REPORT

Coursework #1

SUMMARY

Report about a web-app developed using Python-Flask. Collection of Disney Characters and rating system.

Carine PIC - 40215333

Advanced Web Technologies

Carine Pic

Napier matriculation number: 40215333
Module: Advanced Web Technologies
Coursework #1

Submitted on Sunday 1st November 2015

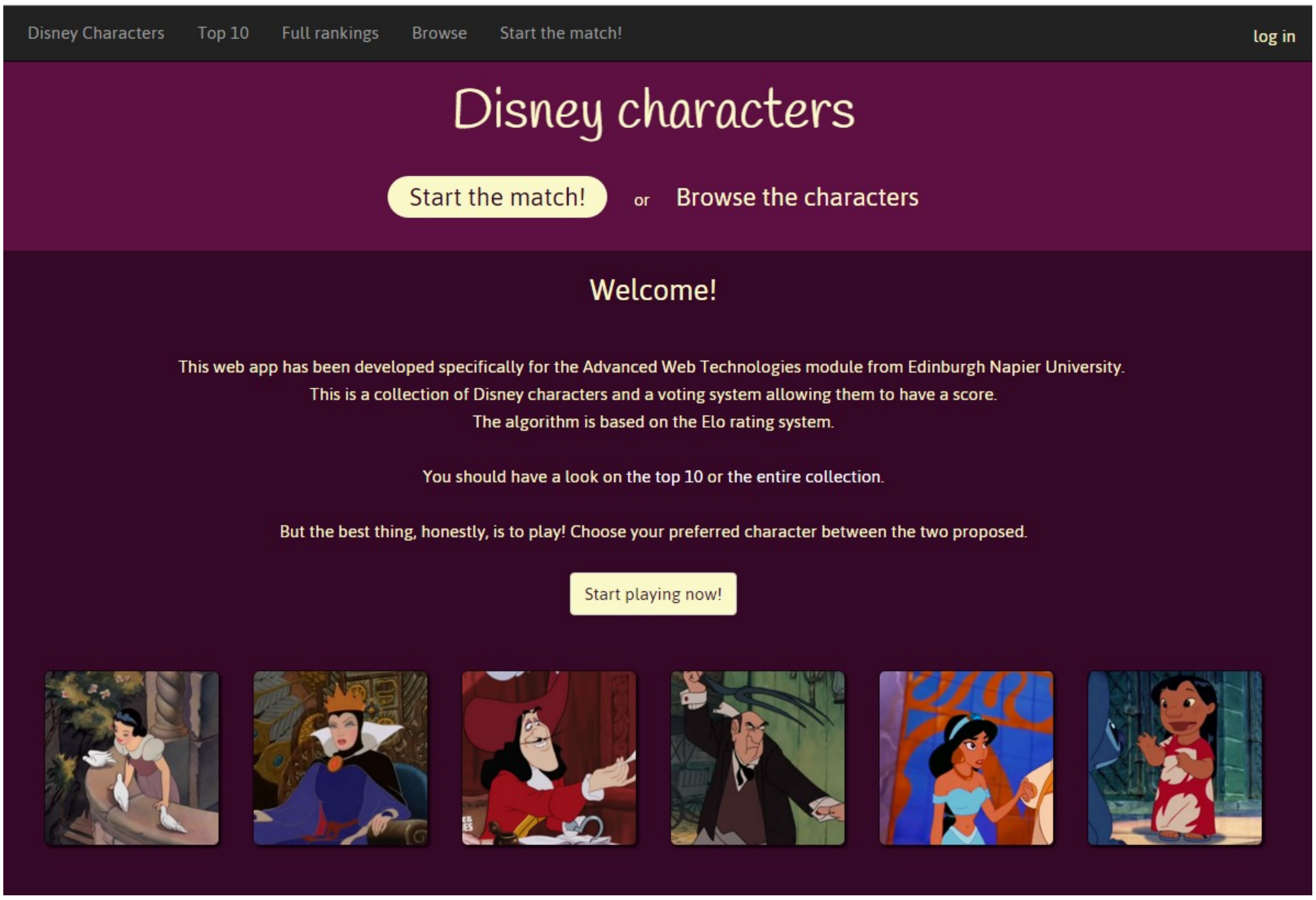
A web-app using Python-Flask

Table of contents

Introduction

This web-app is a collection of Disney characters. The web-app offers the user to browse the characters in a table, or to play a kind of game where you have to choose your favourite character between the two proposed. The characters have a score and the user is able to see the top 10 or the full rankings. The user can also view the details of a specific character and the relative characters (same film) or view all the characters available for a specific film.

