205CDE Project

Lui Cheuk Hin

54777756

20-03-2018

Introduction

This is a system about the operating a online bookstore. This system can be mainly divided into two parts which are the website and the database. In this system, visitors can browse and search through the information of the books in the bookstore with the support of the database, such as the availability, price, stock of the books etc. On the other hand, registered member can even put books into shopping cart, and sent the order to the book company with the website. Shipping and delivery will be made after user has been paid with bank transfer for their orders of books.

The goal of the system is to provide a precise, accurate and convenient online books ordering service for the users. With the system, the quality and pace of service is generally higher than the normal way of bookselling attributed to the aid of centralized, instant live data and streamlined book ordering process.

The Flow of the System

The Home Page

At the top of the browser there is a navigation bar provides a list of actions which can be performed by users. They are the home page button, dropdown of the news, dropdown of book categories, about us button, register button, login and search bar.

At the centre of the page, there is a banner greets the user and provides buttons for user registration and login. Below the banner, there is a carousel showing the latest news of the bookstore.

News

The News dropdown provides only one option at this moment and that is showing the latest arrivals of books. The latest imported books will show in a list format with descending order, the latest being on top. In each single list item, it shows the book name, authors and also the imported date. It provides clear information to users which books have arrived and lately imported to this bookstore with this feature.

Books

The Books dropdown provides a option to browse through the book items with particular category. Users may choose the way to find their books they want in a sorted category. If the header is ‘Books’, it means that this page is currently showing items in all category. There are initially 4 items listed in each page with that category. Users can also change the number of items for each page if they want to browse the books in a faster way. Users can also get a good grasp of the books’ information from each box. They are name of the book, author, price, stock, description. If users want to know more about a particular book, they can click the name of the book and that will bring them to a page which contains more information and comments from buyers. Finally, there is a ‘Add to cart’ button for the logged in members to add the items into their shopping carts.

Page of a Book

This page can be shown by clicking the name of the books in New Arrivals and Books page. This page contains all the information of a single book. Moreover, there is a comment section in this page. User can leave and read comments about the book. Therefore, they can make use of the comments to have a good understand about the book and determine whether they buy it or not. The most important part is that the comment section forster the discuss between the members and it is believed that it will help the click rate and the popularity of this book store slightly.

About us

In this page, it specifies and states the goal of the bookstore and provides the way and contacts and bank account number for the payment

Login

In this page, it provides a option for register users to login. User logged in can leave comments of books, add items to their shopping carts, submit their orders, check the condition of their orders.

Register

In this page, it provides a option for users to register and to be a member of this book store. It requires the name, address, email, username, and password of the user for registration.

Shopping cart

In this page, it provides a option for users to shows the book items that were temporarily store their items in the shopping cart. They can delete items from the shopping cart if they want to. After they confirm the order, they can see the total price of the order and the button of order submission.

Order History

In this page, it provides a option for users to check the condition of their order. Users are supposed to pay the order with bank transfer after submission of the order. If payment is confirmed by the bookstore, the ‘Paid’ condition will change to Yes from ‘No’ and they can check the current condition of the shipping of their orders.

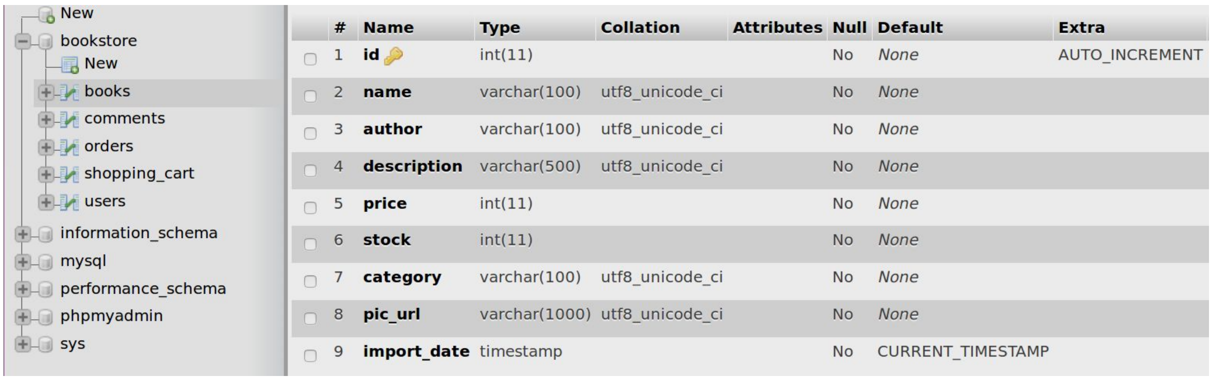
Logout

In this page, it provides a option for users to logout.

Database Structure

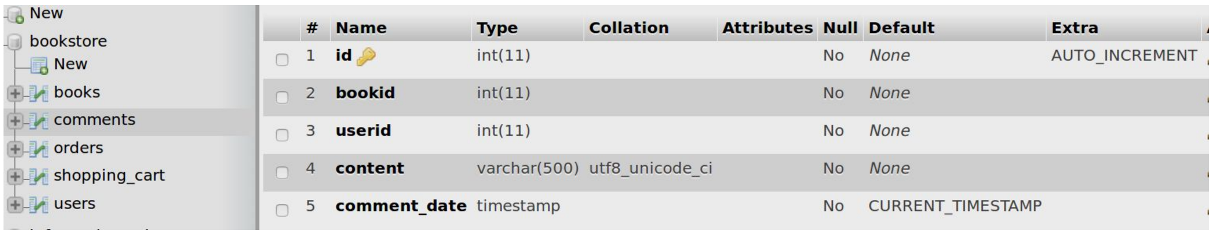
The database of this system consists of 5 tables.

