CMPSC 431W Project

Name: Arpit Singla

PSU email: abs6339@psu.edu

Deployment:

As told in the phase 1, my tech stack consists of Django framework which is a python framework with MYSQL for database handling.

Now first let's install **Django**:

1. First open the command prompt as admin
2. Now we will need to install pip
3. After pip has been successfully installed, we will need to enter the following command to install Django:

   Python –m pip install Django

Now, with this Django has successfully been installed on your local machine

Now, we will need to install **MYSQL**,

In order to do so we will first install MYSQL command line client, this helps us in creating the database and initial setup.

Now to install MYSQL on Windows,

1) Download the MSI installer package from https://dev.mysql.com/downloads/shell/

2) Go to the link, and click Download

3) After installing, go to your MYSQL workbench,

4) If it is the first time for you opening it then it'll ask you to create a username/root and create a password (**NOTE**: please remember this password and username, its needed to connect to the server)

4) Also, whenever you need to access MYSQL, you need to enter your password

Now, let's start Django and create a project

1. First open your command line/ terminal and write

   Django-admin startproject bookstore

2. With this command you have successfully created a Django project which contains some preloaded files by Django framework.
3. Now to run the project, type

Python manage.py runserver

So, now you are connected to the server.

Now, we can connect Django with the MYSQL

1. First, go to your project's main directory and in settings.py file
2. Inside you will see a DATABASES dictionary, to connect to MYSQL, change the sqlite3 to mysql and add user, password, host and port fields as in the following image.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'mybookstore',
        'USER': 'root',
        'PASSWORD': 'root',
        'HOST': '127.0.0.1',
        'PORT': '3306',
    }
}
```

NOTE, here the user and password are values that you saved in the above step while setting up mysql

Now, keep in mind these two commands whenever you want to change the database and update it everywhere i.e. in the Django and MYSQL.

Python manage.py makemigrations

This command checks if everything that you updated in the database is alright and is acceptable according to the dbms rules.

Then second command is

Python manage.py migrate

This command actually makes all the updates, that is in the mysql and Django files.

Now, you have successfully connected Django and mysql database

Before we start writing our code, in Django there is something called apps which one needs to create as per standard practice. Here app refers to a directory where all the main functions and code goes. So, here we have created **mystore** app.

Now after creating app, **we can start writing the code.**

As we have already created our tables in phase 1, we can start adding them in our code.

So, in the models.py file in your Django directory, make a class like in the below image and add the table.

```
# **************registering users***************


class registerdb(models.Model):
    username = models.CharField(
        max_length=30, primary_key=True, blank=False, null=False, unique=True)
    firstname = models.CharField(max_length=30, blank=False, null=False)
    lastname = models.CharField(max_length=30, blank=False, null=False)
    phone = models.CharField(max_length=10, blank=False, null=False)
    address = models.CharField(max_length=100, blank=False, null=False)
    password = models.CharField(max_length=20, blank=False, null=False)
    userlevel = models.CharField(
        max_length=10, blank=False, null=False, default="0")
    isactive = models.IntegerField(default=0)
```

Note, as explained above as you add this table more like a class in your model.py, you will need to run those 2 commands which are used to create and update the database in mysql.

With this we can successfully use this database table and it's attributes in our code.

Likewise, you can add more tables to complement with your functions.

Now, we can start writing the backend part of the code i.e. **core functions of the project.**

Functionalities:

Note, for our project,

Note: We have **admin**/ manager access given to **username**: **arpit7** and **password**: **admin**

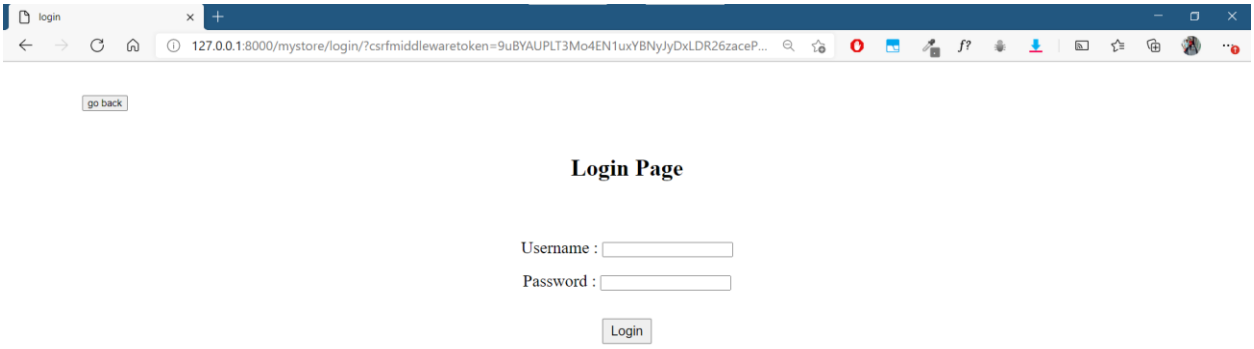
Note: We have **user**/ customer access given to **username**: **vanshs** and **password**: **12345**

User Login:

In the loginpage function under (mystore/views.py), it asks the user to input his username and password, if the username and password matches in the database then we check their userlevel (i.e. are they manager or a customer). If they are manager then we direct them to the manager dashboard and if they are a customer then we direct them to user dashboard. Note, if the username or password doesn't match then we return error message.

```
def loginpage(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')

        all_customers = registerdb.objects.get(
            username=username, password=password)

        if (all_customers.username == username and all_customers.password == password):

            all_customers = registerdb.objects.filter(
                username=username).update(isactive=int(1))
            all_customers = registerdb.objects.get(username=username)
            context = {'all_customers': all_customers}
            print(all_customers)
            print(all_customers.userlevel)
            if (int(all_customers.userlevel) == 0):
                return render(request, "mystore/bookstorehpuser.html", context)
            else:
                return render(request, "mystore/bookstorehpmanager.html", context)
        else:
            messages.info(request, 'Username OR Password is incorrect')

