1.4: Creating a Case Study for Your Portfolio

# myFlix-client

## React App

Payton Provo

# PROJECT OVERVIEW

myFlix is a movie database app to access information on movies, directors, and genres so that a user can learn more about movies they have watched or are interested in. Users are able to create an account, update their personal data, and create a list of favorite movies.

Objective:

Using React, build the client-side for an application called myFlix based on its existing server-side code (REST API and database).

Essential Views and Features:

Main view
- Returns a list of ALL movies to the user (each listed item with an image, title, and description)
- Sorting and filtering
- Ability to select a movie for more details

Single movie view
- Returns data (description, genre, director, image) about a single movie to the user
- Allows users to add a movie to their list of favorites

Login view
- Allows users to log in with a username and password
- Registration view
- Allows new users to register (username, password, email, birthday)

Genre view
- Returns data about a genre, with a name and description
- Displays example movies

Director view
- Returns data about a director (name, bio, birth year, death year)
- Displays example movies

Profile view:
- Allows users to update their user info (username, password, email, date of birth)
- Allows existing users to deregister
- Displays favorite movies
- Allows users to remove a movie from their list of favorites

# Technical requirements

- The application must be a single-page application (SPA)
- The application must use state routing to navigate between views and share URLs
- The application must give users the option to filter movies
- The application must give users the option to sort movies
- The application must initially use Parcel as its build tool
- The application must be written using the React library and in ES2015+
- The application must be written with React Redux (hence respecting the Flux pattern)
- The application must use Bootstrap as a UI library for styling and responsiveness
- The application must contain a mix of class components and function components
- The application may be hosted online

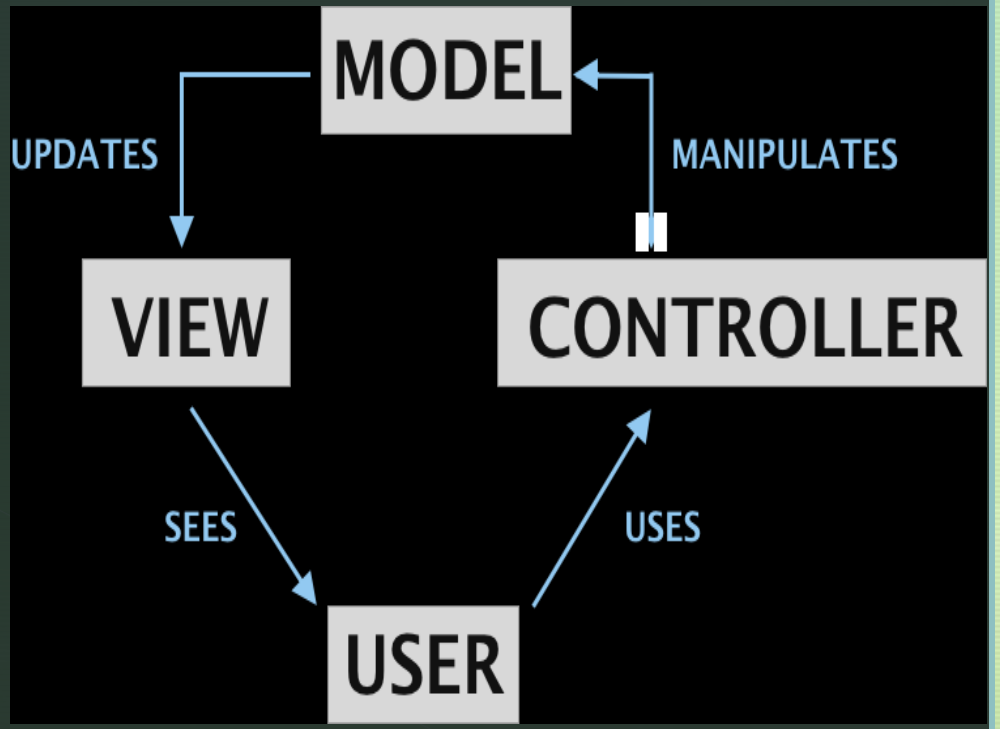
# Purpose and Context/Objective

Purpose and Context:
myFlix was a personal project I build as part of my web development course at CareerFoundry to demonstrate my mastery of full-stack JavaScript development.

