

myFlix Movie App: a case study in fullstack development

OVERVIEW

myFlix by Sarah is a personal project created as part of CareerFoundry's full stack web development program. It took two months to complete and was the first full stack project that I completed. The project uses MongoDB to create databases on the server side and React to create the client side of the app.

The Challenge: To create the server side (back-end) and client side (front-end) of a movie app that allows users to access details about movies and to save movies to a list of their favorites.

The Process: Based on user stories, this app uses MongoDB, Express, React, and Node.js (known as the MERN stack) to create a database of movies and users that users can access through an authentication and authorization process.

The Goal: A responsive web application that allows users to create a profile and log in. Once logged in, users can view a list of movies, search for a specific movie, view movie details, view director and genre details, add movies to their favorites, remove movies from their favorites, and edit or delete their profiles.

KEY DECISIONS

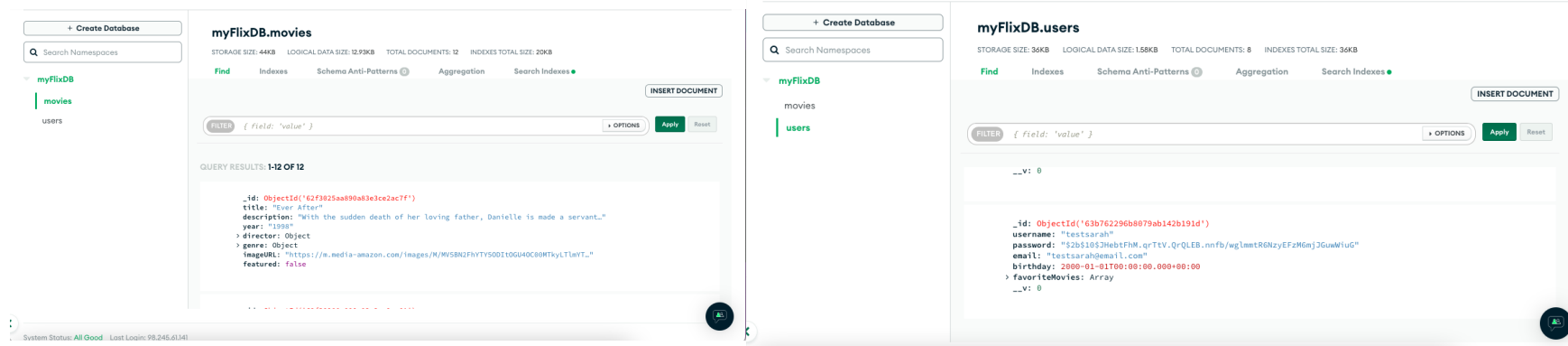
Tools

- MERN Stack
 - MongoDB: non-relational database to hold movie and user data
 - Express: framework that adds functionality to Node.js
 - React: popular JavaScript framework to create the front end
 - Node.js: runtime environment to support server-side programming
- REST API: how data is received, sent, created, and deleted using API endpoints
- Postman: allows for testing of API endpoints
- Heroku: hosts the API and the final live app



