COMP 2406 Winter 2021 Project Report

Section 1 – Initiating database and server.

1. Navigate to the appropriate file directory in a CLI of your choosing.
2. Type and enter into the console “npm install” to install the necessary dependencies for this application.
3. Type and enter into the console "mongod --dbpath=database" to connect to the database.
4. To initialize the base state of the server, type into another console (not the one with an active connection to the database), “npm run db”, which runs two database initialization scripts sequentially. These create initial documents and precomputes some information necessary for the rest of the application. Due to the nature of these scripts, the 2500 Movie set may take some time, but be assured it DOES work and will load in the data after it is finished.
5. At this point, the server is ready to be launched, and is done by typing and entering into the console “npm run start”.
6. Initially, you will be greeted with the log in page, navigate to the signup page by selecting the button, and entering details to create a new account.

Section 2 – Overall Design

Functionality of User Accounts

- To use the application, a user must first be logged in, as such they are only allowed to view the 'Log In' and 'Sign Up' pages for purposes of creating a new user account, or logging into an existing one. Upon logging in/signing up successfully, this user is redirected to their own profile.
- When a user is viewing their own profile, they are allowed the opportunity to change between being a regular user or a contributing user, by selecting the choice they would like to be and pressing the 'Change Role' button. By default, a user's account type will be regular, and only when they are a contributing user, will they see a 'Contribute' button appear on their profile, which allows them to contribute a new person or new movie to the application's database of information. Through regular navigation of the website, there will be no other way for a user to access the contribute page if they are a regular user, and if they try to access the '/contribute' page by manually typing in the URL, they will be denied access. No previously contributed people or movies will be altered if the contributor of these data switches their account type after the fact. By controlling the access of the contribute page in this manner, it is believed that this requirement is fully implemented.
- When a user is viewing their own profile, visible is a subscriptions button which redirects a user to a list of the people and other users they are subscribed to (following). From this page, a user can click on any person or user they follow and be redirected to the respective page. When a user is viewing their own subscriptions page, they can also stop following any person or user that they currently follow.
- Upon initially signing up with the website, a user's watchlist will be empty. After adding movies to their watchlist, this information will be saved in the database until the user removes the movie from the database manually. This will be done by selecting the 'remove' button which is attached to every movie in the watchlist.
- Upon initially accessing the website, a user will not have any recommended movies. After viewing some movies and revisiting their own profile page a user will be recommended a short list of movies, that will be updated upon visiting other movies in the database in the application. The algorithm for this will be discussed later below.
- After subscribing to users or people, a user may receive notifications in the event that:
    - A user they follow posts a new full review on a movie. From this notification, the user can be directed to the movie for which this review was made.
    - A person they follow features in a movie that has recently contributed to the database. From this notification the user can be directed to the newly created movie.

  In both cases, the user can remove this notification from their own profile by simply clicking the delete button on the respective notification.
- From the index page, accessed (after successfully logging in) at `localhost:3001`, or by clicking the 'MOVIE NIGHT' button in the top left of the page, (this button can be accessed from any page) a user is directed to the search page. Per the instructions given, a user can search for movies by either title, actor name, and/or genre keyword. A query is built up from any combination of these parameters given, and the user is redirected to a page containing the

results of this search. Through these results, pagination is supported, however instead of 10 movie results displayed, a design choice was made to display 4 rows of 4, bringing it to 16 movies displayed on each results page.

Functionality of Viewing Movies

- When viewing a movie page, the basic information including title, release year, average reviewing rating, runtime and plot is displayed. Initially, no movies will have an average reviewing rating because there will be no reviews for the movie. After any review is added, (basic or full), this movie will have an average review rating displayed out of 10 on the page. This is updated, every time a new review is added to this movie.
- Each of the genre keywords for the movie is displayed and allow the user to navigate to search results that contain movies with the same genre keyword.
- The director(s), writer(s) and actors the movie has are displayed and this allows the user to navigate to the respective person's page.
- A list of similar movies is displayed on each movies page. The algorithm for this will be discussed later below.
- A user is able to add the movie currently being viewed to their watchlist, by selecting the 'Add to watchlist' button visible on the page. A user may not remove a movie from their watchlist when viewing the respective movie's page, however if they attempt to add it to their watchlist when it is already there, the user will be alerted that they cannot. They may remove movies from their watchlist from their profile page.
- A list of all full reviews made for the movie currently being viewed, is visible on the page. These reviews display the score given, the review summary and the username of the user who made the review, as-well as a link to this user's profile. Upon pressing the attached 'view details' button on any of these full reviews, the user is redirected to the full review which includes the full body of the review.
- A user can submit a basic review for this movie by typing a score between 1 and 10 and selecting 'Submit Basic Review', or a full review by filling in the score, review summary and full review fields and selecting 'Submit Full Review'. If a user attempts to submit either type of review without all of the necessary fields fulfilled, they will be alerted that the form is incomplete, and therefore a review will not be created.

Functionality of Viewing People (directors, writers, actors)

*The phrase ‘person/people’ will be used to describe either a director, actor, or writer of a movie.

- A person’s page, can be accessed via the following options:
    - By browsing through the ‘film-crew’ button in the header which redirects the user to a list of all film-crew present in the database.
    - From clicking the name of the person in interest under a movie in which they have worked.
    - From clicking the name of the person of interest when viewing another user’s subscriptions page.
- When viewing a person’s page, their full work history is displayed whether it be a movie they have directed, wrote, or acted in. It is possible to navigate to any of the movie’s listed in the work history by simply clicking on the card.
- The top 5 most frequent collaborators (if that many) are displayed on a person’s page. The most frequent collaborator information for people adding into the initial state of the database, are precomputed in one of the initialization scripts, and any subsequent people added to the database will not have any frequent collaborators until they are added to a movie.
- When viewing a person’s page, they can be subscribed to by clicking the ‘follow’ button which is displayed under their name. You may not unfollow a person from their page, however you will be alerted if you attempt to follow a user that you already follow.

