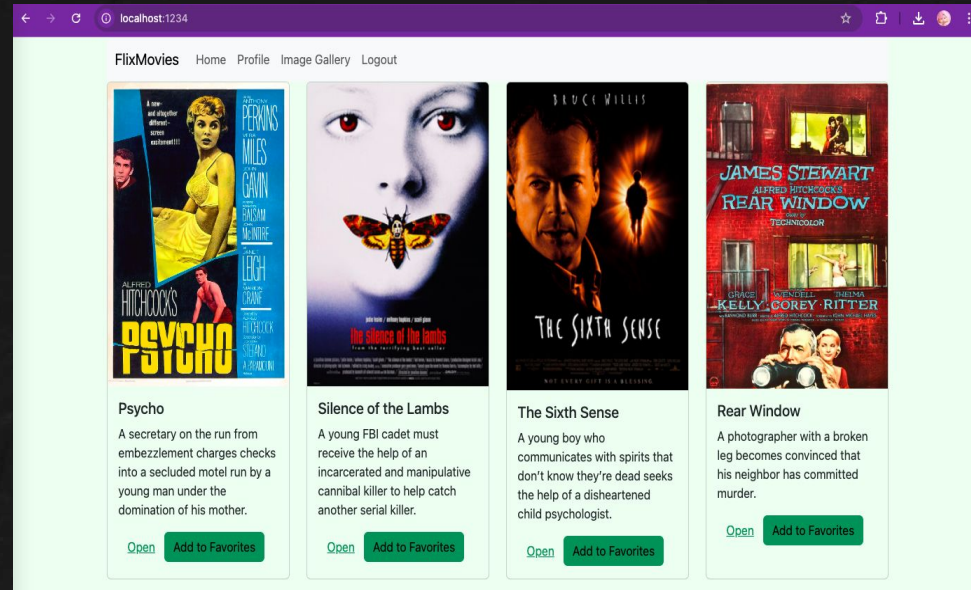


The logo for 'my FLIX' is displayed. The word 'my' is in a white, lowercase, sans-serif font. The word 'FLIX' is in a large, bold, red, uppercase, sans-serif font. The background is dark and textured, featuring a faint image of a film reel and some abstract shapes.

# Overview

MyFlix is a web app, developed using the MERN stack (MongoDB, Express, React, and Node.js) that provides users with access to information about movies, directors and genres. Users are able to create an account, update their personal data and create list of Favourite movies.

## MY FLIX



The background is a dark, textured surface resembling crumpled paper. It is framed by decorative film strip borders: a vertical cyan strip on the left, a horizontal cyan strip at the bottom, and two diagonal strips (one cyan, one magenta) crossing in the upper right corner.

# Purpose

This project was created as part of my Full-stack development course at CareerFoundry to demonstrate 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.



## 01 Role

Lead Developer : Bhavani Penugonda

## 02 Project Scale

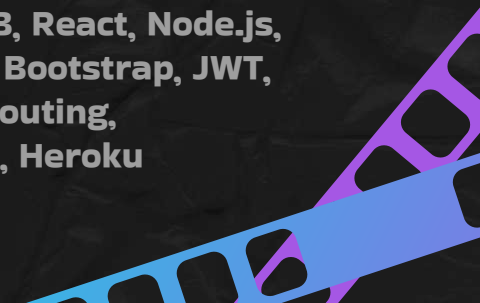
2 Months Project

## 03 Credits

Lead Developer: Bhavani Penugonda  
Tutor: Zaheer Abbas  
Mentor: Sonia Rose Mary Karungi

## 04 Tools/Methodologies

MongoDB, React, Node.js,  
Express, Bootstrap, JWT,  
Redux, Routing,  
Postman, Heroku



## Server-side and Client-side Development

- **Server-side development, also known as backend development, involves creating and managing the code and logic that runs on a server, rather than directly on the user's device (client). This type of development handles tasks like processing user requests, interacting with databases, and generating dynamic content for web applications.**
- **Client-side development, also known as front-end development, involves creating the part of a web application or website that users directly interact with, running on the user's device. This includes designing the visual appearance (HTML, CSS), handling user interactions (JavaScript), and validating forms.**



