Movie Database Project Report

Vadim Boshnyak

COMP 2406

1.**OpenStack**

The instance name is VadimBoshnyak,

IP is 134.117.132.209

Username: student

Password: vadimboshnyak2406

project is at Dekstop/final

cd Desktop cd 'final'

To run node server.js

Everything is already installed and ready to go

The OpenStack seems a little bit slower than when I was running the server on my computer.

ssh -L 9999:localhost:3000 student@134.117.132.209

To run on your computer at localhost:9999

Password is the same: vadimboshnyak2406

Instance should be running currently

2. Functionality Implemented

- **Users** - Login - Ability to log in to an account, account must be created, or log in to an account that is already created. Checks for input, if incorrect it tells you the issue. Works as a form, with username and password. REST API - GET/login (renders login page), POST/loginUser logs in user

- Signup - Ability to create an account with a form, inserting a username and password, checks for input, if incorrect it tells you the issue. REST API - GET/signup (renders signup page), POST/signup creates user
- Switch Contributor - a logged-in user can switch between being a contributor or not on the user page. Works as a form. POST/contributor
- Ability to view the users and people the user follows, all people and users are a link so they are clickable to their user/profile page. GET/users/userid
- Ability to view the users' and people's followers, all people and users are a link so they are clickable to their user/profile page. GET/users/userid
- Ability to search for movies using title, name and genre keywords. Result sends as an array and each movie displayed can be displayed. Also available for non logged in users. GET/movies
- Frequent Collabs is implemented, only works with actors working together, not directors or writers.
- **Viewing movies** - /movie/id page shows all basic information as, title, release year, runtime, plot, directors, writers, genres, similar movies, reviews. GET/movies/movieid
- Genres of movies are clickable as a link, it searches for movies that contain that genre and returns those movies as an array, each movie is clickable and redirects to the movie's page. GET/movies/movieid
- Directors, writers and actors are also a clickable link that searches for that director/writer/actor, when clicked it finds that person then it's clickable as a link to that person's page. GET/movies/movieid
- Ability to see similar movies, the algorithm for similar movies if the movie has 3 of the same genres then it's a similar movie if a movie only has one genre, then it finds movies that have that genre randomly GET/movies/movieid
- Ability to see all movie reviews. GET/movies/movieid

- Ability to add a basic review which is just a number from 0-5. Basic Review is only for logged in users, accessed through a button on the bottom of the page. GET/addReview to render review page. POST/addReview to add basic review
- The ability to add a detailed review that is also accessed by the same button at the end of the movie page requires a score out of 5 and a text review. POST/addDetailedReview to add basic review
- When adding a review, the average rating of the movie gets updated as well
- **Viewing People** - Ability to see what movies the person, directed, wrote or acted in. Each movie is a clickable link that redirects right to each movie's page. GET/person/id
- Ability to search for people by name POST/getPeople
- Ability to follow and unfollow the person by clicking a button, user have to be logged in to follow a person if a user following the person, the user will get a notification at the bottom of the user's page when a person is added to a movie or a new movie is added with that person. POST/followPerson POST/unfollowPerson
- **Viewing Other Users** - Ability to view user's reviews, and view all the people and users that this user is following, they're accessible as a link. GET/users/userid
- If you a logged-in user follow this user, and this user makes a review the user that follows will receive a notification when that user adds a review, will be explained how to test it in section 8. GET/users/userid
- Ability to follow and unfollow user on user page POST/followUser
- **Contributing Users** - Ability to add a person, checks if that person already exists. POST/addPerson
- Ability to add a movie- must have at least the title, genre, at least one director, at least one writer and at least one actor. Also can add a plot, runtime, country, etc. POST/addMovie

- Ability to edit a movie, when clicking Edit movie on the movie page you can add a new actor/director/writer, checks if that person exists if not then add a new one, Extra implementation will be explained in section 3. POST/editMovie
- All Authrozitaion has been implemented too, a lot of the functions check for authorization, some functions only work when a user is signed or a contributor. Also, a lot of pug rendering changes if signed in or a contributor or something doesn't exist. (For example, if the movie has no reviews the movie page will not show the Reviews header and such, there's much more to this.

3. **Extensions**

- More implementations
    - Ability to get the password if the user forgot.
    - Ability to create a new person if the user forgot the password.
    - Removing a movie is possible, on the add/remove page. This also updates everyone's page accordingly.
    - When editing a movie, you can edit the movies' name, plot, country, runtime and year in addition to adding an actor/writer/director. Also comes in a prefilled form which I thought was a nice implementation
    - Also when editing a movie and adding a new actor/writer/director if that person doesn't exist then a new person is created.
    - Log out also implemented but not sure if this was required or not.
    - When looking for a movie in the URL you can just write /movies/movietitle and it will redirect you to the movie page. I don't think this was required. It works for both id and movietitle.
    - All these functions were implemented as I thought they are necessary for a movie database website to have and make the site more complete.

