CAREERFOUNDRY

Achievement 2 Project Brief: myFlix

Create a REST API with Node.js

Objective

To build the server-side component of a “movies” web application. The web application will provide users with access to information about different movies, directors, and genres. Users will be able to sign up, update their personal information, and create a list of their favorite movies.

Context

It’s no longer enough for a JavaScript developer to be skilled in frontend development alone. It’s become essential that you be able to interface with (as you did in Achievement 1) and even create your own APIs. For this reason, throughout this Achievement, you’ll create a REST API for an application called “myFlix” that interacts with a database that stores data about different movies.

In the next Achievement, you’ll build the client-side component of this same application using REACT. By the end of Achievement 3, you’ll have a complete web application (client-side and server-side) built using full-stack JavaScript technologies that you can showcase in your portfolio. The project will demonstrate your mastery of full-stack JavaScript development, including APIs, web server frameworks, databases, business logic, authentication, data security, and more. The complete tech stack you’ll master is known as the MERN (MongoDB, Express, React, and Node.js) stack.

Note that it’s not a requirement that you name your application “myFlix.” You’re also free to decide which movies you wish to include in your project. For example, you may want to build a movie application specifically for period or superhero movies, or a more diverse application with a wider user base. It’s up to you!

The 5 W's

1. Who—Your immediate users will be frontend developers who'll work on the client-side for the application based on what's been documented on the server-side (in this case, this developer is also you!). You should also consider the users of the myFlix application. These will be movie enthusiasts who enjoy reading information about different movies.

2. What—The complete server-side of your web application, including the server, business logic, and business layers of the application. It will consist of a well-designed REST API and architected database built using JavaScript, Node.js, Express, and MongoDB. The REST API will be accessed via commonly used HTTP methods like `GET` and `POST`. Similar methods (CRUD) will be used to retrieve data from your database and store that data in a non-relational way.

3. When—Whenever users of myFlix are interacting with the application, the server-side of the application will be in use, processing their requests and performing operations against the data in the database. These users will be able to use the myFlix application whenever they like to read information about different movies or update their user information, for instance, their list of "Favorite Movies."

4. Where—The application will be hosted online. The myFlix application itself is responsive and can therefore be used anywhere and on any device, giving all users the same experience.

5. Why—Movie enthusiasts want to be able to access information about different movies, directors, and genres. The server-side of the myFlix application will ensure they have access to this information, that their requests can be processed, and that all necessary data can be stored.

Design Criteria

User Stories

- As a user, I want to be able to receive information on movies, directors, and genres so that I can learn more about movies I've watched or am interested in.
- As a user, I want to be able to create a profile so I can save data about my favorite movies.

Feature Requirements

The feature requirements below were extracted from the user stories listed above. **Your project will only be approved if the following "essential" feature requirements are implemented in your Achievement project**.

Essential Features

- 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, password, email, date of birth)
- Allow users to add a movie to their list of favorites
- Allow users to remove a movie from their list of favorites
- Allow existing users to deregister

Optional Features

These are optional features. You can incorporate these into your project through Bonus Tasks as you work through the Achievement. If you don't have time, you can use this list as inspiration for a second iteration of your application once you've completed the course.

- Allow users to see which actors star in which movies
- Allow users to view information about different actors
- Allow users to view more information about different movies, such as the release date and the movie rating
- Allow users to create a "To Watch" list in addition to their "Favorite Movies" list

Technical Requirements

- The API *must* be a Node.js and Express application.
- The API must use REST architecture, with URL endpoints corresponding to the data operations listed above
- The API *must* use at least three middleware modules, such as the body-parser package for reading data from requests and morgan for logging.
- The API *must* use a "package.json" file.
- The database *must* be built using MongoDB.
- The business logic must be modeled with Mongoose.
- The API *must* provide movie information in JSON format.

- The JavaScript code *must* be error-free.
- The API *must* be tested in Postman.
- The API *must* include user authentication and authorization code.
- The API *must* include data validation logic.
- The API *must* meet data security regulations.
- The API source code *must* be deployed to a publicly accessible platform like GitHub.
- The API *must* be deployed to Heroku.

Your Project Deliverables

Throughout this course, you'll be working from Exercise to Exercise to complete your project. For each Task, you'll submit a deliverable that directly contributes to the final product—in this case, the backend component of your myFlix web application.

Make sure each step in your development process is well-documented to ensure you have high-quality deliverables to incorporate into your professional portfolio.

Below is a breakdown by Exercise of your Achievement project deliverables:

Exercise 1: Intro to Server-Side Programming

- Set up your project directory

Exercise 2: Node.js Modules

- Practice writing Node.js syntax

Exercise 3: Packages & Package Managers

- Create a "package.json" file
- Import all necessary packages into project directory
- Define your project dependencies

Exercise 4: Web Server Frameworks & Express

- Route HTTP requests for your project using Express

Exercise 5: REST & API Endpoints

- Define the endpoints for your REST API

Exercise 6: Relational Databases & SQL

- Create a relational (SQL) database for storing movie data using PostgreSQL

Exercise 7: Non-Relational Databases & MongoDB

- Recreate your relational (SQL) database as a non-relational (NoSQL) database using MongoDB

Exercise 8: The Business Logic Layer

- Model your business logic using Mongoose

Exercise 9: Authentication & Authorization

- Implement authentication and authorization into your API using basic HTTP authentication and JWT (token-based) authentication

Exercise 10: Data Security, Validation, & Ethics

- Incorporate data validation logic into your API
- Implement data security and storage controls
- Host your project on the web using Heroku