    return render(request, 'mystore/login.html')
```

Our website view of the login page:

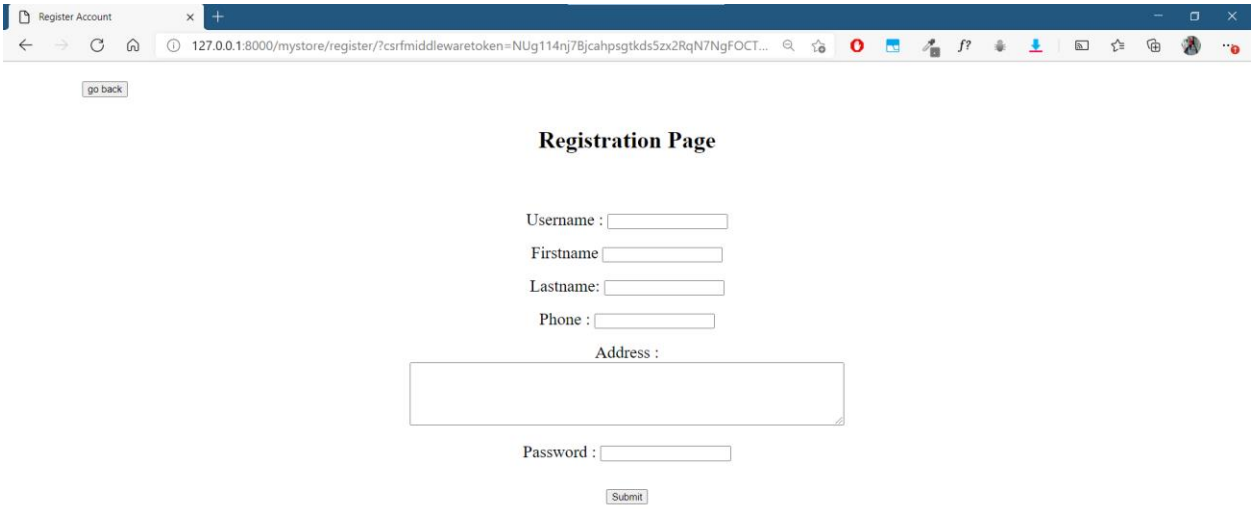


User Register:

In the register function, we take customer information via POST method and fetch their details and store them in the database. Then we use Django's SQL query to insert the username and other information in our registerdb database table.

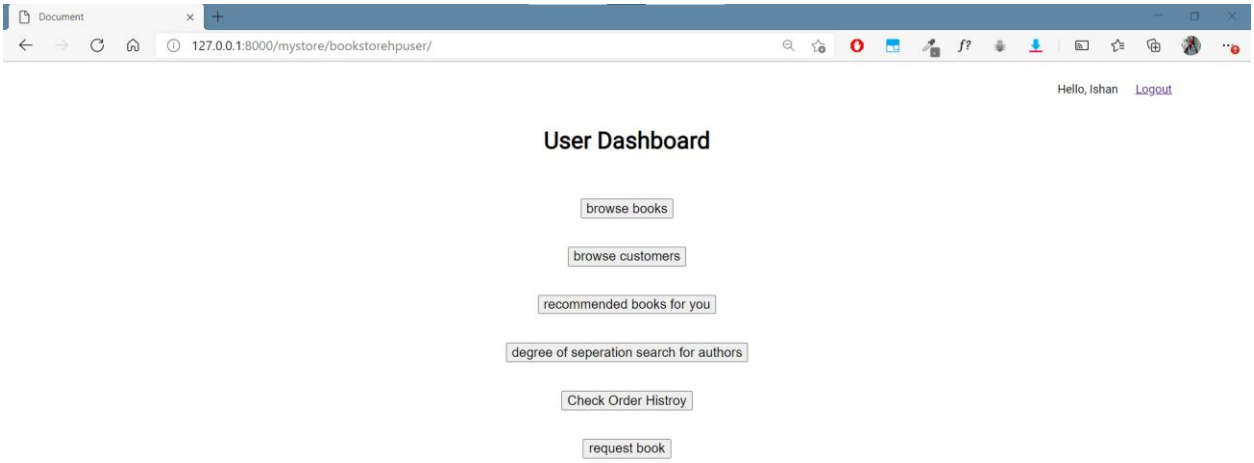
```
def register(request):
    if request.method == "POST":

        username = request.POST.get('username', 'm')
        firstname = request.POST.get('firstname', 'm')
        lastname = request.POST.get('lastname', 'm')
        address = request.POST.get('address', 'm')
        phone = request.POST.get('phone', 'm')
        password = request.POST.get('pass', 'm')

