

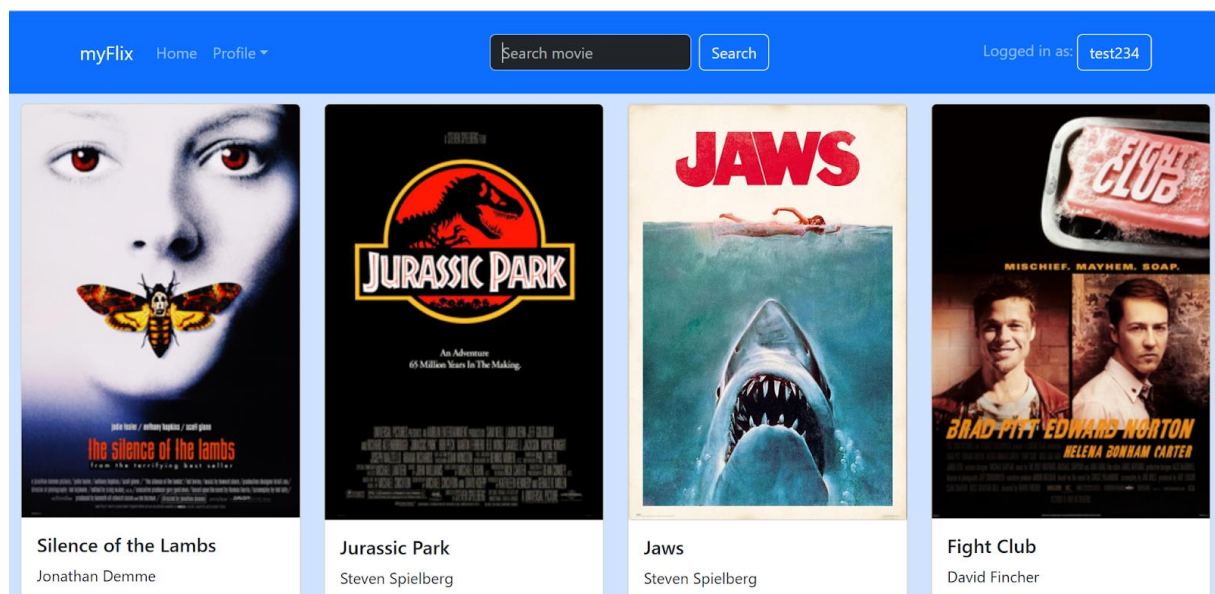
## Case Study:

### myFlix (with React)

#### Project Overview

**myFlix-client** is a React-based web application designed to provide users with a seamless experience for browsing, searching, and viewing detailed information about movies. The app features user authentication, allowing users to create accounts, log in, and manage their profiles. Users can also curate a personalized list of favorite movies.

The front-end is built using React, ensuring a responsive and dynamic interface. The application integrates with a custom RESTful API to fetch and display movie data, and utilizes **React Bootstrap** to create a polished, user-friendly design. The myFlix-client is fully responsive, providing an optimal experience across devices.



#### Purpose & Context:

myFlix was a personal project I built 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.

## Key Features

### 1. User Authentication:

- Users can sign up and log in securely.
- The authentication system is powered by **JWT** for secure token-based access.

### 2. Movie Browsing & Search:

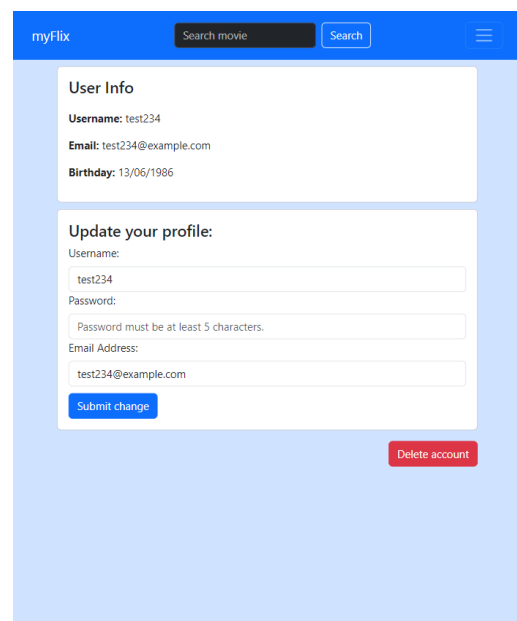
- Users can browse a collection of movies and search by title, genre, or director.
- The data is fetched dynamically from the backend API.

