Jersey Aubin-Déry 101079607
Malcolm Smith 101143404
(Project Submission Done by ***Jersey*** *on cuLearn)*

Final Project Report - Movie Database

OpenStack Information:

```
IP:                 134.117.130.117
Port:               3000
Username:           student
Password:           JerMal2406
```

INSTALL AND RUN INSTRUCTIONS

```
cd --ProjectFolderPath
npm install
node server.js
```

If all else fails, nuke and restart

```
cd --ProjectFolderPath/..
./reset
```

Web Page can be viewed with Google Chrome

-Functionality-

Implemented

- *Ability to reset the database to a default state with reset.js*
  - *Execute reset.js using Node. This will delete the current data and reset back to the default one.*
- *User Accounts*
  - *Create new User*
    - *New created users are not contributor by default*
    - *Checks if username is unique*
  - *Logging in / logging out*

        - *Able to view user profiles*
        - *Contributor / non contributor*
            - *Ability to switch on user page, if signed in*
        - *Manage Follows / Following*
        - *Recommended Movies*
            - *Shown on user profile page*
            - *Based off of followed movies, genres, and users*
    - *Searching Movies*
        - *On home page, ability to search for a movie or actors*
        - *Will bring user to the movie page which most closely resembles the title they entered*
    - *Viewing Movies*
        - *Shows information on movie*
            - *Title*
            - *Release year*
            - *Average rating*
                - *Calculated from reviews attached to movie from users*
            - *Runtime*
            - *Plot*
            - *Genre*
        - *People who worked on the movie*
            - *Includes links to their own pages*
        - *Recommended Movies*
        - *See reviews added to this movie*
            - *Reviews demonstrate*
                - *Score out of 10*
                - *Summary*
                - *Full review text*
    - *Viewing People (directors, writers, actors)*
        - *History of their work,(links to their movies)*
        - *Frequent collaborators*
        - *Follow person button if signed in*
    - *Viewing user profiles*
        - *Shows movies recommended to this person*
        - *Reviews made by this person*
        - *List of followed people*
            - *Linked to each person's page*
        - *Follow user button if signed in*
    - *Contributing Users*
        - *If signed in AND is a contributing user, a create movie button will show on*

the top bar

- *Create movie page*
  - *Adds movie to database*
- ***Ability to edit movie when on the movie page***

Not Implemented

- *Ability to edit movie when on the movie page is not implemented, however the API request to do so exists*

-API-

GET

/movie/:movieID

Returns movie object

/person/:person

Returns person page

/userprofile

Returns userprofile if signed in, otherwise 404

/userprofile/:user

Returns userprofile page of username

/users/:username

Returns user object of matching username

/allUsers

Returns all user objects

/searchUser/:query

Returns array of user objects matching query

/isContributor/:username

Returns value of the contributor property of a user object (true or false)

/getMovie/:movieID

Returns a movie objects via imdbID

/allMovies

Returns all movie objects

/searchMovie/:query

Returns list of movies matching query by title

/searchMovieByDirector/:query

Returns list of movies matching query by director

/searchMovieByWriter/:query

Returns list of movies matching query by writer

/searchMovieByActor/:query

Returns list of movies matching query by actor

/searchMovieByGenre/:query

Returns list of movies matching query by genre

/getSimilar/:title

Return list of movies similar to another, based off of their matching genre(s)

/getRecommended/:username

Return list of movies based off of who the user follows, such as other users and the reviews they made over 5/10, and actors / people they follow and the movies they have been in. The list generated ignores movies the user has “seen” (reviewed) and does not present them in the list

/people/:name

Returns a person object via their name

/allPeople

Returns list of all people objects

/searchPerson/:query

Returns list of all people objects whose names match the query

POST

/login

/createUser

/updateUserFollowage

Increment the followage of a user

/updatePersonFollowage

Increment the followage of a person

/updateContributor

Toggle contributor property of user to true or false

/createMovie

/createReview

/createPerson

-Extensions-

- *Persistent Data functionality via Database*
    - *Database created using a SQLite wrapper for Node, called enmap*
    - *Enmap is short for enhanced map, the map being the basic map structure from JavaScript*
    - *This allows for a database to be created and modified by using calls very similar to the collections/map module included with vanilla JavaScript*

```
const Enmap = require('enmap');
movieDatabase= new Enmap({name: "movies"});
// each new enmap is a new table, stored in /data/enmap.sqlite

let key = "Star Wars";
let value = "10/10 would watch again";

movieDatabase.set(key, value); // this is persistent
movieDatabase.fetch(key); // returns value stored beforehand

// other functions include editing the key value pairs
// in the table or removing them when needed
```

    - *Each of our data types (movie, user, person, reviews) are being stored in separate SQLite tables, and the data is persistent between server restarts*

- *Unit Testing (NOTE: CAN ONLY BE RUN WHEN SERVER IS RUNNING)*
    - *There is a test.js file that can be run with node test.js*
    - *This program will run all of the GET requests available in our API and compare their responses to a set of correct responses*
    - *Will log any API requests that FAIL and will keep track of the number of fails*

-Design Decisions-

- *Template Engine was used to make front end a lot easier*
- *Our project was organized into different categories of files which makes the entire project more modular and easier to manage*
- *We have a toolbar at the top of the page which is present on every page of the*

website, and changes dynamically depending on things such as: the user is/is not signed in, is a contributor, etc.

- *The images / posters used on our pages are imported from the actual imdb, which means that local storage for our site is a lot smaller since we do not have to store thousands of images locally*

-Future Improvements-

- *A frontend library, such as react or angular, would make the user experience a lot more robust and visually appealing*
- *Javascript animations for elements on the webpage*
- *The ability to add custom profile pictures / avatars to a userprofile*
- *Hashed passwords to ensure that they are not stored in plaintext*
- *Embedded youtube videos to show movie trailers*
- *The ability to edit movies from the movie page, as mentioned above*