        if(username == 'm' or firstname == 'm' or lastname == 'm' or address == 'm' or phone == 'm' or
        password == 'm' or username == '' or firstname == '' or lastname == '' or address == '' or phone
        == '' or password == ''):
            messages.error(
                request, 'Error, either something is left empty or is invalid!')
            return render(request, 'mystore/register.html')
        else:
            ins = registerdb(username=username, firstname=firstname, lastname=lastname, phone=phone,
                             address=address, password=password)
            ins.save()
            return render(request, 'mystore/login.html')
    else:
        return render(request, 'mystore/register.html')
```

Our website view of the register page:



User Dashboard:

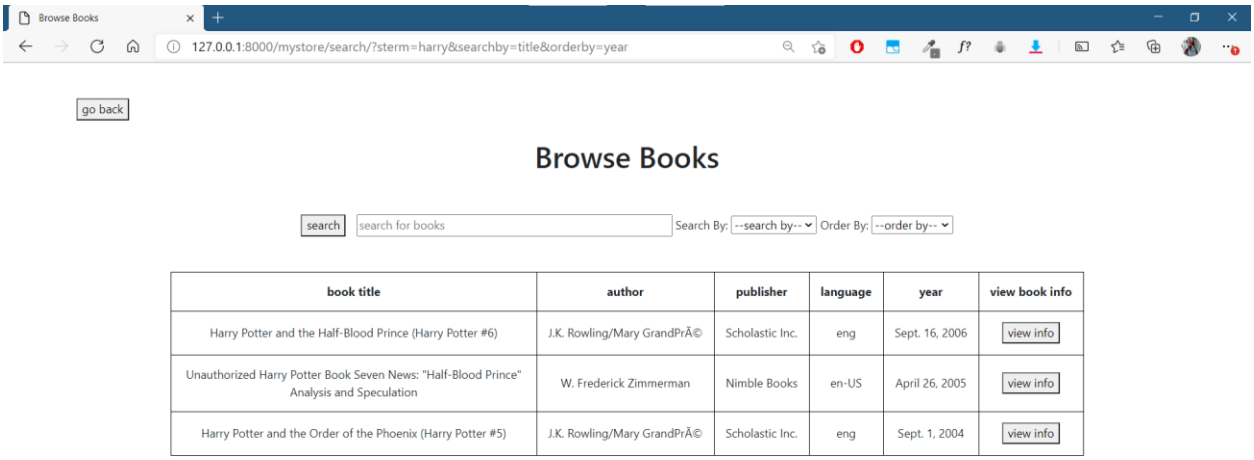
In our homepage of user i.e. 'bookstorehpuser' function in views.py, customers can see all the functions available on our website. Each function is linked to a website where they can make use of the function. So, if you click browse book, then you will see the page where you can browse book by author and other methods. Whereas, the order history allows you to see the orders you placed and their delivery status.



Browse book:

In the browsebook function and search function in (views.py), we see they can search any term by different search by options such as author, title and they can even order them up by year too.

Again, we make use of the Django sql query to filter off the results and show them to the user.

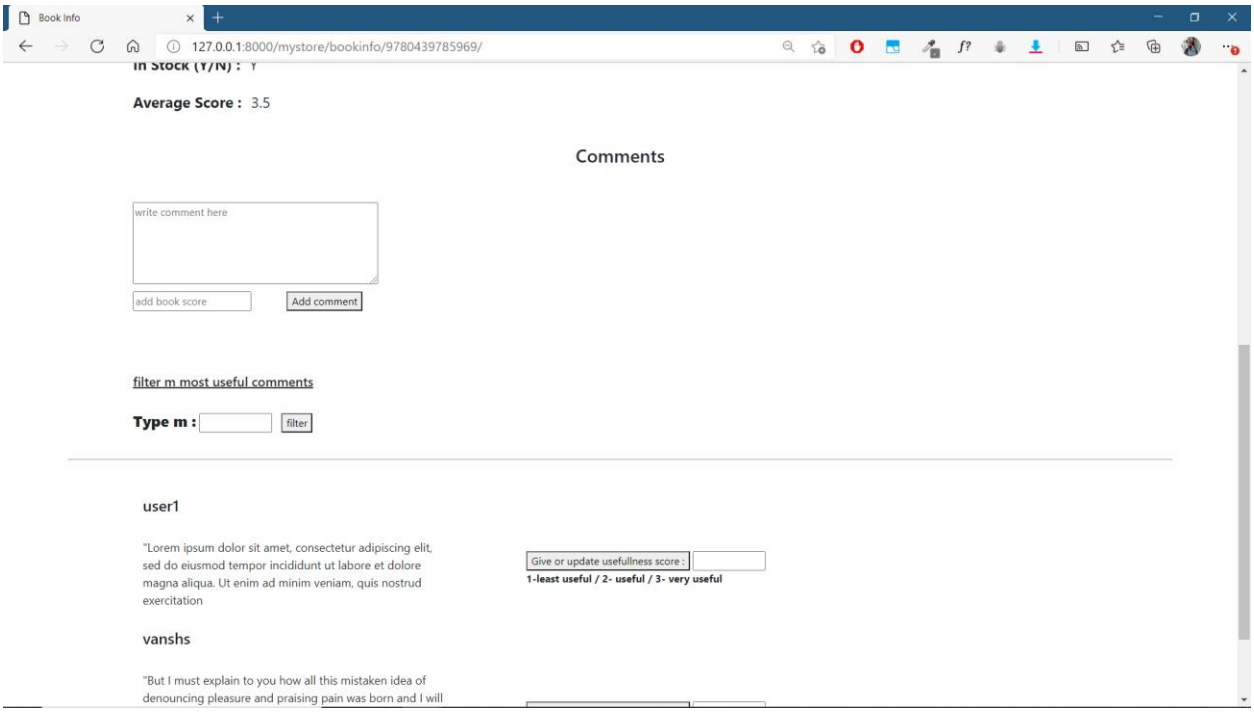


book title	author	publisher	language	year	view book info
Harry Potter and the Half-Blood Prince (Harry Potter #6)	J.K. Rowling/Mary GrandPrÃ©	Scholastic Inc.	eng	Sept. 16, 2006	view info
