Comp2011 Web Application Development

(Coursework-2) -Report (Htoo Aung)

Wicked-Lab

A Wicked-Lab is men's exclusive perfume fictional brand that intends to sell its products to its members. This can be seen from the evidence that the user will not be able to see it's products without creating an account.

In this web application, we assume that the user saves or adds each individual products that are on the sale page to their card so that they can check-out at some point in the future. The user may also remove items that they have saved or added to their cart at any point.

This website use Bootstrap and most of the objects or html tags are stored in the classes: jumbotron or container to ensure that the website fits the layout well and it is responsive. Jinja templating is also used to manipulate the logic of the control flow of the program. Template inheritance has also been used for optimal code-quality.

Log-in credentials

The following is the sample existing email and password for log-in!

Email: sc21hta@leeds.ac.uk

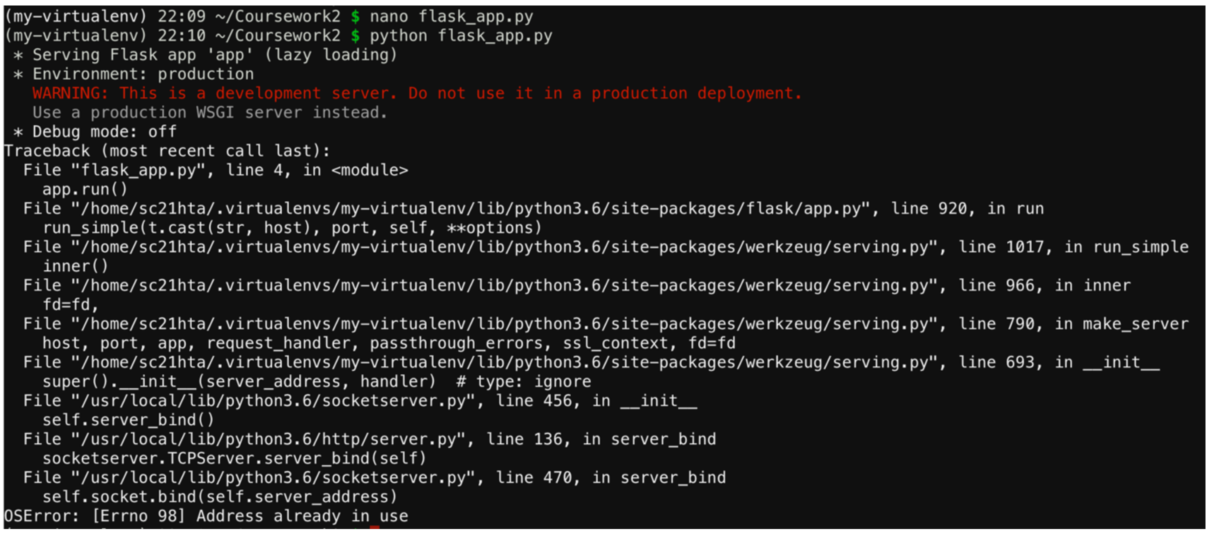
Password: Birthday=131001!

Please do bear in mind that registering a new account and logging to the new account will also work as the web-application is not hard-coded to certain usernames and passwords.

Deployment

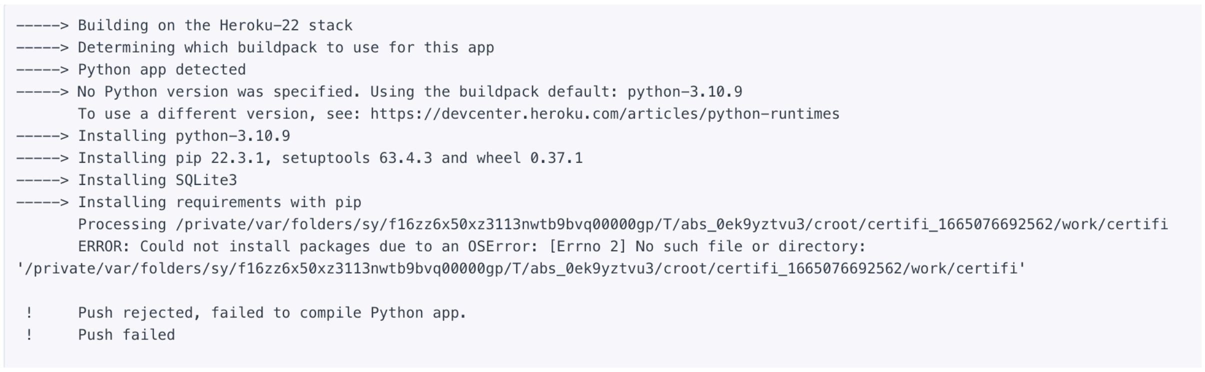
Throughout the development of this web-application, there have been several attempts on deploying this website using the platforms: pythonanywhere.com and Heroku.com. None of these platforms works for my website implementation and both shows the same build error.

