ONLINE FILM DIRECTORY

Set09103_cw01

A simple web app to allow users to search for the favorite films

Ruaraidh Curran / 40209895

40209895@live.napier.ac.uk

Contents

Online Film Directory 2
Introduction 2
Design 2
Login/Register 2
Home 2
Search film 3
Add film 3
See all films 3
Logout 3
Enhancements 3
Adding pictures to each film in the database 3
Displaying films up and coming to cinema 3
Linking films to corresponding IMDB or Rotten Tomatoes review 4
Creating individual forums for each film where users upload their thoughts of the film 4
Critical Evaluation 4
Templates 4
Error Logging 4
Responsiveness 4
Flash Messages 4
HTTP Personalized Error Message 4
Sessions 4
Databases 5
Personal Evaluation 5
References 5
External Learning: 5
Referenced API's and Packages 5
Appendices 6
1. Home Screen 6
2. Navigation Map 7
3. Website Screenshots 8

Online Film Directory

Introduction

I have been tasked by the university to demonstrate my skills and understanding of the Python flask framework by creating a web directory for my chosen subject. This report will cover Design structure, enhancements to the web application I made and ones I would like to implement if given the time, a critical evaluation of the good and bad points of my applications features, and ending with a personal evaluation of how I performed, where I performed well and where I could improve. As a lover of film and cinema I decided to base my web application around this. The base function of my application is to allow the user to easily sort through a number of films by any combination of different metadata types such as film title, director, genre, release date, and run time. I have accomplished the base function given by the coursework with a friendly user interface [1]and also implemented extra functionality to demonstrate my skill with the flask framework and my eagerness to learn.

Design[2]

To access my web app's functionality there must be a user signed in. With the use of redirects and session checkers I have made the login page the first page you will come to no matter what URL the user try's to access, even with pages that don't exist I have made the error 404 page to redirect the user back to the login screen if they aren't logged in already with an appropriate message to advise the user what to do next. Making the login page the default start screen for navigating my web app.[3]

Login/Register

From the login screen the user will be given the option to login with an existing account or to register a new user. To register a new user they simply click "Register Here!" which will redirect them to the register page. This page requires the user to fill in all the fields and enter their chosen password twice to help with any human mistakes that might occur during the process. After completing the user is redirected to the login page where they will be able to enter there login details and continue to the home page.

Home

Throughout the whole website the user will notice a easily understandable navigation bar with appropriately titled links for each operation that the user might want to carry out. On the home page the user is introduced with a personal welcome message and a brief description of the websites purpose and some other data which utilizes CSS Grids to create a comfortable and visually pleasing user environment. From here the user will be able to access all sections of the website through the use of the navigation bar where they can choose to either 'Search film", "Add film", "See all films" or "Logout" each of these URL links taking you to a different page with individual functionality.

[1] View Appendices 1 for home screen screenshot.
[2] Screenshots of entire website found in Appendices 3
[3] View Appendices 2 for the web apps navigation map.

Search film

This page allows the user to query the database for films with a number of different search parameters and search combinations. The user will be able to search the database by any combination of the search params, all with different results given corresponding to the users inputs. By default if the user didn't enter any search params it will redirect the user to a film list URL with all the films in the database. To help with searching I have allowed the searches to be case insensitive further enhancing the user experience.

Add film

From this page the user will be able to add their own films to the database by entering the film information through the simple UI form provided. The films with their details will be added to the database after submission and can be viewable from the website there after. After submission the user is redirected back to the homepage with a message telling them that their film has been added to the database.

See all films

The film list page uses CSS grids to display all the films within the database in a pleasant manner. Individual "divs" for each film have corresponding sections in the grid layout that displays information of each of the movies stored within the database. This page adapts nicely to different screen sizes by changing the number of films shown on each row.

Logout

The logout route simply stops the users session, removing the logged in details and redirects the user back to the login page of the application with a message informing the user of their action. This action is available from all route views making it simple and easy to logout.

Enhancements

There are a few places that I would like to enhance in further development of this application some being small and some a bit more challenging.