Unauthorized Harry Potter Book Seven News: "Half-Blood Prince" Analysis and Speculation	W. Frederick Zimmerman	Nimble Books	en-US	April 26, 2005	view info
Harry Potter and the Order of the Phoenix (Harry Potter #5)	J.K. Rowling/Mary GrandPrÃ©	Scholastic Inc.	eng	Sept. 1, 2004	view info

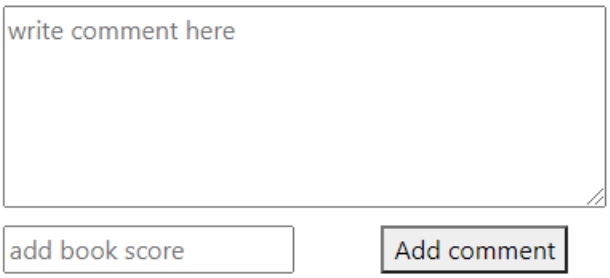
Note there is a view info button on the right side of each book, the customers can click that link to see detailed info about the book and comment as well. Here we are making use of bookinfo function in views.py.

Add Comment:

In the addcomment function under (views.py), we see the detailed information about the book, its **comments**, and here users can even **add comments** on the book and give **usefulness rating to other comments** and **order books** as well

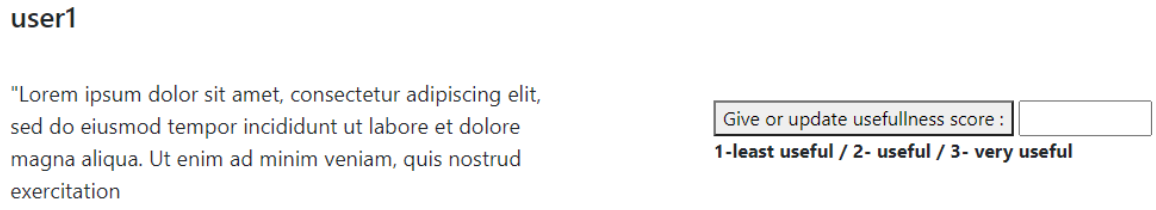


Zoom in View



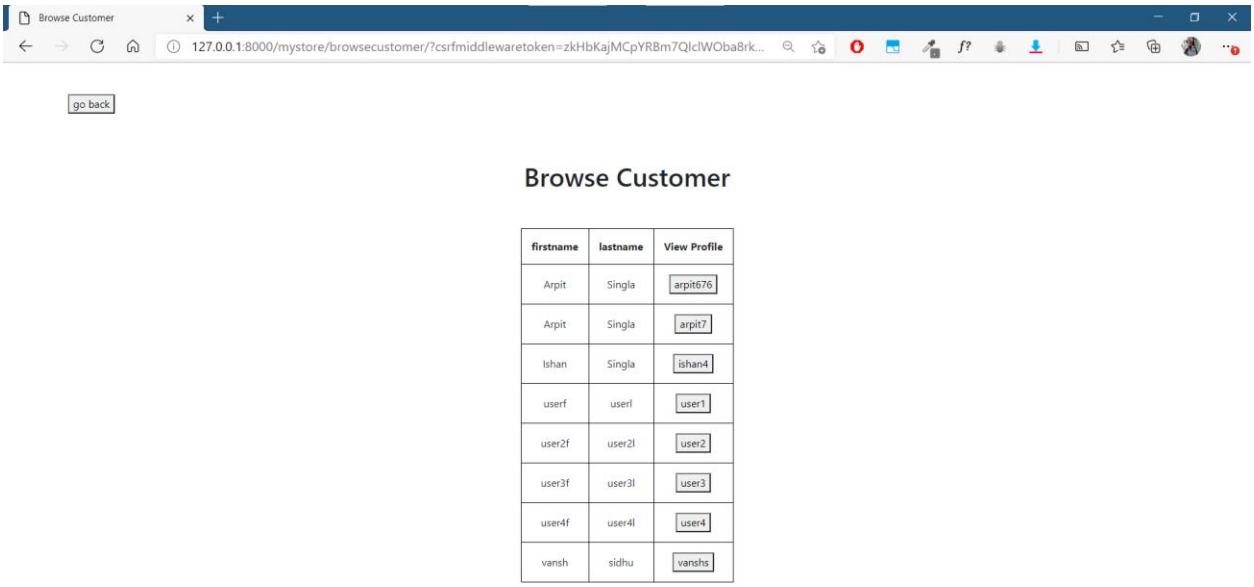
Useful/Useless comments:

User can give his/her rating whether the comment is useful or not. This functionality uses “addusefullness” function in views.py .

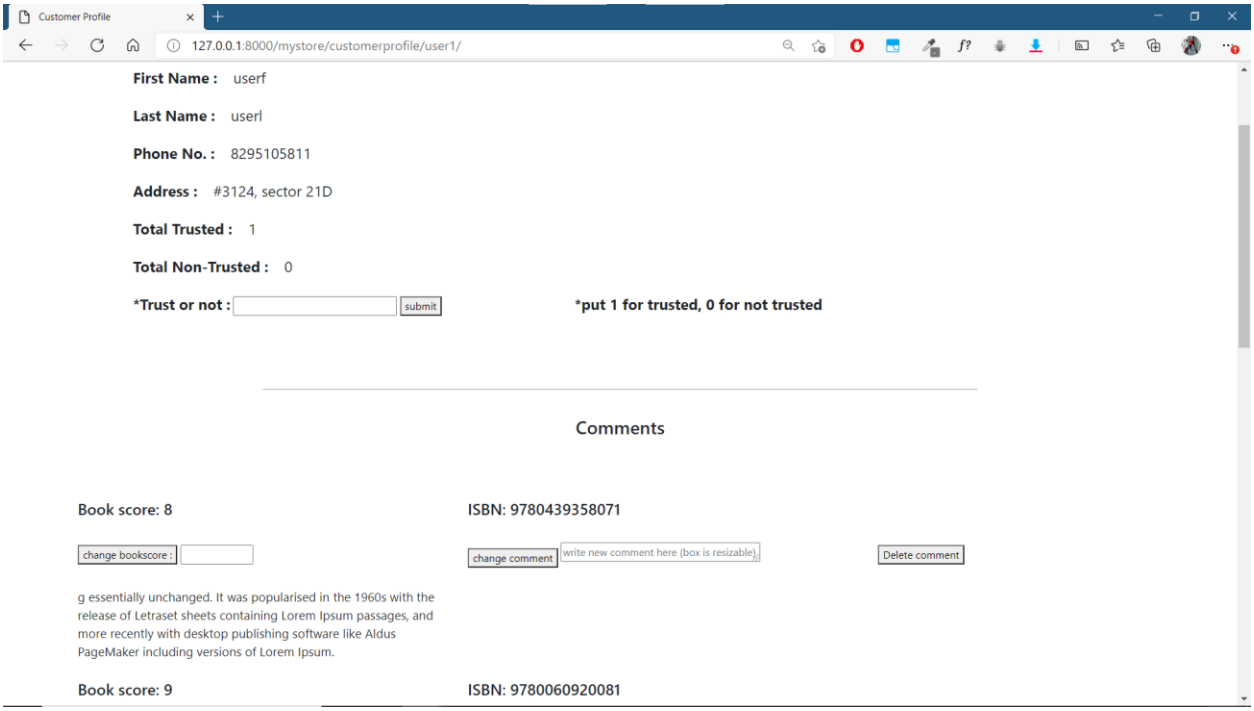


Browse Customer Profiles:

Using browsecustomer function (views.py) from the user dashboard a user can see other user’s profile and the comments they have written on books. Using this function they can even trust or not trust the user. Here a user can click on view profile button and they can instantly see a detailed page with all the other customer’s information.



This is the detail view using customerprofile function, here customers can give **trust rating** and here we even have an option to **edit the comment by deleting, changing it and updating the book score.**



Edit Comment and Delete:

Using editcomment function in views.py

Trust rating:

Using addtrust function in views.py

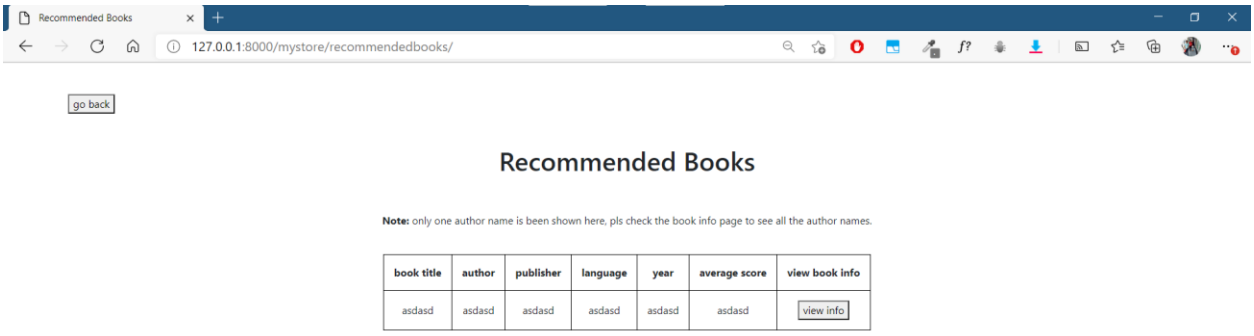
Total Trusted : 1