Pythonanywhere.com:



```
(my-virtualenv) 22:09 ~/Coursework2 $ nano flask_app.py
(my-virtualenv) 22:10 ~/Coursework2 $ python flask_app.py
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
Traceback (most recent call last):
  File "flask_app.py", line 4, in <module>
    app.run()
  File "/home/sc21hta/.virtualenvs/my-virtualenv/lib/python3.6/site-packages/flask/app.py", line 920, in run
    run_simple(t.cast(str, host), port, self, **options)
  File "/home/sc21hta/.virtualenvs/my-virtualenv/lib/python3.6/site-packages/werkzeug/serving.py", line 1017, in run_simple
    inner()
  File "/home/sc21hta/.virtualenvs/my-virtualenv/lib/python3.6/site-packages/werkzeug/serving.py", line 966, in inner
    fd=fd,
  File "/home/sc21hta/.virtualenvs/my-virtualenv/lib/python3.6/site-packages/werkzeug/serving.py", line 790, in make_server
    host, port, app, request_handler, passthrough_errors, ssl_context, fd=fd
  File "/home/sc21hta/.virtualenvs/my-virtualenv/lib/python3.6/site-packages/werkzeug/serving.py", line 693, in __init__
    super().__init__(server_address, handler)  # type: ignore
  File "/usr/local/lib/python3.6/socketserver.py", line 456, in __init__
    self.server_bind()
  File "/usr/local/lib/python3.6/http/server.py", line 136, in server_bind
    socketserver.TCPServer.server_bind(self)
  File "/usr/local/lib/python3.6/socketserver.py", line 470, in server_bind
    self.socket.bind(self.server_address)
OSError: [Errno 98] Address already in use
```

Heroku.com:



```
-----> Building on the Heroku-22 stack
-----> Determining which buildpack to use for this app
-----> Python app detected
-----> No Python version was specified. Using the buildpack default: python-3.10.9
       To use a different version, see: https://devcenter.heroku.com/articles/python-runtimes
-----> Installing python-3.10.9
-----> Installing pip 22.3.1, setuptools 63.4.3 and wheel 0.37.1
-----> Installing SQLite3
-----> Installing requirements with pip
       Processing /private/var/folders/sy/f16zz6x50xz3113nwtb9bvq00000gp/T/abs_0ek9yztvu3/croot/certifi_1665076692562/work/certifi
       ERROR: Could not install packages due to an OSError: [Errno 2] No such file or directory:
'/private/var/folders/sy/f16zz6x50xz3113nwtb9bvq00000gp/T/abs_0ek9yztvu3/croot/certifi_1665076692562/work/certifi'

 !     Push rejected, failed to compile Python app.
 !     Push failed
```

3-tier Architecture

A 3-tier architecture consists of a presentation tier, an application tier and a data tier. In a 3-tier architecture the communication takes between all the tiers.

Presentation tier

1) This involves users filling in the forms assisted by built-in flask library "flask_wtf" which allow classes to inherit from "FlaskForm" where the forms are constructed which are then shown after rendering of the templates and the latter validates the user inputs.

   For example, a class "LoginForm" from forms.py creates a form-label and a form on the front-end.

2) HTML/CSS and Bootstrap were used in this web application to provide smooth and seamless user experience.

Application tier

In this tier, the information collected is processed.

1) Jinja templating has been used to carry out checks and validations and provides the user with the output on the front-end depending on their choices and actions after following a set of rules and conditions.