Table: books



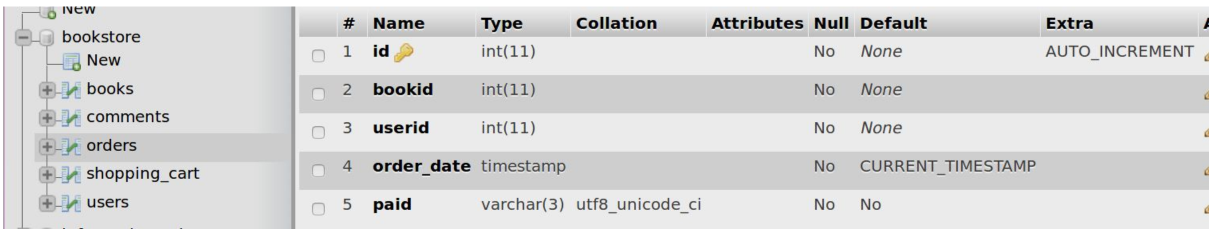
#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**id**	int(11)			No	*None*	AUTO_INCREMENT
2	**name**	varchar(100)	utf8_unicode_ci		No	*None*	
3	**author**	varchar(100)	utf8_unicode_ci		No	*None*	
4	**description**	varchar(500)	utf8_unicode_ci		No	*None*	
5	**price**	int(11)			No	*None*	
6	**stock**	int(11)			No	*None*	
7	**category**	varchar(100)	utf8_unicode_ci		No	*None*	
8	**pic_url**	varchar(1000)	utf8_unicode_ci		No	*None*	
9	**import_date**	timestamp			No	CURRENT_TIMESTAMP	

Table: comments



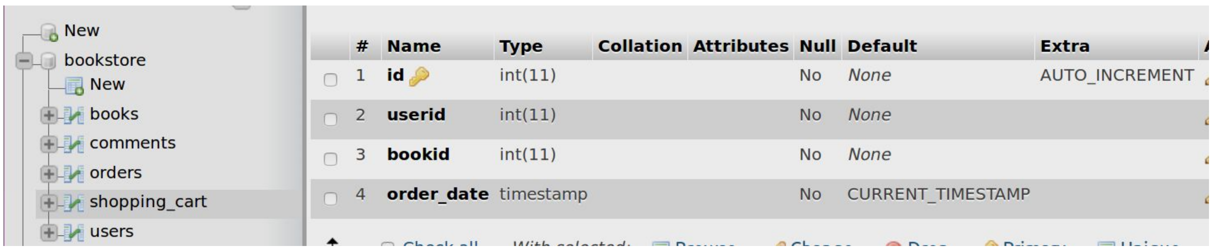
#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**id**	int(11)			No	*None*	AUTO_INCREMENT
2	**bookid**	int(11)			No	*None*	
3	**userid**	int(11)			No	*None*	
4	**content**	varchar(500)	utf8_unicode_ci		No	*None*	
5	**comment_date**	timestamp			No	CURRENT_TIMESTAMP	

Table: orders



#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**id**	int(11)			No	*None*	AUTO_INCREMENT
2	**bookid**	int(11)			No	*None*	
3	**userid**	int(11)			No	*None*	
4	**order_date**	timestamp			No	CURRENT_TIMESTAMP	
5	**paid**	varchar(3)	utf8_unicode_ci		No	No	

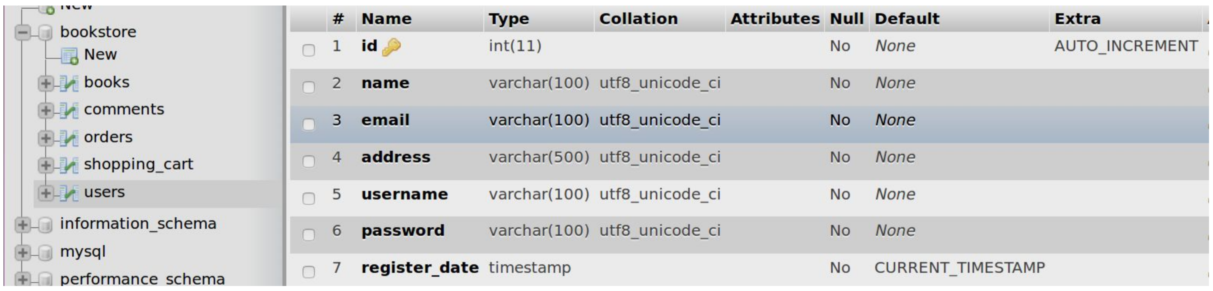
Table: shopp_cart



New
bookstore
New
books
comments
orders
shopping_cart
users

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**id**	int(11)			No	*None*	AUTO_INCREMENT
2	**userid**	int(11)			No	*None*	
3	**bookid**	int(11)			No	*None*	
4	**order_date**	timestamp			No	CURRENT_TIMESTAMP	

