USING PYTHON FLASK WITHIN THE LEVINUX ENVIRONMENT TO IMPLEMENT A WEB APPLICATION

Craig Shaw
40123886

1 Introduction

This web application has been created as an online movie database to display information about movies and cast, including a movie's; title, tagline, overview, genre, runtime, release date, revenue and cast; date of birth, date of death, and biography.

2 Design

URL Structure

Level 1

The root or home page - /

Level 2

The movie and actor selection pages.

/movies

/actors

The add movie and add actor pages are still in development, as it stands a movie can be added to the database, without a poster, but an actor cannot yet be added.

The intention for these pages was to make them only available when logged in as an admin, unfortunately I have been unable to implement a login feature, but I have implemented a secure page at the "/loggedin" URl. You would be able to view this If I had implemented login.

/add_movie

/add_actor - While running this application it is worth remembering that the add_actor URL will redirect to the home page as this has not been implemented yet.

/loggedin – This page is not possible to view as it is restricted for logged in users, logging in and out has not been implemented. It does however give a good example of a custom 401 error page.

Level 3

These URL's can be seen once a movie or actor have been selected from the movie or actor selection page. Once you click on the movie or actor you want to view the URL will display as, http://localhost:5000/movies/selected?movieid=1 or http://localhost:5000/actors/selected?actorid=1 for example.

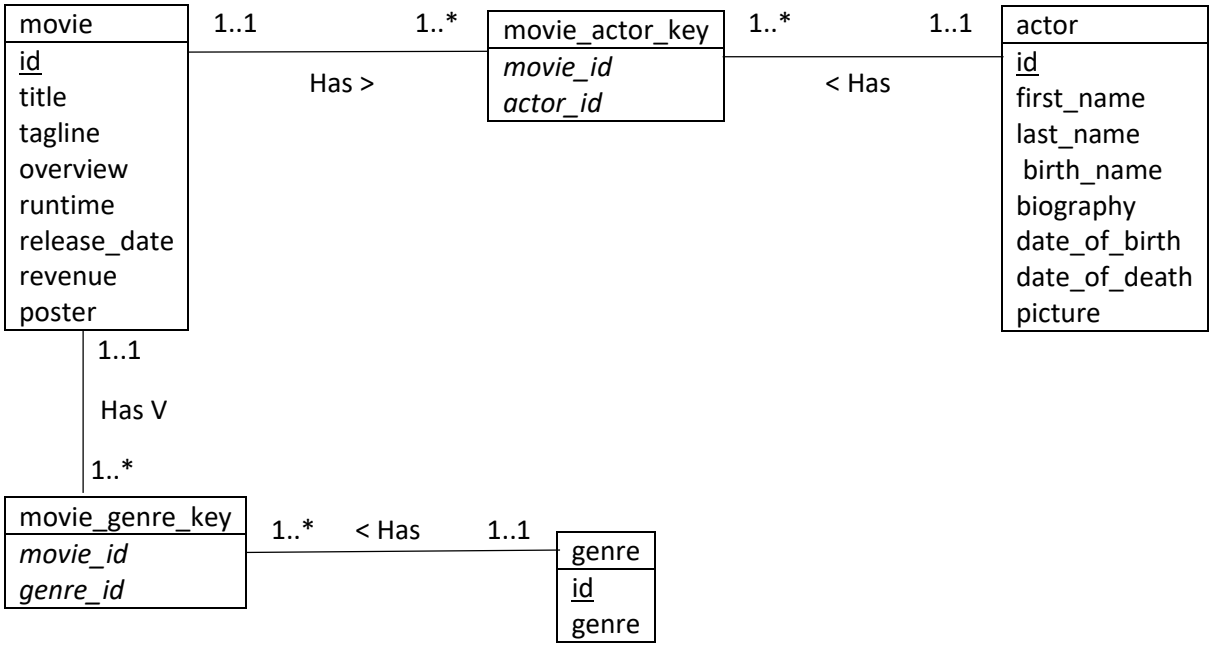
/movies/selected

/actors/selected

The above routes are found in the "index.py" file.

