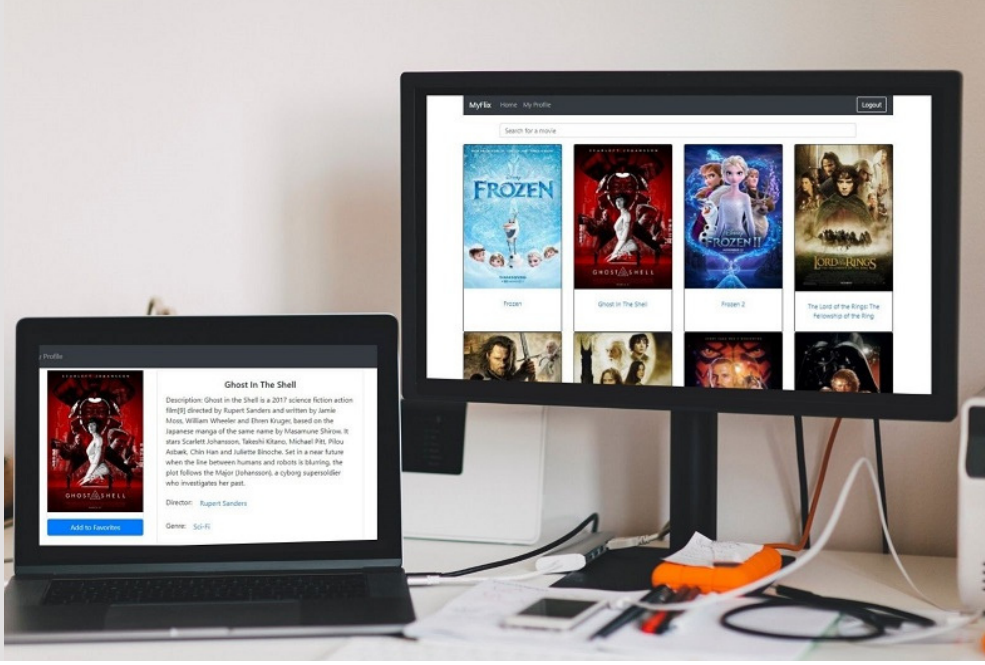
My Flix Movie APP



OVERVIEW

MYFLIX IS A RESPONSIVE, SINGLE PAGE APP THAT DISPLAYS A LIST OF MOVIES AND ENABLES USERS TO SEE MORE INFORMATION ABOUT THE MOVIE, THE GENRE AND THE DIRECTOR. USERS CAN REGISTER, LOGIN, LOG OUT AND CREATE A LIST OF FAVORITE MOVIES STORED IN THEIR PROFILE ACCOUNT.