- Adding pictures to each film in the database.
- Displaying films up and coming to cinema.
- Linking the individual films to corresponding "IMDB" or "Rotten Tomatoes" reviews.
- Creating individual forums for each film where users upload their thoughts of the film

Adding pictures to each film in the database

I thought about this later in production of the application and I believe that I wouldn't be hard to do but unfortunately I had already started using SQLite as my database framework which to my knowledge cannot hold images. I believe It could hold file paths for the individual images on a file system but for a website that could have multiple hundreds/thousands of films on it my hardware could easily become full. One way of overcoming this would be to have the pictures linked to online image URLs that are fetched and displayed as and when they are needed, but keeping this coursework within the specifications of no external files or frameworks I thought it best not.

Displaying films up and coming to cinema

Implementing this would require me to use external API's such as the one provided by "movieglu.com[4]" would be a great feature for my website allowing the user to see cinema film times

[4] Link to movieglu.com found in References

locally to them and links to where to book but sticking to the coursework specifications this was not possible. And creating my own would not be feasible in the timeframe given.

Linking films to corresponding IMDB or Rotten Tomatoes review

There is packages such as "IMDbPy[5]" which would allow me to access and retrieve information about films and display that information on my website. I would likely link them using the "title" datatype in my database but one difficulty would to overcome spelling mistakes made by a user entering data into the database. More thought would be needed to overcome this obstacle.

Creating individual forums for each film where users upload their thoughts of the film

I have created I user forum before where users login and add / edit / delete there own posts to a website. This would have been a nice feature I would like to add to my application but given the timeframe was unable to do so.

Critical Evaluation

Templates

My whole website is designed with HTML templates using the Jinja2 to follow the good practice of no repeating code. All my pages use a base html template that holds all the header information to allow each webpage to access data stored in the static files such as the Media, CSS and JavaScript. Along with any repeating body and footer content. This document I called template.html and is also helps give my website a uniform look.

Error Logging

I have made a simple errorlog.txt document to store any errors that the user might run into for me to view and use to fix any bugs that were not found during the creation of the website. This error log excludes http 4xx Client Error codes which should already be handled by my error catching.

Responsiveness

The whole website has been designed in a way to be responsive to different sized screens and viewports to enhance the UX. Found at the bottom of each page is a professional looking footer which allows the user to direct back to the home page as well as a shortcut key to get them to the top of the page again.

Flash Messages

I have implemented flash messaging throughout my website to update the user on their actions and to advise what they are to do next.

HTTP Personalized Error Message

I have implemented redirects from error pages to an appropriate route with a flash messaging to inform the user of mistakes made and what to do next.

Sessions

I have used flask sessions to allow the user login system to work properly. I have also used a random 12pass key to hide the session information for added security. This session framework allows me to regulate what the users can access.

[5] Link to IMDbPy.com found in References

Databases

I used SQLite3 database with two different tables in my web application. One for user details allowing the user to create new users and log in using existing user credentials. The second to store information about each film which also lets the user to add new films with new data to it.

Personal Evaluation

Coming into the production of this application I had never used python let alone any of its frameworks. I am quite happy to say that I am comfortable in developing using python and more specifically "Python Flask". I think it is a very straight forward and intuitive framework which allows for easy expandability. I believe that I have a strong base knowledge of python now and have a firm grasp of the flask framework and have enjoyed learning everything from routing to sessions to error logging. I know that there is always more to learn and I am excited to continue doing so.

From the start of this module I had never really used a Linux operating system to much extent but as I got into week 3 of the module I decided to install a Linux distro onto my laptop which I am now using on a daily basis with things out width this coursework. I believe that with a bit more practise I will be very comfortable with using the Linux terminal to carry out the majority of tasks in this OS.

If I was to start this project from scratch there might be a couple changes I would make. One of which was planning. I jumped straight into coding without even sketching a layout for theorising what was going to go into my application. I just adapted as I created, luckily this did not cause me any problems but I can see that even slight planning such as a couple of UI design sketches, a navigation map or a list of functions I would like the application to be able to carry out, would help production time and allow me to keep a constant workflow without switching between coding and designing all throughout the development. A second change I would make would be to produce a functioning application but using a template like bootstrap for the initial version. Just to allow me to have a more pleasant graphical interface to use during development. Also I believe that the creation of the application might have been sped up with bootstrap if I was to learn all the different classes to allow for continued coding without stopping and making sure that things fall on the screen properly.

Overall I am happy with my application and think I preformed quite well. I am looking forward to using python flask again for my CW_02 and to implement the changes I would have made to this project to create something special for my final coursework.