If the user is logged in, they are able to add a new character and edit or remove the existing characters.



Welcome!

This web app has been developed specifically for the Advanced Web Technologies module from Edinburgh Napier University.
This is a collection of Disney characters and a voting system allowing them to have a score.
The algorithm is based on the Elo rating system.

You should have a look on the top 10 or the entire collection.

But the best thing, honestly, is to play! Choose your preferred character between the two proposed.

The browse feature

The whole collection is displayed in a table. The user can see the picture (miniature), the name of the character, the film, the score, the number of matches and the number of wins for each character.

The picture is displayed in a circle shape and in a miniature size. However, if you move the cursor over the picture, the size will increase. You can click on the picture to access the character's page as well.
The name is clickable to access the character's page, and you can sort the whole collection by alphabetical order if you click on the arrows at the top of the table next to the headers.
The film is also clickable to access the film's page and you can sort the films by alphabetical order.
The score, matches and wins are not clickable but you can sort the collection using these criteria.

The last feature is the search field. It will search in the name and the film fields and display the results.

The match feature

I have developed this feature to make the coursework more fun, and also because I wanted to test the algorithm used in *the Social Network* film (for the facemash website). After some research, I found that the algorithm used is the **Elo rating system**, mostly used for chess competitions.

For more information about this rating system and how it works, you could visit the Wikipedia page.

In my web-app, you can choose your favourite character between the two proposed. The characters are randomly chosen by the script. Once you have voted, the characters proposed are replaced and you are able to vote again. You can play as long as you want, there is no end to the game. The rankings are updated when you play, and if you want to view the results, there is a page for the top ten, and another page for the full rankings.

Logged in features

You can log in as an administrator and be able to **add**, **edit** or **remove** a character.

The user name is "admin" and the password is "default".

Design & Architecture

The initialisation of the app, the functions and the routings are in the file “*index.py*”. The parameters are stored in the *etc/config.cfg* file.

I have used sqlite3 for the database, so I created a file to store the schema of the database and a function to initialize the database. You can also launch the file “*create_db.py*”. The data are stored in *db/datas.sq3*

There is also a “*elo2.py*” file which contains the Elo algorithm’s class.

The templates are in the *templates* folder. I have created a *layout.html* template and all the other templates override it.

I have used Bootstrap for the styles and some customised styles in the file “*static/styles.css*”.

The URL layout:

/home
/top10
/fullrankings
/browse (GET or POST) or /browse/<sort>/<order> (GET or POST)
/match (GET or POST)
/character/<int:id>
/film/<name>
/add
/edit/<int:id>
/delete/<int:id>
/login
/logout

Enhancements

There are a lot of features that I would add or improve.

- Display the log in form in a modal
- Manage the pictures
    - Create different sizes when uploading
    - Check the extension and the file size when uploading
    - Add several pictures for one character
    - If more than one picture, display a carousel in the character's page
- Improve the validation feature when adding or editing a character (displayed errors, keep the form filled after an error, etc.)
- Add a personalized (and funny) 404 page
- Allow the user to suggest or ask for new characters
- Add a pagination in the browse page
- Films
    - Create a specific table in the DB for every Disney film
    - Replace the "text" field in the add character's form with a "select" field
    - Improve the film's page with a summary of the film's story and extra informations (date, etc.)
- Add enemies and friends feature
- Allow characters that have a low match count to appear more frequently in order to reduce the gap between the top match count and bottom match count.
- Avoid duplicate images when displaying 6 random pictures on the welcome page
- Use the logging feature to help fixing errors
- Add a "back to top" link
- Allow the administrator to reset the score
- Manage the database with SQLAlchemy instead of sqlite3, to allow Object-relational mapping (ORM)
- Allow the user to remove the flash alert once read
- Animate the change of the opponents after voting in the match feature
- Deploying the app online
- And obviously, improve the quality of the code

Critical evaluation

My web-app using Python-Flask has a good base, I think, but it could be improved. As I said in the last section (Enhancements), I have a lot of ideas to improve it and make it much better.

