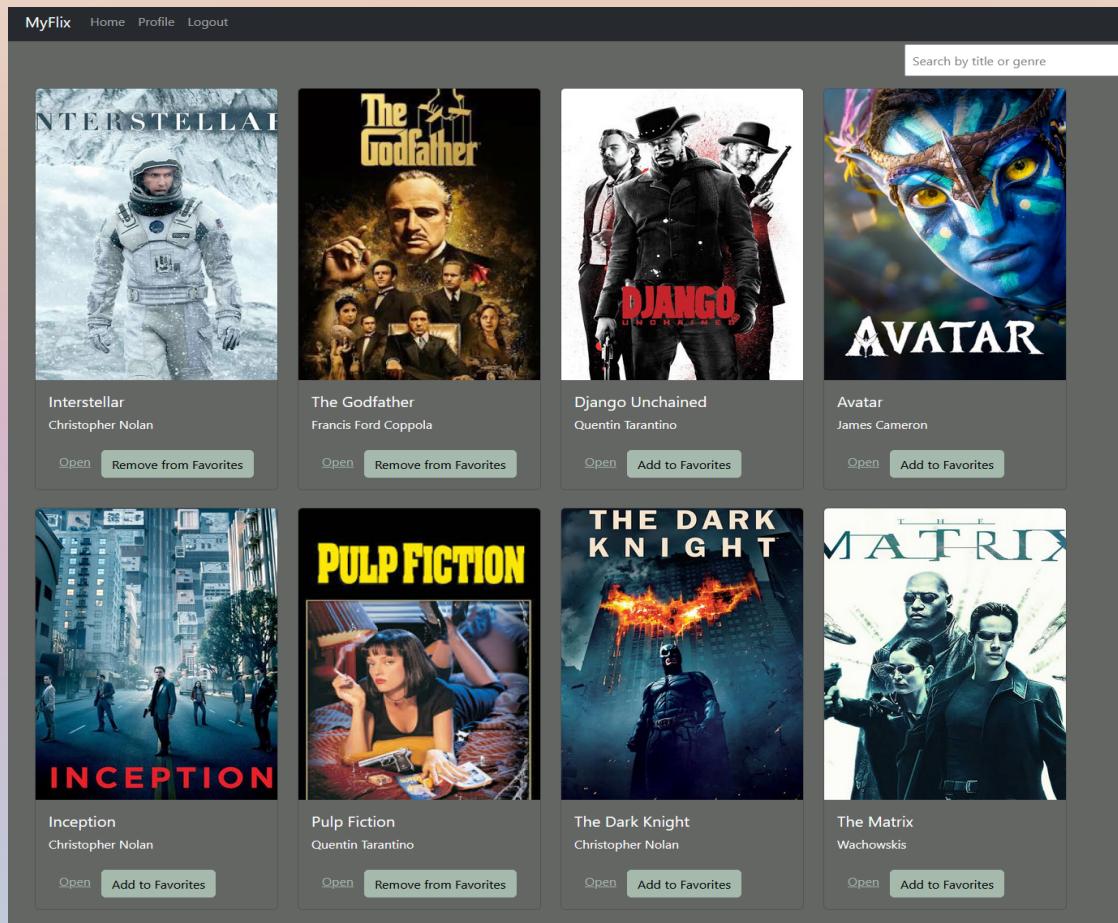


MyFlix

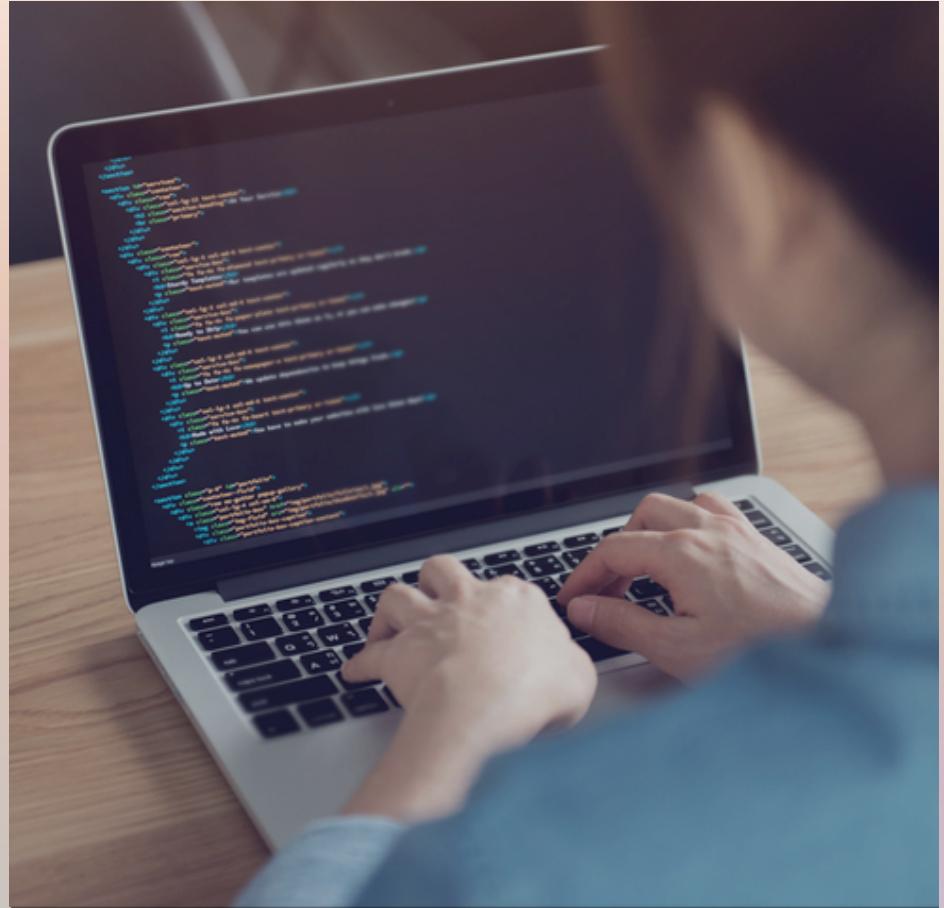
A FULL STACK CASE STUDY



Plumby Eschenbacher

OVERVIEW

MyFlix is a web application build using React. This application allows user to browse, search and explore details about movies, directors and genres, with additional functionalities for user account management and personalization.



OBJECTIVE

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

```
1 import { useState, useEffect } from 'react';
2 import { MovieCard } from '../movie-card/movie-card';
3 import { MovieView } from '../movie-view/movie-view';
4 import { LoginView } from '../login-view/login-view';
5 import { SignupView } from '../signup-view/signup-view';
6 import { NavigationBar } from '../navigation-bar/navigation-bar';
7 import { UserProfile } from '../profile-view/profile-view';
8 import Container from 'react-bootstrap/Container';
9 import Row from "react-bootstrap/Row";
10 import Col from 'react-bootstrap/Col';
11 import { BrowserRouter, Routes, Route, Navigate } from "react-router-dom"
12
13
14 export const MainView = () => {
15   const storedUser = JSON.parse(localStorage.getItem("user"));
16   const storedToken = localStorage.getItem("token");
17   const [movies, setMovies] = useState([]);
18   const [user, setUser] = useState(storedUser ? storedUser : null);
19   const [token, setToken] = useState(storedToken ? storedToken : null);
20   const [triggerMe, setTriggerMe] = useState([]);
21   const [searchQuery, setSearchQuery] = useState("");
22
23   useEffect(() => {
24     setTriggerMe(false);
25     if (!token) return; // Don't fetch if there's no user
26     fetch(`https://movie-flix19-efb939257bd3.herokuapp.com/users`, {
27       method: "GET",
28       headers: {
29         Authorization: `Bearer ${token}`,
30         "Content-Type": "application/json",
31       }
32     }).then((response) => {
33       if (!response.ok) {
34         throw new Error('Failed to fetch users');
35       }
36       return response.json();
37     })
38     .then((users) => {
39       // Find and return the user object matching the username
40       const loggedInUser = users.find(u => u.username === user);
41       setUser(loggedInUser);
42       localStorage.setItem("user", JSON.stringify(loggedInUser));
43       if (!loggedInUser) {
44         throw new Error('User not found');
45       }
46     })
47     .catch((error) => {
48       console.error('Error fetching user data:', error);
49       throw error;
50     })
51   })
52 }
```

