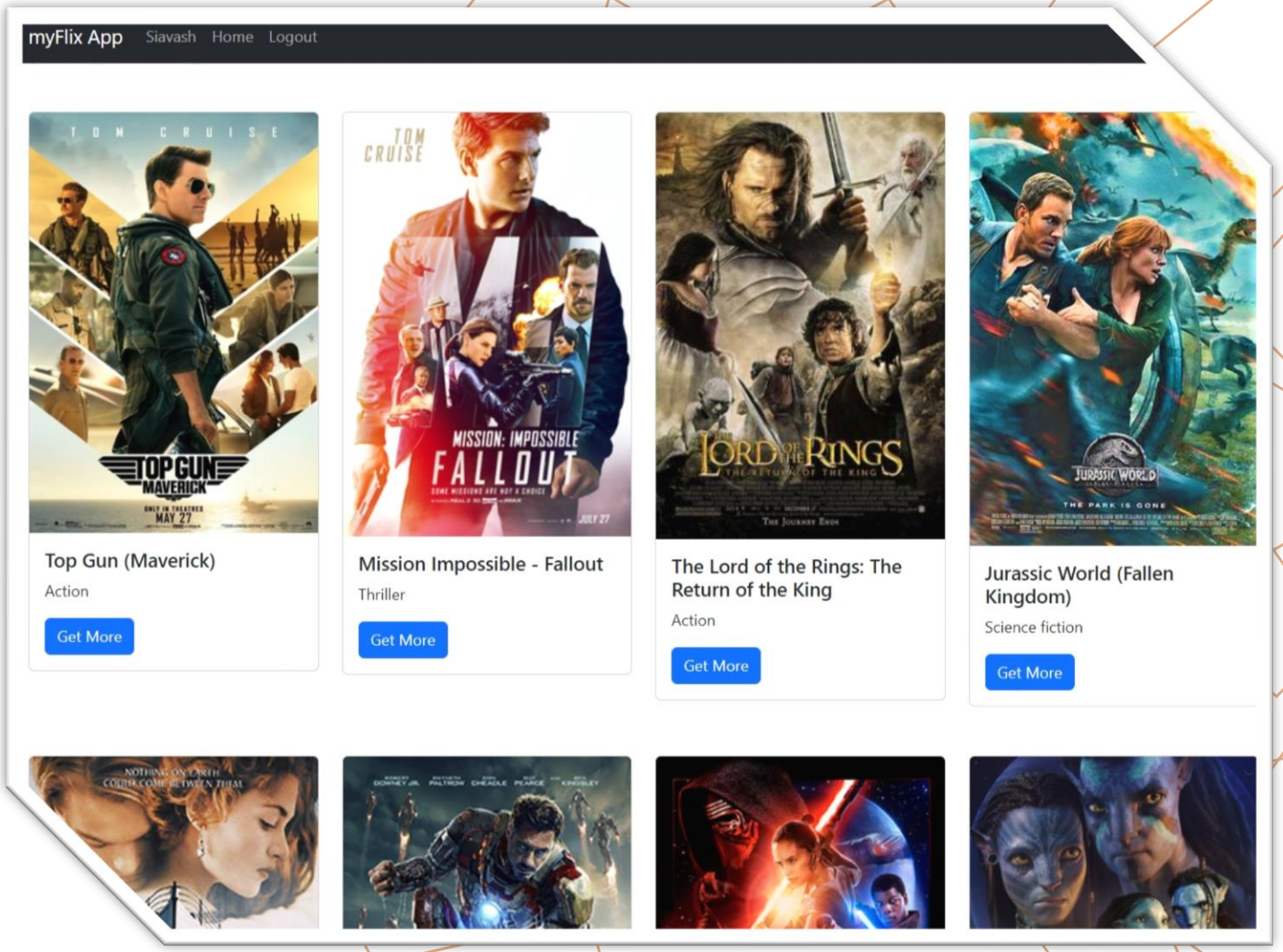
myFlix

A FULL STACK CASE STUDY

Siavash Ebrahimi Darsinouei

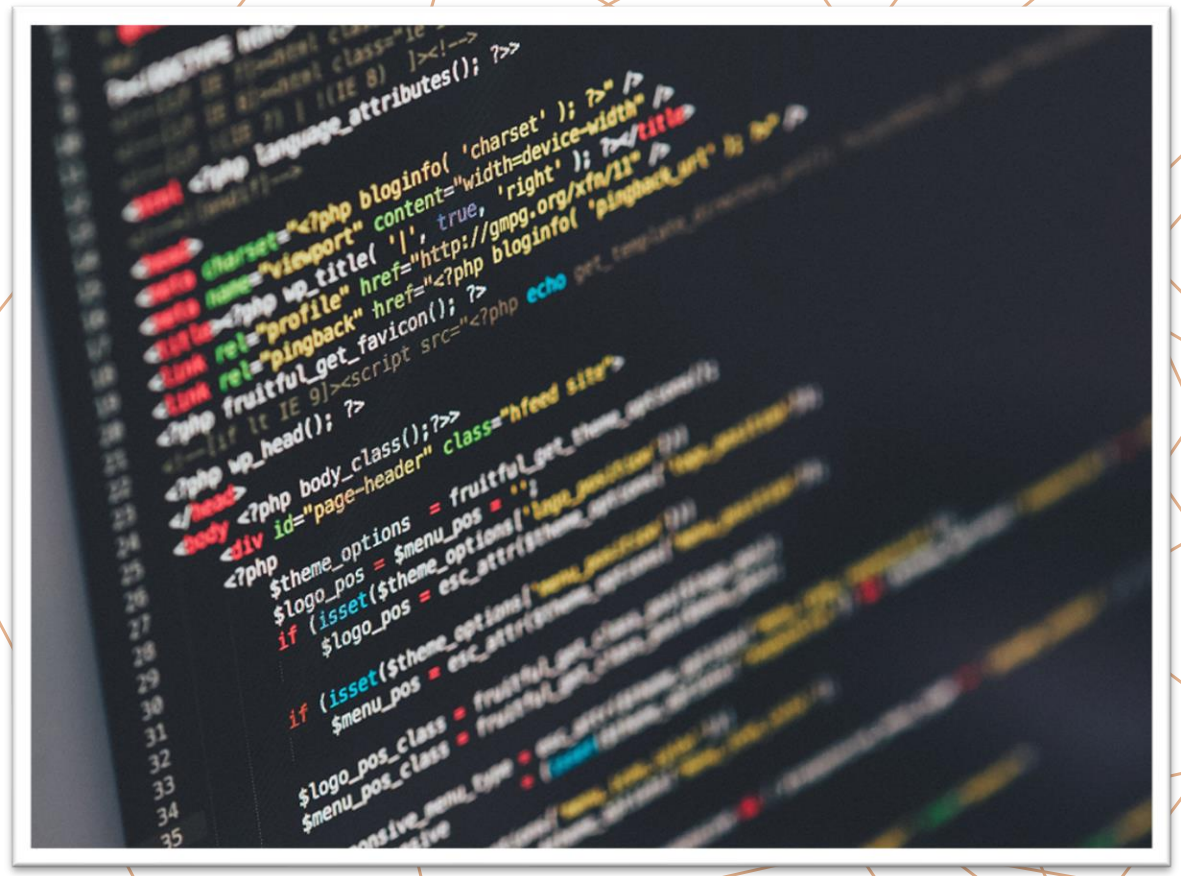
OVERVIEW

The myFlix application is a full stack web application that provides users with a platform to discover and manage their favorite movies. The server-side of the application includes a well-designed REST API and an architected database built using JavaScript, Node.js, Express, and MongoDB. The client-side utilizes JavaScript, React, React Bootstrap, React Redux, and Parcel.



