CSE 4234/5234 – Web Applications

Fall 2024

Final Project: MONGODB | EXPRESS | REACT | NODE

Total Points: 30

Date Assigned: Friday, Nov 15, 2024 **Due Date:** Monday, Dec 2, 2024

Submission Instructions: Please package your node application folder and your react app and submit your work on Canvas as a zipped folder named `cse4234_your_group#_final_project.zip`. Make sure to include the `package.json` and `package-lock.json` files. Do not include any binaries (e.g. node_modules) or unnecessary files. Note that your backend must serve the final build of your React app. This means that it should be sufficient to run only the backend node app to demo the app. However, your react source code must also be included in your folder.

Key Concepts Demonstrated

- Writing full-stack code to facilitate storing movie data in a MongoDB and displaying the data in a React app.
- Writing code in Node.js, Express.js and React.js to interface between a MongoDB and a UI
- Writing code to use sessions/cookies and authentication to manage user interactions in a web app.

© All rights reserved

1. (25 points) Your task for your final project is to create a full-stack (MERN) app that allows a user to register and sign in using a web form and interact with movie data stored in a MongoDB database named *{movies}-{group}-{group#}*. Your full-stack app should satisfy the following requirements:
    - First, download the `movies.json` file from Canvas.
    - Create a Mongoose Schema and model based on the JSON file.
    - Create a Node app to facilitate the addition of movies to a MongoDB database based on your Mongoose Schema. Upon first launch, your node App should create the MongoDB and write all the data to the database using the schema. Do not store your data in remote database.
    - Create a React app that features a Register button and a Sign-In button. Your app should also employ the use of cards to categorize and display the movies in the database by genre. For demonstration purposes, show only the top 10 movies in each genre sorted by rating. Note that one movie may appear in multiple genres. Use a separate view (UserAccount component) to show the movies a user likes/favorites. Your React app should have as a minimum the following components:
        - App
        - Navigation
        - Movies
        - Register
        - SignIn
        - UserAccount
        - User (A component built upon React's `useContext()` Context Provider to store the user's information such as the user's name , which should be inherited by child components)
    - When the *Register* button is clicked, this should present a form that allows the user to register. Only registered users can like/unlike movies. Liking a movie should update the database record for the registered user. Similarly, clicking the Sign-In button should allow the user to sign-in and like movies.
    - Make sure to use HTTP cookies/sessions to handle the user sign in. It is also pertinent to show a cookie consent banner.
2. (5 points) Code should be written and indented properly with code comments where necessary as demonstrated in class. Best practices discussed in class followed accordingly. Make sure to use function-based components with Hooks to accomplish the various tasks.