# Server-side Development

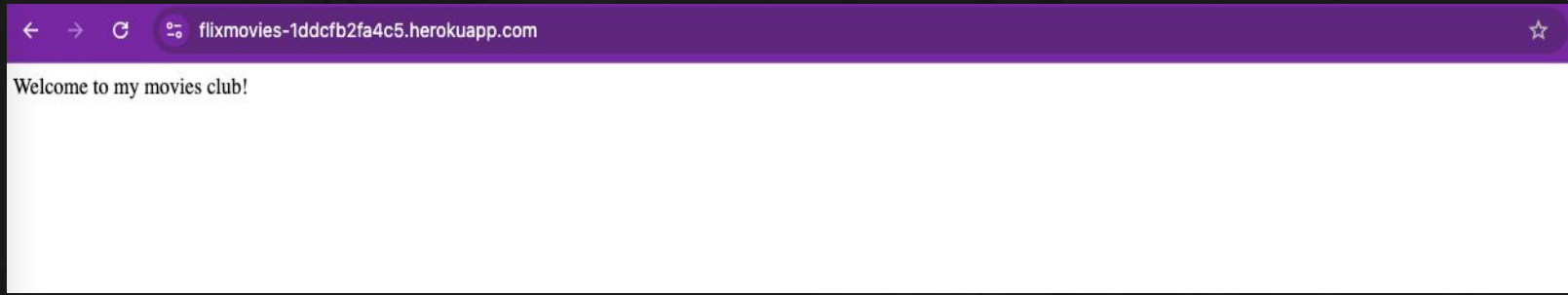
I was tasked to build a **REST API** that serves movie data and allows user authentication and CRUD operations on movies and user profiles.

[CODE](#)

[WEBSITE](#)

## Steps I Took:

- I have set up a **NoSQL database** to store user information and movie data. I used **MongoDB** due to its flexibility with data storage and schema. I designed a scalable database model for users and movies with fields like title, description, genre, and director.
- Next, I have created a functional and secure API that could handle movie listing, user registration, and login. I used **Express.js** and **Node.js** to set up routes for retrieving movie information, user registration, and authentication.



- Users to register, log in, and have their data stored securely, I used **JWT** for creating secure tokens and **bcryptjs** to hash passwords before storing them in the database.
- I tested all the API endpoints using **Postman** and ensured everything was functioning properly.
- Finally, I hosted the MongoDB database on **MongoDB Atlas** for reliable cloud storage and deployed the server-side application to **Heroku** to make it accessible online.



# Client-side Development

I have to build a **single-page application (SPA)** that communicates with the existing **server-side API**, presenting movie data to the users and allowing them to manage their profile and favorites.

[WEBSITE](#)

[CODE](#)

