Movie Seek Application

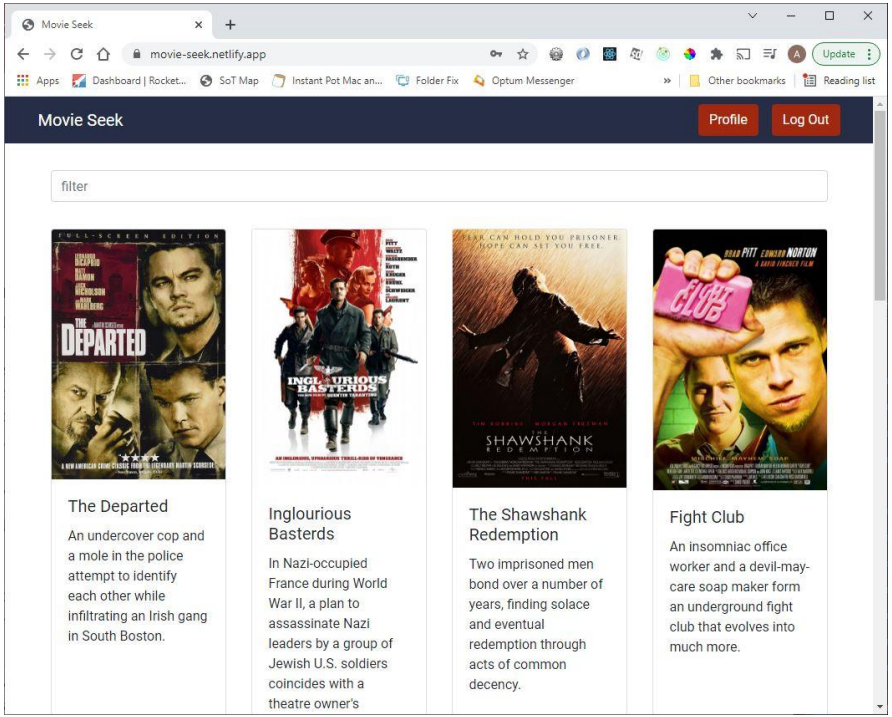
A Case Study by Alex Guizar

Overview

Movie Seek is a personal project developed using the MERN stack. The app provides information on movies, directors, and genres and allows users to create accounts and save movies to a favorites list.

Purpose & Context

I created this project as part of my Career Foundry Achievement project to showcase my capabilities as a Full-Stack Developer.



Objective

The goal of this project was to develop a full-stack application that I could add to my professional portfolio and solidify my understanding of full-stack development. By creating the database, the business logic layer, and the client-side, I could develop and polish my skills as a Full-Stack Developer.

Duration

3 Month Project

Role

Full-Stack Developer

Tools Used

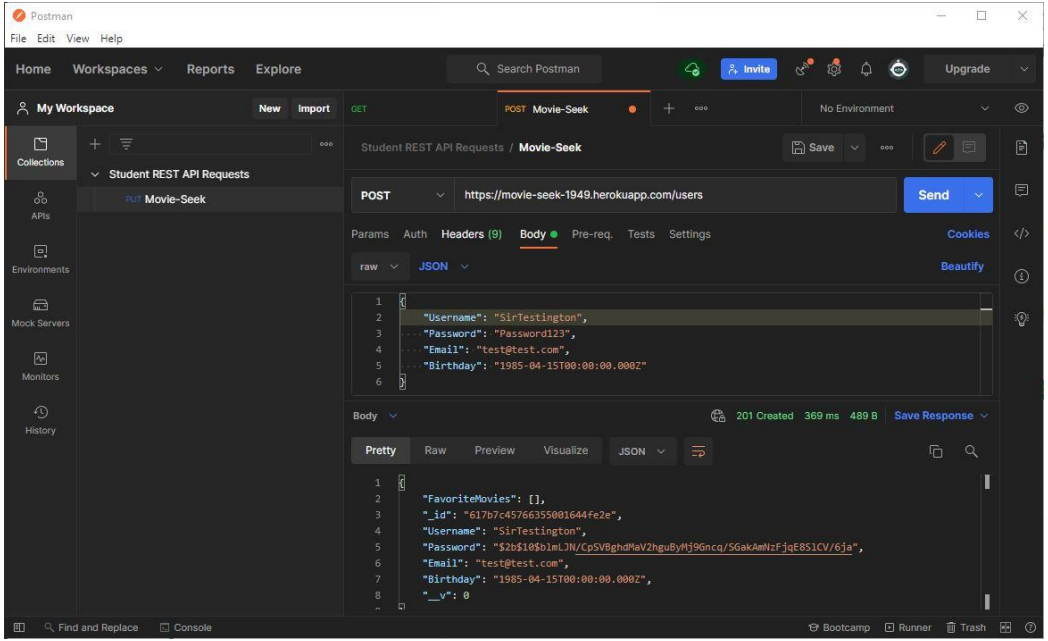
- Node.js
- Express
- Mongoose
- MongoDB
- React
- React-Redux
- React-Bootstrap

