# *Web Programming III (420-H30-HR)*

# *Assignment 2 – React and Express*

Date assigned: Friday, November 22, 2024

Date due: **Friday, December 13, 2024**

**Learning Objectives**

Upon successful completion of this assignment, the student will be able to:

* Work with Express.js
* Work with React

To do:

**General idea of the assignment:**

Use React to call express to read information from a file.

**Details**

1. Use npm to initialize an application called *yourinitials*H30A03, the *yourinitials* for me will be **aadewumi**. In the folder, use npm to add express and nodemon, making sure to update the package.json file with the dependencies as you do so.
2. Create a folder called **data** in the folder and move the movies.json file to this folder.
3. Create an express server running on port 8888. Use express.static to \_\_dirname + client/public as the Webroot. Note: This assumes you are using create-react-app template
4. Add a get endpoint called **/movies** which reads the JSON file and returns a sorted (by movie title) list of all the movies in the file (just the id, movie title and year it was released).
5. Add an endpoint called **/movies/[id]** which reads the JSON file and returns the movie for a specific [id]. All Movie information is returned. Hint: Use array.find() to get a specific record.
6. In the assignment folder, use create-react-app to create a react app in the folder named **client**. Refactor the client application to move things into specific folders for **Components** and **styles**. Also remove all files that are not needed (including those in **public**).
7. The App component will be the main component of your application which is called from your index file. This will be the source of all truth.
8. Add a List component. This component receives a list of movies (only title and year) and displays the list. This is the default component displayed when the App is initially started.
9. Add a Movie Component (which is a child of App) which receives the movie details from express and displays the detailed movie information for an individual Movie (Title, Year, Runtime, Revenue)
10. Add an Actor component which is a child of the Movie Component and displays the actors' names.
11. Add a Genre component which is a child of the Movie Component and displays the genre(s) of the movie.
12. When the application starts use List to display the sorted movies.
13. Provide functionality to select a movie from the list and display the details of the movie including a list of all the actors and genres using the Actor and Genre components respectively. This is NOT to be entered manually, but the movie is to be selected (think hyperlinks).
14. Back in express, add an endpoint called **/actors/[name]** that searches the data for all movies with that actor and returns a sorted list of the movie titles, year and id of the movies that actor was in. If there are no movies found, return an empty (json) array.
15. In the client provide a Component called ActorSelect functionality to allow the user to enter all or part of an actor's name and call the express server to retrieve the information. This function should be available on the main App screen
16. In the client, use the List Component to display these movies. Note, this implies that the same functionality of choosing a movie and seeing the details is provided using the Movie Component.
17. Back in express, add an endpoint called **/years/[year]** that searches the data for all movies released in the requested year and returns a sorted list of the movie titles and id of the movies for that year.
18. In the client provide a Component called YearSelect to allow the user to enter a year and call the express server to retrieve the information.
19. In the client, use the List Component to display these movies. Note, this implies that the same functionality of choosing a movie and seeing the details is provided. Note 2: You should not repeat the year on each line since it is always the same year.
20. In the server, add to the route **/movies** using POST. The route takes in a movie as a JSON object and adds that movie to the database. NOTE: It will first need to determine the maximum key value in the movies.json file and increase that by 1 before writing the new record. You will need to stringify the data before you write it. Stringify can format the data when it is written so that it is readable. The route returns a success or failure message depending on if the write worked or not.
21. In the client, create a component called MovieAdd which displays a form of all the fields of a movie. This form allows the user to enter new movie information. When it is submitted, use a fetch with POST to send the movie data to the server and check the response (remember fetch is a promise…). This function is available as an option on the main app screen.

NOTE: The actors and genres can be entered as a comma separately list but each MUST be sent to the server as an array as per the current data.

1. You MUST provide a nice interface at the start which, by default, displays the list of movies (List component). From this screen the user can:
   1. Select a movie (Movie component) to display the details.
   2. Search for an actor (ActorSelect component) to display the list of movies (List component) for an actor.
   3. Search for a year (YearSelect component) to display the list of movies (List Component) for that year (Movie component).
   4. Add a new movie (MovieAdd component).
2. Use CSS to format the display professionally using proper web techniques. Since there are 1000 movies in the database, this is key.

**The Final 5**

Doing everything about perfectly will get you a maximum 95%. To get the final 5% you need to do at least 2 of the following:

1. Add ability to search by part or all the Movie title.
2. Add another path called /genres/[genre] to return specific genre information and then add the appropriate component(s) on the client side to display the movies of a specific genre.
3. Provide the ability to sort the movies by other fields such as rating, or year released.
4. Add functionality to update the display automatically every 10 seconds.
5. Use Typescript

**To submit**

When you have completed the assignment zip the folder containing all the files for the assignment and copy it to the course page.