Coursework Report

Matthew Lynch
40322639@live.napier.ac.uk
Edinburgh Napier University - Advanced Web Technologies (SET09103)

Keywords – Python, Flask, JSON, HTML, JavaScript, CSS, Jinja2, Coffee, Web-app

1 Introduction

The objective of this coursework was to demonstrate an understanding of the Python Flask micro-framework by creating a prototype web application of an online catalogue. "Matthew Lynch" chose a variety of coffee types as his online catalogue. He achieved this by using Levinux in a virtual Linux server that ran from a USB. Within the Levinux environment, he used the command line and Secure Shell (SSH) to allow him to run his web application. He used Python Flask features such as the debugging tool, JSON and Jinja2 templating in the development of his web app.

2 Design

The web application was architect by using a series of stages including planning, research and implementation of the Flask features to develop to overall outcome.

2.1 Planning

To start his coursework Matthew lay out the directories, which created a visual, understand the overall web app. To allow him to create Figure 1 Matthew used an online diagram software called "draw.io" [1] , which allows him to construct flowcharts and process diagrams.

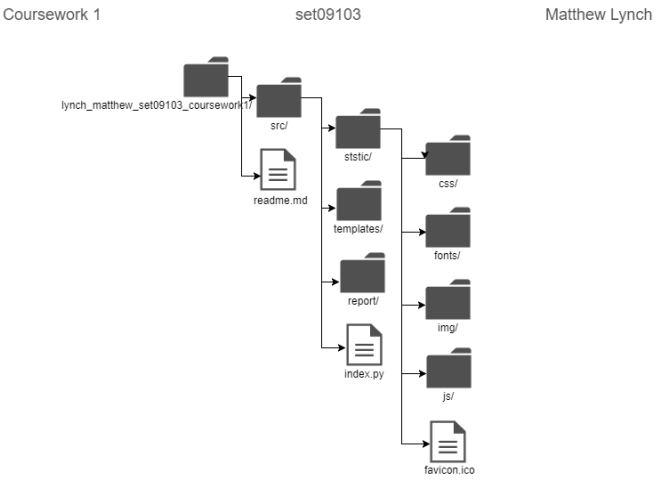

Figure 1: **Directory Hierarchy** Layout of Hierarchy

2.2 Research

Matthew then laid out different possibilities of how the web app could be categorised. For example: espresso, tea, under 200 calories and seasonal drinks. He found that this was an appropriate way to demonstrate categories of coffees and allow the user to retrieve information about the catalogue. He analysed coffee sites like "Cafe Nero" and "Starbucks" to gather ideas and examples of how they organized their coffee catalogue. This allowed him to start the development of the web application.

2.3 Colour

The colour scheme that Matthew has chosen to use on his coffee house web app is simplistic and pleasing. He used the "Bootstrap" framework on elements throughout the web application for the navigation bar and hero image. He also developed a CSS file to give the website more styling and legibility for a better user experience. The CSS file also allowed him to style the images and font in a pleasant way. The images used where acquired from sites "pixabay" [2] and "unsplash" [3] which are royalty free images that can be used for commercial use. The images, CSS and favicon used the flask function "URL FOR"; this creates a URL to the endpoint to provide the content.

```
1 <!-- Links -->
2 <link href="{{ url_for('static', filename='css/bootstrap.min.css↩
      ') }}"
3 rel="stylesheet"/>
4
5 <link href="{{ url_for('static', filename='css/stylesheet.css') }}↩
      "
6 rel="stylesheet"/>
7
8 <link href="{{ url_for('static', filename='favicon.ico') }}"
9 rel="shortcut icon"/>
```

2.4 Flask

The Python file, "index.py" contains each route to the corresponding html template and loads the equivalent JSON data. In Matthew's directory, "src" it contains the main Python file that deploys the web application. It also contains the static directory that stores all the assets. For example: CSS, fonts, images and JavaScript files. The template directory stores all of the html pages.

The html pages consist of many different flask features, which has allowed for effortless development. The feature that Matthew included to help the development process was render template.

```
1 from flask import Flask, url_for, render_template, json, redirect, ↩
      request
2 app = Flask(__name__)
3
```

```
#The Homepage Route
@app.route("/")
def index():
  return render_template('index.html', title='Home')
```

Matthew decided that he would implement a 404 page, in case of any unexpected errors that users may find. This custom 404 page allows the user to re-direct back to the home page and would keep the page stylized for a consistent design across the website.