GOAL

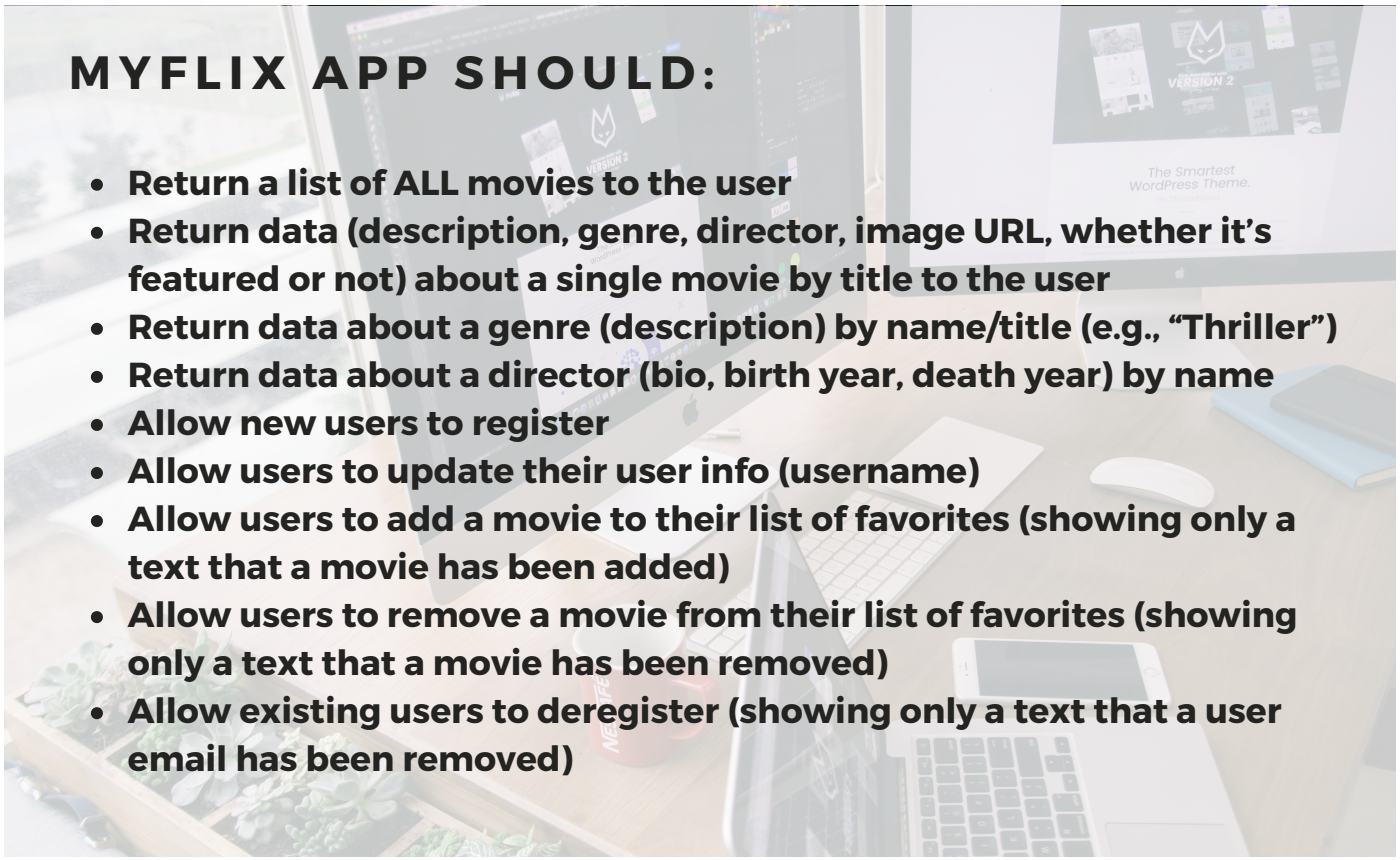
This is a personal project to create a safe, inclusive and responsive movie app using the skills acquired during the full-stack web development course at career Foundry and showcase both front-end and back-end web development skills.

TECHNOLOGY

FRONT-END

BACK-END

OBJECTIVES



MYFLIX APP SHOULD:

- **Return a list of ALL movies to the user**
- **Return data (description, genre, director, image URL, whether it's featured or not) about a single movie by title to the user**
- **Return data about a genre (description) by name/title (e.g., "Thriller")**
- **Return data about a director (bio, birth year, death year) by name**
- **Allow new users to register**
- **Allow users to update their user info (username)**
- **Allow users to add a movie to their list of favorites (showing only a text that a movie has been added)**
- **Allow users to remove a movie from their list of favorites (showing only a text that a movie has been removed)**
- **Allow existing users to deregister (showing only a text that a user email has been removed)**

METHODOLOGY

SERVER SIDE

Restful API and Express

I initialized the project with "package.json" file using npm init. I used Express framework to route HTTP requests, log and serve static files and handle errors. To list all endpoints, methods and formats that will be used to carry out the CRUD manipulations I drafted the Documentation file. For example, Express GET route is located at the endpoint /movies and returns a JSON object containing data about movies. Express Static serves the "documentation.html" file with a description of all endpoints, methods and formats. Morgan middleware library logs all requests. Error-handling middleware function logs all application-level errors to the terminal. All endpoints and their methods were tested using Postman.

Internal Database

I created the database with MongoDB which included the movies and users collections. I used embedded documents to store information about genre and director for the movie collection and user information for the users collection. I created the schema for the database using Mongoose models and integrated the models with the rest of my application. All methods were again tested with Postman.

Authentication and authorization

I used Passport for HTTP authentication and JWT authentication, created a new "/login" endpoint for registered users that contains logic for authenticating users with basic HTTP authentication and generating a JWT token for authenticating future requests. Integrated the Passport strategies as middleware into each of my existing API endpoints so that only users with a JWT token can make requests to my API.