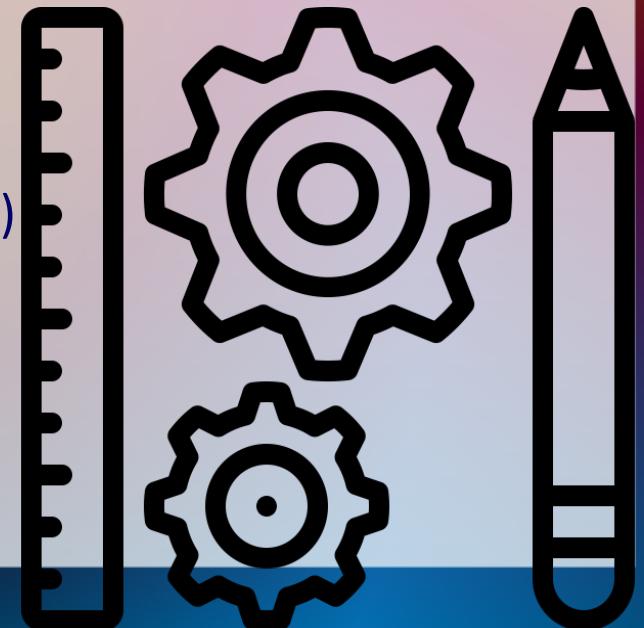
Duration

Developing the client-side took long as much as developing the server-side. Creating backend seems to be more challenging to me because of my first-hand experience in data handling and implementing authentication, on the other hand I found myself enjoying the styling and understanding how React components work together with bootstrap although it consumed a lot of time.



Methodologiest and Tools

- Node.js - JavaScript runtime environment.
- MongoDB - NoSQL database for storing data.
- Mongoose - ODM for MongoDB.
- Postman – for testing and working with API
- Heroku- used for hosting
- Passport - Middleware for authentication.
- JWT - JSON Web Tokens for secure access.
- CORS - Cross-Origin Resource Sharing.
- React - For building the user interface.
- Vite - For bundling and serving the app. (replaces Parcel)
- Sass - For custom styles.
- React Bootstrap- for styling



App Development Process

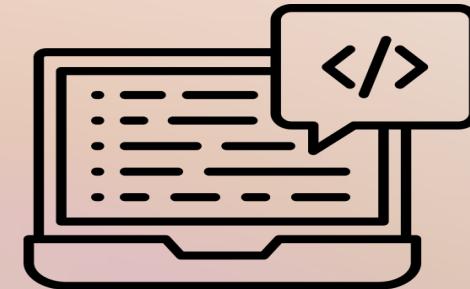
The app was built with the MERN stack and undergo 4 stages that covers backend, frontend, and authentication.

Step 1- Building RESTful Api and utilized by MongoDB, Express.js, React, and Node.js (MERN). I developed the backend routes for creating, reading and deleting data to ensure smooth communication between frontend and backend.

Step 2- Connecting and managing the data base by integrating MongoDB with the backend using Mongoose. Then I handled data operation and ensured accurate storage and retrieval.

Step 3- Frontend Design and Styling. I created responsive UI using the React.js and applied modern styling using Bootstrap as an UI library.

Step 4- Implementing Authentication. I added the user login and registration functionality and secured user data using JWT.



Created by Orange Cat
from Noun Project

BACKEND or SERVER-SIDE

I build a RESTful Api using MERN stack (MongoDB, Express.js, React, Node.js) that interact with database which stores data about different movies. The REST API will be accessed via commonly used HTTP methods like GET and POST. CRUD is being used to retrieve data from the database and store that data in a non-relational way. The API provides movie information in JSON format. To test the API, I used Postman.

