MyMovies-client project case study

Initial statements

Overview
MyMovies-client is a web application built using ReactJS, which gives registered users access to a collection of movies. Users can view details about movie genres and directors, save their favorite movies and update their profile information.

Purpose and context
I worked on MyMovies-client as part of the CareerFoundy Full-Stack Immersion Course. The app's name contains "-client" because it was build as the client-side for another application called myMovies, which is the server-side code storing the movies collection (REST API and database).
This project shows my ability to work with the React library, React Redux and Bootstrap for styling and responsiveness.

Objective
The aim was to add to my portfolio a project build with the React framework, which is widely popular and requested by employers. Full-stack development comprises skills in both the server-side and client-side of web applications. Building both sides from scratch and providing a great user experience are the goals of the overall MyMovies-client project.

Duration
3 months project

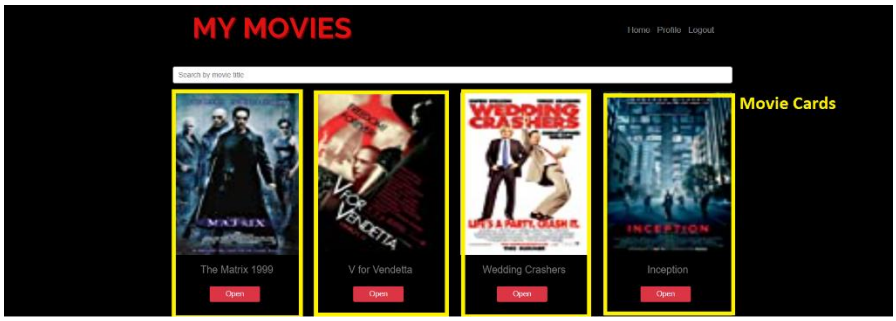
Credits
CareerFoundry Full-Stack Immersion Course

Tools, skills, methodologies
- ReactJS
- React Redux
- React Bootstrap

Building MyMovies-client

React Components & the Virtual DOM

As first step of the project, I got familiar with React's Components, syntactical elements that encapsulates the logic and styling for a piece of a UI. Components split a UI down into its individual parts, making it easier to build, maintain and scale. For example, the main view of MyMovies contains numerous Movie Card components, each of which displays a movie with an image, title, and button to show more information. Rather than writing a new card for each movie, I instead created a reusable Movie Card component.