2.5 JSON

The individual coffee pages consist of data located within a JSON file "coffee-information.json". Matthew had difficulty getting the data in the JSON file to show on the pages. He researched and finally resolved the solution by using a post from stack overflow [4] , which worked.

3 Enhancements

Matthew enhanced the overall functionality and user experience by implementing a search to within his web app as this gives the user a better overall user experience, by allowing them to find what coffee they want by just typing a keyword. Another enhancement that Matthew could implement is social media links. These could be added within the footer and on the individual coffee pages such as Facebook like button. This would allow users to share their favourite coffee with friends and give exposure to Matthew's website.

Matthew could improve the website by implementing a database on the back-end of his coffeehouse site, this would allow for better management and growth. The database would replace any text files that he used instead of a database. Lastly, to enhance the web app Matthew could implant a login and registration function as this would allow for user loyalty, and the ability for users to collect point and receive free extras with their coffee. Overall, these enhancements would give Matthew's website an overall better user experience and add more functionality to his web app.

4 Critical Evaluation

The web application that Matthew has developed meets the overall criteria of the coursework requirement specification. His layout and use of the Jinja2 templating has worked well as this has allowed Matthew not to be repetitive with his code. Within his base.html in the template directory, he has stored his navigation bar, footer and header. This has allowed him not to repeat code unnecessarily as he has brought it in by using the code below in each of his other template pages.

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

{% block content %}

{% endblock content %}
```

Another aspect that has worked well is his overall design of the web application is that Matthew has used a simplistic light colour scheme and layout, this works well in allowing the content to be legible and pop off the page. Matthew's design enables the website to have better usability and accessibility.

Matthew has showed sufficient level of understanding whilst using the Python Flask features. Lastly, another aspect that Matthew could improve on is the JSON implementation and the categories in the catalogue. Within the python file, "index.py", he could have used a better way of implementing the JSON script within the HTML templates. Furthermore, he could have also designed and used better categories to divide his coffees up within the online catalogue.

5 Personal Evaluation

On the completion of the coursework to design and develop a dynamic web application, Matthew has grasped a strong understanding in using Python. He has used Flask, a python library, which has enable his web application to run across different platforms. Matthew was able to familiarize himself with the command line promptly during the class lab sessions. On the occasions when he faced a problem using the command line, Matthew was able to resolve the issue by researching about it on forums' [5].

A challenging part of the coursework that Matthew had to overcome was learning how to use vim: a text editor. "Vim", does not have a graphical interface so Matthew found it hard to construct the web app quickly as he had not used this kind of tool previously. He overcame this by using a mixture of tools available including "Notepad++" and "Atom".

Another area within the coursework that Matthew demonstrated a good understanding for was "Git" and "GitHub" and how they are used. During the coursework, the source code developed by Matthew, was stored on a USB containing the Levinux server. The files then were stored by pushing them into Matthew's personal GitHub repository. [6] Matthew used "WinSCP", "GitHub Desktop" and text editor "Atom" to enable him to efficiently manage and edit his files between a local and a remote computer.

Lastly, the coursework requirement specification wanted Matthew to include a function that extended the lab work. He decided to use a JSON file to load the coffee's name, description and calories into each of the html coffees pages. He overcame this by understanding how to correctly layout his JSON file by using "Jsonlint" [7] which is a JSON validator, and by researching tutorials on how to implement it into his python web application.

References

[1] G. Alder and D. Benson, "draw.io," May 2000.

[2] H. Braxmeier and S. Steinberger, "pixabay," Nov. 2010.

[3] S. L. M. Cho, L. Chesser and A. Woodman, "Unsplash," Aug. 2013.

[4] kOssi, "Flask load local json," Jan. 2014.

[5] user1175807, ""permission denied" error," Oct. 2012.

[6] GitHub, "Adding a file to a repository using the command line," Apr. 2008.

[7] D. Crockford and Z. Carter, "Jsonlint is a validator and reformatter for json," Oct. 2011.