References

External Learning:

YouTube Tutorials. Web accessed between 08/10/2018 and 21/10/2018:

1. TraversyMedia: https://www.youtube.com/user/TechGuyWeb
2. TheNewBoston: https://www.youtube.com/user/thenewboston
3. CoreySchafer: https://www.youtube.com/user/schafer5

Udemy Course. Web Accessed between 19/10/2018 and 21/10/2018:
https://www.udemy.com/python-flask-course/learn/v4/overview

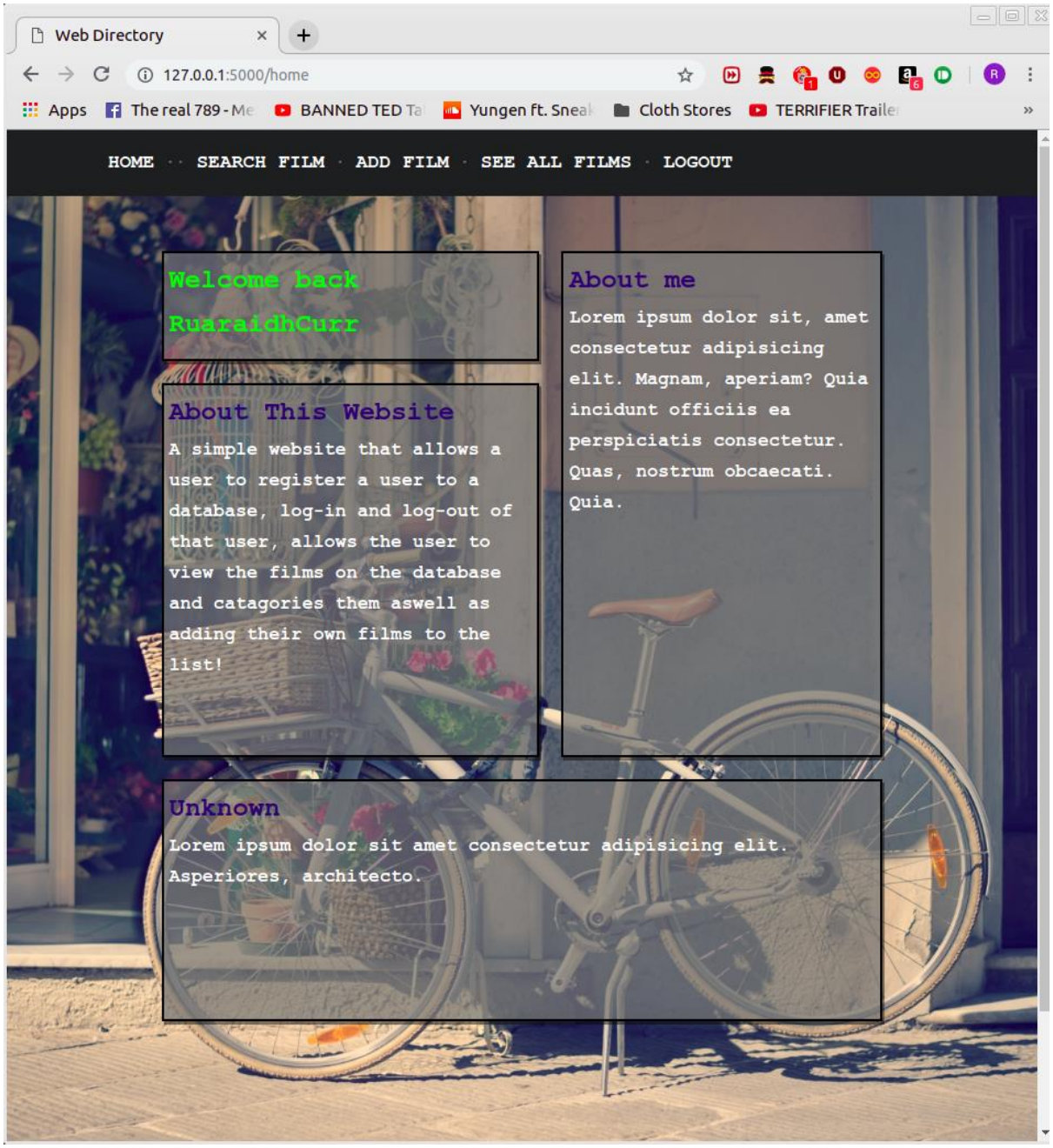
Referenced API's and Packages

Movieglu Movies API. Web accessed 2between 19/10/2018 and 21/10/2018:
https://developer.movieglu.com/

