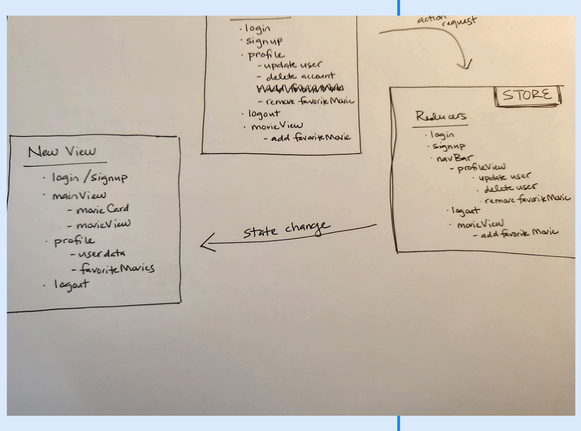
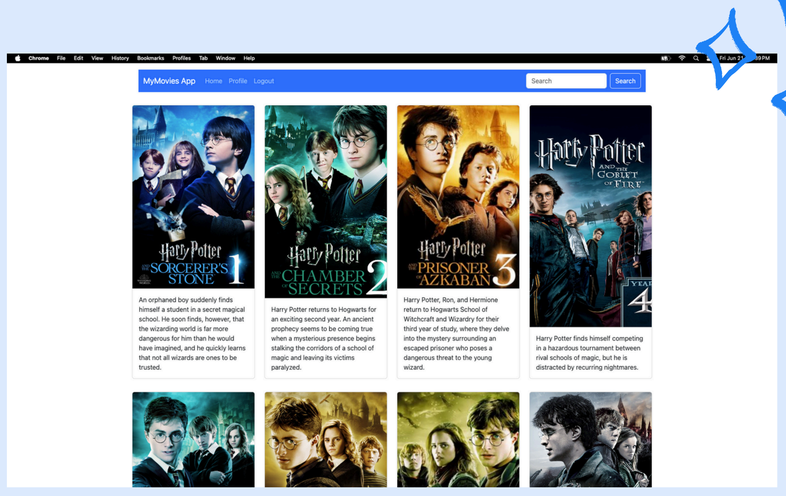
MyMovies App: a case study

Planning

Objective:
As part of my CareerFoundry curriculum, I built a full-stack MERN (MongoDB, Express, React, Node.js) app. The goal was to create a user-friendly movie database app that integrates a custom-built RESTful API, allowing users to securely sign up, log in, and manage their favorite movies/profile.



Conceptualization:
The concept was clear from the outset: a movie app where authenticated users could perform CRUD operations on their profiles and favorite movies. The idea was to make the app useful, visually appealing, and enjoyable to use. Before building this app, I mapped out user sequences and views to create a clear development path.



Development

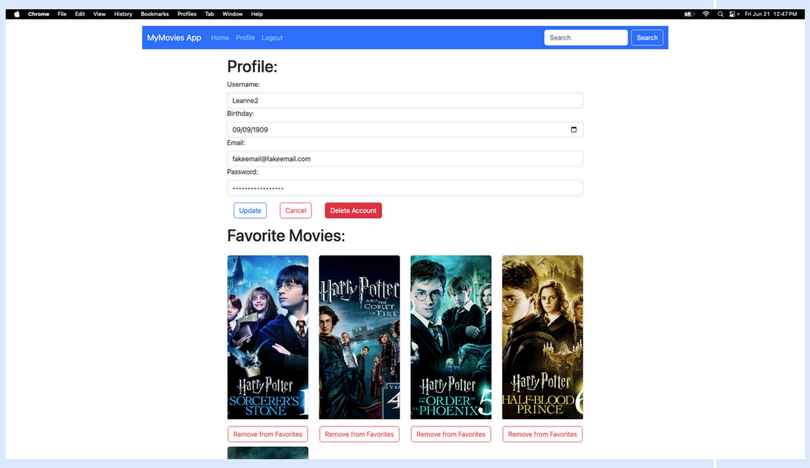
Structure:
I began by constructing the backend using Express and Node.js. The combination of Express's lightweight framework and Node.js's asynchronous programming provided a scalable and efficient way to connect the frontend and backend. This was my first time using the terminal, which was initially intimidating and mysterious. I admit that I continued to use GitHub Desktop and my PC's interface as much as possible.

API Design:
I then created a RESTful API supporting functionalities for both unauthenticated and authenticated users. Authenticated users could perform CRUD operations on their favorite movies and profile information. I found working with MongoDB Atlas to be very gratifying, and a perfect fit for a movie API because of its flexible schema design, cloud-managed environment, and ease of scaling.

Security:
I implemented JWT authentication to ensure secure user data management. This was my first experience with authentication, and inadvertently led to some challenges. Initially, I required users to enter their passwords for every view and CRUD operation without UI prompts, which took a fair amount of troubleshooting to discover.

Design:
Building the frontend with React was a significantly smoother experience. React's component-based architecture felt similar to the organized-based logic of HTML/CSS, making the transition into fluency with this framework easier. By using React in a SPA design, I maximized React's virtual DOM with a SPA's asynchronous data-fetching to give users a dynamically responsive app without page refreshes. I also wanted to give users fuzzy search functionality. A bit of research and documentation reading led me to implement Fuse.js so users could do just that!

Testing:
I used Postman to test API endpoints, ensuring they worked as expected. This was crucial in identifying and fixing issues early in the development process.



Reflection

Conclusion:
Building the MyMovies app was a significant milestone in my coding journey. It introduced me to React (my new favorite framework!), reinforced my problem-solving skills, and demystified the terminal...all while achieving a beautiful and functional full-stack app. This project greatly enhanced my technical skills and deepened my passion for continuous learning. Moving forward, I plan to implement new features, expand the database, and use Vercel for deployment due to its intuitive interface, speed, and auto-deployment features.

Deployment:
Another point of challenge was deployment. Since using the terminal was still rather foreign, deploying and debugging the backend in Heroku took a significant amount of troubleshooting with my tutor. Deploying the frontend in Netlify still took some debugging, but was less of a challenge at that point as the terminal itself was losing a bit of its mystery.

See the frontend repo on Github: https://github.com/leanneduyck/MyMovies-Client.git or visit the live site: main--react-mymovies.netlify.app/
See the backend repo on Github: https://github.com/leanneduyck/MyMovies-Server.git or visit the API: https://my---movies-868565568c2a.herokuapp.com/