Coursework Report

Alexis Guillonneau
40404292@napier.ac.uk
Edinburgh Napier University - Module Advanced Web Technologies (SET09103)

Abstract

This coursework has for objective to demonstrate the understanding of the Python Flask micro-framework. For this web-application I have decided to choose movies as subject. This collection of movies will contains different information such as the duration, the year of the release, the starring, the producers, etc.

Keywords – Report, Coursework2, Advanced Web Technologies, SET09103, MovieSearch, Edinburg Napier University, Alexis Guillonneau, 40404292

Title

Considering the web-application has for collection movies, I pick a simple title so it be simplier to identify the main purpose. Therefore I use a contraction of two word for the final result **MovieSearch**.

1 Introduction

The web-application MovieSearch is composed of many features but the most important remains the research. However for the purpose of the discovery, ten movies chosen randomly are displayed on a carousel as you can see on figure 1. For the research, a side-bar is proposed to the user on the left side to find movies according a certain category (e.g. Action, Horror, etc). A display of all the movies is also available via the tab "Movies". Another feature is to display for example the movies of a certain producer by clicking on his name via the page detailing a movie. Another important feature is the account system because once registred, the user can access to a watch list, a rating system and many more.

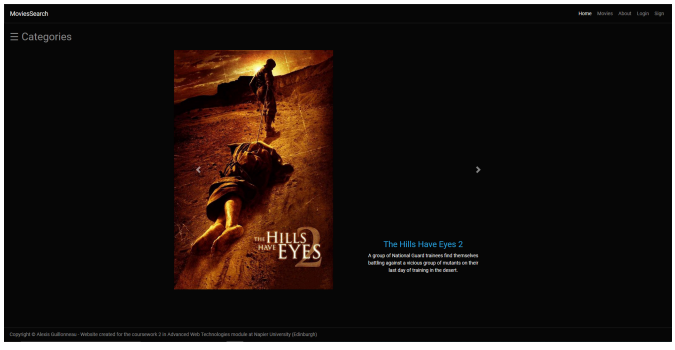

Figure 1: **The Web-application main page** - This main page is mainly composed of a carousel

2 Design

2.1 General design

For this web-application, I choose an URL hierarchy which allows me the simpliest way to find different movies according to the chosen criterion (category, year, producer, etc.). So in the first place, we have the main page located at "/". Of it derives all the others pages as you can see in the figure 2. The different URLs such as "/category/<category>", "/year/<year>", etc. will display movies depending of the string of characters chosen (e.g. "/director/Joss Whedo" will display the movies leaded by Joss Whedo, figure 11). In the same way "/movies/<movies>" will display all the information about the movie chosen (figure 5 and 6). If we go up a notch in the hierarchy, logically the URL "/movies" matches with the tab "Movies", briefly mentioned in section 1. That is the way I will be able to search and display the different information the user wants (figure 13).

To improve the user experience, I added an account system. A simple user can access to the sign in page (figure 10) via "/sign" or the the login page via "/login". Once registered, he can access to his account via "/account" and he can also logout by the route "/logout". To know if is he correctly logged, his email adress is displayed on the navbar (figure 9). By loggin in, the user has access to more features like add a movie to his watchlist, rate a movies, add comments, etc. . .

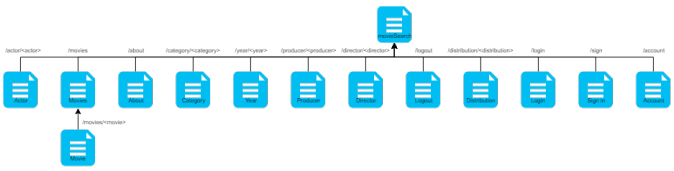

Figure 2: **The Web-application navigation map**

2.2 Architecture

In addition, I have chosen a particular design pattern: the architecture Model-View-Controller. The model is the part that interacts with the database (figure 3). The view, here is our templates, generates output to interact with the user. And the controller is between those two part, it will receive data from the model and will update the view to change information presented to users. Usually, for the model, a database engine such as MongoEngine or SQLAlchemy is used but for our little web-application, SQLite is enough.

It differs from the implementation seen on the workbook, but after some research and some tests, I decided to use the MVC for the coursework.