 © All rights reserved

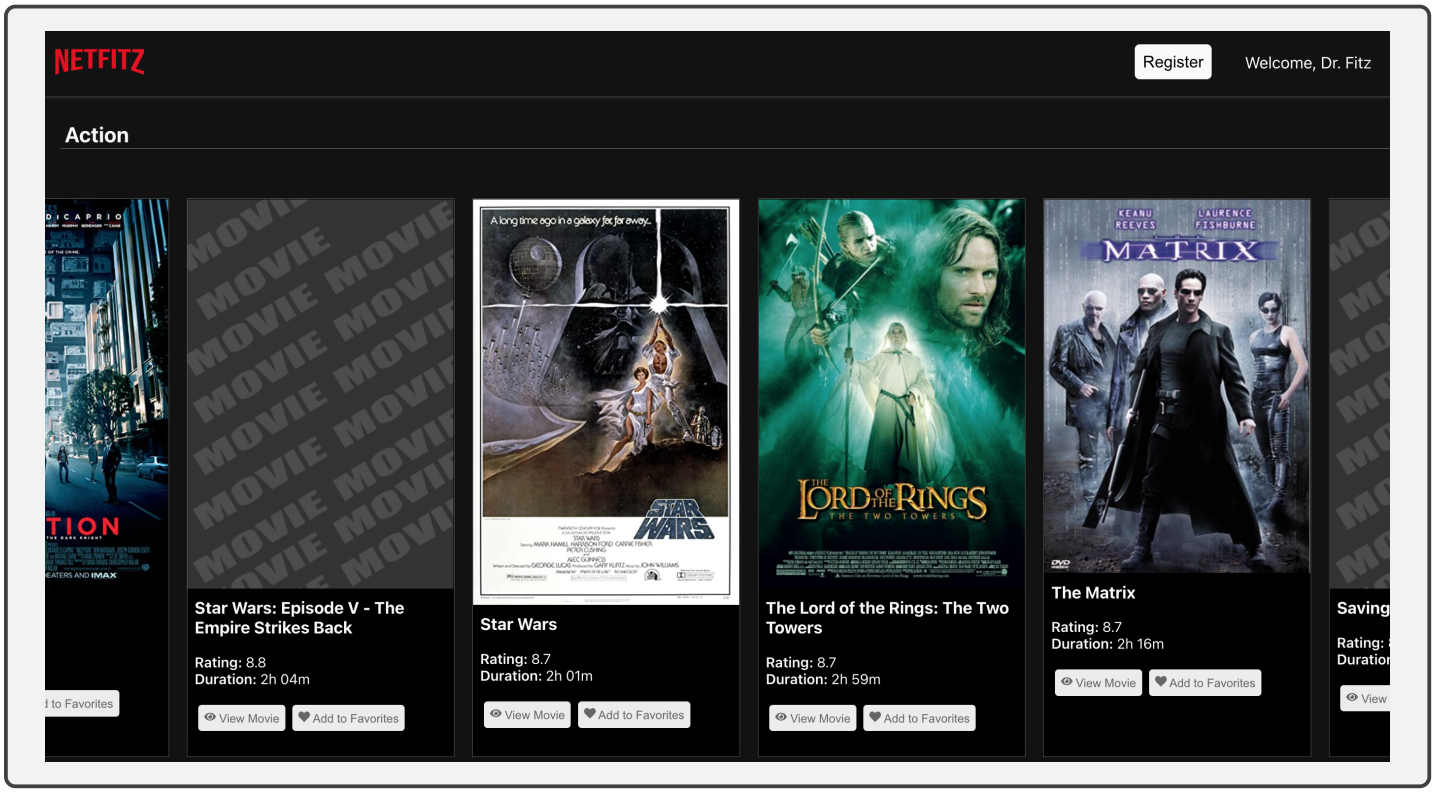

Figure 1: Example Movies App showing movies as cards, user logo, and register button