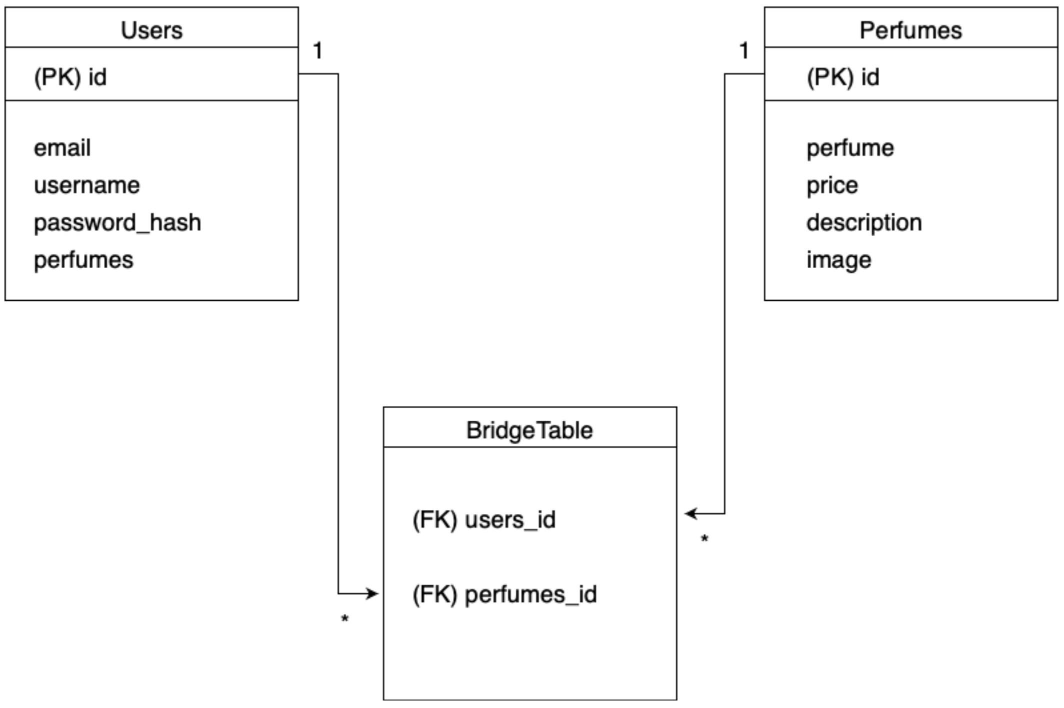
   For example, some nav-items have limited access only to the authorised users. Otherwise, unauthorised or normal-visiting user may have different level of access and privilege.

2) At the very first stage of data-processing before they were accepted or allowed for further processing, they were validated first, this is done by a python library such as "wtforms". For example, email form checks that the email form actually receives an email or not.

   Once all the fields are validated it is stored in the database, this is done by 'POST' method. The data sent from the back-end to the front-end is done by 'GET' method.

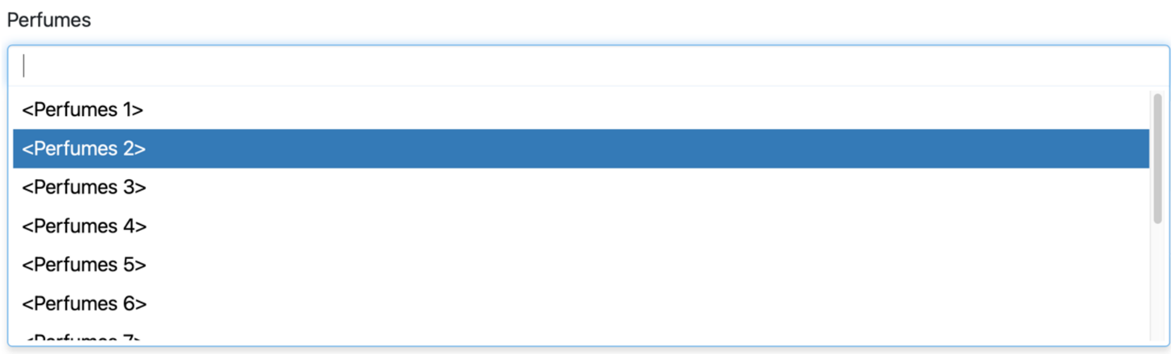
Database tier

In the case of Wicked Lab, a user can have multiple products on its cart and a product can be added to the cart by multiple users. Thus, there exists a many-to-many relationship, by bridging the tables.