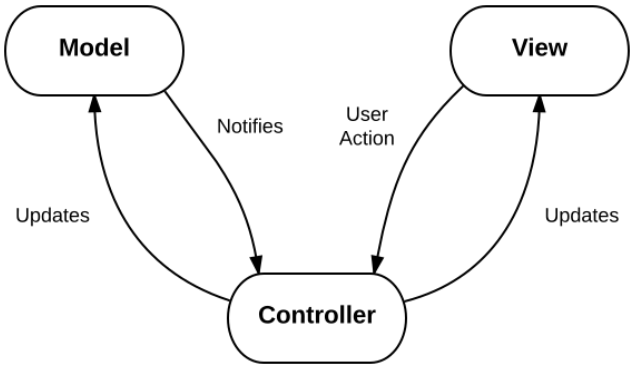

Figure 3: **Model-View-Controller Architecture**

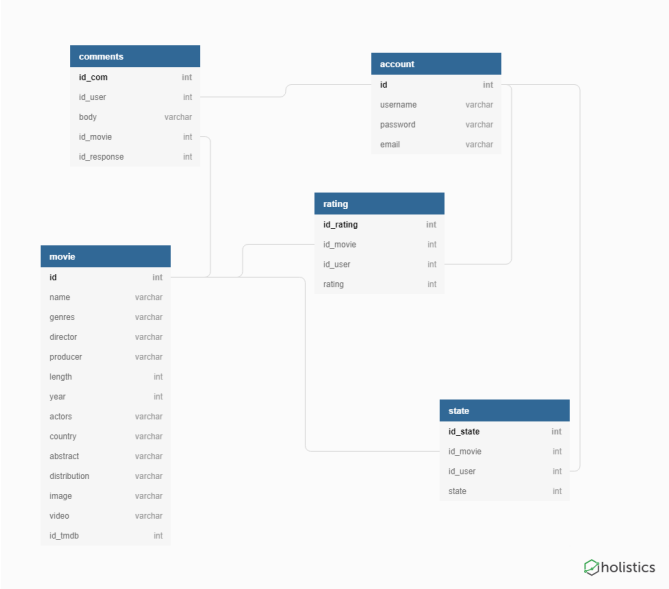

Figure 4: **Schema of the Relational Database**

2.3 Relational Database

For my data, I choose to create a relational database in order to manage the data more easier. My schema for this one is described in the figure 4. The two principales tables are "movie" and "account" because to insert a row in the others tables, at least one movie and one account are needed. In order to have a large database, I extracted almost 5000 movies information by the API provided by *themoviedb.org*. Note that the passwords are already hashed when they get inserted into the database to keep them protected. Some explanations need to be given regarding three attributes of the database :

- id_response in the table comments. If the comment is a response to a comment, this attribute will refer to the id_com of the latter. If it is not a response id_response is equal to -1
- rating in the table rating. A logged user can rate a film out of 5.
- state in the table state. A logged user can add a movie to his watchlist, state will then take the value 1. Or he can tell the web application he has already seen the movie, state will take the value 2. The default value is 0.

3 Enhancements

Firstly, the feature I would improve in the way to sort the movies in tab "Movies" with dynamic Bootstrap tags. This could improve the user's search experience in a more intuitive way. On another side, I would change the way the database is structured. By adding table for the actors, the producers, etc. In this way, I think it would improve the ability to display more information in the future, because right now there is only information about movies in the database but in a sense of quality of the web-application change will be needed. To speed up the process of displaying movies, I would like to separate the movies into different arrays to display for example only 15 per page. We would have a pagination system that will avoid a phenomenon of one page scrolling. Add a search engine could also help the user in his research, at the moment the search function via an text input is really basic. A search engine like ElasticSearch or LunR from Apache could be benefic for the web application, especially since if we update the database regularly, a fast and powerful tool will be required.

Secondly, the feature I would add is an administrator system. Mainly for the use of comments moderation. Indeed, if the web application get more and more users, the stream of comments will become huge, administrators would be here to handle this. It also means an administrator interface for example to add new movie, update or delete data.

Thirdly, I would improve the user account web page. At the moment, he can only see his username and his watchlist. In the future, it will be important to offer the user many tools to customize for example his watchlist, to manage the different comments he posted, etc. . .

