Edinburgh Napier UNIVERSITY

Coursework Report

Jostein Dyrseth
40223535@napier.ac.uk
Edinburgh Napier University - Advanced Web Technologies (SET09103)

Keywords – report, web, flask, python, code, routing

1 Introduction

1.1 The Task

The task was to make an interactive and well structured web app. The framework to be used was the Flask micro-framework. The students must show that they has understanding of the framework and how it renders the different html pages depending on what rout is followed by the user.

1.2 Describing the web-app

The app that was made were a simple, fun and interactive movie-app for people of all ages. It will give the user a list of different categories: action, comedy, documentary, drama and the default one - popular. The popular-list of movies were chosen to be the default, because that is a collection of all genres and is the one people often are interested in.

Figure 1: **Drop Down Menu** - This is the drop-down menu giving the user the available categories.

2 Design

2.1 Explaining the architecture of the web-app

So the base template: 'structure.html' will hold the main structure that the user will see over all of the hierarchy on the app. The main structure is simple with just a header and a background. This was done so that the user don't get confused and also so that they don't get easily lost on the site.

2.2 JavaScript Object Notation

The movies are stored in a .json file that basically is an object or dictionary when read into the python app. The .json file contains the title, vote, id, popularity, path of the cover and backdrop-images, release-date, a small description and some other information as well that were not used in this particular version.

2.3 Templates

The main base or structure has also some other templates on top of that. Using Jinja2 it was easy to implement simple on-top layouts that extends from the base and downwards in the hierarchy-tree of the site. The app used the same template: popular (the first one that was created) - on all the different categories. On these pages there is a for loop that will loop through all of the movie-objects, hence displays them as a grid view for the user to further click on.

The user is free to click on any given movie. When the user clicks on one of them, the specific data given to this movie will be displayed.

2.4 Content

So the user will be presented with a lot of functionality and options at the main page: '/popular', where the 20 most popular movies are listed as the main content, as well as the information and cover photo right above. The header and search-input is also a part of the base-structure of the site which will show up on all of the pages, except the thank you-page the user will get once signed-up and has to press 'home' to get redirected to the home or popular site.

2.5 Signing Up

So when the user is ready to sign up and clicks the "sign up" button, they will be given a 'modal' at that same page, where the content will display a form of four input-values. The values are: user-name, full-name, email and password. These are the least values that any site normally gives the user while signing up. These were chosen because this is the most important ones when dealing with a simple app. Once filled in correctly and the user hits the submit-button, the form will be POST'ed to the server and python will request the form values and store them in the users unique user-name on the server in the .json file. A safe implementation of this was not provided at this stage and version.

2.6 Logging In

The user is now stored on the server, so whenever the user tries to log-in with the right credentials, the user will indeed be granted access hence his/her user profile will be activated

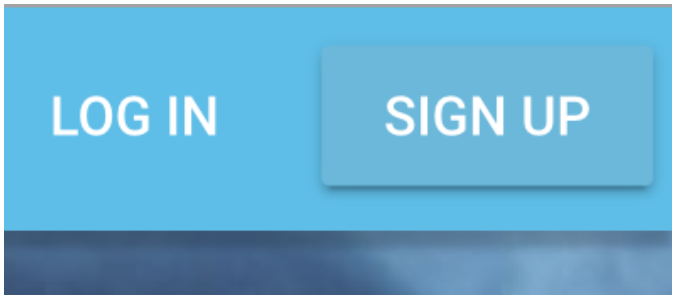

Figure 2: **Search** - This is the search bar. When hovering, the color changes, making the user experience more interactive.

and the "logged-in" HTML template will also be added on top of the current one. The user will now see a small round image of himself in the top right corner. The user can also 'like' any movie and the user's liked movies should also in theory be stored on the server.

2.7 Search Bar

So the search bar is a simple search really. There was not time for anything fancy here. It only works by searching through the list of movie-titles. This is a poor search functionality given that the user must indeed know the exact spelling, whether it is upper- or lower-case and so on. The search must be perfect in other works for it to return any value.

Figure 3: **Search** - This is the search bar. When hovering, the color changes, making the user experience more interactive.

2.8 Menu

The menu at the top left is a menu where the user can click and it will display the categories that are available. When clicked the content of the given category will display.

3 Enhancements

Describing the features that you would add or improve: The Search Bar could have been improved dramatically. The available search items could have some tags for the user to be able to search the item by for example actor, producer, category and so on. This being such a small app though, there is really not that much to improve in a way as well. If the app had many more records, it would definitely be important with a proper search for sure.

3.1 User Account

If more time had been provided, it would be possible to implement a user account with more properties like: liking movies and adding them to favourites, voting on them, writing a review and so on. These functions are often very important for the user so that they want to come back and revisit.

Figure 4: **User** - This is the user icon.