Approach

Server Processes

I began by creating the REST API using Node.js and Express to specify what endpoints to use, using that information to decide what data to provide from my database. With the API constructed, I can jump right into testing with Postman once I have created my database.

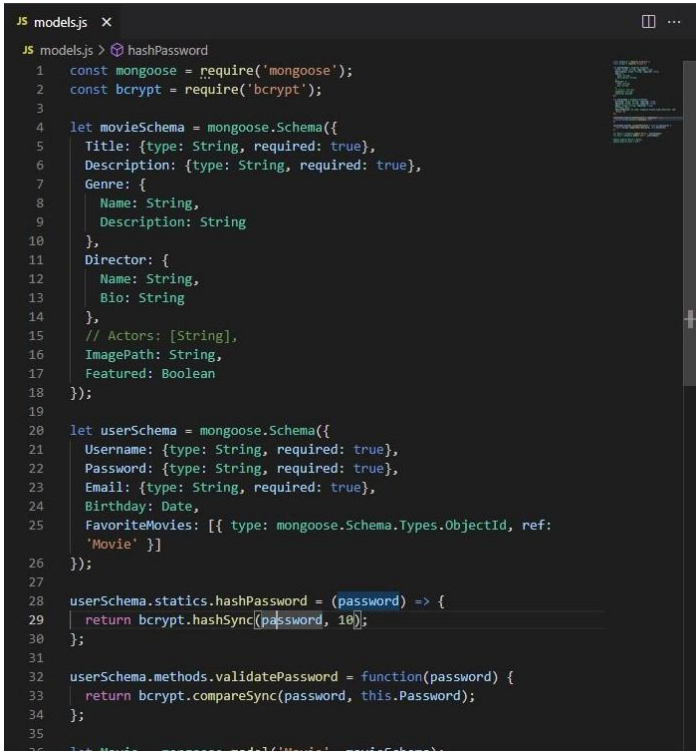
View Endpoints



Database

Once my REST API endpoints were tested, creating my database was my next task to tackle. I utilized the non-relational database MongoDB for its flexibility and expandability should I decide to expand the application. After creating the database, I populated it with a variety of movies from different directors and genres. I then structured the information in a way that made sense and would be easily retrievable.

