Webshop Application Coursework

Richard Borbely
40283185@napier.ac.uk
Edinburgh Napier University - Advanced Web Technologies (SET09103)

1 Introduction

Keywords – HTML, CSS, JavaScript, Python, Flask, Jinja2, Bootstrap, MongoDB, Webshop, Lollipops

The purpose of this report is to describe how this particular web application was produced using various software frameworks and also what functionalities have been successfully or unsuccessfully implemented. The application features a web shop for a small family business that is producing stylish confectionery. Users have the ability to browse through a wide range of products, add them to their shopping carts, register an account and place and order. The business owner is able to list these orders on a private page.
On the server side, *Python 2.7* with the *Flask* micro framework is used to process and serve the required information, which is then used by *Jinja2* templating engine to generate the pages. Registered user accounts are stored in a secure *MongoDB* database hosted by mLab[**?**].

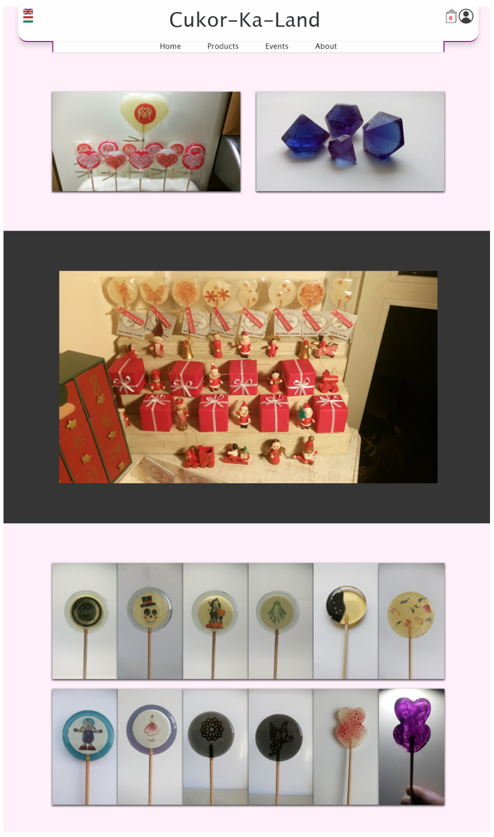


Figure 1: Main page - Navigation bar - Carousels

2 Design

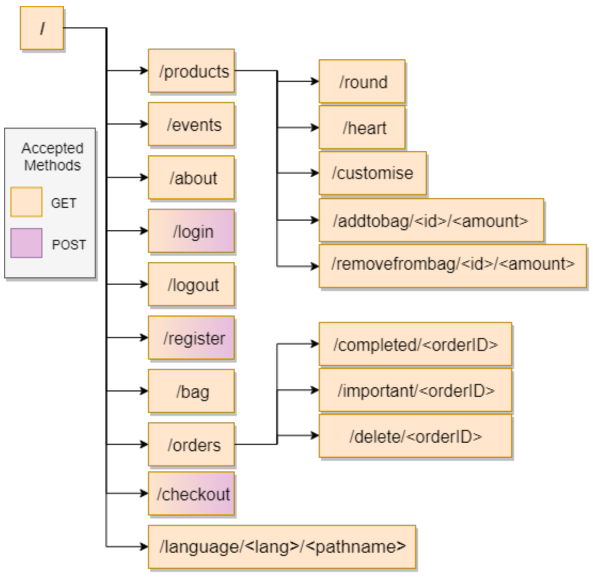


Figure 2: URL hierarchy

The structure of the website is simple and easy to understand, the home page brings a welcoming experience showcasing a range of products. Browsing and adding items to the cart are allowed to anyone landing on the site, while ordering items requires a log-in. The registration and login procedure both require secure handling of user information using encryption and it's implementation is thoroughly described in Section 4.

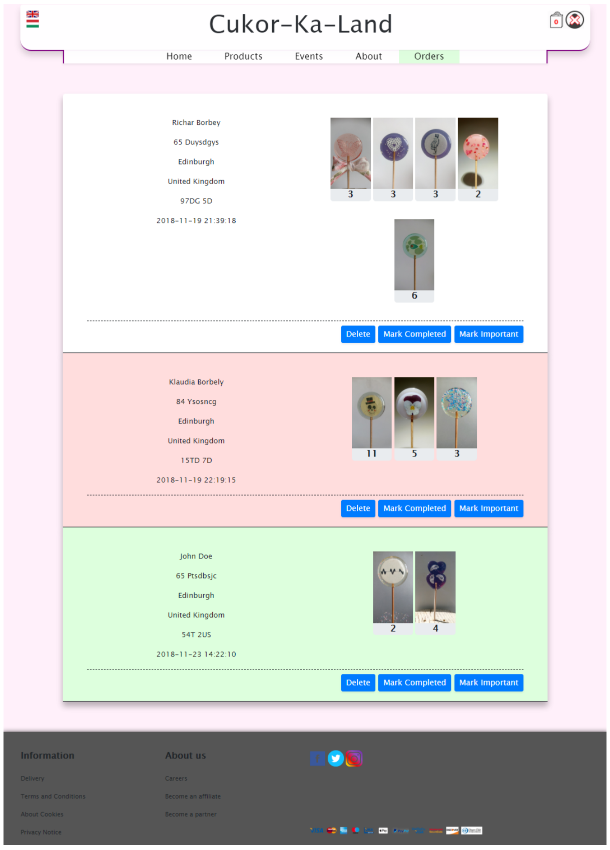
3 Enhancements

The owner of the business needs to be given more control over the application using an interface that is user-friendly. Adding and removing products for example should be an easy task. At this time, it's a tedious process that can only be done by the developer by adding the image of the product to the correct directory using the correct file name. And also the count variable for the number of products needs to be increased manually.