Table: users



bookstore
New
books
comments
orders
shopping_cart
users
information_schema
mysql
performance_schema

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**id**	int(11)			No	*None*	AUTO_INCREMENT
2	**name**	varchar(100)	utf8_unicode_ci		No	*None*	
3	**email**	varchar(100)	utf8_unicode_ci		No	*None*	
4	**address**	varchar(500)	utf8_unicode_ci		No	*None*	
5	**username**	varchar(100)	utf8_unicode_ci		No	*None*	
6	**password**	varchar(100)	utf8_unicode_ci		No	*None*	
7	**register_date**	timestamp			No	CURRENT_TIMESTAMP	

Design feature

Session

Session has been implemented in this system. Since only the users who logged in can access the shopping card, order history, leaving comments, adding items into cart. Apart from the access control, it will be very handy if we stored the information of the users, such as user_id and username, because we can retrieve back the important data anytime we want whenever the users is in any page.

Here is an example of the code:

```
            # Compare Passwords
            if password_candidate == password:

                # Passed
                # Create session for user
                session['logged_in'] = True
                session['username'] = username

                # Create session particularlly to 'id' in table
'users'
                result = cur.execute("SELECT id FROM users WHERE
username = %s", [username])
                id = cur.fetchone()
                id = id['id']
                session['id'] = id
```

This part is in the login process. If the username and password are both matched, it will turn the logged_in flag into True, stored the username and the id of the user.

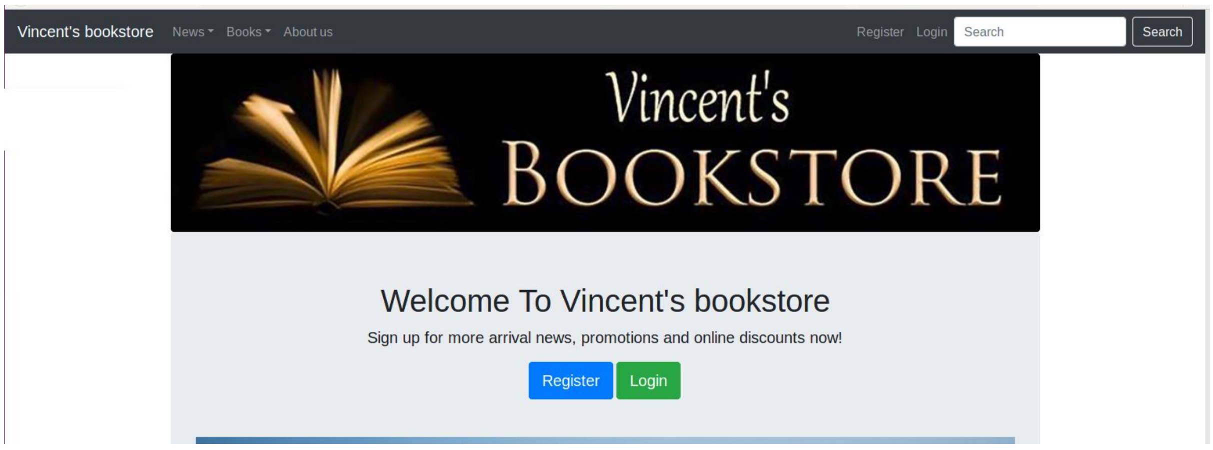
The logged_in flag is used to determine whether the user is logged in. If they do not, they are not granted rights to leave comment, use the function of the shopping cart.

```
    <ul class="nav navbar-nav navbar-right">
      {% if session.logged_in %}
        <li class="nav-item"><a class="nav-link" 
href="/order-history">Order History</a></li>
        <li class="nav-item"><a class="nav-link" 
href="/shopping-cart">Shopping cart</a></li>
        <li class="nav-item"><a class="nav-link" 
href="/logout">Logout</a></li>
      {% else %}
        <li class="nav-item"><a class="nav-link" 
href="/register">Register</a></li>
        <li class="nav-item"><a class="nav-link" 
href="/login">Login</a></li>
      {% endif %}
    </ul>
```

The logged_in flag is used to determine whether the user is logged in. If they do not, they are not granted rights to leave comment, use the function of the shopping cart etc.

In this example, this is the html code of the navigation bar at the top of the page. If the user is logged in, the right hand side of the navigation bar will show ‘Order History’, ‘Shopping Cart’, ‘Logout‘ options. Otherwise, it will only show ‘Register’ and ‘Login’. This kind of action is also done with hiding the comment input form and ‘add to cart’ button if they are not logged in.

