Michael Fleming

CourseWOrk-1

Table of Contents

[Introduction to the web app 3](#_Toc465264120)

[Design Stage 3](#_Toc465264121)

[Typography 3](#_Toc465264122)

[Color Theory 3](#_Toc465264123)

[Layout 3](#_Toc465264124)

[Routing 4](#_Toc465264125)

[Error Handling 4](#_Toc465264126)

[Flask and Jinja 4](#_Toc465264127)

[Generator Function 5](#_Toc465264128)

[Enhancements 6](#_Toc465264129)

[Critical Evaluation 6](#_Toc465264130)

[Evaluation 7](#_Toc465264131)

[References and Resources 8](#_Toc465264132)

# **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 leagues 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 with 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, 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 is 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 off sans-serif. Although serif fonts are more formal and professional than sans-serif Helvetica I still feel it gives of 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 stands out over it. 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 the leagues page they will see three thumbnails one for each league and can see teams currently playing in the division. Leagues are clearly identifiable by the use of the nation’s flag above the name which helps to 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 folder. 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 too 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 choose to help build my online catalogue of football leagues. Using this framework allowed me to use routing, render templates and response objects among other things.

As well as using flask I also used Jinja 2 which is a very well-known template engine. Using Jinja 2 allowed me to parase 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 data for each league and team.



The image above shows the use of Jinja 2 on my teams.html page, t 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 it 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 parase data to my team’s page had been confusing and long. This 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 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. My thinking behind this was if I was to add dozens of more leagues to the app I did not want my file to be filled with complex routes, so I found the function very helpful.





# Enhancements

Enhancements I would make to improve the app would be to 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 was that they were able to add certain leagues to their favorites so that whenever they logged in the 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. I feel this would add much needed functionality to the webapp and I feel with more time and slightly more knowledge would have been possible.

Another feature I would have liked to have added to the webapp would have been to add the current league standings along with information about each team. I feel adding a live league table for each league on the app would add much needed 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 I feel that 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 I have a learnt a lot during it. Hover ever I feel there are aspects of the app that I can work on. 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 time 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 of the app. 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, and are then redirected to their more-info page. Due to this I feel like this is the main reason holding back the functionality on offer from the app. Another negative of the app is that it does not contain any up to date league standing about any of the leagues. Without using an API to pull data from the league it would have to be manually updated which would not be worthwhile.

One positive 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. As my knowledge of the languages I was able to create a functions which render templates and load the json on to 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 to 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 Flask were stretched and at points was very challenging. This became very apparent when trying to parase 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 most challenging functionality I tried to add to the app I feel it has enhanced my Python Flask and jinja skills. I feel now that I have 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 parase through and realized how the code was actually working. I feel 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. However, my knowledge of python and flask was very little. 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 Flask to pull data through to the page. After getting my head around the examples shown in the workbook 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 a 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 would now feel 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. Due to this I would feel a lot more prepared with the knowledge I have gained from this build if I was to take on a similar project.

# 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/>