At last, the sign in system can be improved. At the moment, when the user create his account on the web application, a email is sent to the email address he provided to "welcome him in this web-application". By using this system, a verification system can be created. In the database, a boolean attribute "isVerified" can be added. To be verified, the user will need to click on the link he received via the "welcomming" email. The link will contains for example a unique token create from the user information to be sure that the confirmation link is unique.

4 Critical Evalution

4.1 URL hierarchy

In the web-application I have built, I feel the URL hierarchy work well. Indeed, I allows me to display many infor-

mation depending of what the user needs. He can access for example to all movies distribued by a certain company, etc. This is an implementation that seems clear and easy to use.

4.2 Search tool

At the moment a search tool is offered to the user for the movie research but it is a simple tool. For example, it is not cast sensitive so if you search "superman" instead of "Superman", it will not show you movies related to superman. Furthermore, it search only on the title and the abstract, sometimes it is not working as well as I would.

4.3 Carousel

There are also some problems concerning the layout of the carousel for example or the general layout of the movie information. Besides, information on how to use the web-application are missing, this leaves the user alone to discover the different features.

4.4 Side-bar navigation

However, the sidebar for the categories is working very well wich it allows to quickly access to the wanted information (figure 12).

4.5 Database

For the need of the database, I developed a Python script that allows me to extract information from the API *themoviedb.org* to fill my database with their data. This process is quite long (about 50 minutes for almost 5000 movies) because for one movie, it send four http request to the API (It might be possible to reduce the process time by implement a parallel system with queues but it is not the purpose of the coursework). Once I have all the data in a CSV file, I create a realtional database schema in SQL and after that I insert the data. It is not part of the application but it was useful to me where the evocation in this section.

4.6 Movie page

This is the part that asked me the most work. The following explanations have as visual support the figures 5 and 6. In order to give the best visual experience to the user, the background of the web page is the trailer of the movie. This consists of displaying the youtube video of the latter on the background of the page without the user being able to interact with it. Once finished, it will automatically restart.

One the right part of the page, the user can find all the information about the movie including the average rate of all the users. Only a logged user can interact with the icons. Stars icons are here to allow the user to rate the movie out of five. The heart icon is there for the user to tell the app that he has already seen the movie. By clicking on the add icon, the movie is added to the user wtachlist. All the icons interact dynamically with the user. Move the mouse over the stars will fill them. Clicking on the heart will fill it and the add icon will transform into a checked icon. The update on the database is handle by AJAX request send by the script of the application.

Figure 5: **Layout of the movie details**

Figure 6: **Layout of the movie details**

At the bottom of the page, the user can find the comments part (figure 7). Only a logged user can add a comment. If he desire to respond to a comment, it is possible by clicking on the comment, a textarea will be added (figure 8)

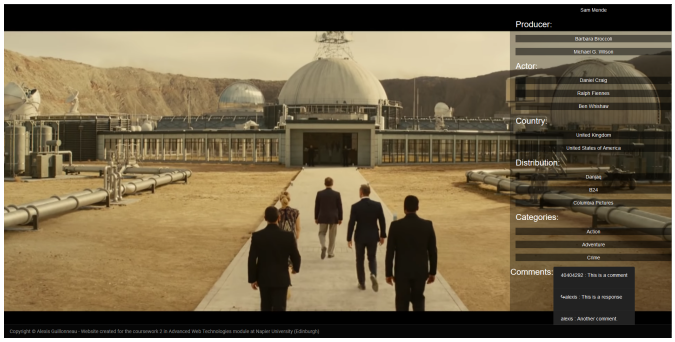

Figure 7: **Layout of the comments part**

Later, I figured out that the layout and the youtube video is not working well in Chrome Web explorer.

5 Personal Evaluation

During this coursework, I discover how to manage a user interface with loggin and password protection. I got a new point of view about security and data sensibility.

Figure 8: **Layout of the response area**

I learned how to do AJAX request with Flask to do a more client side web application.

I also managed a huge amount of data, to choos wich information I will use in my database. I know that 5000 row in a database it is nothing compare to normal database in the web.

I am proud of what I did for the movie page. It was a real pleasure to develop this kind of application with a great visual aspect.

I also think I managed to produce a clean and readable code that could be maintained. Something that I have always had difficulty doing in the different projects that I led.