Functionality of Viewing other Users

- When viewing another user's profile, all of the reviews this user has made are visible on their page, as-well as links to the movies for which these reviews were made.
- The subscriptions page of a user is available to view, which displays both the user's they follow and the people they follow. When viewing the subscriptions page of another user, the current user is unable to manage (remove) any of these subscriptions, only view and navigate to these subscriptions.
- The watchlist of the user profile currently being viewed, is displayed on their profile, but the current user is unable to manage (remove) any of these movies from the watchlist, only view and navigate to these movies.
- A user may subscribe to the user being viewed, by selecting a 'follow' button. A user may not unsubscribe to a user from viewing the user's page, however if they attempt to subscribe to a user that they are already subscribed to, they will be alerted of such. They may unsubscribe from this user in their own subscriptions page.

Functionality of Contributing Users

- A contributing user may navigate to the contributor page, through a link available on their own profile if and only if they are currently a contributing user.
- From this contributing page, a contributor may add either a new person or a new movie to the database. If either the person or movie the user is attempting to add already exists in the database, they will be alerted of this and not be allowed to do so. (Assuming as allowed that all names of people and titles of movies are unique).
- When adding a movie, this entire creation process is done on a single page as required, and through AJAX the user is allowed to dynamically search for people within the database to add to the movie as either an actor, director, or writer without having to type in the full name of the person.

Section 3 – Critique of Overall Design of Application

Frameworks used:

- ExpressJS as the backend framework for Node.js.
- MongoDB to store and query data.
- Tailwindcss which is a utility-first CSS framework.

Overall, I believe that the code quality of this project was clean, efficient, and most importantly effective. Where possible and deemed necessary, complexity of certain functions was abstracted away into general purpose functions to improve readability and overall design of the code. To this end, one area of the project in which I acknowledge could have been done more efficiently with regards to time complexity, would be the second database initialization script in which several precomputations were done to ensure the server at initialization, had less work to do. This script includes several heavily nested 'for' loops, which may not scale the best if the data set had increased further, and if time permitted improvements would be made to account for this. The benefit of getting the precomputation done on this end, was aiding with the minimization of required data transfer for certain requests on the server side and allowing a smoother overall experience in the application once the database had been initialized.

This application follows as closely as possible the restraints of a RESTful system:

- There is a clear distinction between the server and client, in which the server-side stores the necessary resources (data, rules and logic) to respond to any requests made by the client. The client-side on the other hand is responsible for making requests to the server with meaningful data and doing something meaningful with the response.
- The server is stateless, so that any given response from the server is independent of any sort of state, and self-contained (dependent only on data that is held).
- There is a fully functional uniform interface, that follow the four sub-constraints:
    - Resources are organized in a way that they can be manipulated in a meaningful manner.
    - Json is used to represent the resource as it is being exchanged between client and server.
    - Each RESTful request contains enough information to be understood in isolation.
    - Hypermedia as the Engine of Application State

An effort was made to use the most appropriate HTTP status codes when necessary.

Error handling was done in a manner in several parts throughout the application, to minimize the job of any one part of the application. The client-side ensured that the data sent to the server was of the expected types and had no unexpected exclusions or inclusions of data. Subsequently, this was one less task for the server, which only dealt with processing this request on the basis of whether if it was possible to be made even with the correct data. The use of strict mongoose schemas ensured uniform and consistent creation of documents throughout the application, which had all the information needed prior to creation to interact with the application in the database. The use of unique fields in the document models, protected against the creation of 'duplicate' objects, where a field such as the username of an account, or title of a movie/person was intended to be unique and only appear once in the database.

Asynchronous operations were used almost every time the server was interacting with the database, so promises were used to associate handlers with these asynchronous operations. If these promises are prefixed with the ‘await’ handle, the execution of this function is halted until the promise is settled (fulfilled or rejected). This allows for more stable behaviour and avoids circumstances such as race conditions.

Section 4 – Movie Recommendation & Similar Movie Algorithms

Similar Movie Algorithm

To suggest similar movies, the algorithm developed pivots on genres. When viewing a movie, all of the genres which this movie has, is used to query from the database any movie that has at-least these genres. For example, if a theoretical 'Movie A', has three genres, 'Comedy Romance History', any movie found in the database that has at-least these three genres (excluding the movie currently being looked at) will be the first set of movies that populate the array of similar movies, and are concrete. Knowing that there may be several instances of 'Movie A' having a combination of genres that not enough or no other movie has, the next thing done is a match by only one genre. One genre randomly selected from this list of genres that 'Movie A' has, is selected and used to query from the database movies that match this genre since this will likely generate enough results to fully populate a set of similar movies for any particular movie. This also adds a bit of randomization to the process, so upon refreshing a page, the entire set of similar movies will never be the same.

To be noted, is that in the case of a new movie being added, it is added with no genres (as per the spec), so this algorithm is slightly modified, to select random genres from the database's unique set of genres for which to produce this list of similar movies from. At this point, the algorithm is not so similar, and could be improved upon to account for the addition of new movies into the database, perhaps by the film crew working in the movie.

User Specific Movie Recommendation Algorithm

The recommended movies algorithm only kicks in upon viewing a movie page, because only at this stage would the application have any useful information about the user's viewing habits. This algorithm piggybacks on the previously described similar movie algorithm and attaches a subset of the similar movies of the most recently viewed movie to the user's session data. This way, the user's recommended movies will change anytime they view a new movie and is based on what the user was seemingly interested in based on their current browsing session within the application.