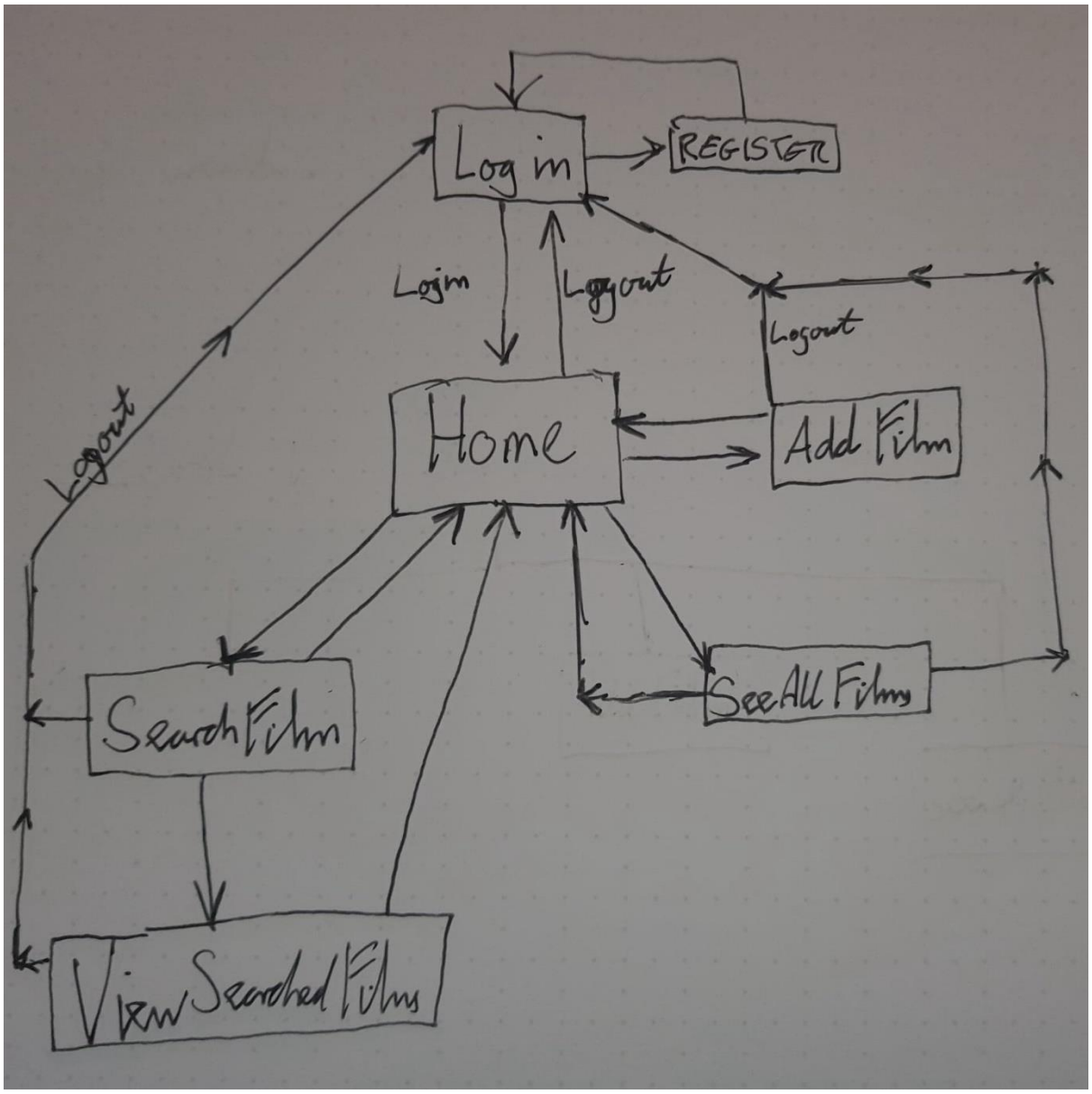
IMDbPY package. Web accessed between 19/10/2018 and 21/10/2018:
https://imdbpy.sourceforge.io/

Appendices

1. Home Screen



2. Navigation Map



3. Website Screenshots