MYMOVIES CASE STUDY

OVERVIEW

Users of the web application myMovies have access to data on movies, directors, and genres. It was created using the MERN stack. A user's personal information can be updated, a list of favorite movies can be made, and an account can be created.

AT A GLANCE

Client-Side

BUILT A USER INTERFACE USING REACT AND REACT-REDUX. THE INTERFACE IS A SINGLE-PAGE, RESPONSIVE APPLICATION THAT OFFERS VARIOUS VIEWS, INCLUDING A MAIN VIEW FOR ALL MOVIES, A SINGLE MOVIE VIEW FOR MOVIE INFORMATION AND FAVORITES, A LOGIN AND REGISTRATION VIEW, AND A PROFILE VIEW FOR USER DATA AND FAVORITES.

Duration

THIS PROJECT TOOK TWICE AS LONG TO FINISH OTHER ACHIEVEMENTS THAN USUAL THIS WAS BECAUSE OF THE CLIENT-SIDE DEVELOPMENT. I NEEDED MORE TIME TO FULLY UNDERSTAND HOW REACT (AND REACT REDUX) FUNCTIONED. MORE IMPORTANTLY, I WANT TO DELIVER A NICE DESIGN TO MY PROJECT.

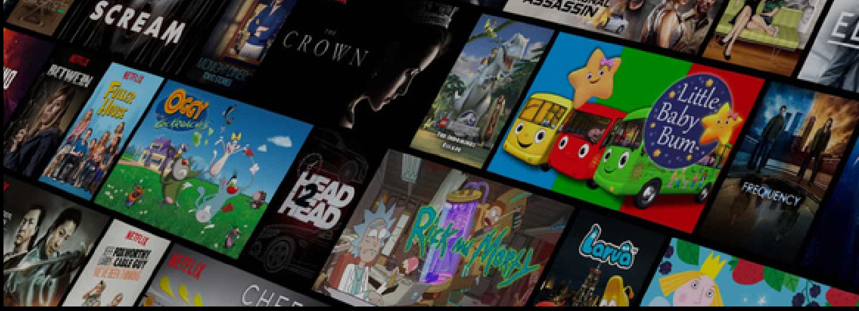
View Demo & Source Code

HTTPS://YOUTU.BE/9TOCTNYJGI8

HTTPS://GITHUB.COM/MEJIABRAYAN/MYMOVIE-CLIENT

Credit

Role: Lead Developer
Mentor: Jonatan Carabello
Tutor: Jason Early



OBJECTIVE

The main goals of this project were to create a RESTful API that would allow users to perform CRUD operations on JSON-formatted movie data, and to include user authentication and authorization using JWT and HTTP authentication

CHALLENGES

One of the challenges I faced during this project was getting familiar with the technologies used to build the API, such as Node.js, Express, and MongoDB. Another challenge was implementing the authentication and authorization mechanisms, as these required a deep understanding of security protocols and best practices. Additionally, testing the API using Postman posed its own challenges, as I had to ensure that all the functionality was working as expected.

OUTCOME

Using Node, I produced a RESTful API. js and Express that communicates with MongoDB, a non-relational database. Common HTTP techniques like GET and POST can be used to access the API. CRUD methods are used to access and store data in the database. JSON-formatted movie information is made available by the API.
I employed Postman to test the API. By using JWT authentication and fundamental HTTP authentication, I also added user authentication and authorization codes.