Database Schema ER Diagram

The following is an entity relationship diagram for the database used in my application, “MoveDb.db”. I created this database using information from imdb and images from google, if this application was being made live information and images would be sourced correctly and the cprrect rights would be obtained to use them.



The database design is quite simple movies join to actors and genres through a key table as there can be one movie with many actors and genres, actors can be in many movies and many movies can have the same genre.

Application

The design of the application is very simple I have use bootstrap throughout to give the app a clean and simple feel. And I have added some simple custom CSS found in the “static/css/movie.css” file.

This application uses a configuration file for host and logging settings, “etc/defaults.cfg”

Logging

The application has full logging of all the pages that are visited and also any errors that occur while the app is being used.

Error Handling

There are some custom error pages, for a 401 and a 404 error.

Other Features

This application is designed to display the movie with the highest id in the movie table on the home page as the “Movie of the Week”

The home page also features two RSS feeds one for film reviews and one for film news, these feeds are provided by film-news.co.uk. They are designed using feedwind.

The movie and actor selection pages can be filtered using the search bar at the top of the app this is done using JavaScript to identify any table rows that contain the search term and hiding rows that do not contain this search term are hidden.

3 Enhancements

To improve this application, I would implement the following;

- I would fully implement the add a movie and actor features.
- I would add the ability to click on the release date of the movie or birth date of the actor to show movies released in this same year or to show actor born in this same year. The same would be done for genre.
- Custom error pages could be made better by adding in a simple JavaScript game for example when the site or page is not available.
- When the database grows it would be impossible to look through a huge list of movies or actors, pagination could be added to display a smaller number of results per page.
- I would implement the ability to log in fully to allow only a site admin to add movie to the database
- It would be good to add a link to Amazon or similar online store to allow users to buy the movie they are looking at on the Online Movie Database.
- Embedding YouTube trailers on the movie page would provide another level of engagement for users.

4 Critical Evaluation

This web application shows good use of Flask library and static files using HTML templates. I feel that the app benefits from logging and custom error pages.

I feel the web application meets the coursework requirement and would be fairly intuitive to the user.

There are however many features I wanted to implement but either because of time or inability to implement there are some features missing. As it stands a movie can be added but with no poster. The reason for this is that once I had added the ability to add a movie I realised that adding the

ability to add a movie would be difficult as movies and actors have a many to many relationship. I decided to leave this and movie back to complete it if I had time at a later date. I unfortunately did not have time to complete this piece of work.

I also feel the application would benefit from the ability to log in which I was unable to implement, I was however able to restrict a page that cannot be viewed within the app. My intention was to restrict a page and then add the ability to log in.

With those points being made I feel the app is still a good application and provides users with decent engagement that can certainly be improved in a later release.

5 Personal Evaluation

I feel my ability to work in the Linux environment has improved greatly after completing the first coursework. My confidence in using it has certainly grown. I also feel my ability to use Git has also improved as during this coursework I have made regular commits and have no major issues with it.

As a whole I feel more confident using Linux and Python to design and create a web app. I look forward to using it again in the future to produce we applications within a working environment.

6 References

Boostrap - http://getbootstrap.com/

My RSS feeds are from Film-News - http://www.film-news.co.uk/rss-feeds

The RSS feeds are creating using feedwind - https://feed.mikle.com/

I used the Flask documentation for reference, particularly around logging in and out although I was not able to implement this. I also read the documentation around jQuery autocomplete although I did not implement this - http://flask.pocoo.org/

I used the Flask mega-tutorial to implement my web form for adding a new movie to the databse and also read the tutorial on user logins - https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-v-user-logins

I also read some of the Jinja2 documentation to help when displaying database records in HTML - http://jinja.pocoo.org/docs/dev/