Total Non-Trusted : 0

***Trust or not :** [ ] submit ***put 1 for trusted, 0 for not trusted**

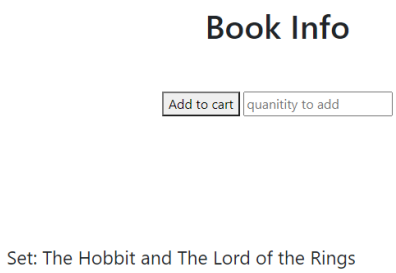
Recommended books:

Here whatever user orders, to that our code makes use of the sql queries to check which other books did other customers bought and recommends them here.



Order Book:

In the book info detail view using the orderbook function, the customers have the option to add the book into the cart. They are asked about the quantity and then they are redirected to the order history page where they can see the total amount of their order and information about all of their past orders as well.



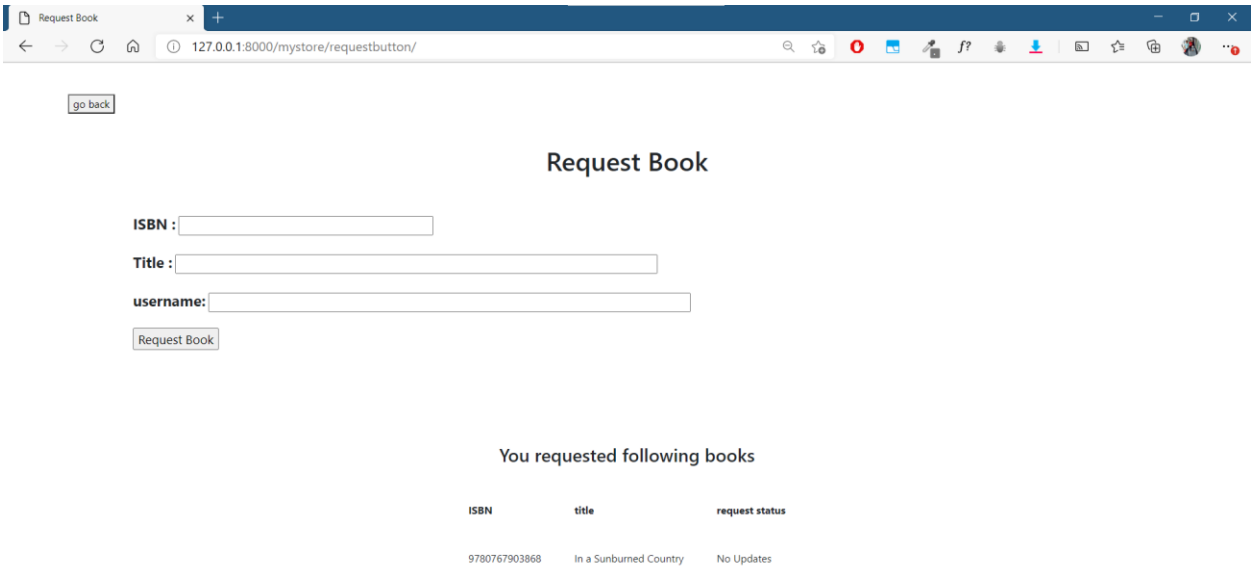
Order History (Extra Function):

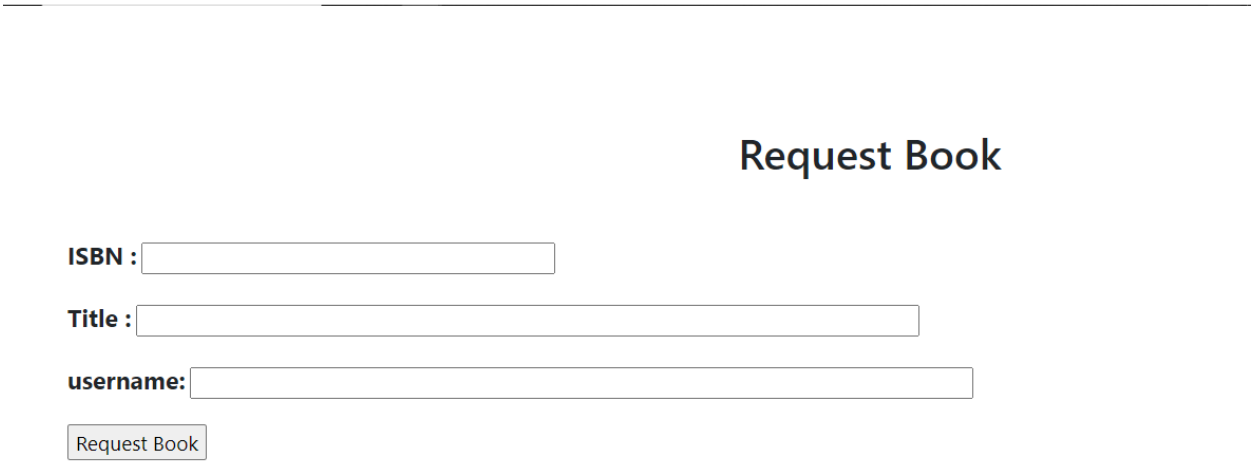
Here the customer can see information about all the past orders using orderhistory function in views.py. We even have order status to keep track of where the order is, is it delivered or pending or not delivered.

Book-ISBN	Order place date:	Copies ordered	Amount in $	Order status:
9780345538376	May 2, 2021	1	100	not delivered

Request Book: (Extra function)

This is the extra function (requestbook function in view.py) where if a user wants any book to be present in the bookstore, they can request the bookstore by submitting this form and the managers can update the status if the book will be available and if not.





Requested book history: (Extra function)

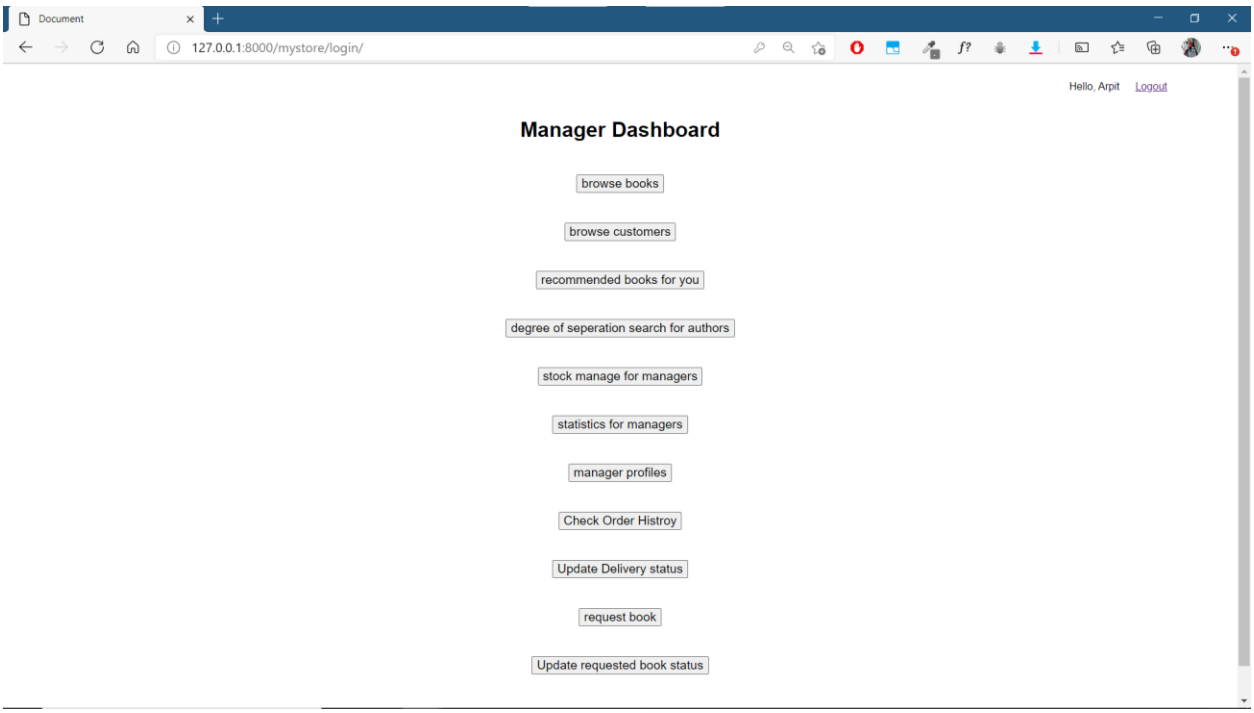
Whatever book users requested from the above form, they can view all those and past requests here and they can even see the status if the book is available or not.

You requested following books

ISBN	title	request status
9780767903868	In a Sunburned Country	No Updates

Manager Dashboard:

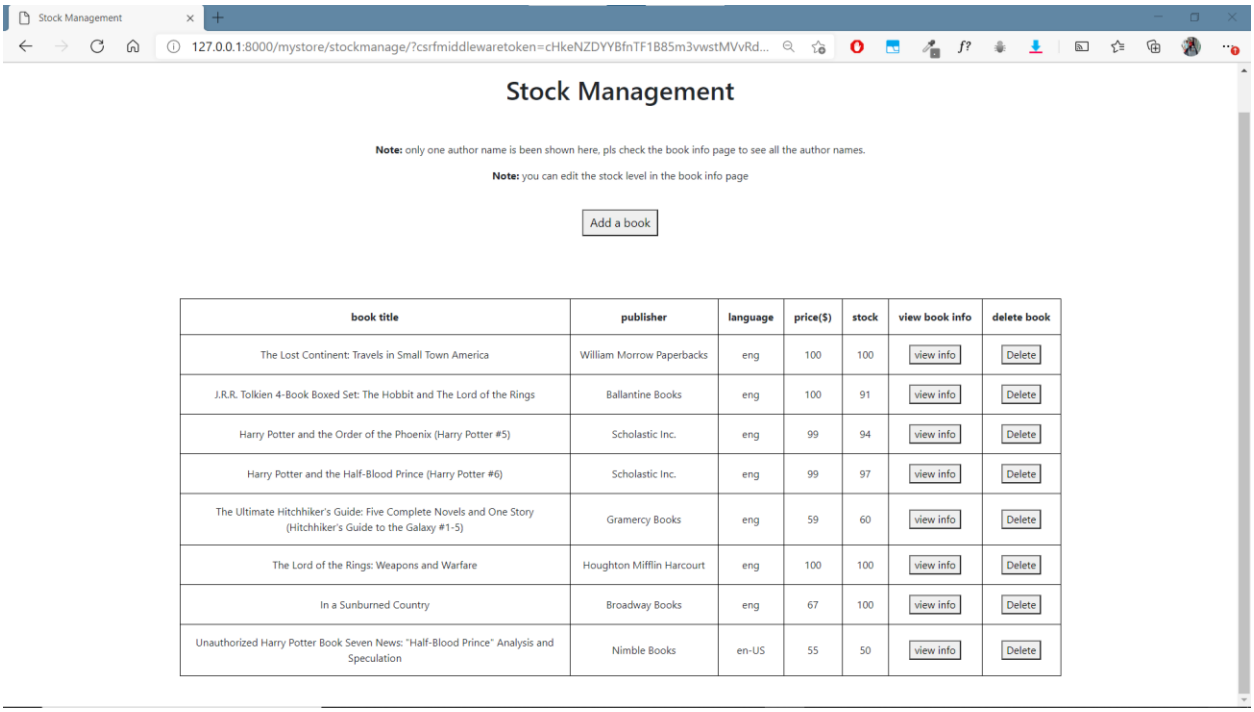