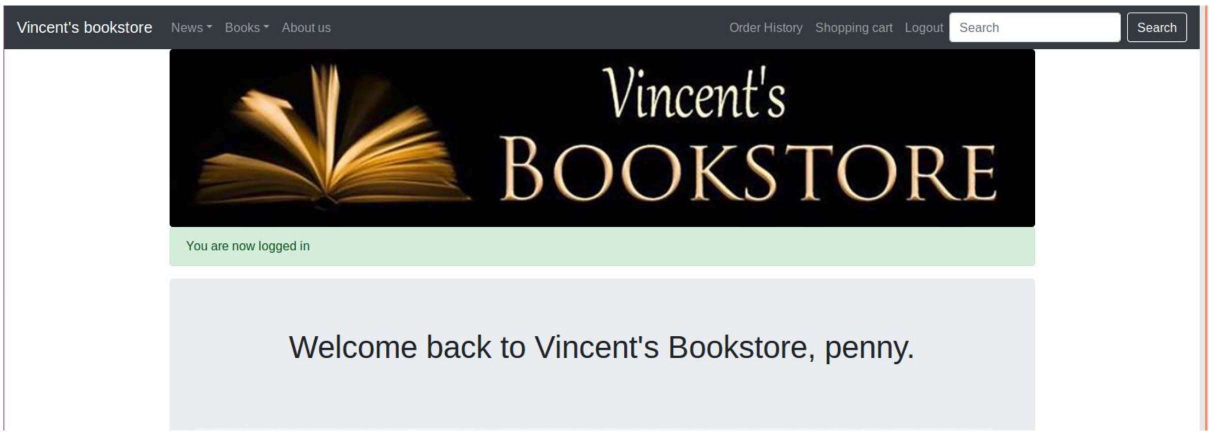
Before logged in



Welcome To Vincent's bookstore

Sign up for more arrival news, promotions and online discounts now!

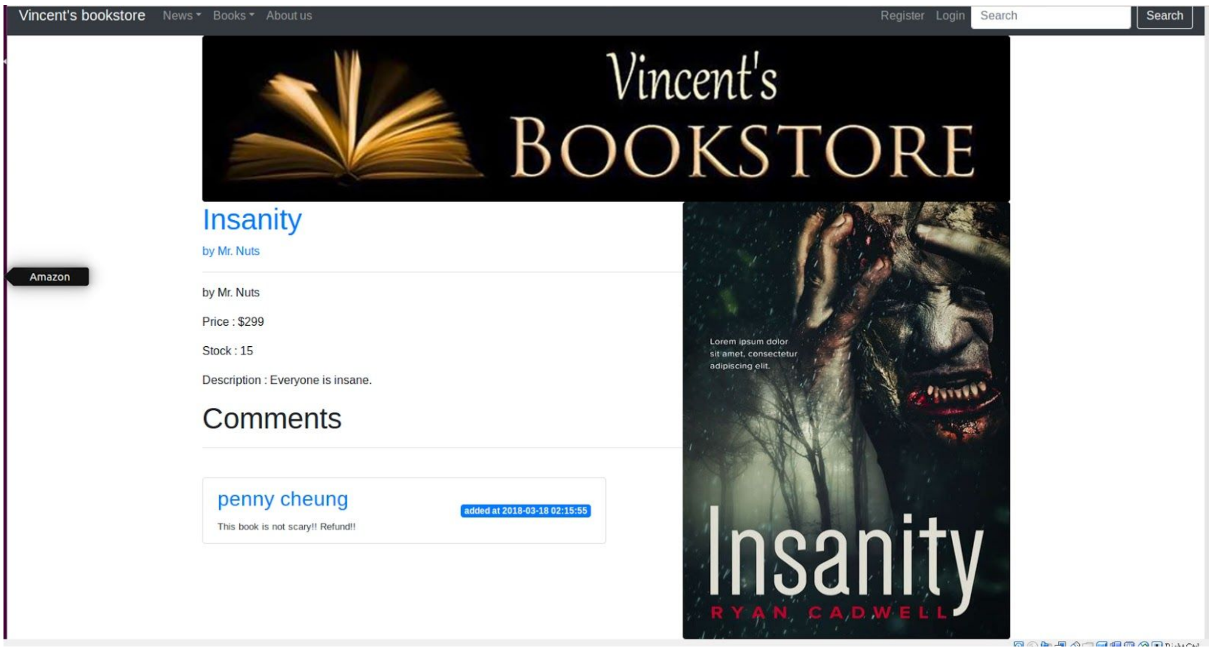
After logged in



You are now logged in

Welcome back to Vincent's Bookstore, penny.

Before logged in:



