CompSci235 Assignment 2 Report Cool New Features

New Feature 1: Adding movie posters to Movies

Whenever there is a movie on screen, it also presents a poster of the movie, adding to the visual appeal of the page and allowing the user to engage more with the page. The image and the title also provide a link to a separate movie page. This increases the user's immersion while using the website.

Design principles used were the single responsibility principle. I used a separate function to retrieve the posters from the link provided. This allowed for easier troubleshooting and allowing for reuse of code, so that it does not need to be repeated elsewhere.

I added omdb 0.10.1 to the requirements.txt file so that it will be installed along with all the other required modules. I then used this to gather information about the movie and along with it, a link to the movie poster. This uses a query through their servers, along with a unique key that has a limit of 1000 uses per day.

The implementation of the retrieval of the poster is using a function that is located in the services.py file in utilities. I made this decision because it is an often used function that is used whenever there is a movie showing.

What the function actually does is it uses a movie object as the parameter, then sets the api key required by the module using the unique key a person gets after signing up to their website. This is followed by using the details in the movie object to query the site which returns a dictionary. I then retrieved the value stored with the key 'poster', however if there is no value, then it just assigns None. This is how the movie link is retrieved.

Another modification made is that I had also added a poster link variable inside the movie class. This allows me to store the link inside the movie object so that it can be accessed directly. This allows for ease of access, so that when the value is sent to the page, the only thing needed is the movie object.

I found this essential, as without it, the site felt dry and stale, without much character and does not leave a lasting impression on a user. Even as the developer it can be easily felt.

New Feature 2: The ability to go to a user profile page

This is basically what the name says. It allows a user to visit a page with their information in it. I find websites that have our accounts much more enjoyable when it lets us see some of our details. Though limited, this provides that satisfaction for a user when they want to keep track of things they are doing / done. The user must be logged in for this feature.

Things that are included in the page are the username, how long the user has spent watching movies on the site, amount of movies watched, amount of movies in watchlist, and the amount of movies reviewed. I have also included links to a watched movies page and the watchlist page.

I used blueprints to create the links to this page which is contained in user_bp. This is a factor that helps us reuse html pages with different sets of data. It also allows us to redirect easily.

Blueprints also help us maintain a separation of view and service concerns. With the service component being allowed to be reused with different views.

New Feature 3: The ability to ‘watch’ a movie and watched movies page

When clicking on a movie there is a button to watch a movie presented. This will let the user ‘watch’ the movie. The user must be logged in for this feature. If clicked, the user will be redirected to the login page.

When the button is clicked, the user is redirected using the user_bp blueprint. I decided to put this in the user blueprint as it is evident on the relevance it has to the user.

What this watch button actually does is brings the user to the link ‘/watch’ with values of the movie appended to the url if the user is logged in. This then processes the data sent through the url and grabs a movie object.

It also grabs the username and comes up with a movie object.

Then using the method watch_movie in the user object, the movie is watched. This means the runtime minutes of the movie is added to the total time spent watching by the users and appended to the list of watched movies of that specific user.

The movie is also removed from the watchlist if it was there and if it wasn’t then it does nothing.

It then redirects us to the watched movies page. This is a page with the user info shown at the top like the profile page, but below it, is the results of watched movies. It has a limit of four per page and it allows the ability to scroll through using the next/back/last/first buttons like the browse page did. Of course it also reuses the same html page as the profile page template did.

I used this redirecting method due to the fact that if a user refreshes the page, they can repeatedly watch a movie. So to prevent accidents this removes that link of when the page is refreshed, data is added again to their watched movies list and time spent watching movies.

This shows the single responsibility principle in multiple ways. Using one link to do its job and then redirecting to a proper page that handles the info for watched movies. This also shows in the use of the classes. Classes are used to their fullest, doing their job well, like having the ability to watch a movie on its and not needing an external class to access its info and add it manually.

This came up as a fun way to show users not just how many movies they've watched, but also what movies they've watched. This ties together the time they spent watching movies, how many movies they've watched and what movies they've actually watched.

New Feature 4: The ability to add a movie to a watchlist and remove it and watch list page

When clicking on a movie, not only is there the watch button, but there is also an option to either add the movie to their watch list or remove it. This feature requires a login. However, when a user is not logged in, the default button they see is the add to watchlist button. If they are logged in and the movie is already in their watchlist, then the button is changed to remove from watchlist. If the button is pressed without being logged in, they are redirected to the login page of the site.

When the button is clicked, the user is redirected using the user_bp blueprint. I decided to put this in the user blueprint as it is evident on the relevance it has to the user.

This works similar to the watched movies button, as it has a link that adds the movie to the watchlist, after grabbing the user and grabbing the watchlist in the user object. After grabbing this object, it then uses the method add movie and adds the movie if it is not there yet.

The opposite is also true for the remove movie from watchlist button. It gets the user, and its watchlist, but then uses remove movie instead of add movie. If the movie is not in the watchlist, then the movie is added to the list.

It then redirects the user to the watch list page where you can view movies in your watch list. It has a limit of four per page and it allows the ability to scroll through using the next/back/last/first buttons like the browse page did.

Just like before, I used this redirecting method due to the fact that if a user refreshes the page, they can repeatedly add a movie to their watchlist.

I added this so that users are able to keep track of the movies they want to watch without having to rely on an external tool, like writing on a piece of paper they could lose. This makes it more likely that a user will keep using the site instead of leaving it.