myFlix

A Full Stack Case Study

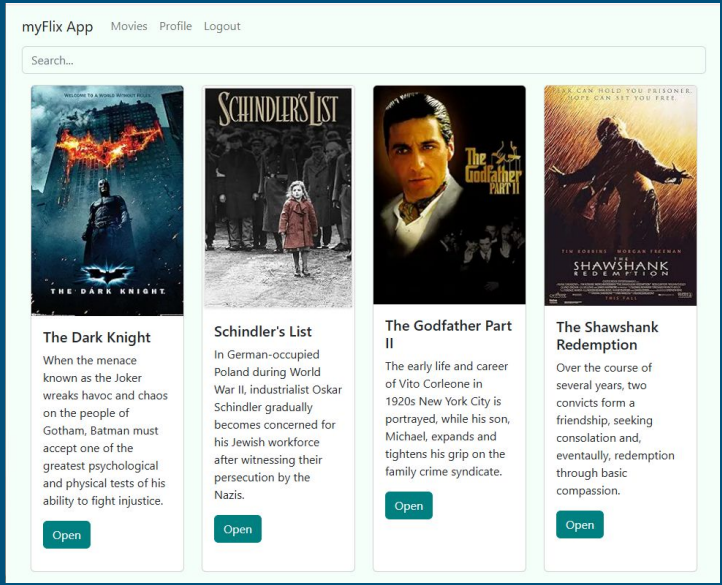
Stephanie Leon

Overview

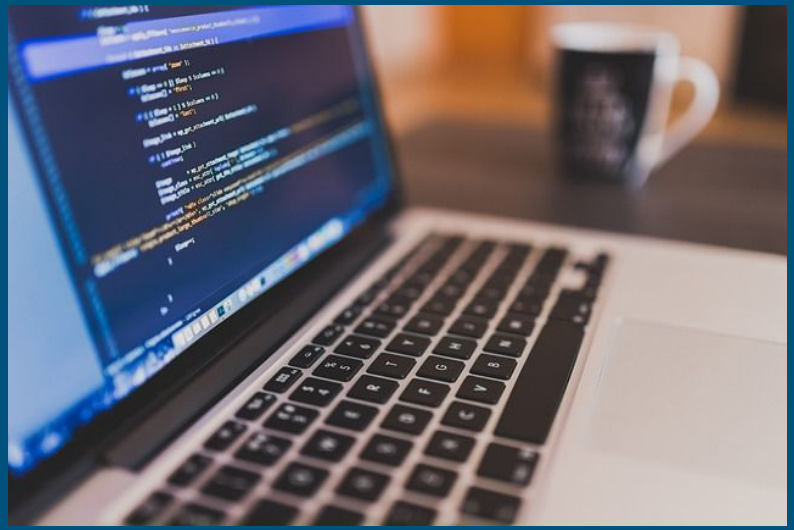
myFlix is a web application designed using the MERN stack.

Through the app, users are able to access information about the film, the director and the genre.

Users can create an account, update their personal information and a save a list of favorite movies.



