

205CDE Developing the Modern Web

Reflective Report

Student Name:

Wong Yat Cheong Jacky

Student ID:

55202288

Due Date:

10th April 2019

Video on introducing my Website

YouTube Link:

<https://youtu.be/VpHVQKTBzPk>

Program Code on GitHub

Window use:

<https://github.com/jackywongboy/flask-online-trading-platform-windows>

Linux use:

<https://github.com/jackywongboy/flask-online-trading-platform-Linux>

During this project, I have use Python, HTML, CSS, JavaScript and MySQL to complete my website. During the coding period, I use Windows and Oracle VM VirtualBox in Linux Platform to code, coding by Visual Studio Code. Connecting MySQL database by XAMPP without any php code.

I use some Flask and Jinja's special functions, such as Template Inheritance and Synopsis and request arguments. On template inheritance, I create a base.html file to be the simple html skeleton document for my website. What was inside is the navbar and the footer.

```
{% block head %}  
{% endblock %}
```

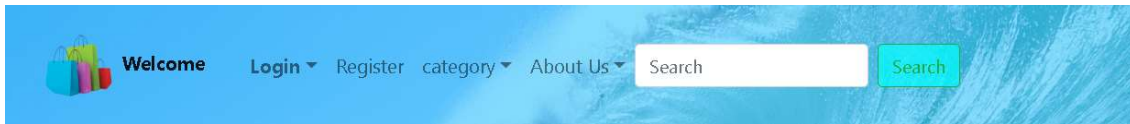
On the other child template, I only need to call out the base.html, then it will have the basic style of my base.html.

```
{% extends "base.html" %}
```

Other new details in different html page just need to code inside the statement.

```
{% block main %}  
.... new details....  
{% endblock %}
```

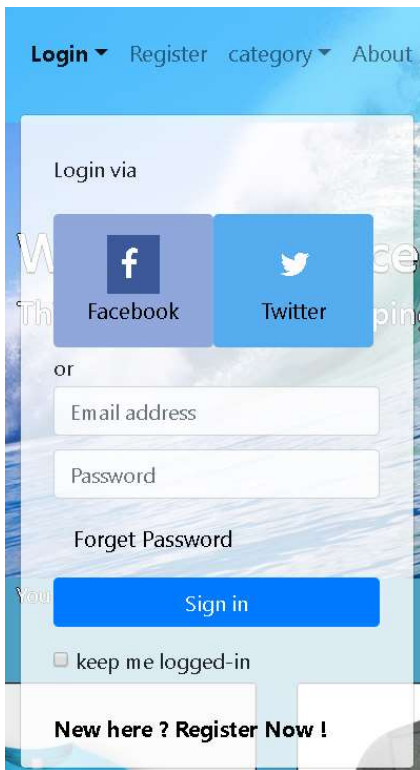
On the nav-bar,



I use CSS style to fix it on the top of the website and small amount of transparency to convenient users no need to row up back to the top and have a better user-interface.

```
.navbar {  
    position: fixed;  
    ...  
}
```

Also, I create a login page in a drop box of the nav-bar. User can login without redirecting to another page, decrease unnecessary nuisance when user login.



Also, I create a search function to search for all products inside the database, this function is based on python by collecting the values inputted by user and search in product's database.

```
@app.route('/search', methods=['POST', 'GET'])
def search(): #search function on the nav-bar , search by products name
    if 'search' in request.args:
        find = request.args['search']
        cur = mysql.connection.cursor()
        query_string = "SELECT * FROM products WHERE pname LIKE %s ORDER BY
pid ASC"
        cur.execute(query_string, ('%' + find + '%',))
```

Finally, I use synopsis statements to change the nav-bar when user logged in and out.

```
{% if session['inputname'] %}
    . . . <!--username and logout -->
{% else %}
    . . . <!--login and register -->
```

Session is also use on the shopping cart function; I also use session to record users what they add into their shopping cart. After logging out, session will be cleared, and the shopping cart record will be cleared too.

```
pid_in_cart = session.get('cart',[])
productno = len(pid_in_cart)
```

On the main page of my website, I call products randomly from database to show on the page.

```
values = 'tshirt'
cur.execute("SELECT * FROM products WHERE category=%s ORDER BY RAND()
LIMIT 4", (values,))
tshirt = cur.fetchall()
```

So that, when the main page is reloaded, different products will appear, which can let user feel more curiosity. Also, I add some jQuery and CSS to make better user-interface, such as fade in the title and animate when users' mouse is pointing on the product card.