4. **Design**

My web application works very fast when I tested on my computer thoroughly, I think scalability wise is very good. The flaw is that I have everything in one server file, instead of

separating the client-side code. I used forms for all of my inputs so that I couldn't do any AJAX operations with input forms, maybe there is a way, but it hasn't been taught in class. I implemented correct RESTFUL API, with all GET and POST operations do what they need to be doing, when creating or modifying something I used POST operations, when getting information to render I used GET operations. Also, I used correct HTML response codes, when adding something successful I used 201 code, when a POST operation modified I used 200 code when an unauthorized user tries to access something I threw 401 codes, when something doesn't exist 400 code, redirecting 304 codes, and 404 codes aswell. I think my design is not the best but it not bad because the web application works pretty fast, I wish I had time to implement a database, so it would work even quicker, and I could add more ASYNC functions. When iterating over big objects like movies object which had 9125 movies and people object which had over 27000 people, I used while loops instead of for loops which improved my server speed by over 1-2 minutes on some operations, one operation went from 2 minutes to less than 2 seconds. Without pagination, I had to splice movie array showing to only 125 first movies, and 250 users. If a URL doesn't exist the website will throw a 404 error thought it was much better than getting random errors.

5. **Improvements**

I would definitely improve the design, using more ASYNC functions and a database. I figured out that my code should be separated into the client and server-side a little bit too late, I really wish the TA's commented on it from previous check-ins but I never got that suggestion, also looking through class code examples I couldn't find any efficient way to separate the code into server and client. Pagination is another that is missing from this assignment, without pagination you can't see all movies in the database unfortunately it would make everything very slow if the page needs to load 9000 elements or 27000 users, therefore every page was spliced to the first 125 elements and 250 users

6. **Modules, Frameworks**

- Bootstrap
    - Implemented a navbar that works on every page, every link is clickable and it changes depending on if the user is logged in or a user is a contributor. When logged in check out the changes and when contributing.
    - A lot of the buttons use bootstrap buttons, and some of the text forms use bootstrap buttons aswell.
    - Bootstrap made my site visually better in my opinion but I didn't like it as much as I felt like it was harder to style everything with it much harder. I used it because I thought some of it looks more visually catching.
- Url module
    - To store the current URL you are currently in. This helped me with an easier way to access data I needed for some functions like adding reviews, getting similar movies, editing movies, adding people, following and unfollowing people and more.
    - I think it improved my design but accessing data easier.
- Sessions
    - For authorizations and authentication reason
    - Much easier to work with the requirement for a certain user, for example, if a user is a contributor or not, or if a user is logged in

7. **What do you like most about your project? What would you say is the best feature(s)?**

I like the project overall and I'm actually impressed with how much implementation we had to put in for this project and I made it all work so I'm proud of the work and glad this project is a thing. I like the visual interface part of the project, I think the website looks pretty appealing since I've never used HTML, PUG and CSS before this project, so making look

how it is right now is pretty impressive to me. I like the number of features I had implemented. The best features of the project in my opinion is logging in/out/creating an account/forget password, editing/creating and removing movies. Similar movies algorithm was also not bad, another good feature I think is all the authorization done, what shows and what not shows depending if you are not logged in or if the user is a contributor.

8. **Testing**

First, you can try accessing stuff without being logged in

There are only 4 users currently. Log in to username: willy2 password: imthebest

This user is a contributor, you can click on any links to the navbar to test whatever you would like. To follow/unfollow any user go to the user's link, then click on one of the users and you can follow/unfollow there. Same procedure with people. To search for a movie go to the movies link on the navbar. The initial movies will only be the first 125 as If I fit more it would become very slow. Same for search results they cap at 125, without pagination this would be super slow. Adding a person is a link on the navbar, adding/removing a movie is also a link on the navbar. Adding a review, go to movies and then select a movie. Then you can start adding either a basic review or a detailed review.

To test notifications - log in to username: animallover pw: dogsandcats, create a review, then log out, then login back into willy2, and on willy2's user page you should be able to see a review, there also should be a console log.

Creating an account and Forgot Password you can do this through the login page there are 2 links below the forum, to create an account or forget a password.

Authorization can be tested by writing /addremoveMovie or /addPerson and more when not logged in.

Searching works with all queries either one by one or all together. For more than one the only way to test is through the URL. for min rating there are only 2 movies with reviews stored, Toy Story and Kingsglaive: Final Fantasy XV 2016, you can test by adding a review, the average rating will update, then new movies will show up with migrating query.