Enabling admin views:

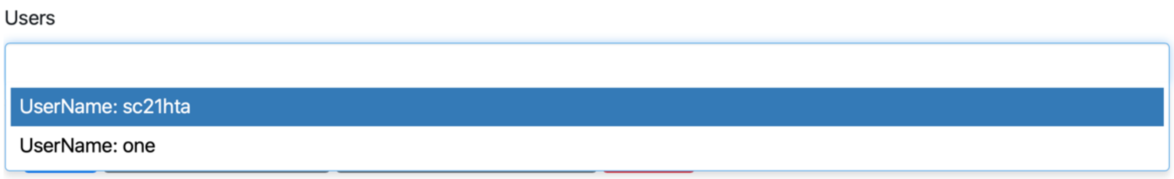
A user can have many perfumes



Perfumes

<Perfumes 1>
<Perfumes 2>
<Perfumes 3>
<Perfumes 4>
<Perfumes 5>
<Perfumes 6>

A perfume can be bought by many users:


Users

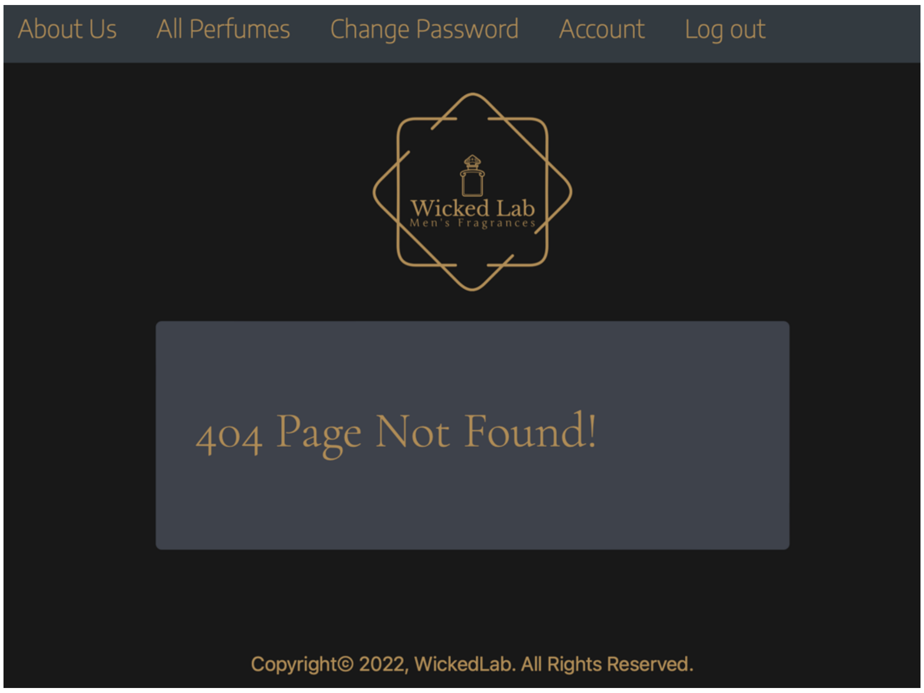
UserName: sc21hta

UserName: one

Wicked Lab’s special features

1.) Most views are protected by “@login_required” this implies that only authorised users have access to certain webpages.

2.) Error 404-pages for pages that are not available on particular user requests!



404 Page Not Found!

Copyright© 2022, WickedLab. All Rights Reserved.

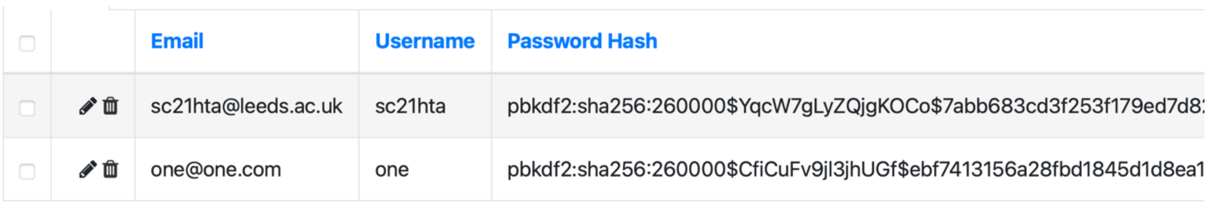
Try typing any random URL after “http://127.0.0.1:5000” such as http://127.0.0.1:5000/fgddg

3.) Error 403-pages were also incorporated in the templates at least in theory, if unauthorised attempts to view webpages that were prohibited from their view. Such eventualities didn’t seem to occur as stated by the first point of this section.

