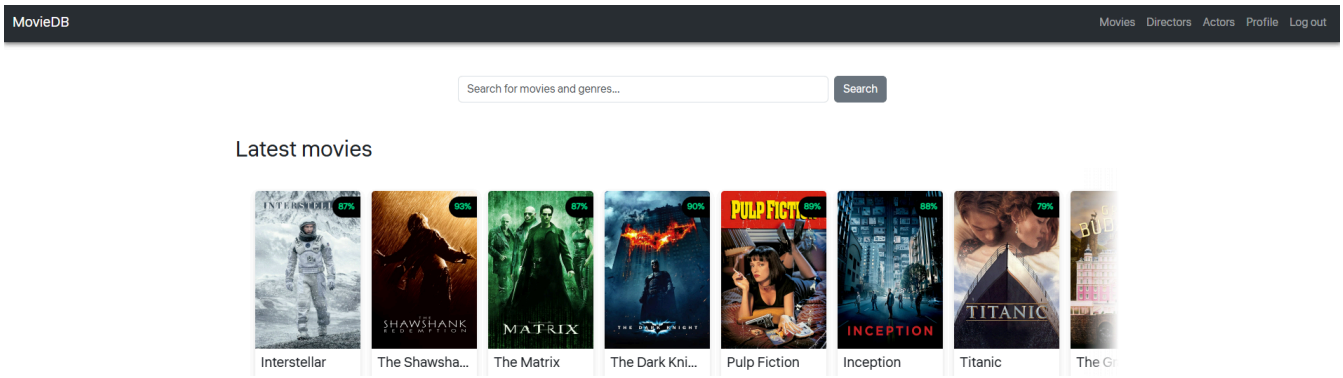
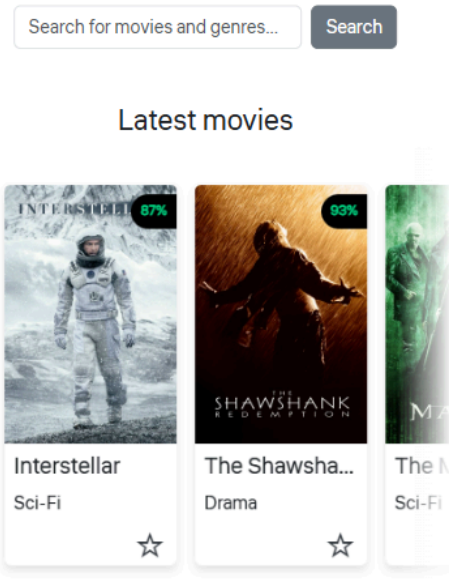


Movie database

Full-stack web app

Overview

Movie database is a full-stack web app I worked on during Careerfoundry's web development course. It's a MERN stack app that allows users to browse movies, view movie information and trailers, as well as information about directors, actors and genres. It also features a user profile view, ability to edit account info and ability to add and remove movies from favourites.



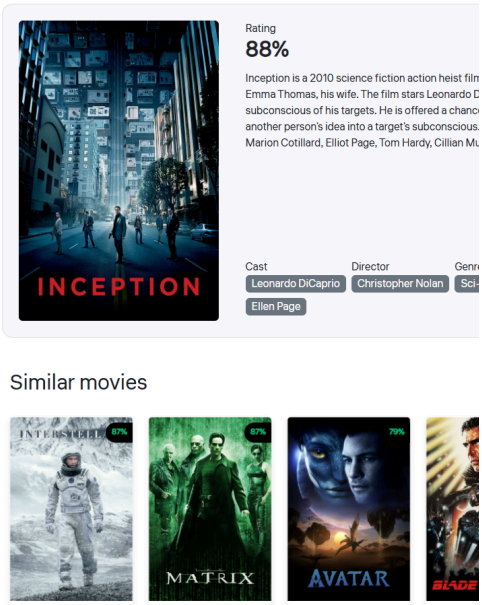
Project goals

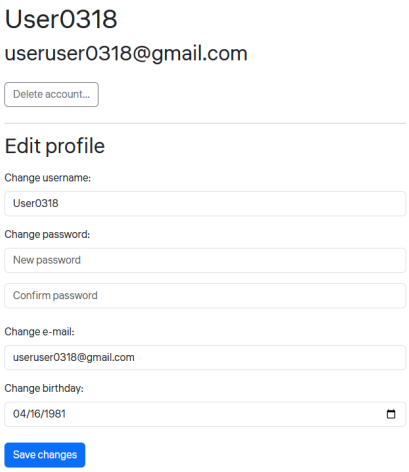
- Learn how to build an API with Node/Express
- Learn how to work with a MongoDb database
- Learn how to use React
- Create a portfolio-ready app