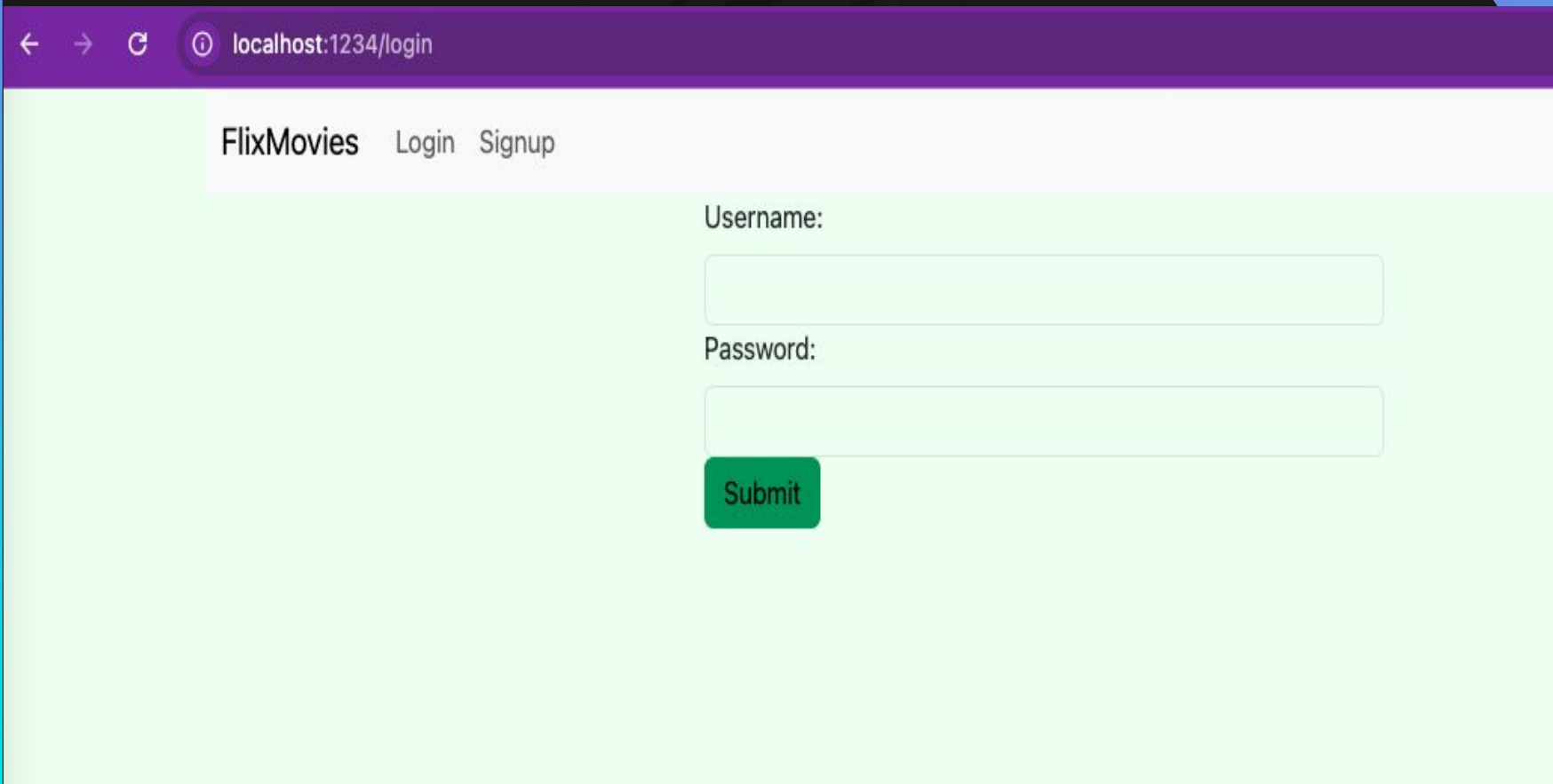
## Steps I Took

- I used React to build a fast, interactive single-page app because its component-based architecture made the code easier to maintain and reuse.
- I connected the frontend to the server-side API I built earlier. So that users could register, log in, and get movie data directly from the backend.

## ..steps

- I created views such as Main View, Movie View, Signup View, Login View, Profile View and integrated **React Router** for view management (e.g., Main View, Profile View).
- Added a feature to Profile view where users can add/remove movies from their profile .
- For styling the UI, I used **React Bootstrap** to design cards and structure the layout of page views.
- I integrated **Redux** to handle shared data across components (like user info and favorite movies), I integrated **Redux**. This made the app's state easier to manage and kept data consistent between different views.
- Finally deployed the client-side app online to **Netlify** so it's accessible from any device.

# Login view



The screenshot shows a web browser window with a purple address bar displaying "localhost:1234/login". The page has a light green background. At the top left, there is a navigation bar with the text "FlixMovies" and two links, "Login" and "Signup". On the right side of the page, there is a login form with two input fields: "Username:" and "Password:". Below these fields is a green "Submit" button.

