Movie-Database Project Report

Name: Minh Thong Mai
ID: 101128263

Project: Movie-Database

Open stack instance information:

username: tommai
password: +Nhatrang1403
floating ID address: 134.117.132.159

Instruction how to initialize and run server on the Open Stack instance

1. Login to the open stack account with username and password above
2. Navigate to the instances name: tommai
3. Click console then click “click here to show only console”
4. On the virtual machine, open terminal
5. Navigate to Public -> Final
6. The database has been generated in the VM, so the TA do not need to generate it again
7. If the TA do want to generate the database, enter Node store-generator. Notice that the process would cost a lot of time (over 20 minutes) because of the heavy load of the data.
8. Enter: Node server.js
9. On local machine, open terminal
10. Enter: ssh -L 9999:localhost:3000 student@134.117.132.159
11. If the password is required, enter: student
12. Open chrome and run URl : http://localhost:9999

Project Assumptions:

This project is designed to not dealing with case insensitive when adding a new person and adding a new movie, so any movie’s title, or person name is written in lowercase letters with a capital letter at the beginning of each word.
For example: **T**om **M**ai

Functionalities that have been implemented in the application

1. Sign up
2. Sign in
3. Search for movies based on title, genre, or year
4. Register/cancel contributing user status
5. View recommended movies based on the movies that people have viewed
6. Viewing a people profile
7. Follow people
8. Viewing another user profile

9. Follow another user
10. Send notification to a user if the current user follows/unfollows this user.
11. Viewing a movie
12. Add a director/writer/actor to a movie if the user is a contributing user
13. Send notification to followers of a person if this person is added to a new movie
14. Add review for a movie
15. Add a new person to the database if the user is a contributing user
16. Add a new movie to the database if the user is a contributing user
17. Log out

Instruction to test the application:

1. Sign up two users to test functionalities in the application
   Username: Tom password: 1234
   Username: Jack password: 1234
2. Test login with username: Tom password: 1234
3. To test Search for a movie, enter a name/year/genre such as Toy Story/1995/Animation in the search bar and click related options that user wants to search then click search button. Note that the words are typed with case insensitive.
   For example: Toy Story ->Title->Search
4. Click to a particular movie to view movie's information
5. To test add review for a movie, navigate to a movie displayed in the movie's page. Adding a rating score and comment then click add reviews
6. To test viewing a people profile, click a specific people in the movie's page or list of followed people in user account page.
7. To test Follow/Unfollow people, click Follow/Unfollow button while viewing a people profile
8. To test viewing another user profile, click a specific user review box in the movie's page or list of followed users in user account page.
9. To test Follow/Unfollow user, click Follow/Unfollow button while viewing a user profile
10. To test Register/Cancel contributing user status, click upgrade/cancel contributing user
11. To test Add a director/writer/actor to a movie if the user is a contributing user, click a movie (ex: Toy Story). Typing name of director/writer/actor (note that the name has to be lowercase with a capital letter at the beginning of each word. Ex: **P**ete **D**octer). Also, the name has to be existed in the database before). Then click add director/writer/actor.
12. To test Add a people to the database if the user is a contributing user, click add people, then press name of people, position directors/actors/writers, add their bio, then click add people. Note that the people could not appeared in the databased before.
13. To test Add a movie to the database if the user is a contributing user, click add movie, the press information related to the new movie and press add movie. Note the title of the movie must be not existed in the databased. Also, name of directors, actors and writers must be appeared in the databased.
14. To test notification system, there are 2 kinds of notifications that has been designed. First, the notifications that send to another user when the current user click follow.

Second, the notifications that send to all followers of a person when this person is added to a new movie.

To test the first kind of notification:
Sign in a user account (such as username: Tom pass:1234) and do some features such as add a review for a movie such as Toy Story, then log out. Sign in another user account (such as username: Jack pass:1234), then navigate to movie Toy Story. Scroll down to reviews list then click to the user namely Tom. After navigating to Tom's profile page, click follow, then sign out. Sign in Tom account again, and there is a new notification informing Jack has followed Tom. Click to the notification and follow Jack if Tom wants.

To test the second kind of notification:
Sign in a user account (such as username: Tom pass:1234) and do some features such as follow a people namely John Lasseter, then log out. Sign in another user account (such as username: Jack pass:1234), then upgrade to contributing user. Do adding a new movie similar like instruction above with a people namely John Lasseter in any fields of director, actor, or writer, then click add movie. Sign out user Jack, then sign in user Tom again. There is a notification in Tom user account page's navbar informing John Lasseter has been added to a new movie.

15. To test log out, click log out button on the nav bar

Summary of design decisions that I feel have increased the quality of the system

1. Applying Restful API constrains when design the system
   a) The server is stateless
   b) Using identification of resource to navigate to a specific user, people or movie
   c) Manipulation of resources through representations (client and server interact via JSON)
   d) Self-descriptive messages (GET, POST)
   e) Using Hypermedia as the Engine of Application State (HATEOAS) to navigate through the application
   f) Using nouns to represent the resources
   g) Using plurals in appropriate cases (users/movies/directors/actors/writers)
   h) Using appropriate response codes (201- OK /401-Unauthorized)

2. Improvements to the system that I think could be made to increase the overall quality of the system:
   a) Using Socket Io to handle notification system to scale it a lot more. Currently, the system is build using a list of notifications in the database, which user must loop through the entire array to display unread message. Using Socket IO to send data in real time would reduce the amount of time and memory compared to the current approach.
   b) Using setInterval() to reload the page after a particular amount of time without sending request and receiving respond from the server. It would help boost the speed of the application.

3. Modules and frameworks that I used in the app:
    a) Express framework help organize the application into a MVC architecture on the server side.
    b) Pug module to handle user interface with simplified syntax compared to HTML, which make the code more readable
    c) Express-session module to store the current user state
    d) Mongoose library to provide a backend database management for the application. It provides a straight-forward, schema-based solution to model the application data.

Describe extension that include in the application

a) Using Mongo dB database to store the system's data. Also using Mongoose to modelized the data in the application.

What I like most about my project:

The project provides me an opportunity to apply theoretical knowledge into a real product using almost all of basic knowledge in the field of web development. I learnt how to handle both front end and backend, so It has broadened my view about the field of web development. The most challenged problem that I have faced is connecting the client side, server side and database together, and using express framework is a great way to deal with that problem. Express gives me a handy tool to organize my application into a model view control architecture. Also, one of the best features in my application is the notification system, which can notify the user about the activity of users, people that this user has followed. The feature teaches me how to build a multi-directional communication system between users in an application.