4 Critical Evaluation

Explaining the features that you feel work well, or work poorly, and why:

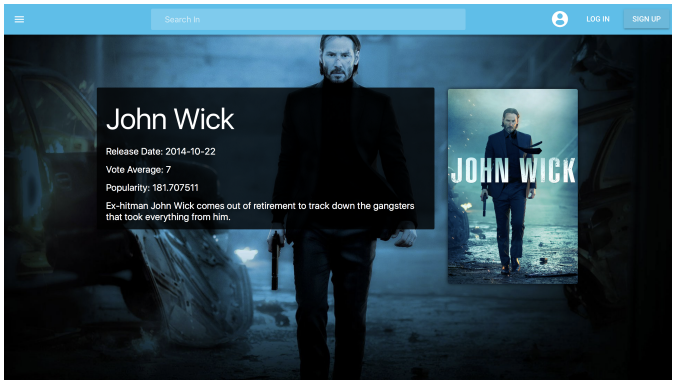

Figure 5: **Layout** - The layout works well and gives the user information about the current movie.

4.1 Works well

The layout works well and the user is not distracted from the different sites easily as this has the same base in all occurring routes. The displaying of the movies are quite neatly shown as a grid view. The elements in the header is lined up and properly laid out. The hover/cursor functionality improves the users interactivity with the page. The sign-in and log-in works well. There is room for theoretically - if scaleable - as many users as wanted. The search is quick since the data is in a dictionary with unique keys being the user-name.

4.2 Works Not So Well

There are a lot of things that does not work so well in this app. the search not work yet unfortunally, but having more time this would defenetly been impoemented with a simple linear search through the dict and possible other lists.

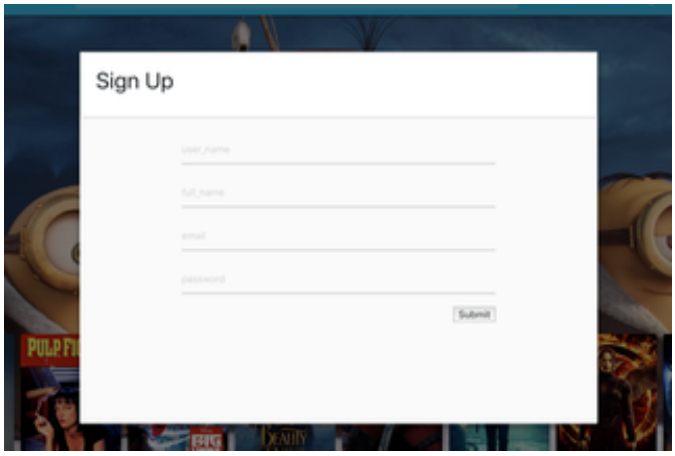

Figure 6: **Modal** - So the modal that was being used will display from a script that runs a simple function which will open to modal using JavaScript. This layout was chosen so the user easily can see at all times where they are in the web hierarchy.

5 Personal Evaluation

Reflecting on what you learned, the challenges you faced, the methods you used to overcome challenges, and how you feel you performed:

5.1 What I learned

So the things that I learned are great. I learned a lot from the workbook as well as Team Treehouse, Stack Overflow and the Python-, Flask-, CSS-, HTML- and Jinja-Documentation. I also learned how to properly deal with JavaScript Object Notation which always comes in handy and how to read and write to the .json object-file. I like the idea of Python being the main "engine" of the app and deals with the back-end, while the HTML and CSS are being rendered depending on what route is being entered by the user.

5.2 The Challenges I Faced

I had a whole lot of challenges. Some of them were because I didn't understand all from the workbook and the lectures entirely, so I had to look a bit online and read about it there as well. Basically I worked through the book, but some of it only made sense when implementing my own site which gave me a lot of "aha"-moments.

5.2.1 Downloading The Images And Backdrops

I had to write a script to manually download the movie-images from the path provided in the .json-file. This took a lot of time. That time could have been better spent on other functionalities like the search-bar.

5.3 Trouble with

I had trouble rendering templates when giving a route of two or more route-levels. I found out that I had to give the .css link a proper relative-path in the head-section of the HTML. I also had some struggles with writing to the .json-file. The file or object gotten from the API was very good in fact, but there were one problem. Basically I had to hard-code and change the .json just a little bit so that the movies were not in a list but structured as a proper dictionary rather. This took me a lot of time to realize and hence restructure my code to deal with this object in a whole other way. This way was better and not to mention faster for the search engine. Instead of doing a linear search through the whole list of movies provided, I could now just easily provide Python with the selected movie as a key and make python fetch the values for that particular movie instead.

5.4 The Methods I Used to Overcome Challenges

I had a lot of challenges, and so I needed some methods to face them. I googled a lot of things and found a lot online. I also used the workbook when I was stuck to try and figure out what the real problem was before proceeding with a solution. I also asked friends and the lab-assistant when I was stuck.

5.5 How I Feel I Performed

Although I were stuck a lot my effort were always there making sure I met the coursework deadline on time. I could have been more pragmatic and not try and solve so many things all at once. One of my bad habits were to leave something half-way done and go over to the next task making the whole code different and re-structured once I got back to the original task, hence making things more difficult for myself.

6 Conclusion

References:
The Movie DB (themoviedb.org),
JsonLint.com: used to validate the json objects,
Material Design Light (CSS),
Bootstrap (CSS),
Materialize (CSS)