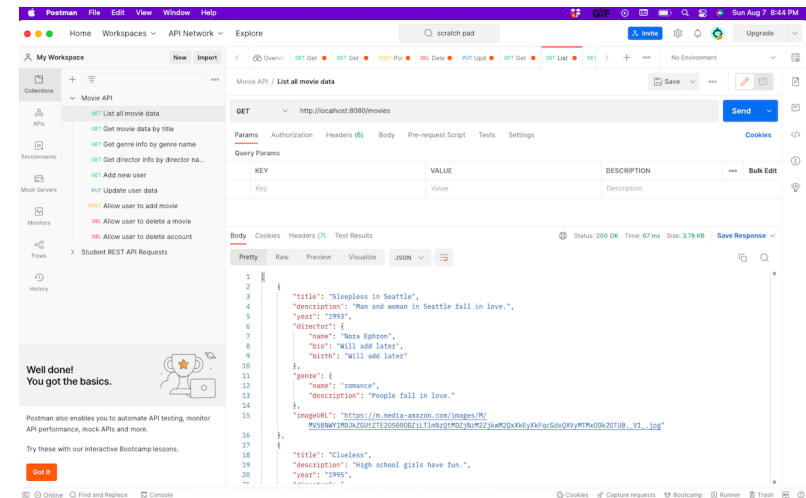
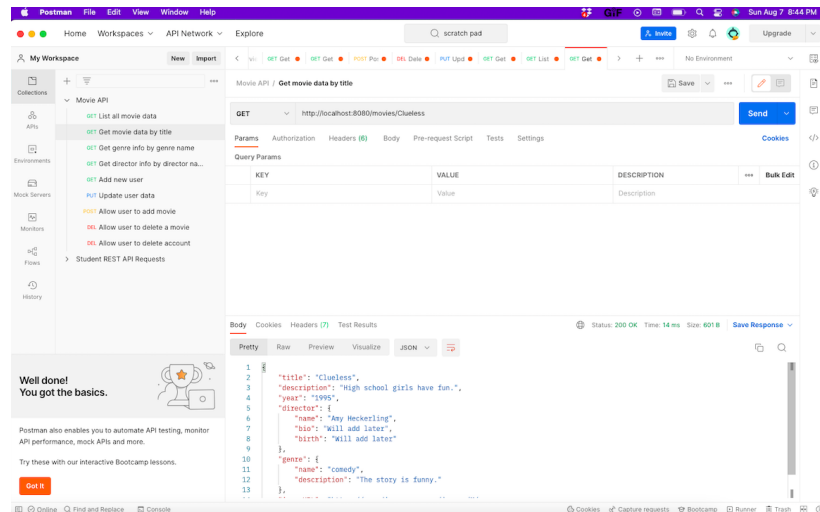
Creating and hosting the back end

This project requires two server side components: a movie database and a user database. The movie database contains all of the data for each movie featured on the website, and the user database contains data on users to allow them to access the key features of the website.



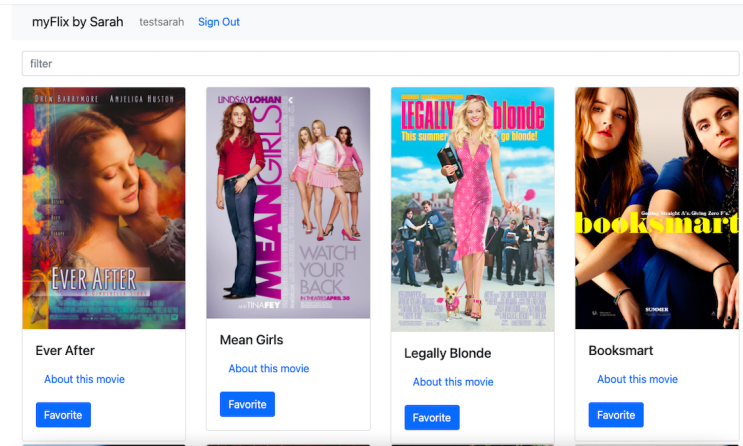
One critical element of the back end of this app is the REST API. The API endpoints, which users see as the URLs, determine the data that users access and how they can engage with that data. ([View the endpoints](#))

Because there was not yet a client side of the app at this point in the process, Postman was used to test that each of the API endpoints functioned correctly.



I used React to create the front end, which was required the following views to fulfill user stories:

- User log in
- New user registration
- Profile view (including list of favorite movies and options to edit and delete profile)
- List of movies
- Search bar
- Single movie details
- Single director details
- Single genre details
- User log out

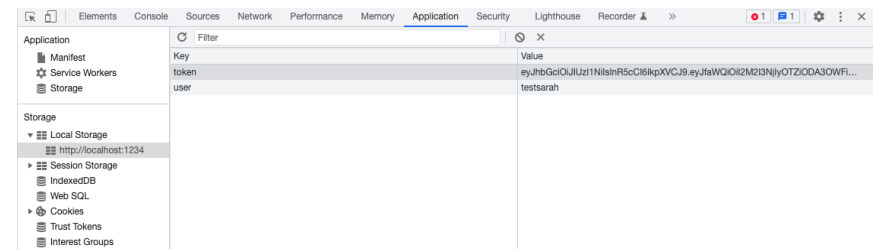


The user log in and registration views are open to any user, but each of the other views require user authentication and authorization, covered in the next section.