### 3. Profile Management:

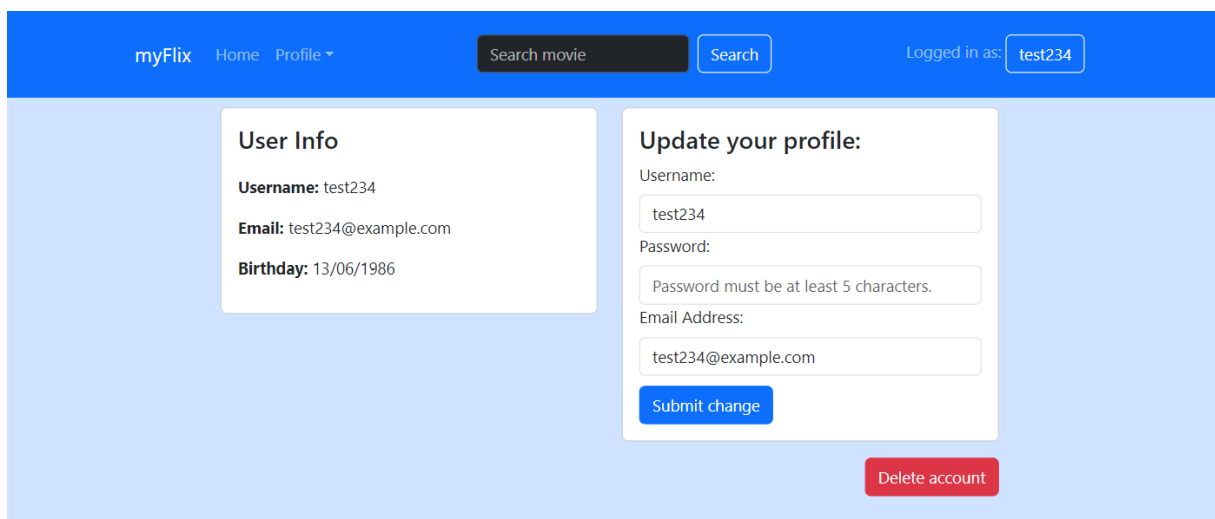
- Users can update their profiles and manage their list of favorite movies.
- Profiles are synced with the database, ensuring data persistence.

### 4. Responsive Design:

- The UI is designed using **React Bootstrap**, ensuring it is visually appealing and fully responsive on both desktop and mobile devices.



The screenshot shows a mobile view of the myFlix profile management page. At the top, there is a blue header with the 'myFlix' logo, a search bar with the placeholder 'Search movie', and a 'Search' button. Below the header, the page is divided into two main sections. The first section, titled 'User Info', displays the user's details: Username: test234, Email: test234@example.com, and Birthday: 13/06/1986. The second section, titled 'Update your profile:', contains input fields for Username (pre-filled with 'test234'), Password (with a note 'Password must be at least 5 characters.'), and Email Address (pre-filled with 'test234@example.com'). A blue 'Submit change' button is located below these fields. In the bottom right corner, there is a red 'Delete account' button.



The screenshot shows a desktop view of the myFlix profile management page. The blue header includes the 'myFlix' logo, navigation links for 'Home' and 'Profile', a search bar with the placeholder 'Search movie', and a 'Search' button. On the right side of the header, it says 'Logged in as: test234'. The main content area is divided into two columns. The left column contains a 'User Info' box with the user's details: Username: test234, Email: test234@example.com, and Birthday: 13/06/1986. The right column contains an 'Update your profile:' box with input fields for Username (pre-filled with 'test234'), Password (with a note 'Password must be at least 5 characters.'), and Email Address (pre-filled with 'test234@example.com'). A blue 'Submit change' button is located below these fields. In the bottom right corner, there is a red 'Delete account' button.

## Used Technologies

- **Front-End:** React, React Router, Redux, React Bootstrap
- **Back-End:** Node.js, Express, MongoDB, Mongoose, REST API, JSON Web Token (JWT) for authentication, CORS for secure access control

## Development Process

Before developing the front-end, I created a comprehensive **RESTful API** using Node.js, Express, and MongoDB. This API serves as the backbone of the myFlix-client, providing endpoints for user registration, authentication, profile management, and movie details. The API includes the following key functionalities:

- Secure user authentication with JWT
- User input validation for robust error handling
- CORS configuration for controlling access from specific origins
- Error handling for graceful degradation in case of failures

The API is deployed on **Heroku**, ensuring reliable hosting with easy scalability.

## Challenges and Solutions

One of the biggest challenges I encountered during the development of this project was learning to work with the **React** framework, particularly understanding how it interacts with the API. Managing state with **Redux** and ensuring smooth communication between the front-end and back-end required a deep dive into asynchronous programming and API integration.

I overcame this challenge by focusing on smaller components of the application first, building simple API calls, and progressively adding complexity as I grew more comfortable with the framework. Debugging tools and detailed documentation were invaluable in this process, and once I understood the flow of data between the client and server, the development became more streamlined.

## Timeframe

The project took me around **two months** to complete, from designing and developing the API to building the user interface and deploying the application.

## Conclusion

myFlix-client demonstrates my ability to build a full-stack web application with modern technologies such as **React** and **Express**. By overcoming challenges like mastering a new framework and integrating a dynamic back-end, this project has significantly strengthened my skills in front-end development, API creation, and user experience design.

## Credits

Role: Lead Developer

Tutor: Timothy Nyabongo

Mentor: Renish Bhaskaran