A2 Flix Skeleton Web App

New Feature:
Watchlist for each user
Functions of it:
When user click on the 'like' button below each movie, the movie will be added to the user's watchlist which can be seen on the navigation bar.
And if user do not click on fresh button, user can switch account, add movies for another account then switch back, the watchlist for both accounts will be kept.

Key Decision:

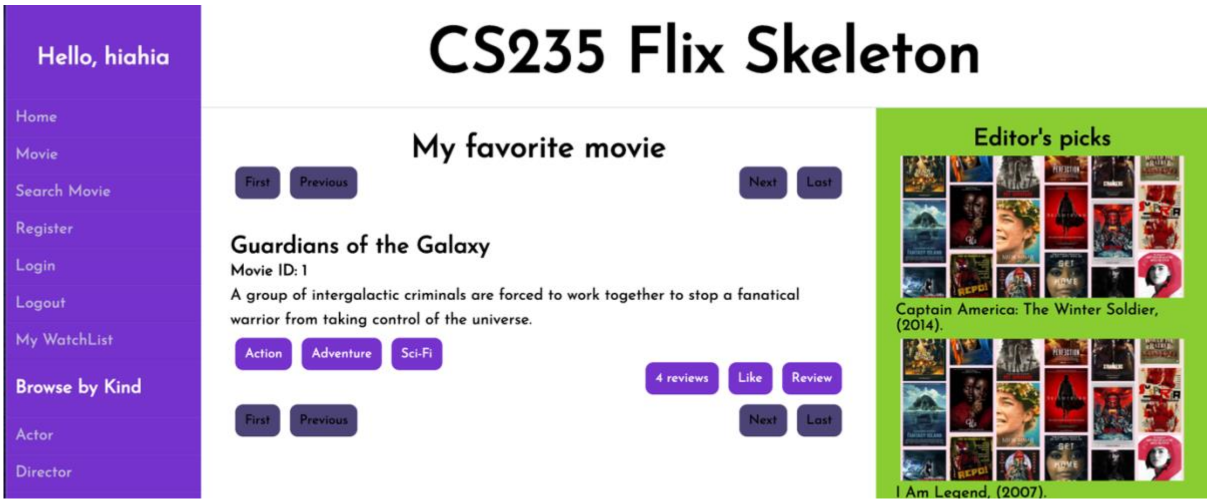
1. Single Responsibility Principle:
   Each html has its own named blueprint and its own service layer.
   It is easy to debug and separate functions to each page.
2. Reflection/ Dependency Inversion
   Abstract repository "Repository" gave the main idea of functions, then 'MainRepository' extends it, and as well as actor- repository, director-repository and genre-repository.
3. Navigation Bar Length
   Navigation bar sometimes is too long in some browser, so I delete the logo.png to fit my window size.
   And put viewing by genres urls into the tab 'genre' to minimize the navigation bar length
4. Pagination
   1) Pagination for actors and directors used 'flask_paginate'
   2) Pagination for movies by year used 'separate movies by year'
   3) Pagination for movies by genre used 'separate movies by genre' and 'cursor' to limit movies in each page
5. For watchlist:
   1) Add one new route named 'add_to_watchlist' and redirect to the page where 'like' button was clicked. It contains the main function to add the movie into user's watchlist. For this function, I made 'login_required' for viewing movies which means users can't see the movie list before they login.

```python
@movie_blueprint.route('/add', methods=['GET'])
@login_required
def add_to_watchlist():
    movie_id = int(request.args.get('movie'))
    username = session['username']
    movie= services.get_movie(movie_id,repo.repo_instance)
    user = repo.repo_instance.get_user(username)
    services.add_movie_to_watchlist(username, movie_id, repo.repo_instance)
    user.watch_movie(movie)

    return redirect(url_for('movie_bp.movies_by_year', year=movie['year']))
```

2) I created a dict to store username and its watchlist. I used to use the ‘watched_movies’ attribute in User object, it can pass all unit tests, service tests I gave it. But when I click the ‘Like’ button it can’t work in real web app. So I store all data in my movie_repository.

```python
    def add_favorite_movie(self, user: User, movie: Movie):
        user.watch_movie(movie)
        if user in self._watchlist_dict.keys():
            if movie not in self._watchlist_dict.values():
                self._watchlist_dict[user].append(movie)

    def get_watchlist(self,user):
        return self._watchlist_dict[user]
```

3) It also extends the functions of movie such as ‘clicking on genre button then you will be redirect to the page contains all movies in this genre’, ‘clicking on reviews button then can see the reviews for this movie’, ‘clicking on review button then can add review for this movie’.



CS235 Flix Skeleton

My favorite movie

Guardians of the Galaxy

Movie ID: 1

A group of intergalactic criminals are forced to work together to stop a fanatical warrior from taking control of the universe.

Captain America: The Winter Soldier, (2014).

I Am Legend, (2007).

4) If the movie is more than 10 then the buttons
   ‘first’, ’previous’, ’next’ and ‘last’ will be able to click.

Test Case:

1. In test_movie_repository.py :
   1) test_add_favorite_movie
   2) test_get_watchlist
2. In test_services.py :
   1) test_add_movie_to_watchlist
   2) test_get_watchlist