Authentication and authorization

This app requires both an authentication, confirming that a user exists, and an authorization, determining what a user can do, process. The overall process includes the following:

1. When users register, their user information, including username and an encoded password, are added to the user database.
2. When users log in, HTTP authentication is used, meaning that the username and password are encoded and sent with the request. The API checks that this user exists, then returns a JSON Web Token (JWT) token which is saved in users' local storage.
3. Any time users try to access parts of the app, the JWT token is sent with any requests to the API.



Once users are authenticated and authorized, they are able to view, add, or delete different content based on what the API endpoints allow.

FINAL REFLECTIONS

Challenges

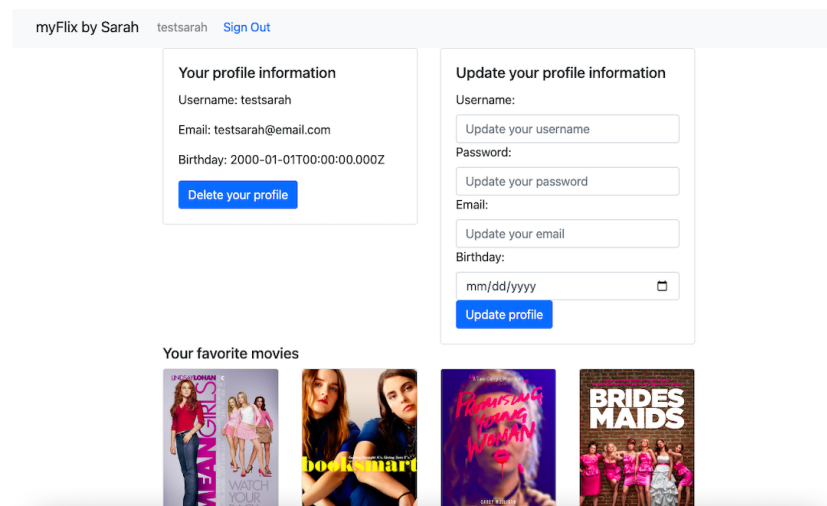
The primary challenge I faced with this project was the range of work required. Having to manage the server side, the client side, and the range of related tools for each was a unique challenge when compared to projects that focus on one side or the other.

An additional challenge was learning React and Redux while building the app. One of the key takeaways that I have implemented in other projects is the importance of taking initiative in answering the questions and addressing challenges that come up along the way. I could not only rely on CareerFoundry resources but needed to access React documentation and Stack Overflow in order to be successful.

Future Plans and Improvements

There are two areas that I will focus on in the future to improve this app and other similar projects. The first is around the UI of the app. While the app does what is required, there are tweaks I would make to the styling to make it more visually appealing.

The second adjustment would be around the options for users to edit and delete their information. I noticed that updating the user databases while a user is still logged in leads to some errors in rendering profile information, so that is an area that requires more time and attention.



The screenshot shows a web application interface for a user profile. At the top, there is a navigation bar with the text "myFlix by Sarah", the username "testsarah", and a "Sign Out" link. Below this, the page is divided into two main sections. The left section, titled "Your profile information", displays the user's details: Username: testsarah, Email: testsarah@email.com, and Birthday: 2000-01-01T00:00:00.000Z. A blue button labeled "Delete your profile" is at the bottom of this section. The right section, titled "Update your profile information", contains input fields for Username, Password, Email, and Birthday, each with an "Update" button. Below these sections, there is a section titled "Your favorite movies" which displays four movie posters: "Mean Girls", "Booksmart", "Promising Young Woman", and "Bridesmaids".