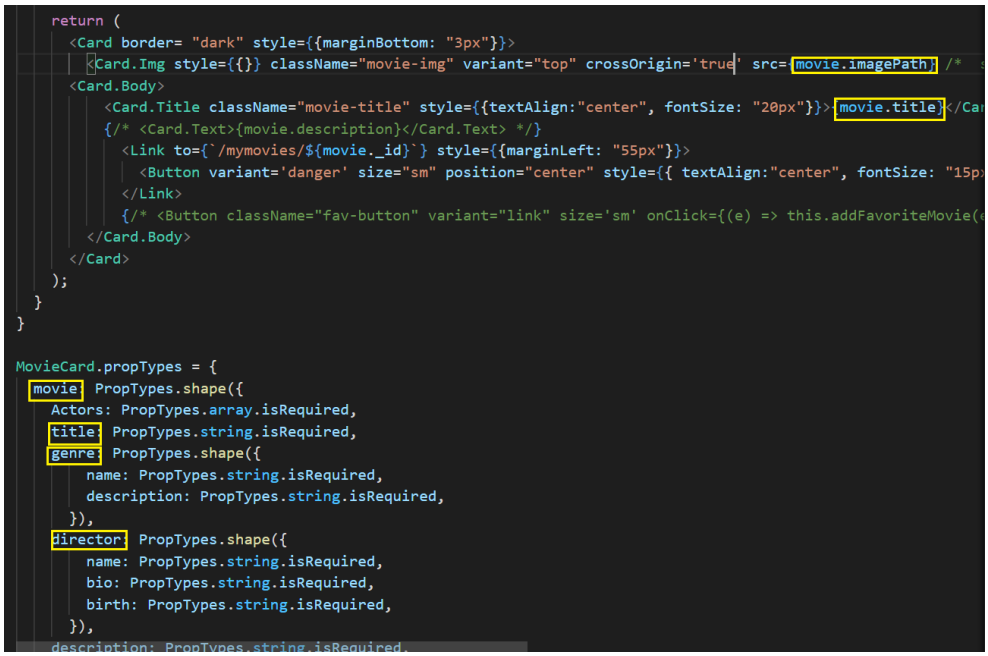


The same logic applies to all the other parts of the web app whose UI can be reused. A movie's description is shown in the Movie View Component, directors' information is stored in the Director View Component etc. I learnt to always follow the industry standard for development using React, which is to set up and organize the project folder structure by giving each Component its own folder, and storing all Components in the same folder. As components kept adding up with time, the folder became quite lengthy. To avoid confusion I made sure to give Components a clear name and function.

```
myFlix-client
    - package.json
    - src
      - index.scss
      - index.jsx
      - index.html
      - components
        - login-view
          - login-view.jsx
          - login-view.scss
        - main-view
          - main-view.jsx
          - main-view.scss
        - movie-card
          - movie-card.jsx
          - movie-card.scss
        - movie-view
          - movie-view.jsx
          - movie-view.scss
        - registration-view
          - registration-view.jsx
          - registration-view.scss
        - genre-view
          - genre-view.jsx
          - genre-view.scss
        - director-view
          - director-view.jsx
          - director-view.scss
        - profile-view
          - profile-view.jsx
          - profile-view.scss
```

Fetching movies with Axios

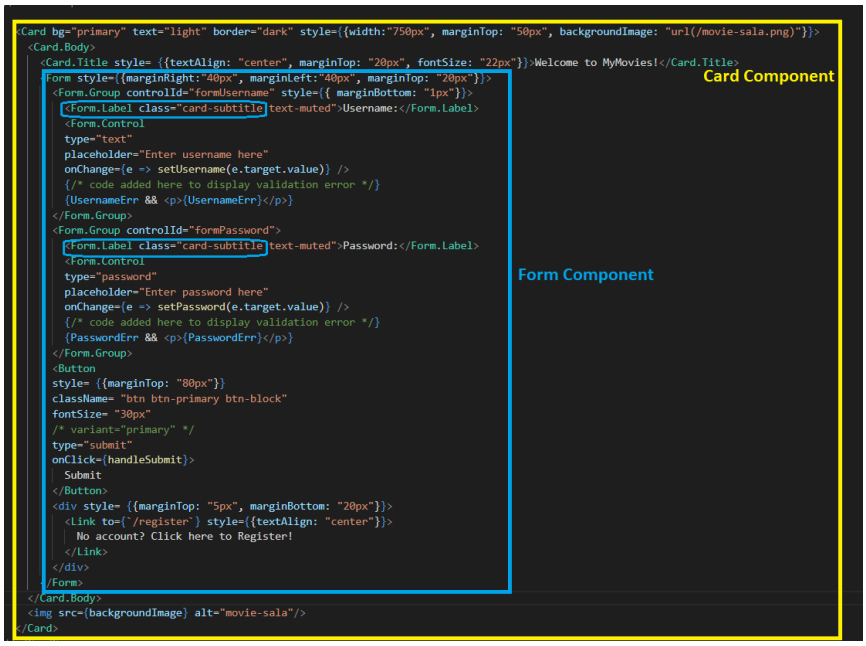
After creating its initial Components, MyMovies-client had to connect to the back-end movies collection in the MyMovies database. The JavaScript package Axios made this possible, by hooking up the MyMovies API. The challenge was to use the correct return statements and props matching the backend code. The errors I encountered were mostly typos and code whose function I did not recall. For example, in the Movie Card Components I wrote props with capital letters “Director.Name”, while the code in the Database was in lowercase “director.name” (the image below shows an example of the correct Movie Card View props). This taught me the importance of leaving descriptive comments and notes tracking the logic behind the code. It is part of developers’ job to make it as easy as possible for themselves and others to pick up a project that’s already been written.