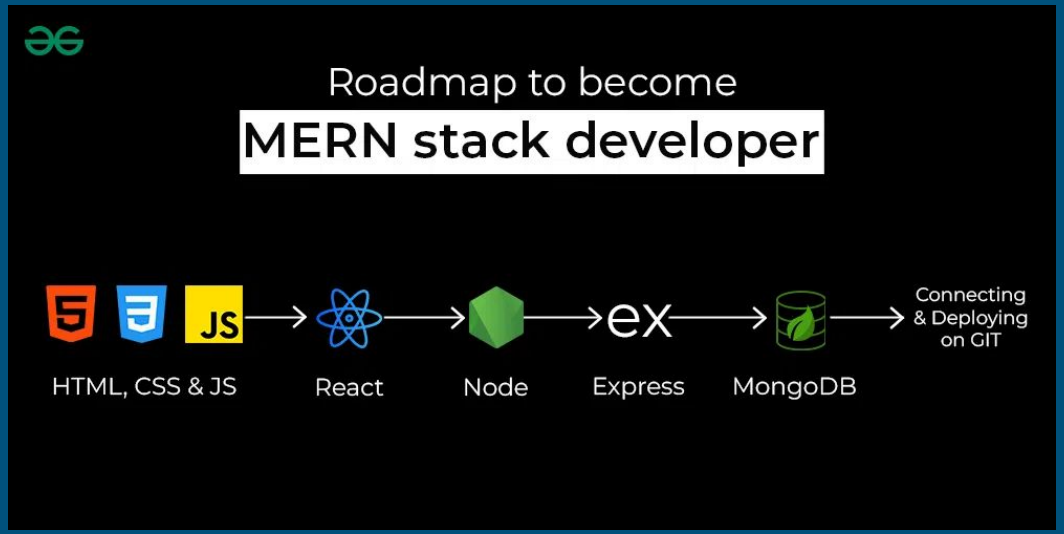
Purpose & Context



myFlix was created as part of a full stack web development course I completed through CareerFoundry. This was my first opportunity to create a full stack application.

Objective

The aim of this project was to build a complete app with a server-side and client-side using a MERN stack that can be added to a professional portfolio.



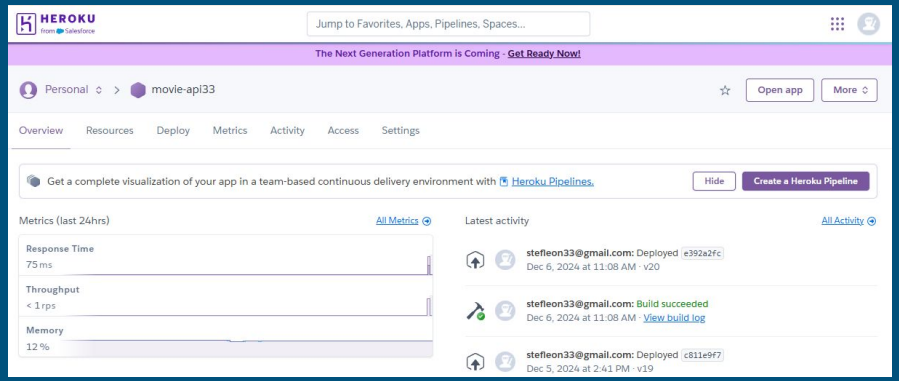
Designing the Solution: Server-Side

Objective: Develop a RESTful API to support a movie application's data operations.

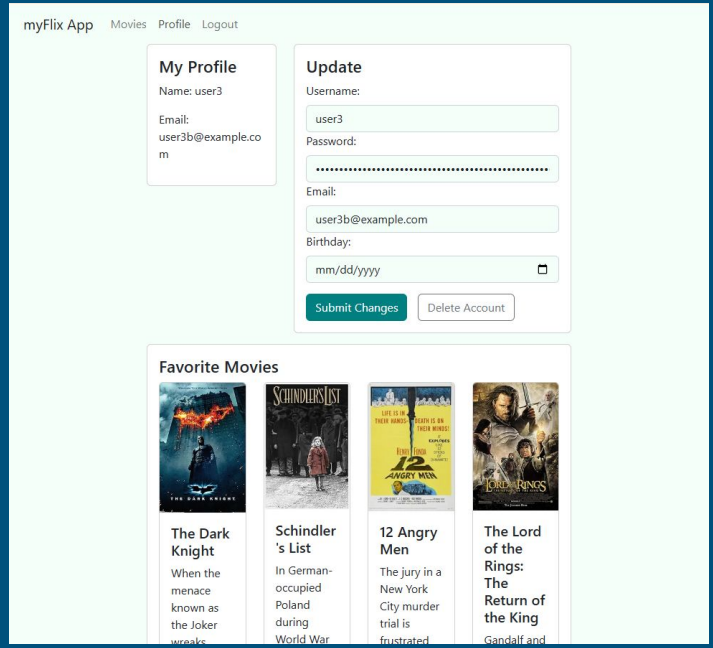
Technologies Used: Node.js, Express, MongoDB, Mongoose

Core Features:

- CRUD operations for movies, genres, and directors.
- User authentication with JWT and data security protocols.
- Data validation and error handling.
- Deployment to Heroku for public accessibility



Designing the Solution: Client-Side



Objective: Create a single-page React application as the client-side of myFlix.