Insanity
by Mr. Nuts

by Mr. Nuts

Price : $299

Stock : 15

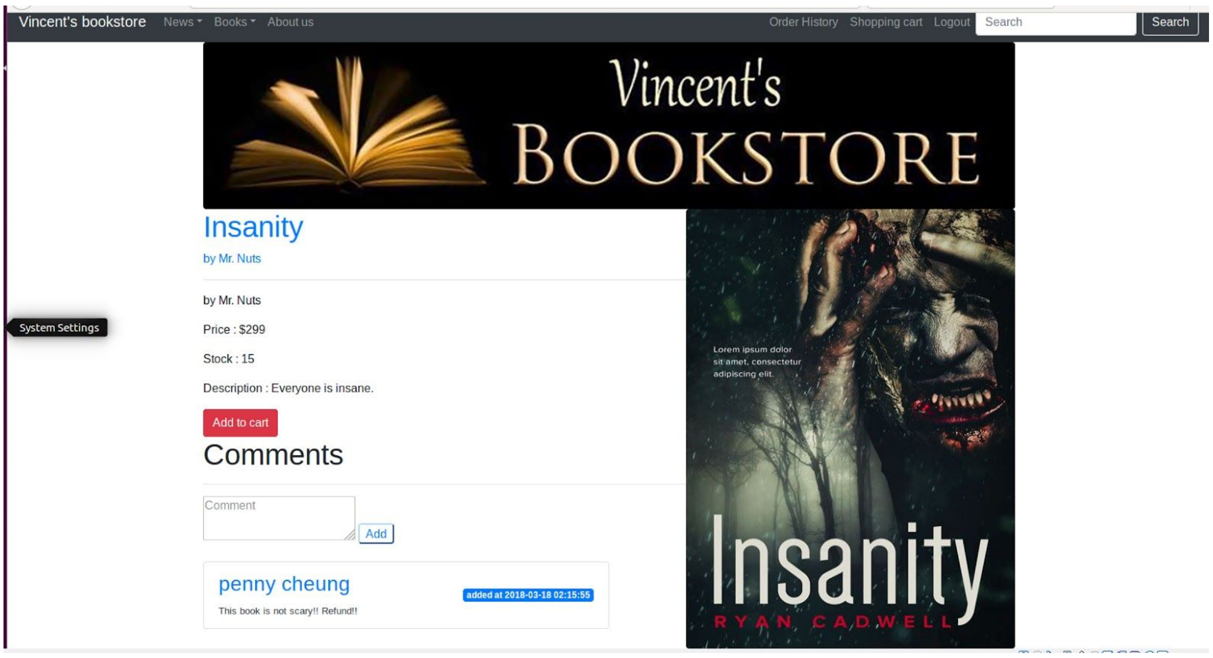
Description : Everyone is insane.

Comments

penny cheung added at 2018-03-18 02:15:55

This book is not scary!! Refund!!

After logged in:



Insanity

by Mr. Nuts

by Mr. Nuts

Price : $299

Stock : 15

Description : Everyone is insane.

Add to cart

Comments

Comment

Add

penny cheung

added at 2018-03-18 02:15:55

This book is not scary!! Refund!!

```
     # Keep looping until all items in shopping cart are sent to
orders
    while True:
        # Get number of items in shopping cart
        no_of_records = cur.execute("SELECT * FROM shopping_cart
WHERE userid = %s",[session['id']])
        # If there is no item
        if no_of_records == 0:
            break
        data = cur.fetchone()
        user_id = data['userid']
        book_id = data['bookid']
        # Insert item 'X' to table 'orders' and delete the item
'X' in table shopping cart
        cur.execute("INSERT INTO orders (userid, bookid) VALUES
(%s, %s)", [user_id,book_id])
        mysql.connection.commit()
        cur.execute("DELETE FROM shopping_cart WHERE userid = %s
AND bookid = %s", [user_id,book_id])
        mysql.connection.commit()
```

The username and id of the user is used to identify the user. If they want to submit the items from their shopping cart, we need to identify who is the user to put into the shopping cart. Therefore, it can make a good use of the session to record the user’s id into the tables in the database.

WTForms

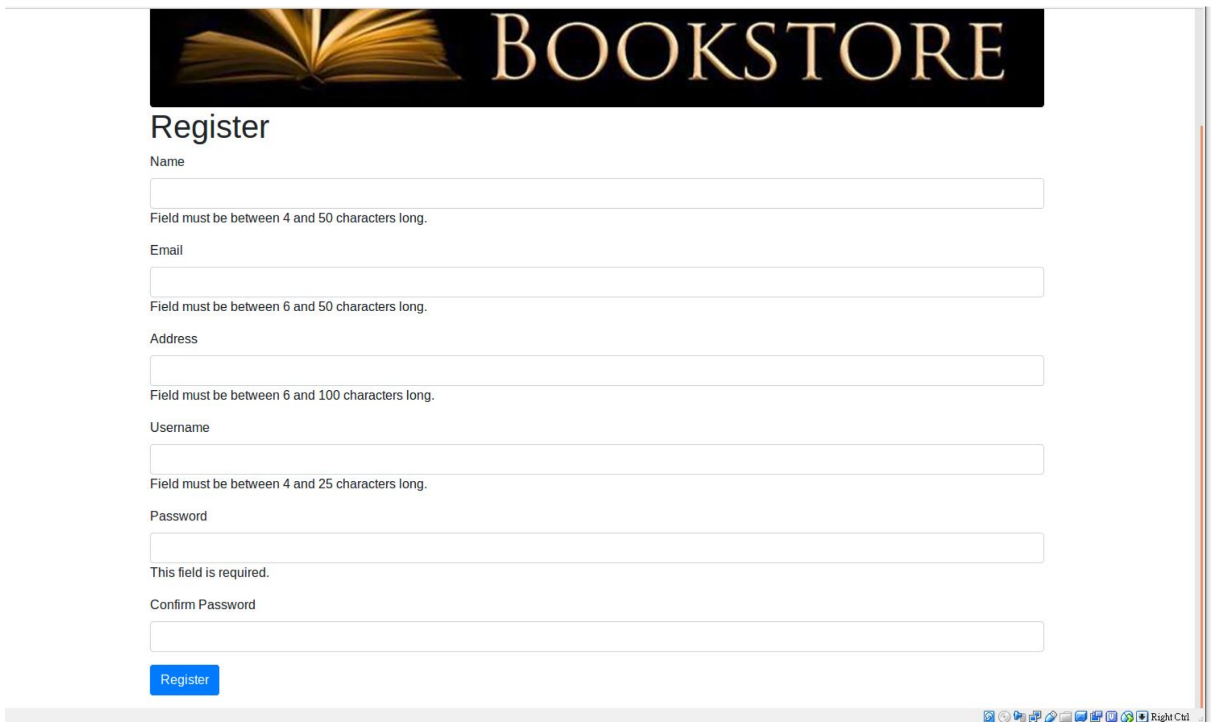
WTForms has been implemented in this system. When new users register, they need to input a list of valid data for registration, so WTForms has been used to control the input of the user.

```
# Register Form Class
class RegisterForm(Form):
    # Wtforms
    name = StringField('Name', [validators.Length(min=4, max=50)])
    email = StringField('Email', [validators.Length(min=6,
max=50)])
    address = StringField('Address', [validators.Length(min=6,
max=100)])
    username = StringField('Username', [validators.Length(min=4,
max=25)])
    password = PasswordField('Password', [
        validators.Length(min=4, max=25),
        validators.DataRequired(),
        validators.EqualTo('confirm', message='Passwords do not
match')
    ])
    confirm = PasswordField('Confirm Password')


# User Register
@app.route('/register', methods=['GET', 'POST'])
def register():

    # Instantiate object of RegisterForm
    form = RegisterForm(request.form)

    # Create cursor
    cur = mysql.connection.cursor()

    # Check if there is a same username
    result = cur.execute("SELECT * FROM users WHERE username =
%s", [form.name.data])
```

```
    # Close connection
    cur.close()

    # If the username has been used
    if request.method == 'POST' and form.validate() and result >
0:
        error = 'This username has been used. Please pick another
one.'
        return render_template('register.html', form=form, error =
error)

    # if the data is valid
    elif request.method == 'POST' and form.validate():
        name = form.name.data
        email = form.email.data
        address = form.address.data
        username = form.username.data
        password = form.password.data

        # Create cursor
        cur = mysql.connection.cursor()

        # Insert a new record to table 'users'
        cur.execute("INSERT INTO users(name, email, username,
password, address) VALUES(%s, %s, %s, %s, %s)", (name, email,
username, password, address))
```

