**CMPT 470**

**Project Report**

**Overview**

The main goals of our project is to build a web application that randomly generate movie titles using a local database with extensive search feature and a movie management system for users. The project was conducted by a team of four during the last month of the semester. The technology stack that was employed is React front-end with Bootstrap Material UI Design, Express back-end and MySQL database, deployed with Docker and AWS.

**Team members and responsibilities**

Rico Chao (**rchao**) – Front-end developer, Dashboard and filters

Yoel Yonata (**yyonata**) – Front-end developer, User Movie List

Jason Nguyen (**pdnguyen**) – Back-end developer, server-side controllers and SQL

June Shim (**jys2**) – Back-end developer, server-side security/authentication and DevOps

**List of features**

* Secure user authentication
* Random movie generator based on elaborate filter options (such as type, rating, genres, year released, runtime, vote count, title includes)
* Manage user movie list (add/remove a title)
* Mark a movie as Watched/Not Watched/Favorite
* *Trending*: Provide list of movies that are most favorite (based on users data)
* *Most Favorite/Watched*: Provided list of most favorite/watched movies (based on users data)

**Technologies and Implementation**

Here is the detailed list of frameworks and libraries used in the development of the project:

Client-side:

* ReactJS
* Material UI Design

Server-side:

* ExpressJS (with jsonwebtoken)
* Bookshelf ORM
* Knex query builder
* bluebird (Promise library)
* MySQL server
* node-fetch for making external API calls

DevOps:

* Amazon Web Service (AWS)
* Docker

A local database was populated by processing a text data set provided by [IMDb](https://www.imdb.com/interfaces/) (for free) that includes movie titles and movie ratings, separately. The two tables was loaded into a SQL database using a **tsv** parser in JavaScript and then combined together to produce the main data set (titles\_with\_ratings) that we can query on.

Client-side is built with RESTful design that makes requests to server routes. The requests are signed with **jsonwebtoken** for later authorization and handled by different server controllers. These controllers take care of parsing the requests, authorizing users and making secure SQL queries using ORM and query builder. The results were then embedded with some details retrieved from an external source (by making synchronous API calls to [TMDb](https://developers.themoviedb.org/3/getting-started/introduction)) and sent back to client-side with appropriate response status code.

The application was then deployed on AWS using Docker.

**Limitations and Constraints**

Currently, our movie database is entirely static since the process of parsing the text data set (of few GBs) and loading into SQL database was not streamlined. This will be improved in the near future to keep our local data set up-to-date. The design and complexity of the front-end interface was intentionally simple but was very minimal given the time constraints and knowledge limitation of framework used (by some team members). For example, the Alert component should be implemented instead of browser window alerts that sometimes give users uneasiness when navigating the site.

There are also some incompatibilities between our local movie database (from IMDb) and the database that was used for requesting poster images (TMDb). Users will occasionally see blank posters (or even incorrect) when searching for movies due to this limitation. IMDb itself does not provide a free and public API for making image requests (hopefully they will soon).

On server-side, the responses are sometimes delayed due to joining of large tables in the database. On average, each request would take 1.5 -2 seconds to process and get the results back. This could be another hindrance that annoys the users.

**Alternative applications**

Both IMDb and TMDb offer users advanced search features based on their own movie data set. However, NO options for generating random movies was found on both site. There are a few websites that include this feature but the filter parameters were very limited. Here are some examples:

<https://www.randomlists.com/random-movies>

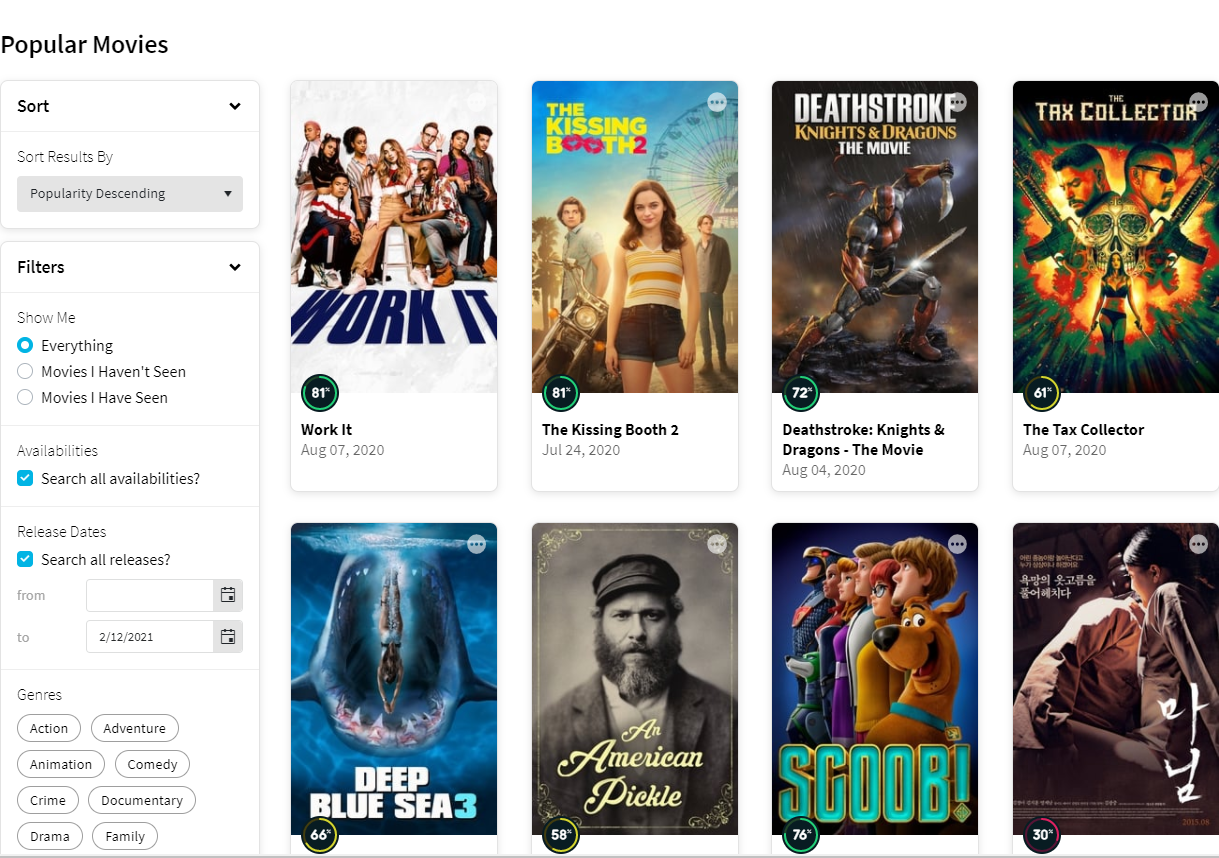
<https://www.generatormix.com/random-movie-generator>

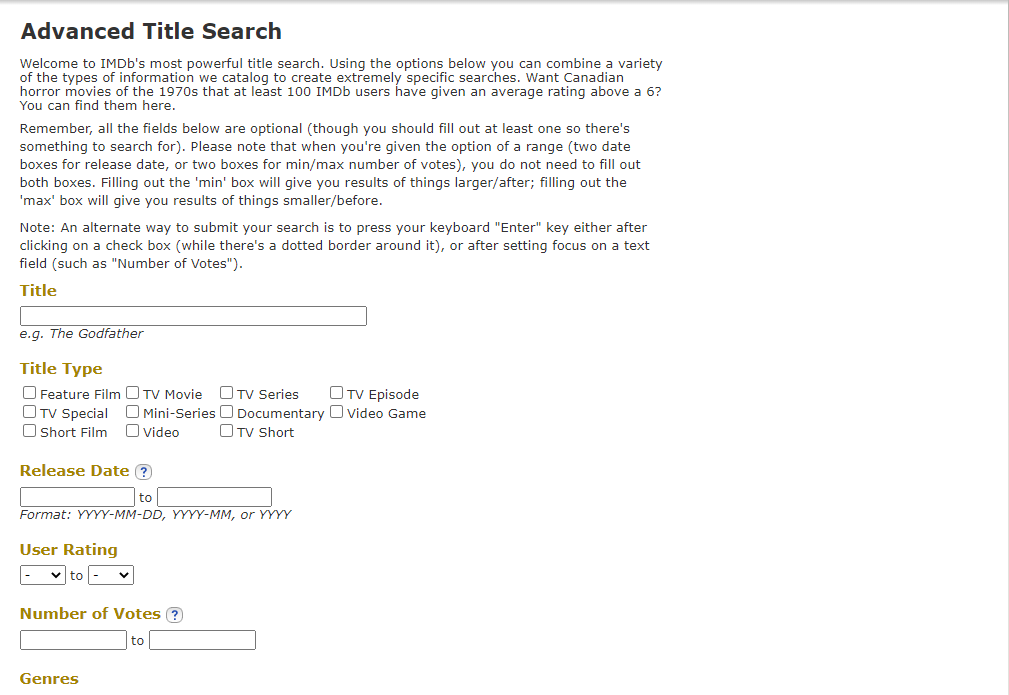
<https://pickrandom.com/random-movie-generator/>

Our application was created to try to solve this paradox, by giving the users more freedom in choosing what they can filter out (creating personalized filter) and combining the two most popular movie data sets on the Internet.

<https://www.imdb.com/search/title/>

<https://www.themoviedb.org/movie>





*Image:* Both IMDb and TMDb have their advanced search options.

**Future Developments**

Development of the project will be carried on after the course to build more features into the web app and to create more usable front-end components. Extended functionalities might include a Search All option with pagination, Save filters for each user, a Sort by option and Search function for movie title lookup in User Movie List. The ultimate aim of this project is to deploy the website to public use with a domain name and HTTPS.