The most important thing that fails is the code, which is not really well thought out and maybe a bit redundant sometimes. I know that and I've tried to make the code clearer but it was a bit hard.

Regarding the features, I have added an extra-feature which was not asked for the coursework. I am happy that this feature works well, as it was kind of a challenge for me to develop it.

The initial demands of the coursework have been, in my opinion, answered well by the browse feature and the different pages (character's page, film's page), as I allow the user to find and retrieve the information.

I have tried to develop a good user experience when using my web-app, and I hope it will be appreciated.

Personal evaluation

As I didn't know Python before this project, I was really happy to learn a new language and especially a new way to code (using levinux and vim).

I used to code with a lot of languages in my previous experience, like HTML, CSS, PHP, SQL, javaScript or even Java but always with a text-editor (Notepad++) or recently an IDE (eclipse, SublimeText and PhpStorm which is my favourite).

As I used to code a lot and I had habits with IDEs, it was hard to code using vim, mostly because I wanted to use shortcuts and features that don't exist with vim. Sometimes I have had troubles using a shortcut in the shell, wanting to go too fast, but in the end, I think I managed to overcome the difficulty.

The second challenge I faced was to find out how to implement the ideas I had in my head using Python-Flask that I knew were possible with other languages (mainly PHP). I read loads of tutorials, documentations and forums about Python and sqlite3, in order to develop exactly what I wanted to do (I'm a bit stubborn and perfectionist…).

In particular for the rating system I wanted to develop, I was not sure of how to do that, and I didn't want to spend too much time on it, so I looked for an existing script using Python.

I found a perfect script, but using SQLAlchemy instead of sqlite3. I had a panic-moment when I found that SQLAlchemy was better than sqlite3 (especially for the use of ORM) but as I have developed all the app using sqlite3, I decided to continue using that and to put SQLAlchemy in the improvement section. At the end, I managed to disregard the ORM by adapting the original script, but with a heavy code which doesn't really please me. *Please see the references section for more information about the original script.*

Otherwise, I was happy to use Git on this project, but I think I'm not comfortable with the commit feature yet: I always forgot to commit my changes and when I thought about it, it was the end of the day and sometimes it represented several hours of work. I will work on it and try to develop good habits and practices using Git.

The last thing I would like to say is the frustration I had not being able to deploy the app online yet. I tried to look for information on how to do that but I gave up for now, as it was not really important for the coursework and I didn't have time.

A quick paragraph about the issues I had using levinux with a French computer and a French keyboard. I finally overcame the difficulty but I would have prefer not to spend so much time on it and concentrate more on interesting things instead.

To finish this report, I would say that I really enjoyed working with Python for this project, and thanks for the really well-written workbook that helped me a lot to learn Python and vim. ☺

Resources and references

All the pictures are Disney films' extracts.

I've used Bootstrap, which is open-source. The code is licensed under MIT (see licence file in the Bootstrap folder).

I've also used the work of Christoph Gerneth for the Elo rating system script. His work is available on Github: https://github.com/c7h/facemash

I've read a lot of documentation, tutorials and forums, so I can't provide all of these here, but the main were:

- Bootstrap documentation: http://getbootstrap.com/
- Python Flask documentation: http://flask.pocoo.org/
- Using SQLite3 with Flask : http://flask.readthedocs.org/en/0.9/patterns/sqlite3/
- Information about ORM and SQLAlchemy: http://pythoncentral.io/introductory-tutorial-python-sqlalchemy/
- SQLite3: https://docs.python.org/2/library/sqlite3.html#