LIKE '%%%s%%' OR author LIKE '%%%s%%' OR intro LIKE '%%%s%%' OR publisher LIKE '%%%s%%' ORDER BY %s DESC"%(keyword,keyword,keyword,keyword,sort))
```

9. Useful feedbacks: For a given book, a user could ask for the top n most useful feedbacks. The value of n is user-specified (say, 5, or 10). The usefulness of a feedback is its average usefulness score.

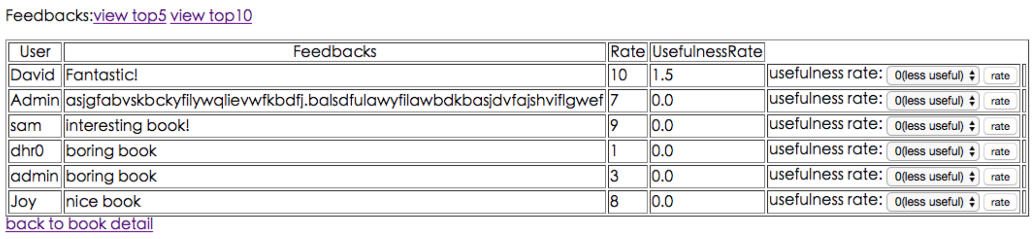


Rate: 6.33

Introduntion: '"There is a plot, Harry Potter. A plot to make most terrible things happen at Hogwarts Schc

copy: 0 Order

Feedbacks:view top5 view top10



Feedbacks:view top5 view top10

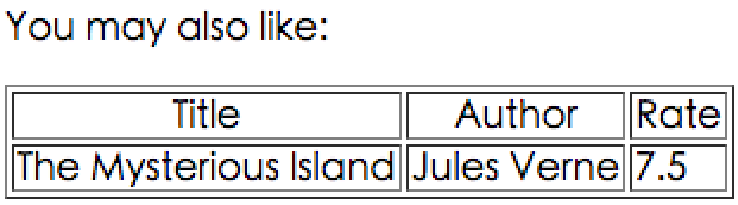
User	Feedbacks	Rate	UsefulnessRate	
David	Fantastic!	10	1.5	usefulness rate: 0(less useful) rate
Admin	asjgfabvskbckyfilywqlievwfkbdfj.balsdfulawyfilawbdkbasjdvfajshviflgwef	7	0.0	usefulness rate: 0(less useful) rate
sam	interesting book!	9	0.0	usefulness rate: 0(less useful) rate
dhr0	boring book	1	0.0	usefulness rate: 0(less useful) rate
admin	boring book	3	0.0	usefulness rate: 0(less useful) rate
Joy	nice book	8	0.0	usefulness rate: 0(less useful) rate

back to book detail

CODE:

```
feedbacklist = Feedbacks.objects.raw("SELECT * FROM BookStore_Feedbacks WHERE book_id=%s ORDER BY avgrate DESC"%(b_id))[0:top]
```

10. Book recommendation: Like most e-commerce websites, when a user orders a copy of book A, your system should give a list of other suggested books. Book B is suggested, if there exist a user X that bought both A and B. The suggested books should be sorted on decreasing sales count (i.e., most popular first); count only sales to users like X (i.e. the users who bought both A and B).



You may also like:

Title	Author	Rate
The Mysterious Island	Jules Verne	7.5

```
    with connection.cursor() as cursor:
        cursor.execute("SELECT DISTINCT book_id FROM BookStore_Orders WHERE
user_id IN(SELECT DISTINCT user_id FROM BookStore_Orders WHERE book_id=%s)
AND book_id != %s AND user_id != %s"%(b_id,b_id,request.user.id))
        booklist = cursor.fetchone()
        if booklist is not None:
            recommendationlist=[([0]*2)for i in range(len(booklist))]
            count=0
            for book in booklist:
                cursor.execute("SELECT SUM(copy) FROM BookStore_Orders
WHERE book_id=%s AND user_id IN(SELECT DISTINCT user_id FROM
BookStore_Orders WHERE book_id=%s) AND
user_id != %s"%(str(book),b_id,request.user.id))
                recommendationlist[count][0] = int(book)
                recommendationlist[count][1] = cursor.fetchone()[0]

                count += 1

            sortdata=np.array(recommendationlist)

            idex=np.lexsort([-1*sortdata[:,1]])

            recommendationlist = sortdata[idex, :]
```