Michael Fleming 25/10/16

CourseWOrk-1

Table of Contents

[Introduction to the web app 3](#_Toc465285792)

[Design Stage 3](#_Toc465285793)

[Typography 3](#_Toc465285794)

[Color Theory 3](#_Toc465285795)

[Layout 4](#_Toc465285796)

[Routing 4](#_Toc465285797)

[Error Handling 4](#_Toc465285798)

[Flask and Jinja 5](#_Toc465285799)

[Generator Function 5](#_Toc465285800)

[Enhancements 6](#_Toc465285801)

[Critical Evaluation 7](#_Toc465285802)

[Evaluation 7](#_Toc465285803)

[References and Resources 8](#_Toc465285804)

# **Introduction to the web app**

The webapp I have designed will be used to display information about top football leagues from around the world. These leagues include the English Barclays Premier League, German Bundesliga and the Scottish Ladbrokes Premier League. The app has a league page where users will be able to select which of the three leagues they wish to view. Once the user has selected their specific league each team will have a little information along with their kit badge. The information included along with the kit badge is full club name, abbreviation, Short name and squad market value.

# Design Stage

The webapp was designed with one main file coursework.py where all routes and functions would be kept. Other folders include templates and static. The static folder contains images and static files such as CSS and JavaScript. The templates folder is where all HTML files for leagues and team’s pages are kept. During the build of the webapp one of my aims was to keep a nice and tidy folder structure with images in the correct folders etc, so all files could be easily found if they had to. Keeping on top of folder management also meant I did not encounter many errors where Jinja templates did not render correctly because they could not be located.

The following tools and languages were used during development

* Twitters Bootstrap Framework
* Flask
* Json
* Python
* Jinja

## Typography

The main font type I plan to use on the web app is Helvetica because it is one of the most famous and popular fonts in the world, but also very easy for users to read on the web due to its dpi (dots per inch), compared to that of sans-serif. Although serif fonts are more formal and professional than sans-serif Helvetica, I still feel it gives a sophisticated and professional feel to the webapp. Helvetica is also a very easy font to read so users will have an enjoyable reading experience on the webapp. Due to these reasons I intend to use Helvetica as the main font choice for the webapp.

## Color Theory

The color scheme I plan to use on the website is a fairly basic one which is overshadowed by the dominant background image which is more prominent. The color scheme I used was a monochromatic scheme. This allowed me to use different shades, tones and tints of my chosen color. The color I choose was #EEEEEE. I felt using this color helped to make the webapp look professional and convey a modern webapp to users. The use of the color #EEEEEE is clear on the landing page of the webapp with a different shade used on the leagues page. The different shade used on the leagues page helps the country’s flags stand out to the user.

## Layout

The layout of the app has been kept basic with the home page not containing much information other than found in the jumbotron. Users are given the option of viewing leagues where they can pick which one they want to see. Once on the leagues page, they will see three thumbnails one for each league and will also see teams currently playing in the division. Leagues are clearly identifiable by the use of the nation’s flag above the name which helps make it stand out the user. The use of the background image is also found on all pages including the 404. Using this image which is vibrant and colorful helps to make the page more appealing to the user.

## Routing

All routing on the webapp I have designed takes place in the coursework.py file. The main routes on the app include /leagues which loads the leagues.html page where users are able to select a league of their choice. Another example of a route is /english\_teams which would serve up the English premier league. If a user was to type /Scottish\_teams, it would load the Scottish league.

## Error Handling

Error handling on the app includes a 404 page which alerts user if they have requested a link or URL that does not exist. If users type a route in the URL search bar in their browser which does not exist, they will be redirected to the 404 page I have designed. On this page will be a brief message stating the page does not exist and the navigation bar will give them options to return home or directly to the leagues page. The 404 route can be found in coursework.py.



## Flask and Jinja

Flask is the micro framework for python that I chose to help build my online catalogue of football leagues. Using this framework, allowed me to use routing, render templates and response objects.

As well as using flask I also used Jinja 2 which is a very well-known template engine. Using Jinja 2 allowed me to parse Json data on to my HTML pages for users to see. Rather than hard coding each individual HTML page using jinja allowed me to call data from Json files which contained all the information for teams in the league.



The image above shows the use of Jinja 2 on my teams.html page. It shows a mixture of HTML and Jinja 2 tags which are indicated by {{ }}. An example of Jinja being used on the page is {{ team.shortName }}. This tag takes team declared at the top of the page and shortName which is the name of a variable inside one of my json files. Using Jinja 2 here saved me a lot of development time as the page was dynamically created with data inside a json file created earlier. Hard coding the page would have meant that I would have to had placed 20 thumbnails on the page. However using a for loop in jinja, did all the hard work for me, helping to save a lot of development time.

### Generator Function

The generator function is something that I added to the webapp in order to cut down on development time. Previously my function to parse data to my team’s page had been confusing and long. The function here simply loads (json. load) the Json file as data file and returns it on the teams.html page. The 1st image below shows the routes I created for all the leagues on the webapp. By returning the page it loads the specific json file declared in the path and outputs onto teams.html which was declared in the generator function. I felt creating this generator function would make it much quicker to add routes for different leagues and saved me a lot of time in this part of the design. I now feel adding more leagues to the webapp in the future would be no problem at all and now requires very little code to do so. The main reason for me creating the generator function was to cut down the complexity of each team route. I achieved this by using the generator function I created.