Here I am assuming that the if someone wants to be a manager then first he will need to sign in as a customer and then the super user can change the other user's user level through they can be given manager level access i.e. access to this dashboard.



Note: because the manager is also a customer therefore he also has access to all the tabs from user dashboard as well.

Stock Manage:

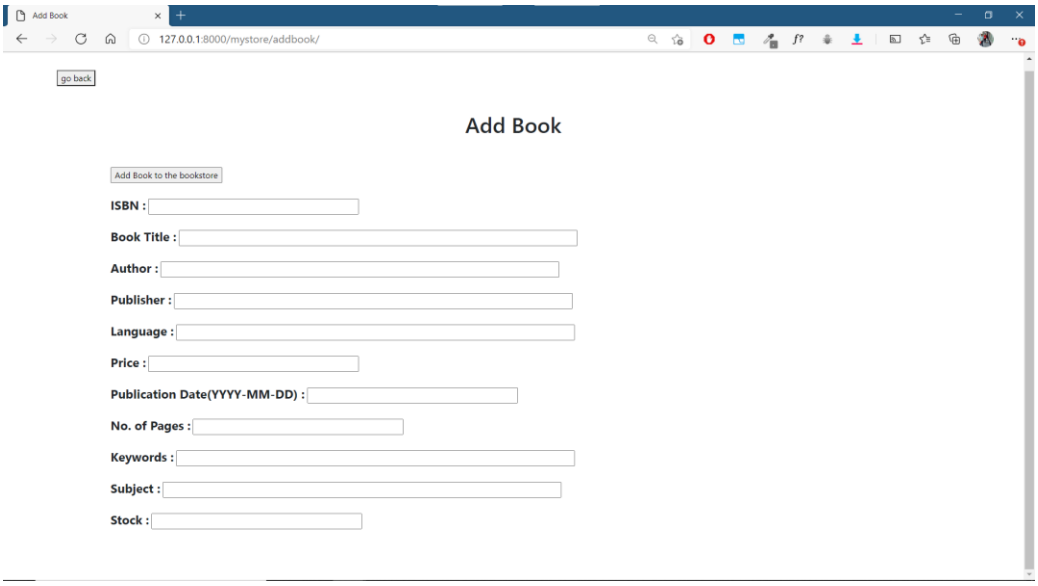
Under this tab using stockmanage function, the manager goes to the page where he can add the book and edit the stock of a book and even delete the book.



book title	publisher	language	price($)	stock	view book info	delete book
The Lost Continent: Travels in Small Town America	William Morrow Paperbacks	eng	100	100	view info	Delete
J.R.R. Tolkien 4-Book Boxed Set: The Hobbit and The Lord of the Rings	Ballantine Books	eng	100	91	view info	Delete
Harry Potter and the Order of the Phoenix (Harry Potter #5)	Scholastic Inc.	eng	99	94	view info	Delete
Harry Potter and the Half-Blood Prince (Harry Potter #6)	Scholastic Inc.	eng	99	97	view info	Delete
The Ultimate Hitchhiker's Guide: Five Complete Novels and One Story (Hitchhiker's Guide to the Galaxy #1-5)	Gramercy Books	eng	59	60	view info	Delete
The Lord of the Rings: Weapons and Warfare	Houghton Mifflin Harcourt	eng	100	100	view info	Delete
In a Sunburned Country	Broadway Books	eng	67	100	view info	Delete
Unauthorized Harry Potter Book Seven News: "Half-Blood Prince" Analysis and Speculation	Nimble Books	en-US	55	50	view info	Delete

Add a book:

Here the manager can write each and every detail of information about the book he wants to add in the inventory of his bookstore. We are making use of the addbook method in views.py.



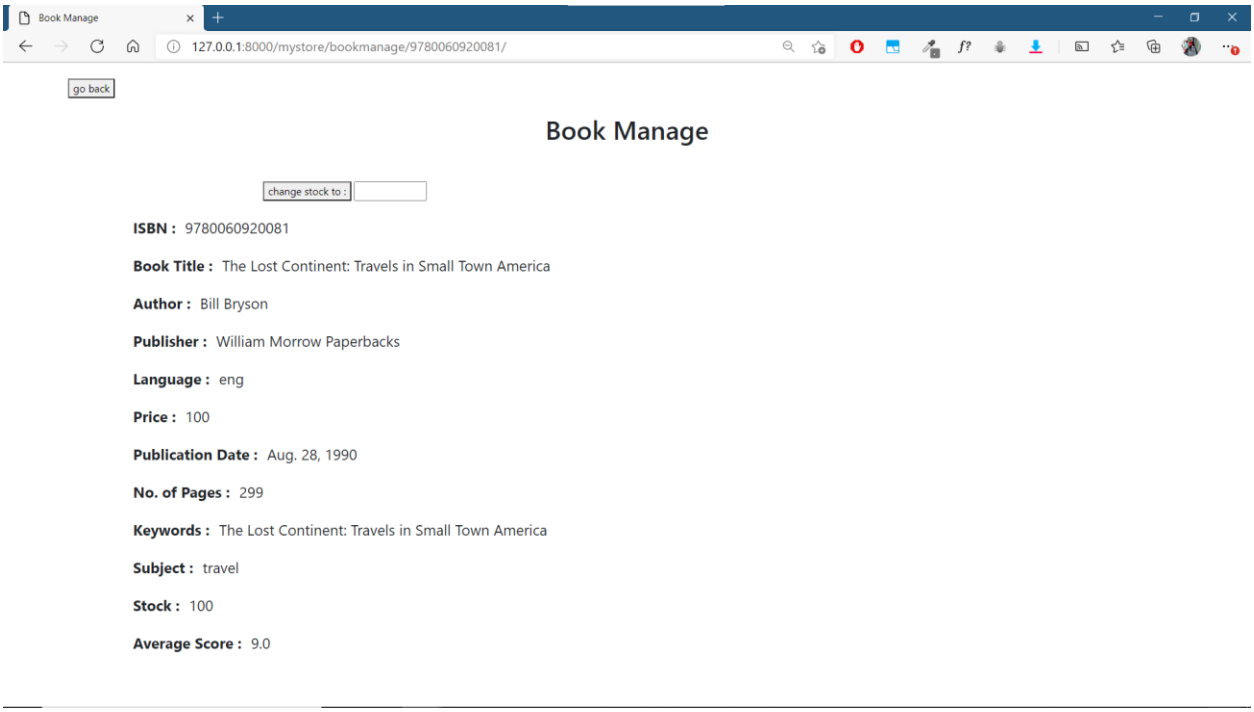
Update the stock:

After clicking on the view info button below,

