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](mailto: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:

A screenshot of a computer

Description automatically generated with medium confidence

Heroku.com:

Graphical user interface, text, application, email

Description automatically generated

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.

1. 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.

1. 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.

Graphical user interface

Description automatically generated

Enabling admin views:

A user can have many perfumes

A picture containing graphical user interface

Description automatically generated

A perfume can be bought by many users:

Background pattern

Description automatically generated with low confidence

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!

Graphical user interface, application

Description automatically generated

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

1. 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.

1. Only hashed-passwords are saved in the database.

from werkzeug.security import generate\_password\_hash,check\_password\_hash

Graphical user interface, text, application

Description automatically generated

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

Text

Description automatically generated

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

{{form.hidden\_tag()}}

Graphical user interface, application

Description automatically generated

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>