On category page, it will print out all the products which designation by category.

```
values = 'tshirt'
cur.execute("SELECT * FROM products WHERE category=%s ORDER BY pID ASC",
(values,))
tshirt = cur.fetchall()
```

When user click on the product, it will redirect to a product details page. This is a method of request arguments. First, make a link and set the link by product ID after '?', as '?' is a definition for request function to get the product ID.

```
<a href="/{{product.category}}?pid={{product.pID}}">
```

when user click on it, it will redirect to viewproduct.html and get the product ID which was found in the link. After that, it will search and print out details from the database to viewproduct.html by the product ID.

```
if 'pid' in request.args: #pid is product ID
```

There is a full-scale image of the product, user can click on it to zoom more bigger to look more details about the product, this function is created by JavaScript.

Non-clicked:

Clicked:



```

$(img)
.addClass('zoomImg')
.css({
    position: 'absolute',
    top: 0,
    left: 0,
    opacity: 0,
    width: img.width * magnify,
    height: img.height * magnify,
    border: 'none',
    maxWidth: 'none',
    maxHeight: 'none'
})
.appendTo(target);

```

This function is worked by JavaScript to get the position of user's mouse and calculate the zoom scale of the picture and change the size of the picture by changing the CSS style. So that it can show out a zoomed picture when user's mouse is click on different position. But, if the picture's resolution is too low, this function will work.

When user click the order button to buy, user will need to input this Full name as receiver and choose the quantity. After testing the information, Flask will upload this order record to order's database.

In user personal page, user can change different personal details, including username and email. Also, user can update his profile picture in personal page. The function of profile picture is based on Flask-avatars, user upload an image. After that, user can crop it into a size which fix the scale. Finally, Flask will save the image in a path which already set and upload the link of the image to the user's database.

On changing personal information, Flask first get information of the user by the cookie which created after the user logged in, then search his personal information from user's database and return this information back to the HTML, so that the form will show out the details which user already set when he registers. Therefore, the user can change his information more convenient.

Username Name	<input type="text" value="qwe"/>
User Email	<input type="text" value="qwe@qwe"/>
Mobile Number	<input type="text" value="123123123"/>
Address	<input type="text" value="password:qwe"/>

On Change password function, as password is a secret item, I had hashed all password and upload hashed password to database which can make sure that no one can see the real password including admin.

userpw

```
$2b$12$7xixMct2tLAcUmTQMCiOSeh.CZxqvBzYrKj/bVNjBB1...
$2b$12$WTeHS0MleXE0Z5r4v6.sm.1nOOHrORYT.YH7qdGyFSr.
$2b$12$CIBRiSulAXpln0LOIfM4AOoIWPsxckyvmtykSq3j/li...
$2b$12$20i8ErhDOJt7.m3sdiwVQ.f1S9nob40P/d1X2ljmmdB...
$2b$12$VUIRjHm./cIJW8uxUfIJYO35eLwGqrsS8GC0dq56qJc...
$2b$12$8Ag3ThJ12ziCpVPr5VYs/.iwFAMfNM7TRvKI86RH3Y...
$2b$12$NNqxoar9rtIra5Wl0ilBmXe3pHrbafDJ/WIS/q4FCsHD...
```

Because of hashed password, testing method will be more complex. First, I will hash the values inputted by users, then check these two hashed codes aren't same to find out is the user input value a correct. If he input a correct password, I will hash the new password and upload back to the user's database. To be careful that the hashed password is so long, make sure that the database has enough space to save the whole hashed password.

On the forgot password function, I use Flask-mail to complete it. As hashed password cannot reverse back to un-hashed, I choose to generate a new password for the user. First, user need to input this register email, I will check this email isn't presence from the database, if yes then send an email with 4 random code generated by python random function as a new password to the user. Meanwhile, update the new password to the user's database.

On register function, I use python code to test the information inputted by the user isn't correct, also, I will check for the username and user email isn't duplicate with other users. If the above information is incorrect, I will redirect to an error page with setting an error cookie to print out in the error page.

```
else:
    error = "Username has been used "
    session['error'] = error
    return redirect(url_for('error'))
```

After checking all the information, Flask will upload a new user into the user's database and set the personal profile link as a standard one. So that even the user doesn't upload image, it will also show a standard image.

