COMP2406A
Project Report

Student Name: Sheikh Fahim Anwar
Student Number: 101141744
Project – Movie Database
Go through the README as well as reading this project report before using the application. README details the testing of the program. The project's code has also been well commented explaining more specifics of the functionalities which you can take a look at if you have time and need further clarification.

1. OpenStack Details and Instructions for Running the Server

a.

OpenStack instance information –
Instance Name - SheikhFahimAnwar
Public I.P address - 134.117.132.31
Username - student
Password - solomongrundy1

b. No database for the application
c. No initialization script required to run manually
d. All the dependencies for the system are already installed, to run the server –

After making the ssh connection, there will be a directory called project in the starting directory itself (in /home/student). Navigate to the project directory using cd project. Then use the command node server.js to run the main server. When starting the server, it'll take some time to actually run because it's initializing the server with the movie data. It'll print out when it's done initializing and it's running. Then in another terminal window create the ssh tunnel and connect to server through your local computer.

The documentation for the Public JSON API is available in the README as well as the directory structure for the project.

2. Summary of Functionality Implemented

All the functionality required in the project specification has been implemented.

- Users Accounts
  1. They are able to make new accounts by signing up on the sign-up page by specifying a username and password. Username should be unique. Only one user can log into the website in a single instance/session. The previous user will need to have logged out. Users cannot perform any other actions except accessing home/sign-in/sign-up page without signing into the website.
  2. If they already have an account, they can sign in through the sign-in page.
  3. Users can access their profile page which shows them their username, user type, followers, users and people they're following and their recommended movies.

4. The users and people they follow or are followed by can be navigated to, to view their pages. The movies have links too which can also be navigated to.
5. Users are Regular users by default, they can upgrade to a Contributing user by clicking the 'Switch Account Type' button at the bottom of their profile page.
6. There is also a 'Contribute to Cinephile database' button which allows ONLY Contributing users to access a page which allows them to add new movies and people.
7. Anytime any user they follow adds a new review, a new notification is added to list of notifications. Also, if a person they follow has a new movie added, user gets a notification. However, this does not use polling, but instead new notifications are only loaded when page is refreshed or navigated to. There is a 'Clear Notifications' button which user can use to delete all notifications.

- Searching and Viewing Movies
  1. Users can search for movies with the search bar available on the navbar which only appears after users sign in. When the search button is pressed, user will be taken to a search results page showing the results if any and the number of hits the search had. The movies have links which takes the user to their movie info pages
  2. In the movie view page, users are able to view all the basic information of the movie including – title, average user rating, runtime, plot, list of genres it belongs to, the director, writers and actors involved in the movie. The page also shows a list of similar movies to this one whose movie pages can be navigated to. The user can also navigate to the people's pages involved in the movie. The movie page also shows a list of full reviews that have been made for this movie. Furthermore, it has a section for adding either a basic or full review. Both involving giving 0-10 rating, but the full review also involves adding a summary and full text for the review. The reviews also show the user that made the review.
  3. The user can click on the genre keywords of the movie to get a list of movies of those genres similar to the search results page.

- Viewing Users and other People
  1. When viewing a person's page, the user can follow/unfollow the person. The person's roles can be seen too, i.e. – Actor, Writer, Director. A list of the person's movies/works can be seen which the user can navigate to. A list of frequent collaborators for this person is also available. If the user follows this person, anytime a movie with them in it is added, user will get a notification.
  2. When viewing another user's page, the current user can follow/unfollow them. They can view all the full reviews made by them. They can also see all the people that they follow. If the current user decides to follow this user, anytime they make a review, the current user will get a notification.

- Contributing Users
  1. By default, users are regular users when they sign-up. They can change to contributors from their profile page. Contributors can do everything regular users can.
  2. In addition, contributors can access the 'Contribute to Cinephile database' page from their profile which allows them to add a new movie to the database if it doesn't exist already and only if the people in the movies already exist in the database.

3. Contributors are able to add new actors and writers to movies if they exist already. This can be done on the movie view page.

