Project Report

OpenStack Instructions

IP address: 134.117.129.200
username:student
password:helenmathew

1) Open cmd/terminal and type in "`ssh student@134.117.129.200`" and enter
2) type in the password (helenmathew)
3) Open another terminal window, type in "`ssh -L 3000:localhost:3000 student@134.117.129.200`", hit enter, and type in the password
4) Type in "`cd CheckIn3`" (full filepath: "`home/student/CheckIn3`")
5) To start and initialize the database, type in "`node init.js`", hit enter
6) Running the init.js file logs the number of movies, people, users and reviews that have been added to the database. Next, you can start the server by running `node projectserver.js`
7) Go to http://localhost:9999/ on the web browser to view the project.

I also have a file named `generator.js` which uses the data from the provided movie-data.json file to create a random set of users and reviews (random lorem text) for my database to use. The init.js file uses the data created by the generator file (`movie-trimmed.json, userdata.json, reviewdata.json`), can be found in the data folder) to create the database. You do not need to run the `generator.js` file unless you would like to create a whole new dataset with new users, and new reviews. People and movies remain unchanged, unless you play with the code, for example, change the number of movies being used from the original data. If you'd like, you can use the following credentials to log in for testing, brandy is a contributing user:

- username: brandy
- password : brandy

Note that brandy will no longer exist if you run the generator.js file once again since it creates new random data.

Functionality

Successfully implemented

- 100 users are created on the system initially, these users can log in by clicking the 'Login' button on the navbar. To create a new user, there will be a link on the login form to a signup page or you can go to the /signup url. Only unique usernames will be accepted.

- Once the user logs in, their name will be displayed on the navbar, clicking this opens a dropdown menu which contains a link to the user's profile. On their profile, they will have the option to edit their name, username, password or account type. The dropdown menu also contains links to the notification page, logout, and if the user is a contributing user, they will also be shown links to add a new movie/person.
- The profile will also contain links to view their followers as well as the people they follow. These pages contain links to the profiles of everyone they follow/follow them. On their profiles, the user can follow/unfollow them if they wish to do so.
- The user will be shown recommended movies on the home page. When no one is logged in, the homepage displays a specific set of movies. But once a user logs in, this set of movies change to display the user's recommendations instead. I generate the recommendations based on the people the user follows. The user will be recommended movies that the people they follow have worked on.
- The user is able to search for movies/people/users. I have a general search bar on the navigation bar. The user can select the category they wish to search (movies/people/users, default is movies) and enter some text and results will be displayed. Empty searches are redirected to an error page. The same result can be achieved using the url. Entering http://localhost:3000/search?category=movies&search=t into the url box for example.
- In addition to this general search, there is an advanced search option where the user can mention the movie's title, genre, year or minimum rating to search for a movie. There is an 'Advanced search' link below the search bar which takes you to a separate page in which you can enter the fields and search for movies. The page also has forms to search for people and users by entering their names. This can be achieved using the URL as well. Sending a request to http://localhost:3000/movies/?title=toy&genre=animation&year=1995&minrating= displays Toy Story since that is the only movie that matches all those queries. If the title parameter was just "t" for example, it would match all the movies containing 't' in their title. In both general and advanced searchers, the user can navigate to the movie's page by clicking any of the results. http://localhost:3000/movies/ displays all movies, "/users" displays all users etc.
- Each movie's page contains its title, release year, rating, plot, runtime, average rating, genre and the people involved. The people and genres are links. Clicking any of the people takes you to their own page. Clicking any of the genre searches for movies of that genre, so clicking 'animation' displays all the movies in that genre.
- There is also a link to view movies similar to the current movie. I defined similar movies as movies that contain any of the genres of the movie in question. If a certain movie's genres are animation and fantasy, the program returns all movies whose genre list contains animation and fantasy.
- Anyone on the website can view reviews but only logged in users can add reviews to a movie.
- When viewing a person's page, the user will be able to see their previous work and frequent collaborators. They can also choose to follow this person.

- When viewing another user, the user can view any reviews they might have made, their follower/following list, and they can follow them.
- The user receives a notification every time a “person” they follow is added to a movie, and when a user they follow has created a new review. They also have the option to clear their notifications. However, a page refresh is needed to view the changes, which I understand is a limitation.
- Contributing users can add people and movies to the database. They will not be able to add a movie/person that already exists.
- All the minimum JSON REST API requirements are met; all the GET requests return the expected results, and the POST/movies successfully adds a movie to the database as long as it passes the validation. The validation check involves checking if a movie with the same title already exists, if any of the required fields are empty, or if the year and runtime are not numbers.

Partially implemented

- Frequent collaborators of every person includes anyone they’ve ever worked with, not just the people they have worked with the most.
- Notifications are added to the user’s queue, but they are only displayed on the notifications page after a refresh.

Extensions

All of the data my project uses is stored in MongoDB, I have also used Mongoose. My interface adapts to various screen sizes (still not very attractive though), I have also used some colours to make it look somewhat nice

Design decisions

- I modified the provided movie data into a more usable format. This includes splitting the writers, actors, directors, and genres into arrays, as well as removing parts of the strings like “(original story by)”. I also deleted many properties that are not required. All the functions I used to modify the data is in the `modify.js` file, although most of it is commented out just to be safe.
- Most of the validation of user input in POST requests is done client side to reduce the load on the server, for example, ensuring the input values aren’t empty, and type checking for inputs that need to be numbers. The request is only sent if these requirements are met
- I used template engine partials to add the same stylesheets and scripts to every page used. I also added a navigation bar this way. This reduced code duplication and made it easier to make any changes required, since I only have to edit one file.
- The user is able to clear notifications in case they have too many.

- I used regex for search patterns. So it’s easier to match movies according to the searches made by the user. I also used regex in POST requests to /movies and /people to check if a movie/person exists already with the specified title/name.
- I have implemented pagination for the search pages, this gives the user a better browsing experience rather than having to scroll forever.
- User has the option to add posters for the movies they create, but it has to be a link, file uploads do not work
- In the `generator.js` file, I used the faker npm package to create random user data and review data to populate the database. This file can be run multiple times to create entirely new sets of data every time.

Improvements that could be made

- The way I set up notifications is inefficient. Data is not seen as soon as it is available. A better approach would be using Socket.io since it enables event based communication between the client and the server. Polling is another possible approach, but Socket.io would be more efficient.
- When creating a new movie, the user has to type different people/genres separated by “, “ in order for the server to parse the input properly. For example “tom hanks, robin williams” would work fine but “tom hanks,robin williams” would not. This is because I split the string using the “, “ separator. However, this would mean that the user has to be extra cautious while inputting information in order to ensure that everything works as expected.

Other tools used

- faker: I used this package to create fake user data and review data.

Best features

I like the general search bar on the navigation bar, since it can be used for quick searches. I also like that when the user searches for movies, the posters are also displayed, and the pagination is nice as well. Clicking the logo from any page redirects you back to the home page, which I also find to be very convenient.