On unload product page, I use Flask uploads function to upload picture in to file by a path which set already. The link of the product will be upload to database will in product ID, so that every time I call out this product in database, I can get its ID and other details include its link.

```
file = request.files['picture']
if name and price and description and quantity and category and file:
    pic = file.filename
    photo = pic.replace("'", "")
    picture = photo.replace(" ", "_")
    if picture.lower().endswith(('.png', '.jpg', '.jpeg')):
        save_photo = photos.save(file, folder=category)
        if save_photo:
            cured.execute("UPDATE products SET pname=%s, pprice=%s,
description=%s, quantity=%s, category=%s, plink=%s WHERE pID=%s ",
                (name, price, description, quantity, category, pic,
product_id,))
            mysql.connection.commit()

            flash('Data updated', 'success')
            return redirect(url_for('uploaded'))
```

To make sure that the file is support on most browser, I create a test for the picture. First, I will replace all symbol to absolute types. Next, check the file by checking the ending by its file types. If the file type is support, then will save the photo to the path which set already. After that, return Flash messages in the HTML.

On Flash message, I first create a html for Flash format which can able to show a good style flash message in HTML.

```
<!--- Flask function Flash use page --->
{% with messages = get_flashed_messages(with_categories=true) %}
  <!-- Categories: success (green), info (blue), warning (yellow),
danger (red) -->
  {% if messages %}
    {% for category, message in messages %}
      <div class="alert alert-{{ category }} alert-dismissible my-4"
role="alert">
        {{ message }}
      </div>
    {% endfor %}
  {% endif %}
{% endwith %}
```


Here I use new WITH format to combine two html together as extends function is already used.

On product details page, I create a comment function.

Comments : (5)

Please enter


Send



qwe

oh yeah , buy it with discount XDDDD


2019-03-16 15:57:38



dfg

kisfydgkiSfegkuiSYGfdkuiSYfdkguySGFDffffffffffffslkdf
ygKUSFYDGkuSfydgikuSfydgokUSIfydgukSfydgkusfydgk
uSfydgkuSfdgkuSdgfkuSdgfkuSYGdfkusgkuSfydgkuSYGf
dkuSfdgkuSydgfkuSfydgkSUDfygkSUfydgkUSFDG

2019-03-16 13:00:58



dfg

yeayrayaasdasdasjkhgdjkkashvfgcxjakhgsfdujkavhsyxgcjay
gvfsdjkaGFSXZckujahgsfvdkuzYgf

2019-03-16 12:52:31

The comments saved in database and have its primary key of comment ID, so even the website reload, the comments will not be disappeared. Also, the comment database is linked with foreign key by User ID, which means that when user change their username or profile picture, comments details will also be updated.

On the login box, there is a button for login by social media, click on it and will redirect to admin login page. Admin account is separate from user's database, which means that user may not able to login as admin or register an admin account. After logged in as admin, there are 3 buttons, they are "all user", "all product", "all orders". Same as user, admin may able to check all users' details, all products' details and all orders' details and able to update or delete the records. To show clearly, I used a Bootstrap table to list out different information.

On reflection, I admit that I didn't use well on Flask-WTF to create forms for register and login, WTF provides a more convenient testing function on the word length and more, next time I will try to use more on it and shorten the duplicate coding on testing.

On the database, I should make a greater database for products, including more product details, such as product level, the color, pattern, size, material etc. Which can let users to search products and organize the products in a more details way. Also, I may expand the products image link to let product have more than one image to save in the database. More images about the product is better than only one image.

On user-interface, I should add more animate like CSS animate and jQuery to get richer on visual enjoyment, decrease the boring on non-movement words. Also, besides the comment function, next time I may try to add a marking function too. User who has bought the product can give a mark for the product, which can let other users to evaluation this product not only by comments. On seller side, I may try to create a 'shop' for sellers, seller can upload products in his 'online shop' and their followers can follow the 'online shop' and can get notice when the shop upload new product. Search function can also able to search by shop name or owner. This idea can be more user friendly.

Also, I would like to add discount function on my next website project. For example, calculate the user's buying records, if he bought more than 10 items from this website, give him a discount coupon. Also, gives coupon as a present by user's birthday. On seller side, give seller to create discount product, able to promotion different kinds of products. The kind of function may able to stimulate user's purchase intention, increase the website exposure.

Finally, in the future, I would like to add an online chat room between the buyers and sellers, also between users and admins. Instant messaging is popular and useful nowadays, it should be a main function in real projects and be one of the most changeling function for me to try.

Thank you for reading!