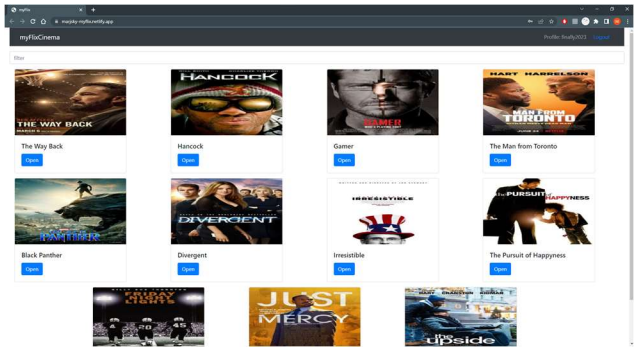
Case Study MyFlix full-stack

Overview



Web application for MyFlix built with open-source JavaScript stack to build a dynamic web site, similar to MERN stack, which includes MongoDB, Express, React, and Node. The application is a movie library where users can create an account and log in to access movies through the database. In addition, the application provides features such as selecting a single movie containing information about the director and genre, adding or removing favorite movies from the listing, and updating profile information or deregistering.

Purpose and Context

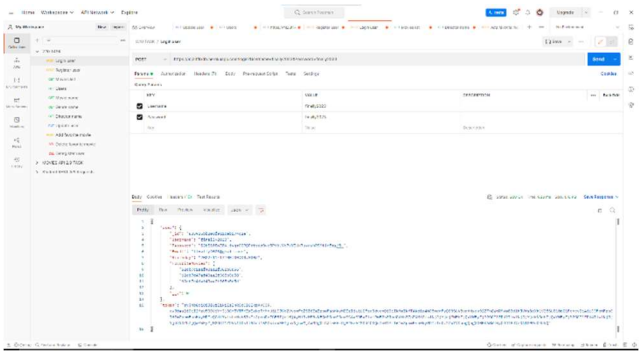
I developed MyFlix as part of CareerFoundry's full-stack immersion course. In my role as lead developer, my responsibility was to develop business logic for REST APIs respecting databases and building responsive React single-page applications.

Objective

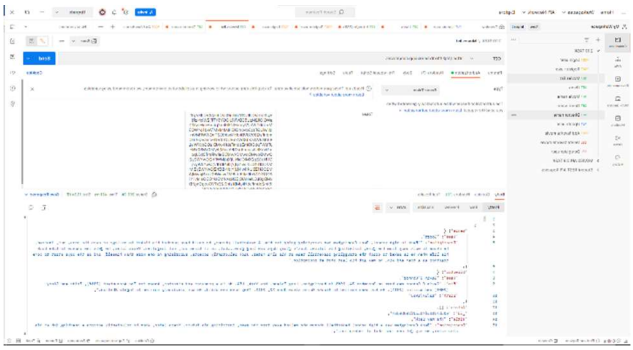
This project focused on building a merit full-stack application from scratch using the MERN stack to showcase my skills and open doors to potential employers.

Approach

Server-Side



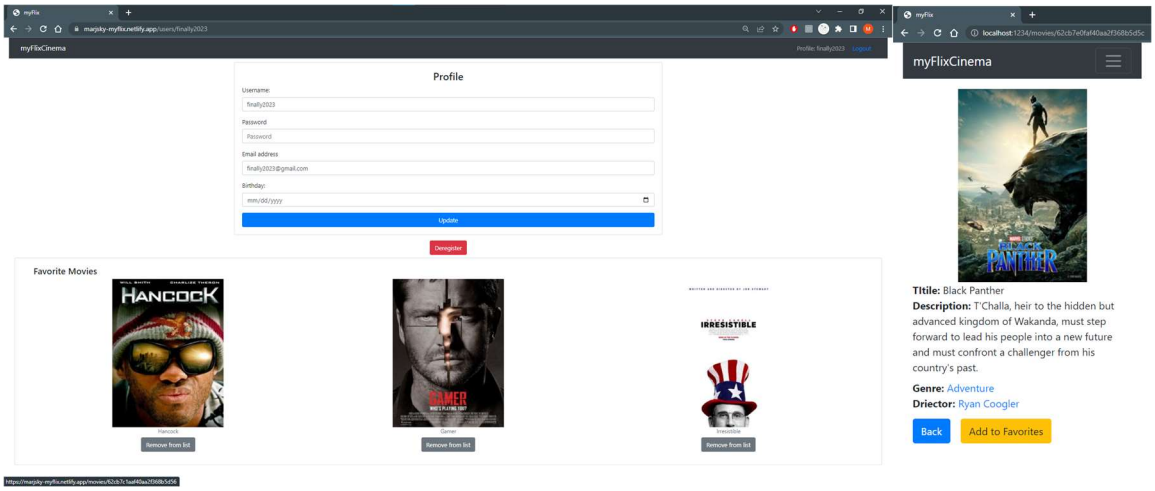
My first step was to develop a RESTful API using Node.js and the Express framework to interact with MongoDB, a non-relational database. Database construction involved establishing two collections: movies and users. Movies contain director, genre, and synopsis information; users contain a list of registered users.



By storing data and retrieving it from the database using GRUD methods, it can be accessed using HTTP methods GET, POST, PUT, and DELETE. I use Postman to test the API's functionality and validate its performance. Afterward, add user authentication and JWT authorization code to the HTTP form. As a final step, I deployed the API on Heroku and connected it to Mongo Atlas through Mongoose.

Client-side

After the API is fully functional, build user interface components using the React framework. These components include: login view component to access the user's account database, registration view component to add new accounts to the database, main view component to display all movies in the database, movie view component to display individual movies, genre view component to display information specific to genre, director view component to display information specific to director, profile view component to display personal information, add or remove favorite movies from the list, and update upon user request.



Challenges

For me, this project is the most exciting because it presented unprecedented challenges that required me to learn a lot of libraries in order to run the software smoothly. I discovered that my codebase was out of date while I was learning and building my app. Therefore, I became confused and realized I was using an out of date structure. My first step was to backtrack and use the previous architect codebase for components more carefully. Furthermore, Redux was undergoing updates and it was difficult to grasp the concept. Therefore, I did research and practice until I was confident and I was able to swap the upgraded version for the old Redux library version. Thus, I completed an enormous project, gained valuable knowledge, and gained confidence in my abilities.

Duration

I developed myFlix over the course of nine weeks for a CareerFoundry achievement. The majority of this period was spent on client-side development using the React framework because numerous libraries were ongoing upgrades and learning updated information from official documentation.

Credits

Role: Lead Developer
Tutor: Jay Quach
Mentor: Sebastien Filion