book title	publisher	language	price($)	stock	view book info	delete book
The Lost Continent: Travels in Small Town America	William Morrow Paperbacks	eng	100	100	view info	Delete
J.R.R. Tolkien 4-Book Boxed Set: The Hobbit and The Lord of the Rings	Ballantine Books	eng	100	91	view info	Delete

Book Manage View for the managers

the manager sees a new window here, he sees the detailed information about the book and here he has the option to edit the stock level.



Using this he can update the stock level easily, note also that whenever a book is purchased the stock level is automatically decreased.

50920081

Delete Book (Extra Function):

Here on the stock manage page using deletebook function, we are also giving the manager to delete the book i.e. completely remove the book from the inventory of the bookstore.

book title	publisher	language	price($)	stock	view book info	delete book
The Lost Continent: Travels in Small Town America	William Morrow Paperbacks	eng	100	100	view info	Delete
J.R.R. Tolkien 4-Book Boxed Set: The Hobbit and The Lord of the Rings	Ballantine Books	eng	100	91	view info	Delete

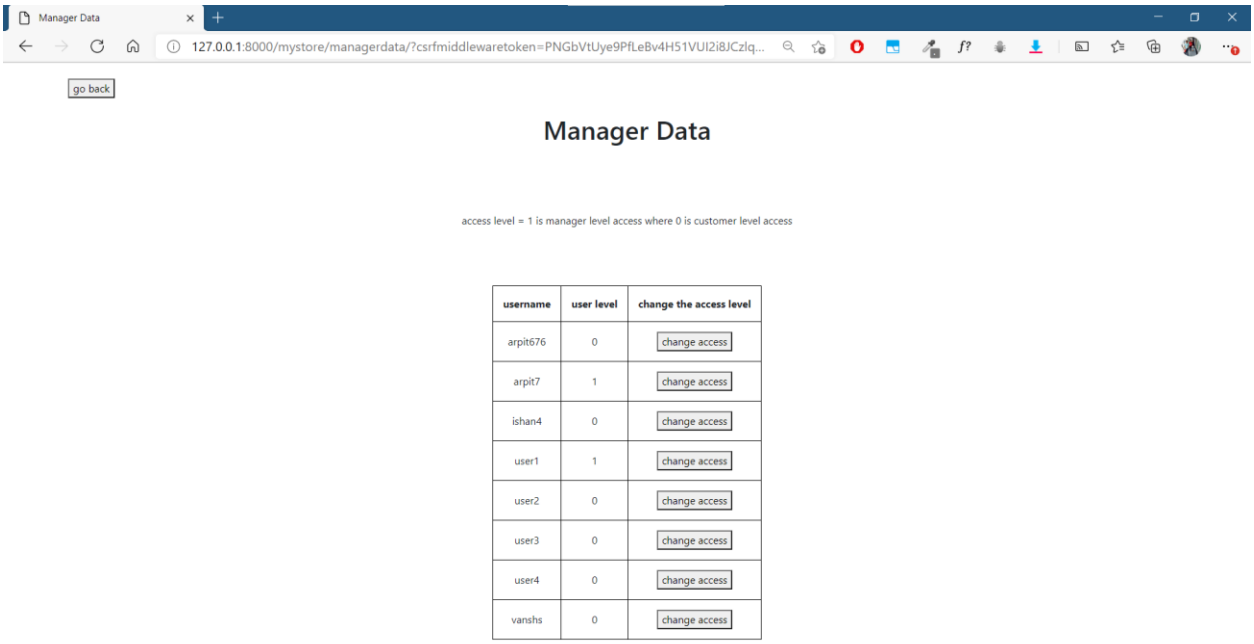
Statistics:

Inside this using statistics function specifically userstatistics and bookstatistics, the manager can use various tables available in our MYSQL database to find some useful stats about the most popular book and most useful and trusted user.

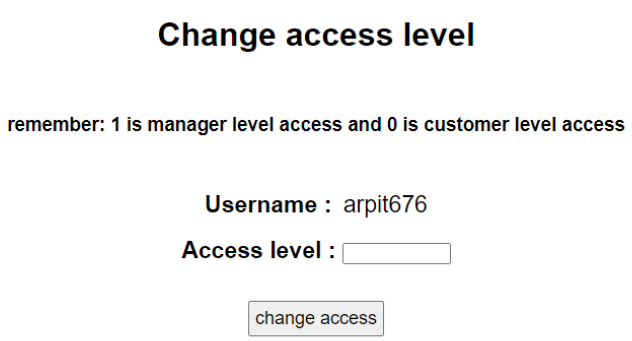
Manager Access:

Here, in manager data function, the manager can see all the users registered on the bookstore website. And here the super user can change the access level of any customer to provide them manager level access.

Note, ‘1’ here is manager level access and ‘0’ is user level default access.