Every input data in each field has different limitation. The field name needs to be in length 4 to 50 characters.The field email needs to be in length 6 to 50 characters. The field address needs to be in length 6 to 100 characters. The field username and password need to be in length 6 to 100 characters. The confirm field is required to be identical to the input in the password field.

If the registration is failed, the error messages will be shown.



Register

Name

Field must be between 4 and 50 characters long.

Email

Field must be between 6 and 50 characters long.

Address

Field must be between 6 and 100 characters long.

Username

Field must be between 4 and 25 characters long.

Password

This field is required.

Confirm Password

Wrap

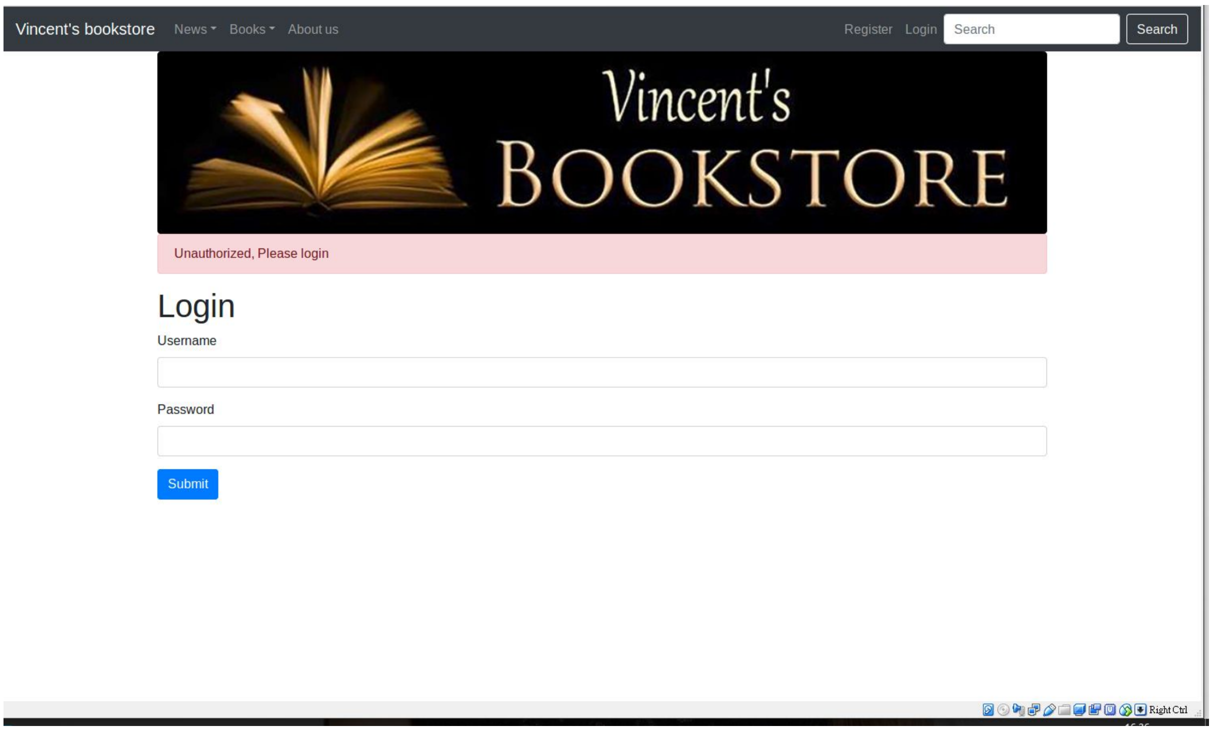
Wraps has been implemented in this system. Wraps is used to prevent user reached certain pages without logged in, such as the shopping cart page, and the order history page. If the users' session is not logged in, they will be prompted a message 'Unauthorized, Please login'

```python
# Check if user logged in
def is_logged_in(f):
    @wraps(f)
    def wrap(*args, **kwargs):
        # If user is logged in --> continue
        if 'logged_in' in session:
            return f(*args, **kwargs)
        # If user is logged out and want to go to some pages like 
shopping cart
        else:
            flash('Unauthorized, Please login', 'danger')
            return redirect(url_for('login'))
    return wrap
```

