Adrian Alexander, 101150602

Instance information:

- Public IP: 134.117.130.102
- Username: student
- Password: arar0625

No database, as it uses JSON files to store information

To run, to the main directory (~/Project/Movie-Database) and run with the command “node server.js”

Testing instructions

- Go to http://localhost:3000 to start. Click on the movie icon to proceed to the login page.
- Click on the register button to register a new user. Take note that you are required to fill in all the fields or else it will not let you to register for the site. Also try to register with an already registered user (admin@gmail.com). It won’t let you. Once you made a user, click register. It will redirect you to the login page.
- Login with the user you just made. Copy the URL (should be http://localhost:3000/private/home), and log out. Try to access the URL again. It will redirect you to the login page. This is my implementation of user sessions.
- Log back in, but with these credentials (email: admin@gmail.com, password: admin). You will notice that if you keep refreshing, it shows a different list of recommended movies each time. That is because the movies are based off the user’s watchlist and favourites list.
- Go to ‘All Movies’. If you scroll all the way down, you can see an arrow that will bring you to the next page. Click on the arrow to proceed to the next page. If you scroll down again, you can see that there are two arrows now. You can also change the page number/the number of movies being displayed by changing the URL parameters. E.g. You can set pageNumber to any number greater than or equal to 0 (the last page is 456 with a page length of 20, but if you try to set to a number more than 456 it won’t give an error, it will just show a blank screen. Setting the number to a number less than 0 will give an error though.)
- Click on any movie image. You can see that it uses the imdbID as it’s unique ID, so there won’t be any overlap on movies and you can also paste an imdbID of another movie in the URL and it might be in the database. You can see all the basic movie information like name, plot, genres, and list of people involved in the movie.

- Click on any of the links, and then press search. It will take whatever is in the search parameters and search by the type, which is also in the URL. These links are also clickable.
- Click on a movie again, and click on a person. You can see list of movies they were involved in and what their role was. All these links are also clickable.
- Navigate to the Search bar. Input 'Star Wars ' and set the filter to movie. It will show all the movies that include the keyword 'Star Wars' that match the type 'movie'. It also isn't case sensitive, so you can input 'star wars' or 'sTaR wArS', and it will still work. Note that the page doesn't refresh, because it utilizes AJAX to render the search results.
- Navigate to 'My Account'. This takes your user session's information, and renders your information like your favorite movies, watchlist, and people/users you are following. Since you're logged in as me, I already have things in my favorite movies and watchlist, as well as am following people and a user. The top 2 lists are links to movies. You can click on any of them. On the movie page, you will notice that you can add or remove the movie to your watch list and favorite movies.
- Go back to 'My Account', and click on any of the links under 'People Following'. You can see that you can choose to follow the person or not.
- Go back to 'My Account', and select the user under 'Users Following'. In the URL you can see he has a unique ID, as it is generated based on time. (Date().toString() or something like that) You can see the same details as your account has. You can also choose to friend the account or not.
- Go back to 'My Account', and click on "Click here to change account type to normal' and then click 'Contributor options'. It will redirect you as you are not of type 'contributor'.
- Click 'Click here to change account type to contributor', and then click 'Contributor options'.
- Click on 'Add a movie', and fill in the fields with any random gibberish. If you submit and check the server console, you can see that it says "A person doesn't exist in the database". Also try adding a movie again, but set the imdbID as 'tt0027660'. If you submit and check the server console. You can see that now it says "Movie already exists in the database". Try to add a movie in one more time, but this time set the writer as Joe Jonas, the actor as Nick Jonas, and the director as "Chris Evans". Submit, and you can see that it successfully added the movie to the database.
- Now, click on 'Add a person', and type in any person. If its not in the database already, the log will say 'Success!', but if you try to add the same name to the database again, it will say "Person already exists in the database".
- Go back to a movie page, and make sure you're a contributor. First, try to add a review. Once you're done and click submit, reload the page (I couldn't get it to work without refreshing the page unfortunately) and you can see your review.
- Go back to the movie page, and
- You can log out now, the step-by-step instructions is done now lol

Functionality I have implemented:

- Implemented the ability to create accounts with an email and a password, with error-checking for email uniqueness.
- Implemented user sessions, as well as an authentication route so if a user tries to access the website without a login, they will simply be redirected to the login page.
- Allowing users to change between a regular and contributor account, where the contributor access will allow them to add people and movies to the database.
- Allowing users to view and manage the people and users they follow.
- Implemented searching for movies via a search input and a type, such as title, name, genre, and year, and making clickable images that lead to the movie page.
- When viewing a movie page, the user can see the basic movie information such as title, release year, average rating, runtime, and plot.
    - The movie page also includes genre keywords that, when clicked, will bring you to the search page to search the genre keyword by the genre type.
    - The movie page also includes a list of the people involved (director, writers and actors)
- When viewing a person page, it lists the movies they are involved in, categorized in what role they play as.
- When viewing a user page, it shows their details such as their watch list and favourite movies.
- When viewing a user's own page as a contributor, they can click a button to go to the contributor panel, where they are able to add a person or a movie. There is error checking for both functions, as contributors can only add a person if they aren't already in the database and can only add a movie if the movie's people are already in the database and if the movie itself isn't in the database yet.
- Created a public API to search for and view a movie, person, or user's information, and add a movie to the database
- Made a recommended movies feature that is based off the user's favourites list and watchlist. Shows 5 random movies that the user might want to check out.
- Added the ability for any user to create movie reviews on the movie page. Users can give a written review, and/or a numerical review, and it will be on display for all users to see.

Functionality I have not implemented:

- Related movies on the movie page
- Frequent collaborators
- Notifications for movies added that involves a person whom the user has followed

- Notifications for following users' reviews
- Ability for contributors to edit movie details, except for the poster and imdbID, in case they would like to add people

Design decisions:

I first want to talk about the search design. So, the search function takes either the text from the search field and the filter list, or if there is URL parameters, it will take those. This is done so that movies can easily be searched whether the user arrived on the search page via the navigation bar, or clicked on by a link from a movie page. The search algorithm itself is pretty simple. It takes the lowercase version of the search input, and goes through all the movies. If the search term matches any title, genre, people, or year, and if the type matches what the user used, it will add the movie information to a filtered list, and push that onto the search page. This means that searching 'George Lucas' with type 'director' and searching 'George Lucas' with type 'Writer' will yield different results. Overall, I am happy with this design, especially because I can just add another if statement to add another condition, so it's pretty scalable in that sense. Another really well designed part of my website that I'm proud of is the addPersonScript script. Originally, I used it to get all the people from the movie-data file quickly, but I ended up improving it's speed and using it as an important part of my website. It is used to update any person I added from a movie that doesn't actually have a list of movies they worked on yet. It uses a map, so each name is unique. Finally, even though it's a minor design decision, but I'm glad I used the Nord theme. It's easy on the eyes because it's not too hard to read (not too light, not too dark, so it can be viewed on any time of the day), and the Nord color pallet and font is just the best theme in my opinion.

Improvements:

Honestly, there can be a lot of improvements to be made. Starting off with the CSS design. I heavily dislike the way I designed the looks and aesthetics of this website, and I have not put in enough time to learn how to make the website beautiful. The only thing I really put effort into were the buttons, which I used all over the site because I liked them so much. I also wanted to include smooth animations when transitioning from one page to another use AJAX more to increase user experience and include a night mode (even though the site's background is already pretty dark).

Onto functionality improvements, there are many I would like to list. The major one is the reuse of common code for some of the routes. For the movie, user and person routes, I have a function to get the current user session, and a function to check whether or not the current user session is following the user or person, or has the movie added their watchlist/favorite movies. I could have put these functions in their own file, export it, and call them with specific parameters whenever I use them to save space/time. Same thing with the EJS files, as most of them were either labels and inputs, or tables with a bunch of table rows and data cells. I

probably could've found a way to auto-generate them. The business logic is also something I can improve on. A lot of the code is getting the path, using fetch to send a GET request to the path, getting the current user session, and checking whether the user is following a person/user or not, or has a movie in their watchlist/favorites or not. Another improvement I would make is obviously to use a database instead of JSON files, as I could've replaced checking for the current user session and opening and parsing user/movie/people data completely. Unfortunately, my edit movie function isn't working, and I'm not sure why. It worked in Postman, but not for the web client for some reason.

Another thing I should've implemented are alerts/notifications. I don't think I have any notification system to tell you if there's something wrong or not, just console logs. I wasn't sure how to implement those, even though I heard they were easy, so I will learn them for the future.

Finally, I also need to improve on the search function. Currently, it meets the spec's expectations, but it could be condensed. And it reads and goes through the file each time a search is conducted. This is done because a movie could be added/updated while the site is up, but again I could've used a database to speed it up a bit faster.

Modules, frameworks and other tools:

- Body-parser: I used this a bunch to parse req.body requests.
- EJS: I used this and not Pug because even though I think everyone recommends Pug, I like just writing HTML and then wrapping it in EJS, and also it was just really simple to learn.
- Express: Express is a really simple middleware to learn and use. I enjoyed learning how to use this, and will continue using it in the future.
- Nodemon: really handy for auto restarting the server.

Final thoughts about the project:

The thing I liked the most about the project was making the login and register CSS and logic. It was interesting learning about authenticating and how to set up a login system with sessions. The best feature about my website is the search functionality. I believe it is very scalable, and I would probably use the same algorithm should I choose to make another search function for a different website.