References

[1] Aggarwal, *Flask Framework Cookbook*. Birmingham,United Kingdom: Packt Publishing Ltd., 2014.

[2] Dwyer, *Flask By Example*. Birmingham,United Kingdom: Packt Publishing Ltd., 2016.

[3] Maia, *Build Web Applications with Flask*. Birmingham,United Kingdom: Packt Publishing Ltd., 2015.

Appendices

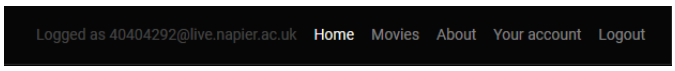

Figure 9: **Layout of the nav-bar once logged**

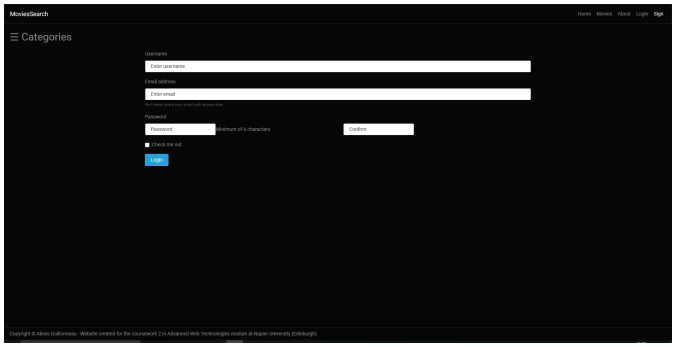

Figure 10: **Layout of sign in form**

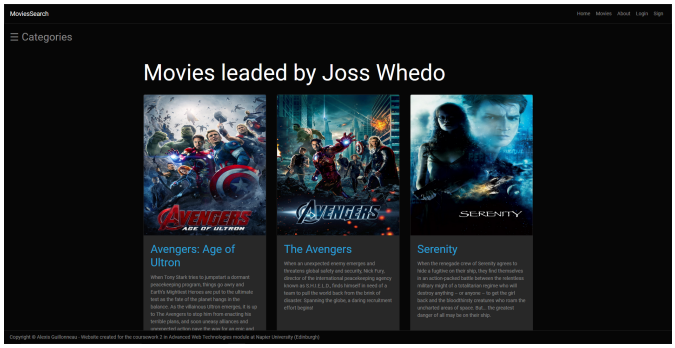

Figure 11: **Joss Whedo example** - Layout of the movies leaded by Joss Whedo

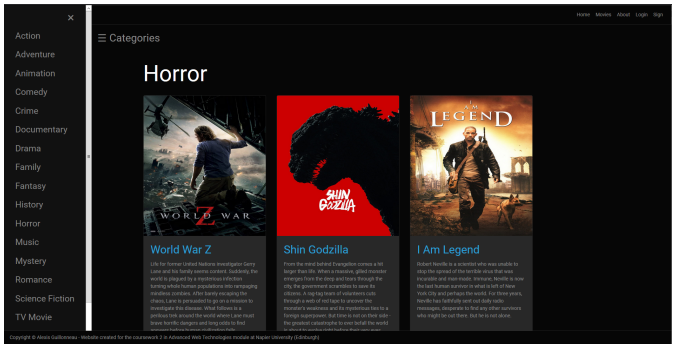

Figure 12: **Example : Horror movies** - Layout the category Horror

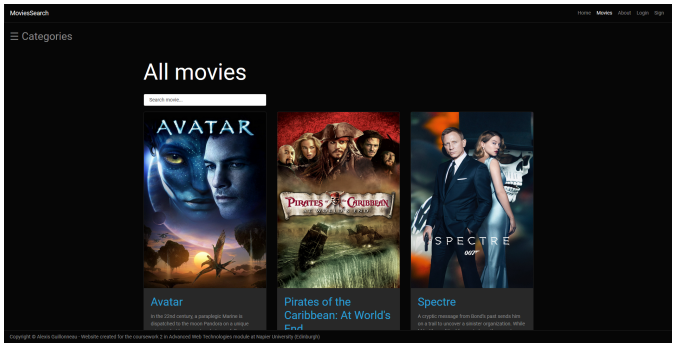

Figure 13: **Layout of the tab "Movies"** A search input-can help the user to find desired movies