If there is an @is_logged_in under the @app.route, it means users need to be logged in in order to manipulate the function below.

```
# Add to shopping cart
@app.route('/add-order/<string:category>/<string:item_per_page>/<string:page_no>/<string:bookid>', methods=['POST'])
@is_logged_in
def add_order(category,item_per_page,page_no,bookid):
  # ...
```

When we attempt to go to shopping cart by typing 127.0.0.1:5000/shopping-cart, it will show:



Unauthorized, Please login

Login

Username

Password

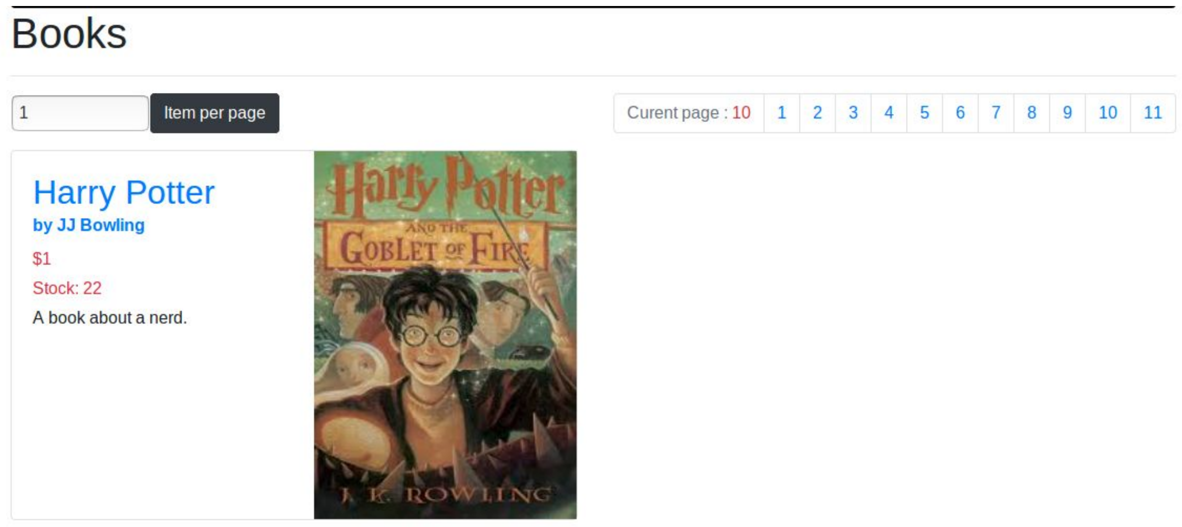
Dynamic Url

Dynamic url used this system. When it is in the ‘Books’ page, it shows a list of books. Since the number of books will change very often, it is not possible to make certain number of templates of html to spare for certain number of books. If the number of books increase, the page number should increase by itself. Therefore, there is only page to show all the books. However with the dynamic url, the page can determine to split to the books to how many page, which item should be shown and hided whatever the user like to have how many items for each page.

```
# Category of books
# Dynamic url
@app.route('/category/<string:category>/item-per-page=<string:item_per_page>/<string:page_no>', methods=['GET','Post'])
def category(category,page_no,item_per_page):

    if request.method == 'POST':
        item_per_page =request.form['item_per_page']
    item_per_page=int(item_per_page)

    if item_per_page <= 0:
        flash('Cannot be shown smaller than 1','danger')
        item_per_page = 4
```

```
    # Create cursor
    cur = mysql.connection.cursor()

    # Get books matched that category
    if category == 'books':
        cur.execute("SELECT COUNT(*) FROM books ORDER BY import_date DESC")
    else:
        cur.execute('SELECT COUNT(*) FROM books WHERE category = "%s" ORDER BY import_date DESC' %category)

    data = cur.fetchone()
    no_of_books = data
    no_of_books = data['COUNT(*)']
    # Below is very similiar to /book/<page_no>
    # Calculate how many page to render
    # Calculate which item should start to render


    if no_of_books % item_per_page == 0 :
        no_of_pages = no_of_books / item_per_page
    else:
        no_of_pages = no_of_books / item_per_page + 1


    if category == 'books':
        cur.execute('SELECT * FROM books ORDER BY import_date
DESC')
    else:
        cur.execute('SELECT * FROM books WHERE category = "%s"
ORDER BY import_date DESC' %category)

    all_book_info = cur.fetchall()
    all_book_info = list(all_book_info)

    cur.close()

    show_list = []

    if page_no == 1:
        for one_to_four in range (0,item_per_page):
            try:
                j=all_book_info[k]
                show_list.append(j)
            except IndexError:
                break

    else:
        temp_page = int(page_no) -1
        start_item = temp_page*item_per_page
        for k in range(0,item_per_page):
            try:
                j=all_book_info[start_item+k]
                show_list.append(j)
            except IndexError:
                break

    if no_of_books > 0:
        header = category[0].upper() + category[1:]
        return render_template('category.html', item_per_page = 
item_per_page,header=header,category = 
category,no_of_pages=no_of_pages,all_book_info=all_book_info,page_
no=page_no,show_list=show_list)
    else:
        error = 'Category: "'+category+'" Not Found'
        return render_template('home.html', error=error)
```