Technologies Used: React, React Router, Bootstrap, Parcel and Redux

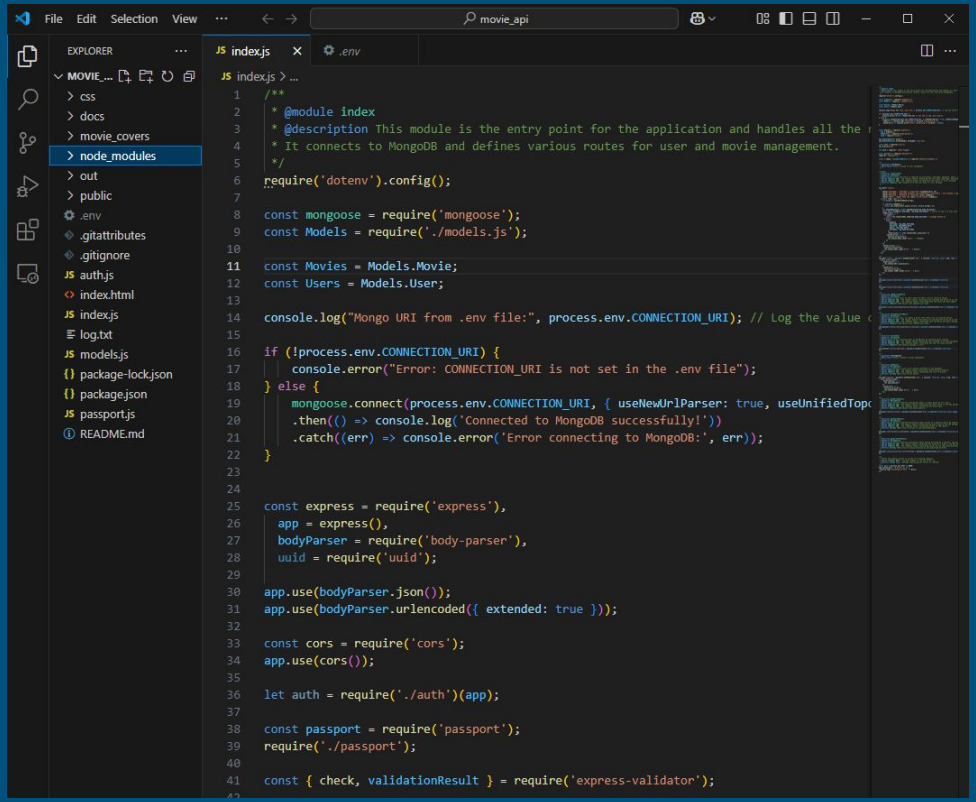
Core Features:

- Interactive views: Login, Signup, Main, Movie, Profile
- Responsive design for seamless use across devices
- Integration with the backend API for real-time data handling

Development Process: Server-Side

Backend Implementation:

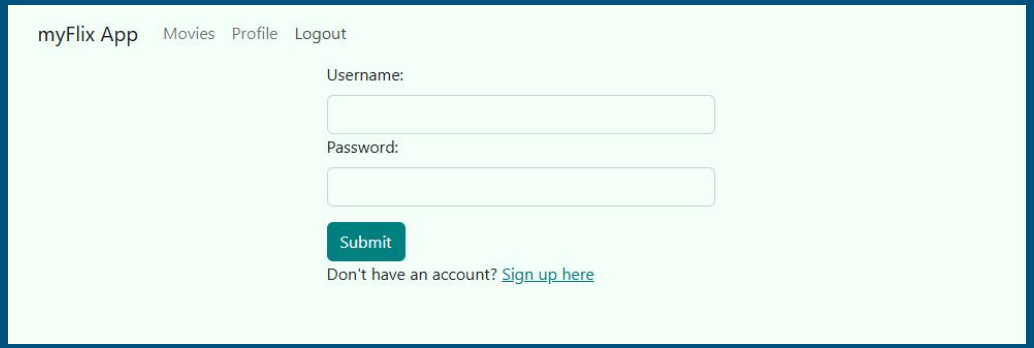
- Set up the Node.js and Express server and defined RESTful endpoints
- Connected the backend to a MongoDB database using Mongoose for data modeling
- Integrated JWT-based user authentication and authorization
- Validated and secured data with middleware like body-parser and bcrypt
- Deployed the backend to Heroku