3. Extensions to the application

In addition to meeting the base requirements for the application, I placed more effort to the UI and UX for the application. I designed the UI and front-end in hopes of it being more pleasant to look at and browse. I also designed the navigation and usage of the website making it more intuitive for the user to use and have appropriate feedback for any actions the user performs. I incorporated and took into account the problems I face as a user when browsing the web to make my application better. Few of them will be explained in the 'Design Decisions' of the program after this section

Some of the UI and UX decisions –

- I decided to use a two-toned color pallet for the website, with dark grey navbars and footers with a lighter gray for the screen in between. I opted for darker colors for the UI as people generally tend to prefer dark mode and because lighter colors tend to cause greater eye strain.
- I designed the navbar and footer to be responsive and show feedback when hovering over the elements. The navbar also shows which page you're currently on to give feedback to the user.
- I also tried to make navigating between pages as fast as possible and tried to ensure the fewest number of clicks is needed to get to or do something.
- When a user isn't signed in, the user will be redirected to the sign-up or sign-in page indicating they need to authenticate with the system to proceed.
- I browsed Google fonts to pick the fonts that were most pleasing to look at instead of using the default ones. I also made sure to not use more than 2 different fonts in a page as it can get very distracting otherwise.

4. Design Decisions of the Program

- A user who hasn't logged in to the system cannot access any page except homepage, sign-in and sign-up page, if they do try, they'll be redirected to the sign-in page. The search bar for movies which is usually available on the navbar also doesn't appear till they sign-in
- Once they sign in, they'll be redirected by their profile page, they will no longer be able to access the sign-in or sign-up pages
- After logging in the sign in button will become a logout button
- When user logs out, they're brought back to the home page.
- I've separated the usage of the system through the interface and through the public API into two - so to access the public JSON API and the functionality mentioned in the project spec, routes will have to start with /api. It's detailed further in the README

- The button to contribute to the database i.e. - Adding movies/people, is present in the profile page and when clicked takes the user to that page only if they're contributing users, which they can switch to with a button on the same page
- The fields and the button for editing movies by adding actors/writers becomes visible only when the user becomes a contributor. Furthermore, user can add only actors and writers to the movie because I took the assumption of only one director per movie
- When a user tries to access a /user/:user route corresponding to themselves, they'll be redirected to their profile page rather than a general view of the user page
- Anytime a contributing user adds a person or movie to the database, they're redirected to the respective pages
- I've tried to limit client-side JS code as I want to expose as less as possible to end-users, also used HTML forms where possible
- I also ensured when sending objects to the pug pages, only the necessary information is sent, and passwords are never sent

'Similar Movies' algorithm –

When the movie JSON data is parsed, for each movie, an array of similar movies is generated. This is generated by iterating through all the movies for each movie and checking if they include the same actors, director or writer and add the movie if they do. The array is a max of 10 movies. If the array doesn't reach 10 movies, similar genres are used to add additional movies. I chose to use having the same director/actor/writer as a criterion for similarity instead of only having similar genres because I feel like that represents similarity better. Genres can be quite wide, and a lot of those movies would fit that criteria. Instead, users are usually more interested in watching movie with the same actors in it or having been directed and written by the same directors and writers.

'Recommended Movies' algorithm –

When users first sign up for the website, they are required to enter 3 of their favorite genres. The application then uses those genres to generate an array of movies the user might like. The array is then shuffled with the unbiased Fisher-Yates algorithm to shuffle the results and returns 10 movies, this method ensures the users get a new set of recommended movies and not only from the start of the movies array which stores the movie objects. New movies are recommended each time the user logs in, so they get fresh new recommendations each time. I felt like the method I used was the best combination of being simple yet effective in recommending movies.

5. Improvements that could be made to the system

Currently everything runs on the RAM, there is no database for the application. Adding a database to the system would have added persistence to the system and greatly increased the scalability of the system. However, due to time constraints I wasn't able to add a database to the system as it would involve completely overhauling the business logic I had in place. Right now, while running on the RAM, a large number of users couldn't use the system at the same time.

Another improvement that could've been made is using AJAX more often for partial page updates. Right now, I rely a bit too heavily on reloading the page completely. It is best to have as less data transferred as possible over the network to speed things up.
I also had a filter functionality in my business logic (you can still see it) which allowed the user to filter search results using three criterias – Runtime, Year Released, Average Rating, and in both ascending and descending order. However, I wasn't able to fully implement it into the actual web application in time.

6. No additional modules have been used except for the base ones which include pug, express, and express-session.

7. What I like most about my project?

I like that it has robust error handling, combined with express's error handling and thinking of edge cases I was able to handle a lot of errors that could happen and provide proper informative responses for the errors to the users. I also like that my system is fairly intuitive (at least I hope so) to use because of the effort I put into the user experience and the layout of the UI. The user gets proper feedback for each and every action they perform. Furthermore, I also quite liked how the design of my website looked in the end.