← → ↻ ⓘ localhost:1234/login

FlixMovies Login Signup

Username:

Password:

Submit

# signup view

← → ↻ ⓘ localhost:1234/signup ☆ 📁 ⬇ 🌐 ⋮

FlixMovies Login Signup

Username:

Password:

Email:

Birthday:

 📅

Submit

# Main View



localhost:1234



FlixMovies [Home](#) [Profile](#) [Image Gallery](#) [Logout](#)



## Psycho

A secretary on the run from embezzlement charges checks into a secluded motel run by a young man under the domination of his mother.

[Open](#)

[Add to Favorites](#)



## Silence of the Lambs

A young FBI cadet must receive the help of an incarcerated and manipulative cannibal killer to help catch another serial killer.

[Open](#)

[Add to Favorites](#)

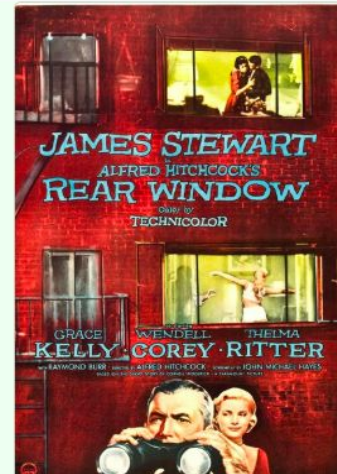


## The Sixth Sense

A young boy who communicates with spirits that don't know they're dead seeks the help of a disheartened child psychologist.

[Open](#)

[Add to Favorites](#)



## Rear Window

A photographer with a broken leg becomes convinced that his neighbor has committed murder.

[Open](#)

[Add to Favorites](#)

# Profile View

FlixMovies [Home](#) [Profile](#) [Image Gallery](#) [Logout](#)

## Your Profile Information

**Username:** chinni

**Email:** bhavani02@gmail.com

**Birthday:** Tue Feb 04 1992

## Update Profile

Username:

Password:

Email:

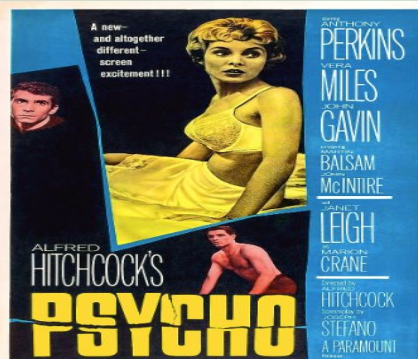
Birthday:



[Update Profile](#)

[Delete Account](#)

## Your Favorite Movies



### Psycho

A secretary on the run from embezzlement charges checks into a secluded motel run by a young man under the domination of his mother.

[Open](#)

[Remove from Favorites](#)

## The Most Challenging Parts

- I found it challenging to securely manage login, signup, and private routes, ensuring users could only access their data.
- Setting up the database on MongoDB Atlas, deploying the server on Heroku, and connecting both to the React client was complex and taught me a lot about production environments.
- Ensuring API communication works for all CRUD operations (create, read, update, delete)
- Integrating user authentication (login, signup, protected routes) securely
- Connecting the client to the REST API and handling API errors
- Debugging CORS issues and configuring environment variables
- Making sure the token was correctly included in the **Authorization header** of every protected API request so that users could access their data reliably.

# Retrospective

I successfully built a full-stack movie app, **MyFlix**, with both server-side and an interactive, user-friendly client-side. The final product allows users to browse movies, manage favorites, and personalize their profiles all within a smooth, responsive interface.

The original goal was to create an app that makes it easy for users to explore and save movie information I achieved that. The most challenging part was managing state across multiple components. Adding **Redux** helped simplify this and made the app more scalable. Deploying both the server and client online taught me valuable lessons about **deployment workflows** and **cloud databases**. Building a **MERN** stack app strengthened my full-stack JavaScript skills. I improved my understanding of **RESTful API integration**, **React architecture**, and **state management**.

Moving forward, I would like to add more features such as **"To Watch" List** which gives users the ability to maintain a separate list of movies they plan to watch later and so on.