Development Process: Client-Side

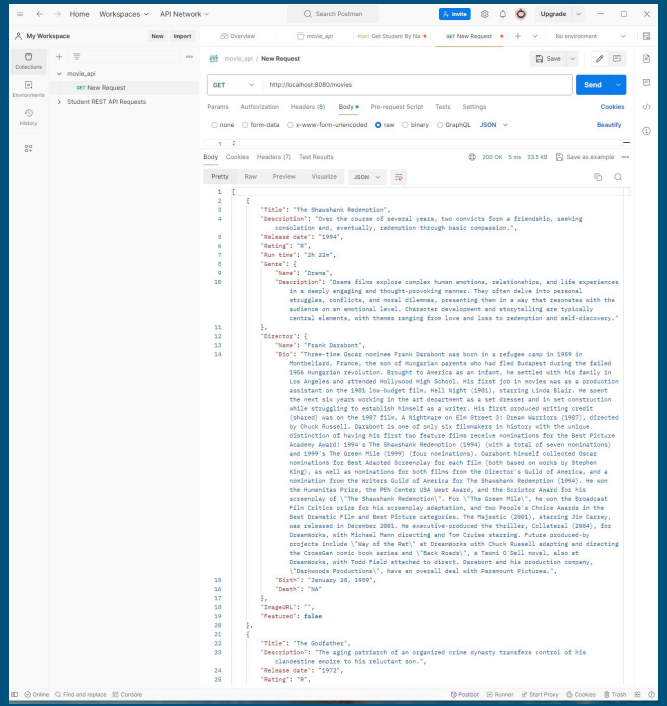
Frontend Implementation:

- Structured the React application with distinct components for each view
- Implemented routing for seamless navigation
- Integrated the React app with the REST API to handle CRUD operations
- Deployed the frontend online for public access



Testing and Iteration

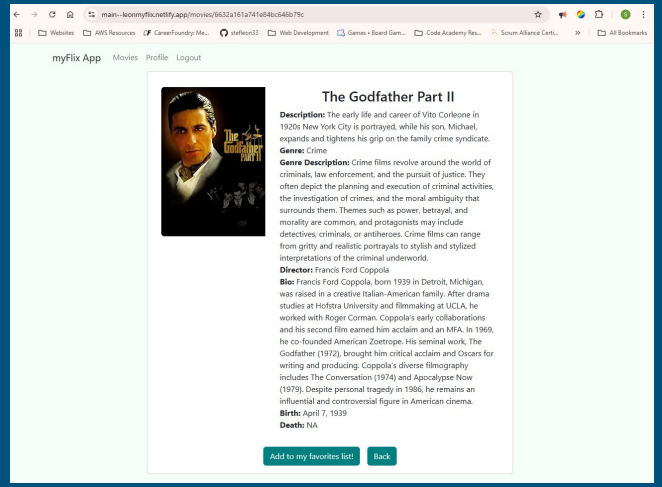
- Conducted rigours API testing using Postman to ensure the backend was error-free
- Tested the client-side views for user experience, responsiveness, and functionality
- Gathered feedback to refine both server-side and client-side implementations



Final Solution

The completed myFlix application provides:

- A robust backend supporting movie data and user operations
- A polished, responsive, and user-friendly front end allowing users to:
  - Browse and search for movies
  - View detailed movie, genre, and director information
  - Manage user profiles and favorite movie lists



Lessons Learned

Backend:

- Stick to the exercise brief:
  - I spent a lot of time setting up the movie data in second exercise when it wasn't necessary, then had to set the data up in the proper format later in the achievement.
- Pay attention to the details when coding
  - the placement of quotes, the letters that are capitalized, the exact name of variables

Frontend:

- If the frontend code should be working, and isn't, take a look at the backend code.

Future Iterations

- Additional features like actor information, movie ratings, “To Watch” lists
- Improved scalability and performance through serverless architecture

Duration:

I completed the server-side in 2 months and the client-side in a month and a half.

Credits:

Lead Developer: Stephanie Leon
Tutor: Jesus Diaz
Mentor: Shreyansh Kumar