If user chooses to have 1 item per page, there is 11 books and it will generate 11 pages.



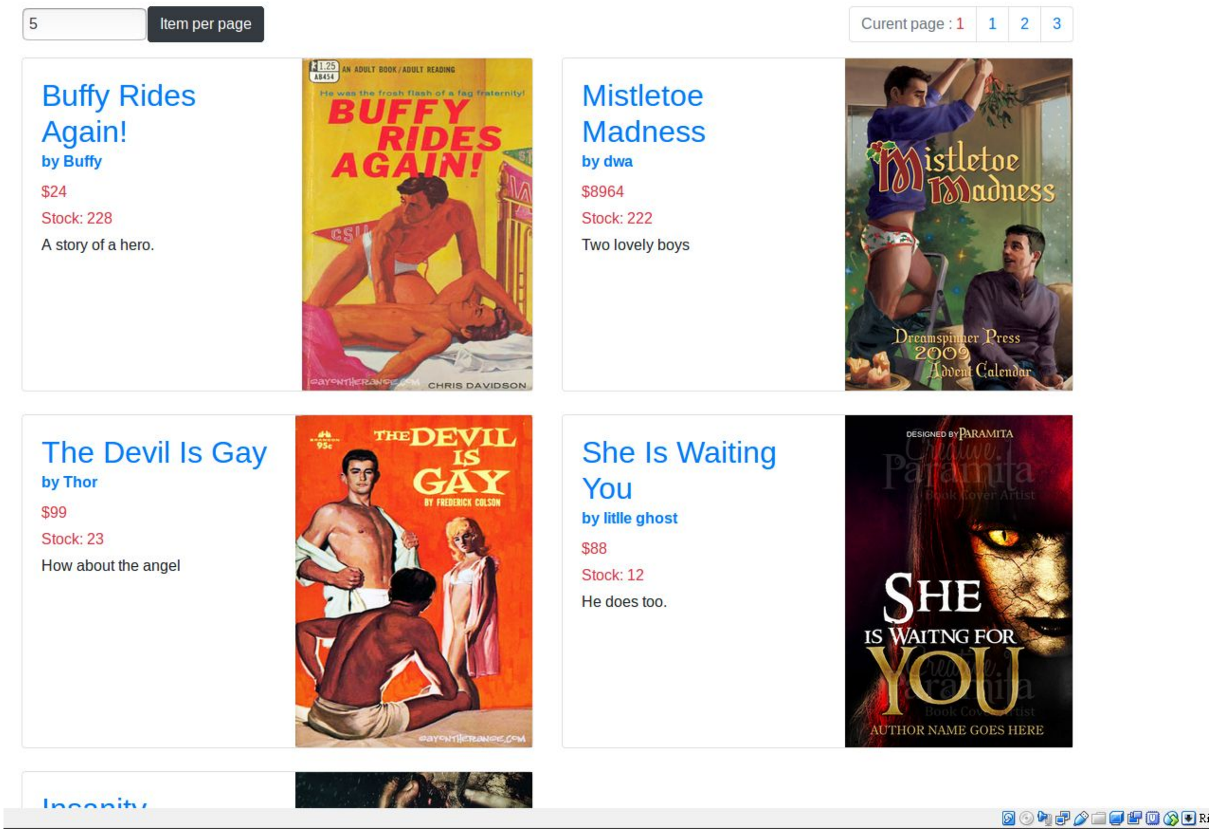
Books

1 Item per page

Curent page : 10 1 2 3 4 5 6 7 8 9 10 11

Harry Potter
by JJ Bowling
$1
Stock: 22
A book about a nerd.

If user chooses to have 5 item per page, there is 11 books and it will generate 3 pages only.



5 Item per page

Curent page : 1 1 2 3

Buffy Rides Again!
by Buffy
$24
Stock: 228
A story of a hero.

Mistletoe Madness
by dwa
$8964
Stock: 222
Two lovely boys

The Devil Is Gay
by Thor
$99
Stock: 23
How about the angel

She Is Waiting You
by litlle ghost
$88
Stock: 12
He does too.

Improvement

The biggest flaw of the system is that if the staff wants manage the system, they are required to have techniques of MySQL. It is believed that the system will be greatly improved the staff can manage the system with in the interface of the website. It can be achieved by making a interface on the web page and it can support the staff the amend the books, price and stock within the web page with the staff account.