Objective:
The aim of the project was to have an ambitious full-stack project I can add to my professional portfolio. The problem I wanted to solve is to build the complete server-side and client-side for the application from scratch.

# USER FLOWS

MVC is an architectural pattern—or design pattern—that defines the structure of an application, or part of an application, by separating it into specific parts with specific roles.




Using MVC architecture, this would work as follows:
Model: Loads movies from the server.
View: Displays the movies, each with a special toggle for users to click on if they want to "Favorite" or "Unfavorite" a particular movie.
View → Controller: Each time the user clicks on the toggle to add a movie to their "Favorites" list, the controller is triggered.
Controller → Model: The controller sends the user's request to favorite the movie to the model and instructs the model to refresh.
Model: The model is refreshed. Back to step one.

# Technologies Used

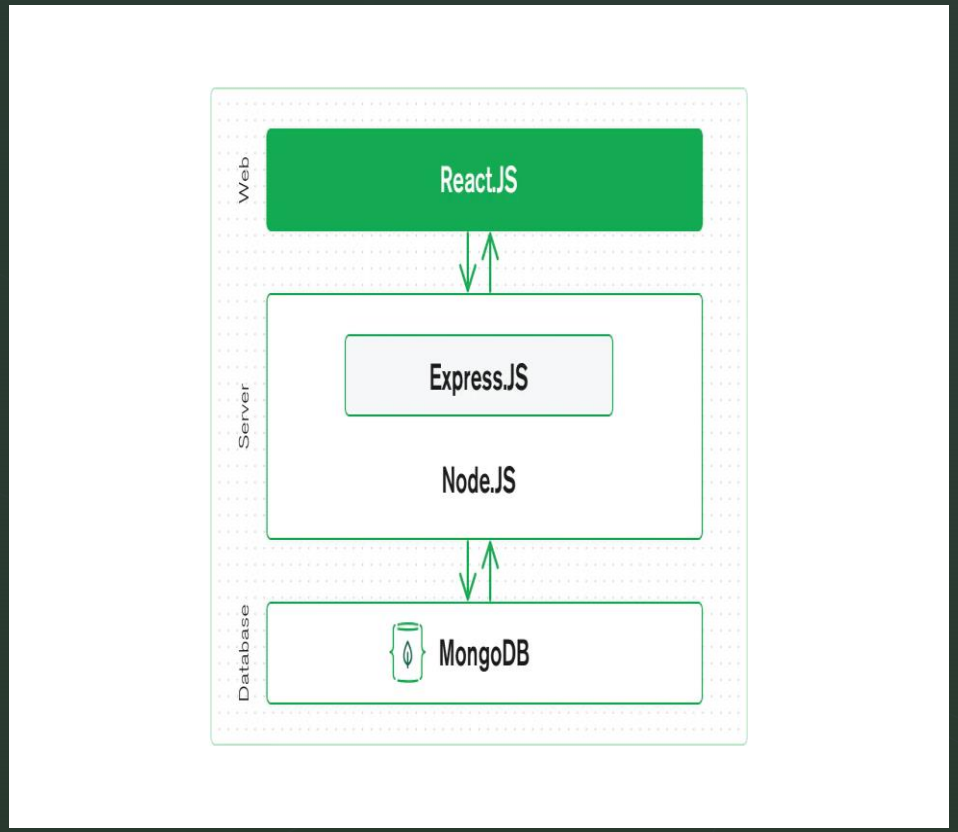
MERN:
MERN stands for MongoDB, Express, React, Node, after the four key technologies that make up the stack.

MongoDB - document database

Express(.js) - Node.js web framework

React(.js) - a client-side JavaScript framework

Node(.js) - the premier JavaScript web server



React:
React JS is a JavaScript library used in web development to build interactive elements on websites. React will design simple views for each state in your application and will efficiently update and render just the right components when your data changes.