The screenshot shows the Postman application interface. A GET request is made to `http://localhost:8080/movies`. The response status is `200 OK`, and the response body is displayed in JSON format:

```
[{"Title": "Twilight", "Discription": "When Bella Swan moves to a small town in the Pacific Northwest, she falls in love with Edward Cullen, a mysterious classmate who reveals himself to be a 108-year-old vampire.", "Director": {"Name": "Catherine Hardwicke", "Birthdate": "October 21, 1955"}, "Genre": {"Name": "Romance", "Description": "The romance genre is a category of literature, film, and other media that focuses on love and romantic relationships between characters as the central theme"}, "ImageUrl": "https://en.wikipedia.org/wiki/Twilight_(2008_film)#/media/File:Twilight_(2008_film)_poster.jpg"}, {"Title": "Princess Diaries"}]
```

```
152 //READ
153 app.get('/users', (req, res) =>{
154   |   res.status(200).json(users);
155   |
156 })
157
158 //CREATE
159 app.post('/users', (req, res) => {
160   |   const newUser = req.body;
161
162   if (newUser.name) {
163     newUser.id = uuid.v4();
164     users.push(newUser);
165     res.status(201).json(newUser);
166   } else {
167     res.status(400).send('require name')
168   }
169 })
170
171 //UPDATE
172 app.put('/users/:id', (req, res) => {
173   |   const { id } = req.params;
174   |   const updatedUser = req.body;
175
176   let user = users.find( user => user.id == id);
177
178   if (user) {
179     user.name = updatedUser.name;
180     res.status(200).json(user);
181   } else {
182     res.status(400).send('id cannot be found!')
183   }
184 })
185 //CREATE
186 app.post('/users/:id/:movieTitle', (req, res) => {
187   |   const { id, movieTitle } = req.params;
188
189   let user = users.find( user => user.id == id);
190
191   if (user) {
192     user.favoriteMovies.push(movieTitle);
193     res.status(200).send(`${movieTitle} has been added to user ${id}'s array`);
194   } else {
195     res.status(400).send('name cannot be found!')
196   }
197 })
198
199 // //DELETE
200 app.delete('/users/:id/:movieTitle', (req, res) => {
```

FRONTEND or CLIENT-SIDE

I built the user interface using React.js and Bootstrap as a UI library for styling and responsiveness. The app is a single-page application, it provides several interface views including a movie view, a registration view, log in view, and a profile view (where users can update their data and list of favorite or deregister themselves).

User Stories

- As a user, I want to be able to access information about movies 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.

Key features:

- ✓ Returns ALL movies to the user (each movie item with an image, title, and description)
- ✓ Filtering the list of movies with a “search” feature
- ✓ Ability to select a movie for more details
- ✓ Ability to log out and allows users to log in with a username and password
- ✓ Returns data (description, genre, director, image) about a single movie to the user
- ✓ Allows new users to register or deregister



Search by title c



Interstellar

Christopher
Nolan[Open](#)[Add to
Favorites](#)The
GodfatherFrancis Ford
Coppola[Open](#)[Add to
Favorites](#)Django
UnchainedQuentin
Tarantino[Open](#)[Add to
Favorites](#)

Avatar

James Cameron

[Open](#)[Remove
from
Favorites](#)

Title: Titanic

Description: A romantic drama set during the ill-fated maiden voyage of the RMS Titanic.

Director: James Cameron

Canadian filmmaker known for epic films that push the boundaries of technology.

Born: 8/16/1954

Genre: Drama

A genre focusing on realistic narratives and emotional themes.

[Back](#)[Remove from Favorites](#)

Main View

Movie view and Movie details

Page Views

PAGE VIEWS

MyFlix Login Signup

Firstname:

Lastname:

Username:

Password:

Email:

Birthday: mm / dd / yyyy

MyFlix Login Signup

Username:

Password:

Login and Signup View

MyFlix

User Profile

Name: lukika

Email: cefgrghr@wefe.com

Email

First Name

Last Name

Password

Username

Favorite Movies:



AVATAR



TITANIC

Profile View