Currently, having a user account gives no additional functionality to the user, other than saving their shopping cart. Additional information should be stored within a user account, like delivery address and payment informations. This would provide a smoother experience when putting through multiple orders. Placed orders should also be accessible for the user with the ability to edit or cancel them.

4 Critical Evaluation

Orders When the owner logs in to the website an additional menu is displayed to be able to see and manage the orders. Each order is printed with delivery information, pictures and counts of the requested products. They are stored in the database and can also be marked completed (*given a green background*) or important (*given a red background*).



Richar Borbey
65 Duysdgys
Edinburgh
United Kingdom
97DG 5D
2018-11-19 21:39:18
3 3 3 2 6

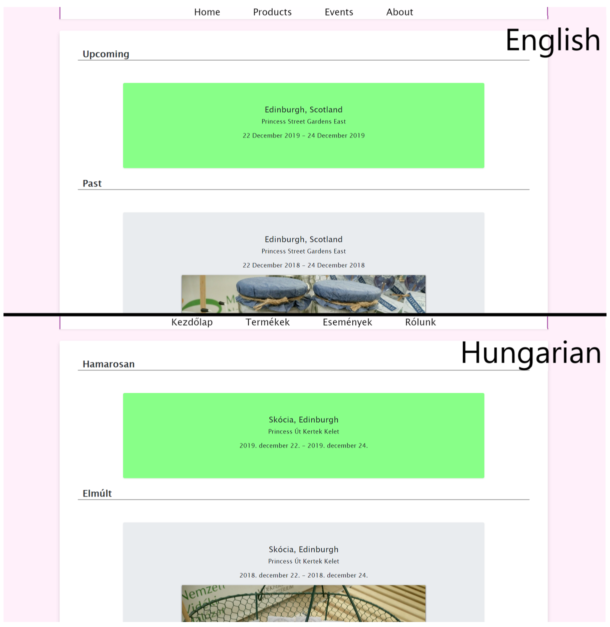
Klaudia Borbely
84 Ysosncg
Edinburgh
United Kingdom
15TD 7D
2018-11-19 22:19:15
11 5 3

John Doe
65 Ptsdbsjc
Edinburgh
United Kingdom
54T 2US
2018-11-23 14:22:10
2 4

Figure 3: Orders page

Shopping Cart As a guest user begins to put products in their shopping cart, those products are stored within a cookie (*session*). Once the user proceeds to log-in, the cookie data gets uploaded to the database to be stored with the user data. This way if the account is accessed in a later time or from another device, the shopping cart will still contain the same items.
Cookie data will not overwrite the database, in case the user has already got some items that are stored in the database, but logs-in with products in the shopping cart, the database is going to have priority and will overwrite the local data.
Perhaps this functionality could be changed so that the local and online carts will merge, rather than losing the local data.

Language The business is based in Hungary, but the production has already expanded to Germany and it is likely to expand even further, therefore adding multi-language support was a must. A fantastic library called *pybabel* has been implemented to achieve this, which not only includes language, but date formatting as well.
Pricing is dynamic, depending on the selected language, which was achieved by simply passing through the correct base price for the products.



Upcoming
Edinburgh, Scotland
Princess Street Gardens East
22 December 2019 - 24 December 2019
Past
Edinburgh, Scotland
Princess Street Gardens East
22 December 2018 - 24 December 2018

Hamarosan
Skócia, Edinburgh
Princess Út Kertek Kelet
2019. december 22. - 2019. december 24.
Elmúlt
Skócia, Edinburgh
Princess Út Kertek Kelet
2018. december 22. - 2018. december 24.

Figure 4: Multi-language support

Configuration Application configuration has been set-up in order to support multiple deployment environments. When launching the application, the required configuration file name needs to be stated as the first given argument, if there is none given it will launch with minimal default configurations.

Mobile Support In the style sheet for the application, everything is defined by using either percentage amounts or '*VW = viewport width*' which results in the web application displaying exactly the same on mobile devices.

5 Personal Evaluation

I very much enjoyed working on this project, knowing that with further development it might be used in the near future. I have faced plenty of challenges implementing features through the course of development, but managed to research and implement them successfully. One of the most interesting challenges was implementing encryption for the stored user data. When successfully registering an account, the E-mail address and password are encrypted and stored in the database. E-mails are encrypted with AES encryption using a 128 bit key, which is hard coded into the application. When trying to log-in the encrypted E-mail value is decrypted and matched against the submitted one. Using the *PyCrypto* library with the help of *sentdex*[**?**]'s YouTube video on AES encryption made it super easy to implement this feature.
The workbook provided sufficient information on password hashing and it's implementation; passwords are hashed and the resulted hash value is stored, which is matched against the submitted password's hash value in the login process. This results in a high-level of security since the application doesn't need to work with the actual password, only the hashed value.

6 References

1. mLab. [online] Available at: https://mlab.com

2. Author: 'sentdex' Title: Python Encryption Tutorial with PyCrypto - *Published on 24 May 2013* [online] Available at: https://www.youtube.com/watch?v=8PzDfykGg_-g [Accessed on 19 Nov 2018].