After manager clicks on change access, they come to this page and here they can update the access level making a user to manager and a manager to a user level access only. Here we make use of the changeaccess function.



Delivery status update: (Extra function)

Here the manager can see all the orders placed and their details as well such as the quantity ordered and can update the delivery status of the order using the deliverystatus fucntion. So, this change is directly reflected in the user view as well where they see the order history and therefore they also see the order status.

Delivery Status

id	Book-ISBN	Order place date:	Copies ordered	Amount in $	Order status:	update Order status:
1	9780345538376	May 1, 2021	2	200	delivered	update status
2	9780439785969	May 1, 2021	3	297	delivered	update status
3	9780439358071	May 1, 2021	1	99	delivered	update status
4	9780439358071	May 1, 2021	2	198	not delivered	update status
5	9780060920081	May 2, 2021	1	100	not delivered	update status
6	9780060920081	May 2, 2021	3	300	not delivered	update status
7	9780345538376	May 2, 2021	1	100	not delivered	update status
8	9780345538376	May 2, 2021	1	100	not delivered	update status

(one more function on next page)

Update requested book status: (Extra function)

We are providing customers a function where if they want a book to be present in the bookstore then they can request a book.

That request is sent to the managers and they can see them and accordingly update the request status as in if the book is being included on the bookstore or not.

Requested book status

ISBN	title	request status	update request status:
123456789	Book 1	No Updates	update status
45789012	Book 2	No Updates	update status
9780517226957	Book 3	No Updates	update status
9780439785123	In a Sunburned Country	on the way to bookstore	update status
9780767903868	In a Sunburned Country	No Updates	update status