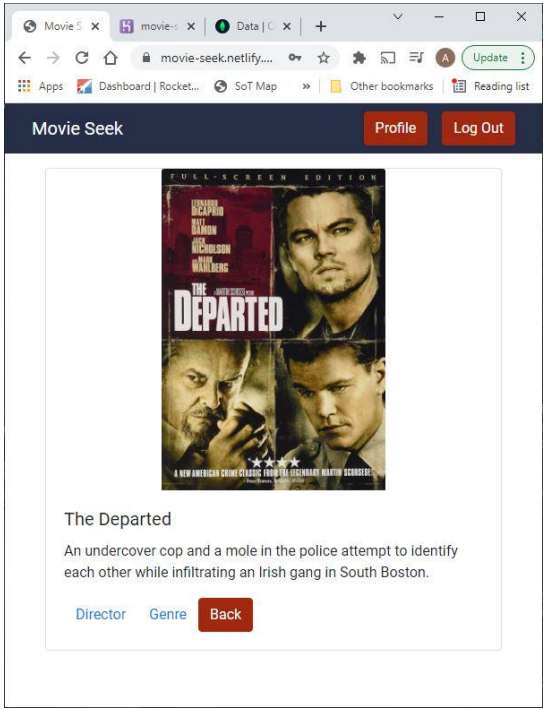
Business Logic



Now that I had my data ready, I moved to construct the business logic. Using Mongoose and its focus on models, I created structures for the database to be mapped to when used by the API. These models allowed me to ensure the data is structured consistently for the database.

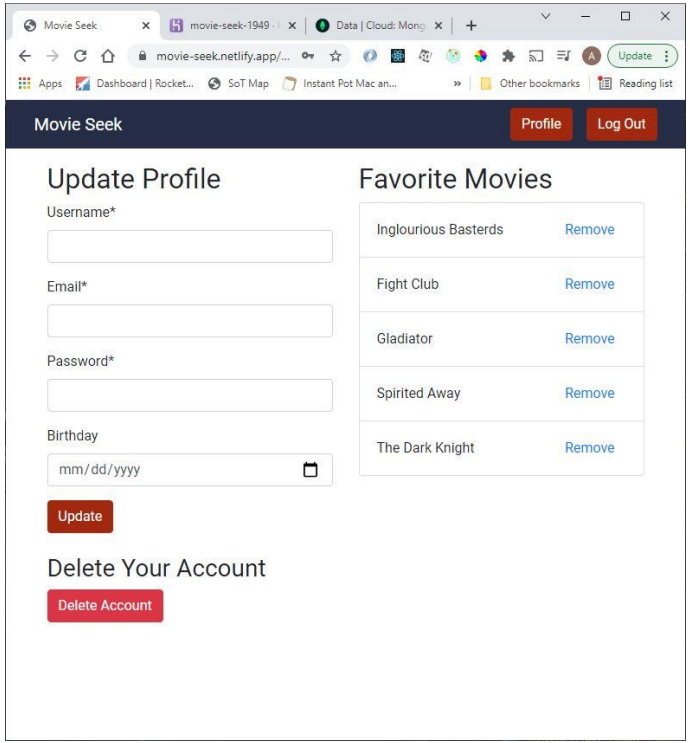
Client-Side

With the back-end in place, I now know the type of information to be provided to the user and how they will get it. I selected React as my JavaScript framework as its virtual DOM ensures faster rendering when moving from view to view. React's scalability will allow me to expand my application should I decide to expand upon it, and its popularity makes sure there will be support for a long time. Combining this with React-Router allowed me to create navigation for my Single Page Application (SPA). React-Redux was selected to have an application-wide state and have that state flow in one direction.



Client-Side (Cont.)

Now that I had my structure and navigation in place, I moved on to the design. I chose React-Bootstrap to apply quick and clean styling in a short time frame.



Summary

With the back-end in place, I now know the type of information to be provided to the user and how they will get it. I selected React as my JavaScript framework as its virtual DOM ensures faster rendering when moving from view to view. React's scalability will allow me to expand my application should I decide to expand upon it, and its popularity makes sure there will be support for a long time. Combining this with React-Router allowed me to create navigation for my Single Page Application (SPA). React-Redux was selected to have an application-wide state and have that state flow in one direction.

For my second iteration, I plan to overhaul the UI/UX. I was not satisfied with the end design of the project and will make it a focal point in my goal for a better user experience. I will also increase the amount of data per movie that is available. This increase in data will allow me to include a more fine-tuned search option as well.