Example:

```
import React from 'react';
import PropTypes from 'prop-types';
import Button from 'react-bootstrap/Button';
import Card from 'react-bootstrap/Card';
import { Link } from "react-router-dom";

import "./movie-card.scss";


export class MovieCard extends React.Component {
  render() {
    const { movie } = this.props;
```

CSS:
Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

Example:

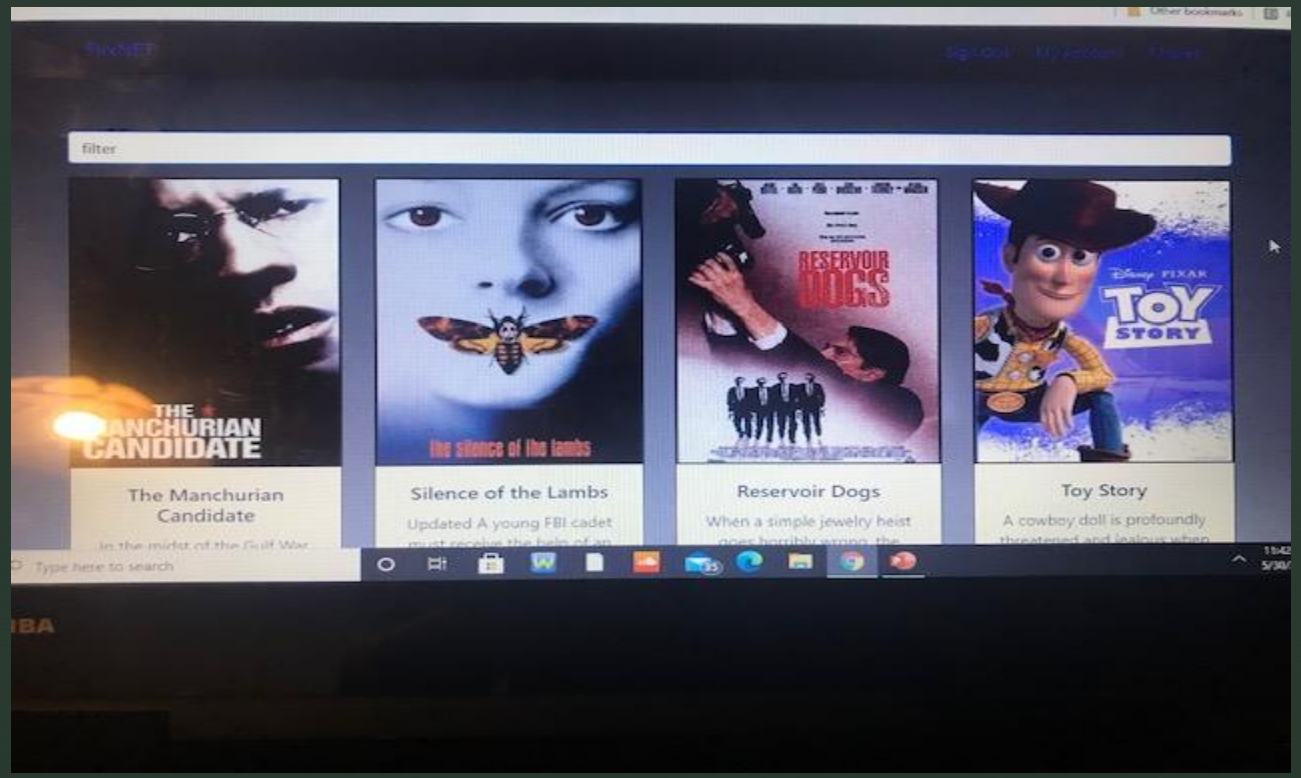
```
#profile-page-title {
  font-size: 2.5rem;
  text-decoration: underline;
  cursor: default;
}

#profile-details-title {
  font-size: 2.5rem;
  text-decoration: underline;
  cursor: default;
}
```

# My Application

Here is my link to my GitHub repository:

https://github.com/1998-creator/myFlix-client



# Challenges

- Learning how to understand React and Redux for the first time was challenging since I hadn’t worked with anything like that before
- Figuring out which format of the app I wanted and which CSS styles fit that format
- Implementing the API that I built into this app
- Making sure I keep a good schedule on working on the project

# RETROSPECTIVE

What went well:
After the completion of the project, no errors were found. The simplicity of the site allows a user to easily pick up how the app functions. If a real movie database was added to it, and with some tweaks to the site, it could be a decent app.

What didn’t go well:
I had some problems with a few of the CSS stylings and formatting decisions that I was able to get help from my course mentor.

What can be improved:
Some more CSS stylings could be added to make it look more like a legit app. Also adding in an Actors view and a watch list for users to pick out a movie to watch in the future.