Security mechanisms

I implemented CORS into the app, ensuring that all domains are allowed to make requests to my API. Added password hashing to the user schema and integrated it into the login and registration HTTP handlers to ensure that passwords aren't stored in my database without first being hashed. Added data validation in the form of alphanumeric data to login and registration endpoints. If entered data doesn't meet the requirements, an error message should be sent as an HTTP response back to the client.

Deployment

Final step of the server side project was deployment. I deployed the application to Heroku, uploaded the database to MongoDB Atlas. Connected my Heroku application to my MongoDB Atlas database by adding the new connection URI with the use of environment variables. Then, pushed the changes to Heroku and tested that my Heroku app loads and can connect to the database. Tested the endpoints in Postman to ensure that everything is working correctly between the app and database.

CLIENT SIDE

React

React was used to build the client-side of the application. The app contains several components that represent the “views” of the app. The “main view” component renders movies as “movie cards”. “Movie card” component displays the image and the title of the movie. Clicking on the MovieCard takes users to the “movie view” component which renders the full information about the movie, including description, genre, director and a button that adds the movie to a list of favorites. Clicking on the director or genre opens a “director view,” and “genre view,” I also developed a “profile view” where users can edit their information, delete the account and view their list of favorite movies. A back button has been implemented in all views to take users to the previous page. In addition, “login view” and “registration view” that are function components with hooks have been designed and ask users to input data to either login or register if they are a new user.

React Bootstrap

React Bootstrap was used to add a responsive grid and incorporate all UI elements like buttons, form elements and navigational components. I made sure I applied consistent styling across all views of the app.