Challenges

This project was my first time learning about Node/Express, MongoDb and React and all of the associated topics, so trying to put together a functional app with a pleasant user experience while learning was challenging.

I implemented features little-by-little, revising my code as I learned more about each topic. I would also tweak the visual design of the front-end over time.





Takeaways

By the end of the project I felt I had a good understanding of React and Node/Express fundamentals. Aside from learning the basics of the MERN stack, I gained valuable experience about how to write code in both back-end and front-end and what kind of solutions I could employ to problems.

The back-end

Node / Express / MongoDb / Mongoose

The first part of the project consisted of putting together an API that accessed a database hosted on MongoDb Atlas. While we were taught about SQL databases, for this project a non-relational database was deemed more suitable.

This involved creating all of the API endpoints for managing movies and user information in Node/Express. Mongoose was used for interacting with the database. Authentication for accessing each endpoint, as well as logging the user in, was done with Passport. The method of authentication was JSON Web Tokens.

All API endpoints were tested with Postman, which was crucial for both verifying that they worked in the first place, but also for debugging.

The API was hosted on Heroku.

The front-end

React / React-Redux / React Bootstrap

The second part had me working on the front-end, which was done with React. React Bootstrap was used for styling. The app is a single-page application, so React Router was used for switching between different views.

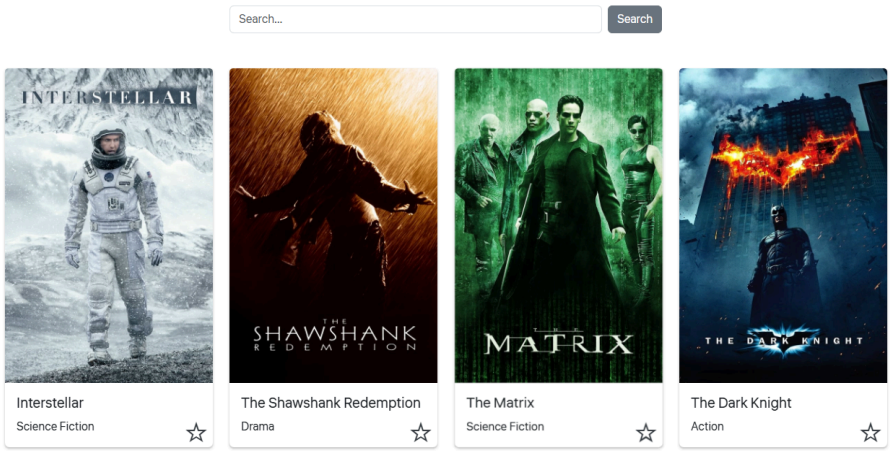
The bulk of the work consisted of creating components for all of the necessary views like MainView, MovieView, ProfileView, etc, as well as implementing API fetch requests for getting data from the database and then displaying it on the page with help from Bootstrap.

For styling I chose to go with a “less-is-more” approach with no frills, since I felt it served the purpose and was easy to maintain.

Towards the end of the project I implemented React Redux, which made state management simpler and more reliable.

I also implemented a search function and client-side pagination.

The site was hosted on Netlify.



Conclusion

I was ultimately happy with the result and with how much I'd learned about everything in the MERN stack. The experience gave me the confidence that I could create such an app and I wanted to continue to learn and get better at it.

With the benefit of hindsight, I would perhaps revise my API code to be a bit cleaner, and at the time of writing I still might. I would also like to implement server-side pagination which is more suitable for larger datasets (many movies).