This shows somewhat robustness of a system on a basic-level.

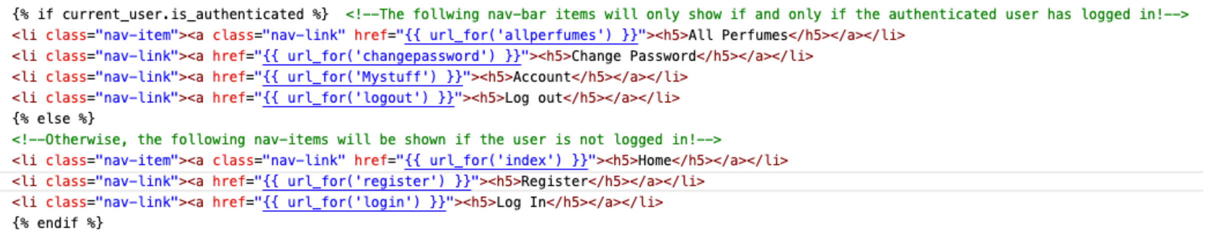
4.) Only hashed-passwords are saved in the database.

```
from werkzeug.security import generate_password_hash,check_password_hash
```



		Email	Username	Password Hash
☐	✎ 🗑	sc21hta@leeds.ac.uk	sc21hta	pbkdf2:sha256:260000$YqcW7gLyZQjgKOCo$7abb683cd3f253f179ed7d8
☐	✎ 🗑	one@one.com	one	pbkdf2:sha256:260000$CfiCuFv9jl3jhUGf$ebf7413156a28fbd1845d1d8ea1

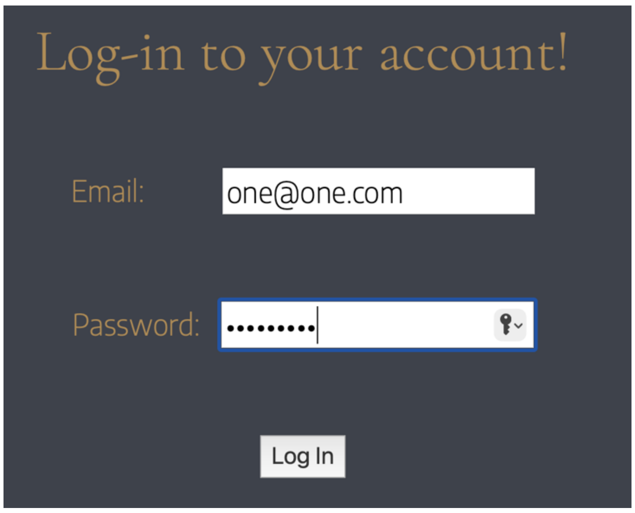
5.) After logging out right after logging in, the user will see limited features and functionalities provided by this web-application.



```
{% if current_user.is_authenticated %}  <!--The follwing nav-bar items will only show if and only if the authenticated user has logged in!-->
<li class="nav-item"><a class="nav-link" href="{{ url_for('allperfumes') }}"><h5>All Perfumes</h5></a></li>
<li class="nav-link"><a href="{{ url_for('changepassword') }}"><h5>Change Password</h5></a></li>
<li class="nav-link"><a href="{{ url_for('Mystuff') }}"><h5>Account</h5></a></li>
<li class="nav-link"><a href="{{ url_for('logout') }}"><h5>Log out</h5></a></li>
{% else %}
<!--Otherwise, the following nav-items will be shown if the user is not logged in!-->
<li class="nav-item"><a class="nav-link" href="{{ url_for('index') }}"><h5>Home</h5></a></li>
<li class="nav-link"><a href="{{ url_for('register') }}"><h5>Register</h5></a></li>
<li class="nav-link"><a href="{{ url_for('login') }}"><h5>Log In</h5></a></li>
{% endif %}
```

6.) This hides the form thus limiting the threats from some spywares, not entirely secure on its own.

```
{{form.hidden_tag()}}
```


Log-in to your account!

Email: one@one.com

Password: ••••••••••

References:

1.) Bootstrap: https://getbootstrap.com
2.) Flask: https://flask.palletsprojects.com/en/2.2.x/
3.) Fonts: https://fonts.google.com
4.) Deployment, Heroku: https://www.heroku.com, PythonAnywhere: https://www.pythonanywhere.com
5.) Pictures: https://www.pexels.com, https://www.freepik.com