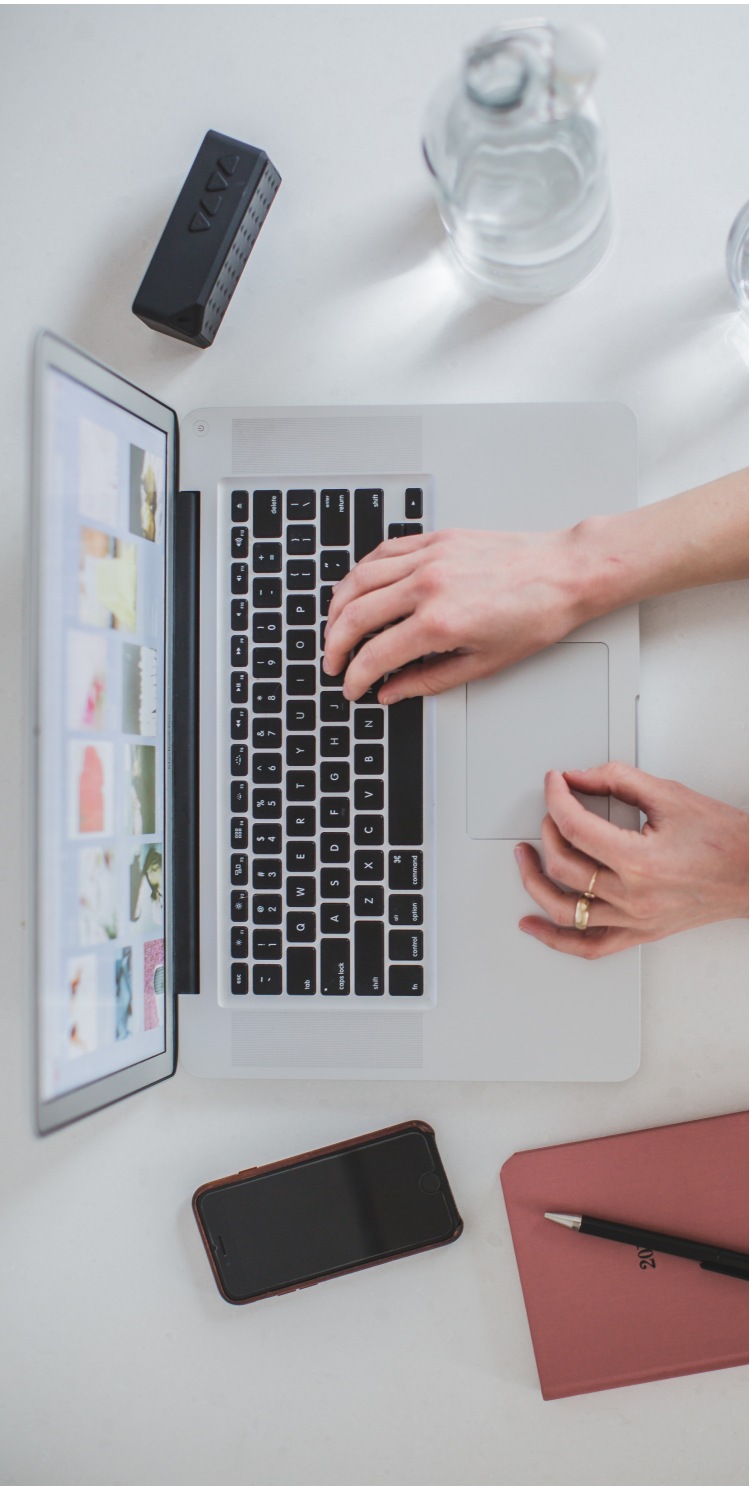
Connecting client side to server side

React Router was used to enable routing in the application that corresponds to the endpoints designed in the server side of the app. I also implemented the necessary code to store and send a JWT token alongside client requests for authenticated users and client-side form validation for the login and registration views based on the server-side validation I implemented in the server-side of the app.

React Redux and final steps

As part of training I restructured my app using React Redux and Flux to manage the application’s state in order to make the project more scalable. The movies, user currently logged in and the filter for the movie list is handled by stores and modified through actions. Finally the app was tested in multiple browsers to ensure it is running smoothly and the app will be hosted on Netlify.

CONCERNS



DATA SECURITY

Data security is one of the things to have in mind when building an app. It is important to make sure the users are safe to use the app, without worrying that their data might be stolen with malicious intent.

ACCESSIBILITY

No one should be excluded when it comes to accessibility of your app. We should always consider how our apps are perceived by different people and to ensure no one is discriminated because of their abilities or disabilities.

USER EXPERIENCE

There are so many apps out there now. Due to this competition developers need to ensure their app is not only accessible but also performs well, meaning it is fast, easy to use and is responsive otherwise the user will simply switch to a different app.

SOLUTIONS

DATA SECURITY

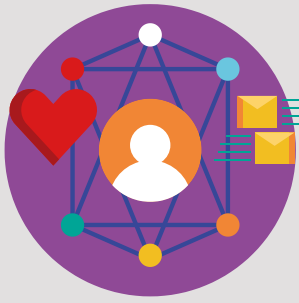
To ensure data security, I included user authentication and authorization code in the form of basic HTTP authentication and JWT authentication. Users have to create a unique log in, Only alphanumerical characters are allowed when registering, to minimize the probability of anyone trying to inject malicious code in the log in form that may harm the app. Each time the user logs in, they are assigned a JWT token, which expires after a certain period of time. The password is hashed, which reduces the probability of anyone stealing the password data.

ACCESSIBILITY

My goal was to create an app that is accessible for everyone. I used contrasting colors so it is easy to read the text and view the content. I made user the app is easy to use- all buttons are highlighted and the app uses an intuitive design. I have also included the alt text on all elements of my apps, so the app is easy to navigate for people who use a screen reader which will read out all elements like buttons, images and decorations.

USER EXPERIENCE

To ensure the app performs well, I ensured that all frameworks and libraries I use are reliable, have a big following on github, and are up-to-date. I used best coding practices introduced to us by career foundry, ensuring there is no redundant code, the images are small enabling the fast loading times of the app. I used Bootstrap to make the app responsive so that users can view the app on any device without the app losing on its design. Finally users have full control over their account and can update or delete it as necessary.



SUMMARY

MY FLIX

A MOVIE APP WITH THE LIST OF MOVIES AND HAS MOVIE, DIRECTOR, GENRE, LOGIN/REGISTER AND PROFILE VIEWS. USERS CAN CREATE A LIST OF FAVORITE MOVIES

LEAD DEVELOPER:

VERONIKA ROGATSKAJA

TECH STACK:

MERN

DURATION:

3 WEEKS

This project has been a true test to my skills. Every technology used here was completely new to me and thus I did extensive additional research to make sure I adhere to best coding standards while maintaining the positive user experience.

I made use of all the resources available to me, google, slack community for the course, github community for React I reached out to my tutors and mentors and having had the approval from them, I can confidently say that MyFlix App is a great example of a full-stack project and it strives to highlight and enhance the safety, accessibility and user experience.