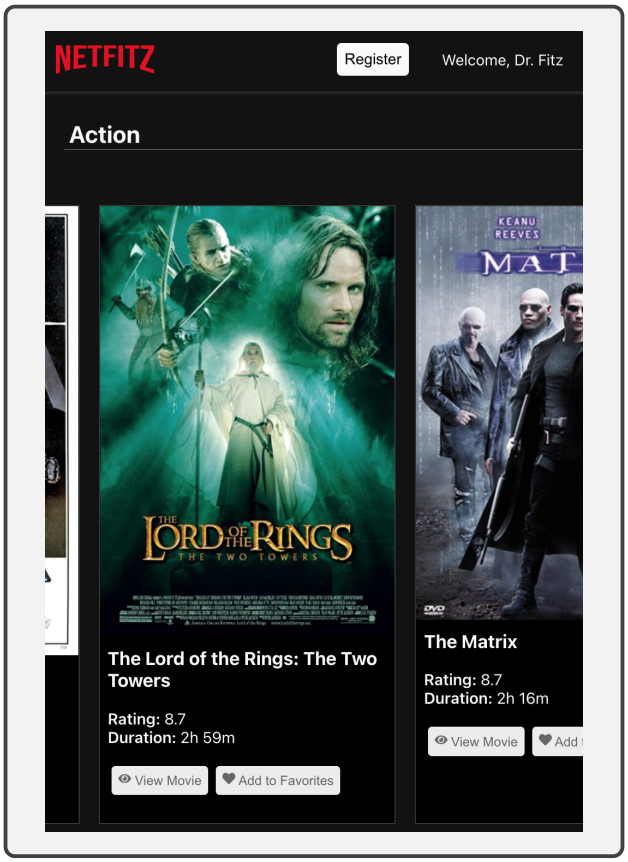

Figure 2: Example Movies App showing movies in mobile view

© All rights reserved

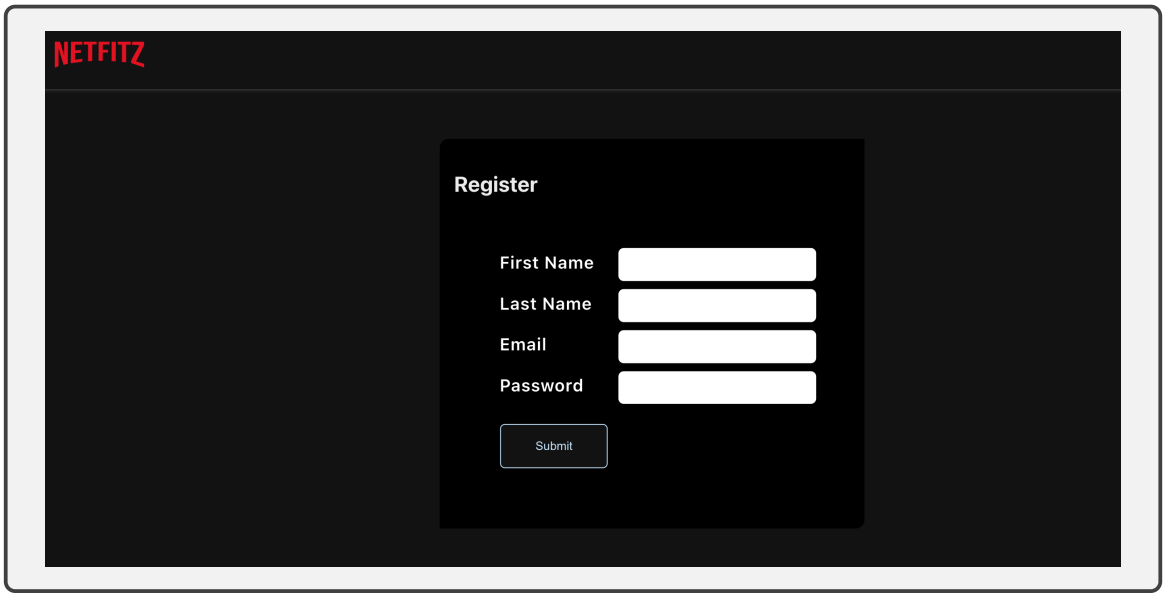

Figure 3: Example Movies App showing Registration Form

 © All rights reserved

Suggested Division of Labor

Example 5-Person Division of Labor

Member 1: HTML and CSS with responsive design, App and Navigation Components and EC2 deployment and testing

Member 2: Node/Express App that parses the JSON file and uses a Mongoose schema to write movies to the database and manage routes for end-points

Member 3: Register and SignIn Components that feature forms that allow the user to register and sign-in by passing data to the backend using the JavaScript fetch API

Member 4: Movies Component that displays movies as cards in the React frontend and the User Component that interfaces with the UseContext hook as well as the cookie banner.

Member 5: UserAccount Component that displays liked/favorited movies as cards as well as the code that uses the Mongoose Schema to retrieve the necessary data from the database

Example 4-Person Division of Labor

Member 1: HTML and CSS with responsive design, App and Navigation Components and the UserAccount Component that displays liked/favorited movies as cards as well as the code that uses the Mongoose Schema to retrieve the necessary data from the database

Member 2: Node/Express App that parses the JSON file and uses a Mongoose schema to write movies to the database and manage routes for end-points, EC2 deployment and testing

Member 3: Register and SignIn Components that feature forms that allow the user to register and sign-in by passing data to the backend using the JavaScript fetch API

Member 4: Movies Component that displays movies as cards in the React frontend and the User Component that interfaces with the UseContext hook as well as the cookie banner.

© All rights reserved

Bonus - 10pts (Not for the faint of heart)

- Include a view/page that displays important details about each movie when the *View Movie* button is clicked. This information includes the release-date (MM/DD/YYYY), the plot, and a list of actors with small rounded profile photos. These pieces of info should feature appropriate icons such as a calendar icon for the release-date, etc. You may need to we-scrape the actors photos
- Allow the user to delete their account using a settings menu.
- Use the Nodemailer middleware backed by OAUTH security to send emails to the user when they register to confirm their email address.

© All rights reserved