# Enhancements

Below are the enhancements I would like to make to improve the app.

To improve the functionality, I would add a basic login system. My plan for this would be to ask users on the home page to sign up and login as users. One of my ideas for logged in users is that they are able to add certain leagues to their favorites so that whenever they logged in, they could quickly see the leagues which they had a specific interest in. If I had more time I would have added more leagues for users to view. At present allowing users to add leagues to their favorites would not be worthwhile as there are currently only 3 leagues on the webapp. Adding the login system, I feel would give more control and customization to the user as they can see the leagues which interest them. This would add much needed functionality to the webapp and with more time and more knowledge this would have been possible.

Another feature I would have like to see implemented to the webapp would have current league standings along with information about each team. Adding a live league table for each league on the app would improve the content to attract users. Adding this feature to the webapp would require a web api rather than the static json files which it is using at the moment to pass data to the page. Using a web api would allow the page to dynamically update based on current standings in the league giving users the most up to date information.

One improvement I would like to make to the app is creating a dynamically more-info page for each team. This would allow users to click more info on any team they are viewing then see more detailed information about the team they select. At present there is only basic information on each team in the league adding more content about each team would enhance the webapp for users and allow them to see more.

# Critical Evaluation

Overall I feel the build of the web app has been successful and has enhanced my skills. However, I feel there are aspects of the app that I can work on and improve. One negative side to the web app is its lack of functionality to offer users more details about each team. At present users only see limited information about each team but I would have liked to create a more-info.html page where users could see detailed information about each club on their own dedicated page. This is one downside to the app which greatly limits the functionality. If the app did offer this kind of functionality my next step would have been to add a search functionality where users can search for a team, this would have resulted in the user being redirected to the teams dedicated page. Another improvement that could be made is to use an api to update league standings, otherwise it would have to be manually updated which is not scalable.

A positive feature of the app, is its dynamically created pages for leagues and teams. At the start of the project I thought I was going to have to hard code individual HTML pages. Due knowledge of the languages I was able to create functions which render templates and load the json onto the page. This saved myself a lot of time during the build as I was able to directly pull data from json files. Pulling data from json files rather hard coding each page meant I only had to create one teams.html which could then be rendered by each route, and in future would easily allow me to add more leagues to the app.

I also feel that the app is very easy to navigate for users and makes it clear which leagues we have on offer and how to view them. To help make the app easy for users to navigate I kept the design consistent throughout with the navigation bar kept in the same place. I also kept error handling pages much the same as other pages on the app so users would not feel lost after reading an error handling message.

# Evaluation

Building this webapp has been challenging and enjoyable but has furthered my knowledge and understanding of Python and has extended my technical experience. At the start of the coursework I had very little knowledge of Python and other tools used along the way such as Vim. Throughout the build of the webapp my basic skills of Python and Flask were stretched. This became apparent when trying to parse the Json files to my HTML pages using jinja. This was by far the most challenging aspect of the build and by far the most time consuming part as well. Although this was the most challenging functionality I tried to add to the app, I feel it has enhanced my Python, Flask and jinja skills. I now have a much deeper understanding of how the code works after spending a lot of time trying to implement it. To overcome this issue, I researched examples online and looked back at examples in the workbook. After doing this I finally got the team name to parse through and realized how the code was actually working. Persisting with the jinja here was worthwhile as I now have a deeper understanding of how the code works.

During the build of the web app a lot of languages were used including HTML, CSS, JavaScript, Python Flask and Jinja. At the start of the project I had knowledge of HTML, CSS and JavaScript due to previous experience of using them in website builds. Although my understanding of Python, Flask and Jinja were much lower than that off HTML and CSS. This made the start of the project hard and progress was slow, and I felt adding any form of functionality such as pulling team data from json files was out of my skill set and would not happen during this build. As the build of the app progressed and I also worked through the workbook, I began to get my head around how jinja integrated with Python and Flask to pull data through to the page. As my skills increased , I set about trying to implement this functionality into my app. Once I successfully got jinja pulling the correct data through, my next aim was to cut down on the complexity of each team route in my coursework.py file. At this stage my team routes were confusing and large. At this point I came up with the idea of using a generator function which is described in detail above. Once my generator function was created I was able to easily add new routes in half the time in a much simpler and clearer way.

After completing the build, I am much more confident about creating another web app in the future. A lot of development time during this build was just spent reading error messages and testing to see what I could actually get working. I found that my skills improved over the project and this allowed me to build a sleek and modern web app which sets me up well for future projects.

# References and Resources

Bootstrap - <https://getbootstrap.com/>

Flask Documentation - <http://flask.pocoo.org/docs/0.11/>

Json Documentation - <http://flask.pocoo.org/docs/0.11/api/#module-flask.json>

Python Documentation - <http://docs.python-guide.org/en/latest/>

Background Image (main\_image.jpg) - <https://images4.alphacoders.com/251/251764.jpg>

All team badges and information sourced from individual Wikipedia team pages.

Country Flags

Scotland - <https://upload.wikimedia.org/wikipedia/commons/thumb/1/10/Flag_of_Scotland.svg/2000px-Flag_of_Scotland.svg.png>

England - <https://upload.wikimedia.org/wikipedia/en/thumb/b/be/Flag_of_England.svg/1280px-Flag_of_England.svg.png>

Germany - <https://en.wikipedia.org/wiki/Flag_of_Germany#/media/File:Flag_of_Germany.svg>