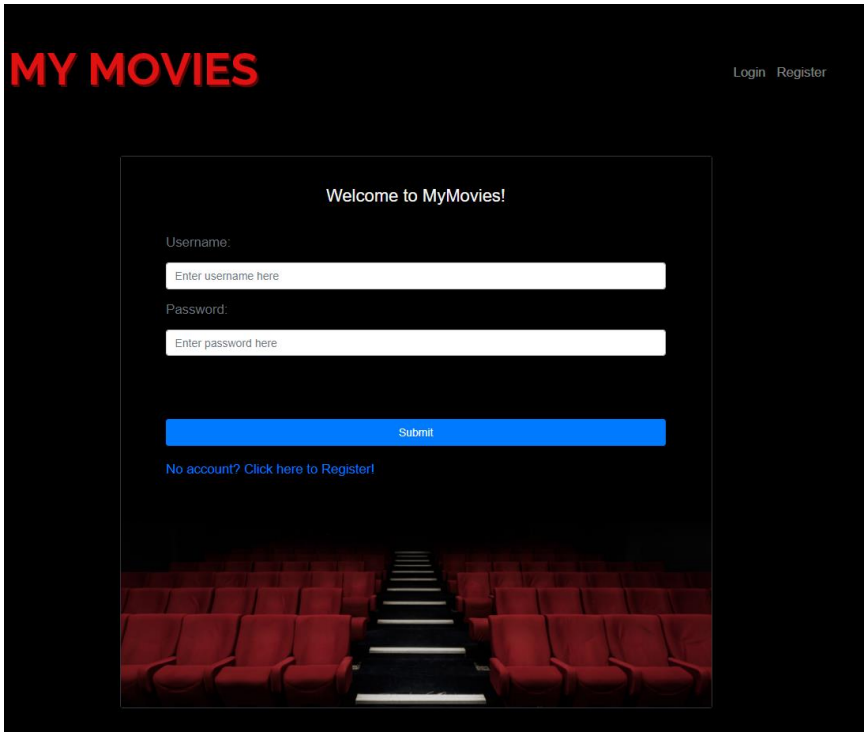


React Bootstrap

I used React Bootstrap Components, in particular Cards and Forms, to polish the application's appearance. Bootstrap Components are flexible and extensible content containers with options for title, subtitle, images, headers and footers, colors, display etc. I found them easy to learn and extremely useful. I also appreciated how they made setting the UI a smooth and efficient process.

The images below show the Login View component: its code is above and its appearance is below. I used a Card Bootstrap component as framework, because of its default margins, to wrap a Form Bootstrap Component, where the user is required to input username and password. The code highlighted in blue '<Form.Label class="card-subtitle"/>' is an example of a React Bootstrap class in a Form Component. The class "card-subtitle" displays the words "Username" and "Password" with default grey color, making the Card title stand out. The Form.Label also positions "Username" and "Password" automatically above the Form.Control Component.





Hosting MyMovies on Netlify

In order to make the web app accessible to everyone, I used the hosting service Netlify. This was a major challenge, because numerous attempts failed in the course of two weeks. The main error I could not figure out was “@parcel/core: Failed to resolve 'react' from './src/index.jsx'”. The solution included various steps, from deleting and re-installing mode_modules and package-lock.json, to deleting double importing statements of Components within other Components. This taught me the importance of periodically checking importing statements to spot mistakes in advance, especially when the app increases in size and Components tend to get out of hand. I also learnt to think in terms of environment, meaning that issues can lie in the app packages and not in what I coded myself. Upgrading packages and checking compatible versions periodically can save time and complications.

Final remarks

MyMovies-client was one of the most challenging and rewarding projects I worked on during the CareerFoundry Full-Stack Immersion Course. The complexity of connecting the back-end to the front-end required me to grasp the steps and tools behind setting up a database and fetching correctly the information it stores, defining clearly the big picture of the web app functionality. Afterwards, I was able to help other students encountering similar issues, and I was happy with my improved knowledge of API fetching and React.

The final project fulfills its original objectives. I implemented all the required React Components, fetched the movies correctly from the database and provided a good and smooth user experience through app styling and responsiveness.