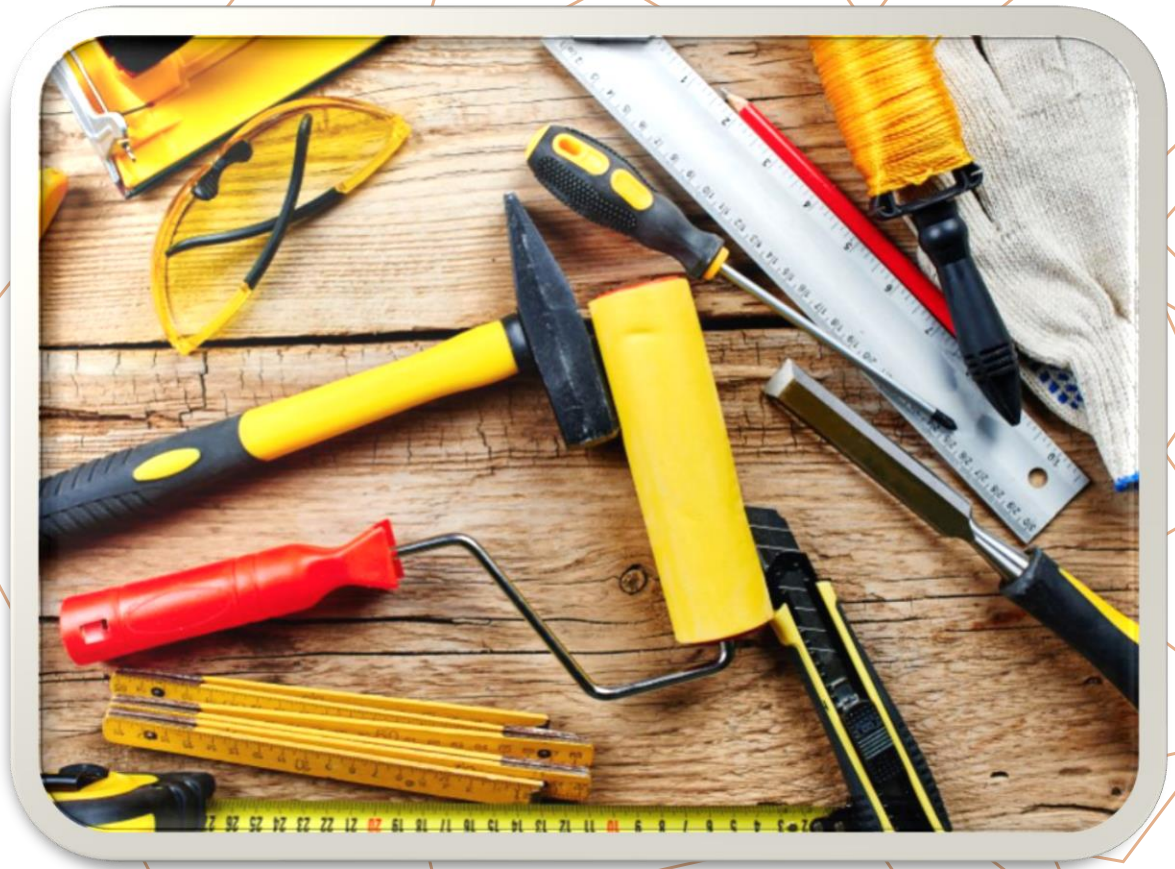
OBJECTIVE

The objective of this project is to create a user-friendly and feature-rich movie application. The server-side aims to handle data management, authentication, and user interactions through a REST API. The client-side focuses on providing an intuitive and visually appealing interface for users to explore and interact with movies.



DURATION

The development of the myFlix application is estimated to have taken two months, including planning, development, testing, and deployment phases.



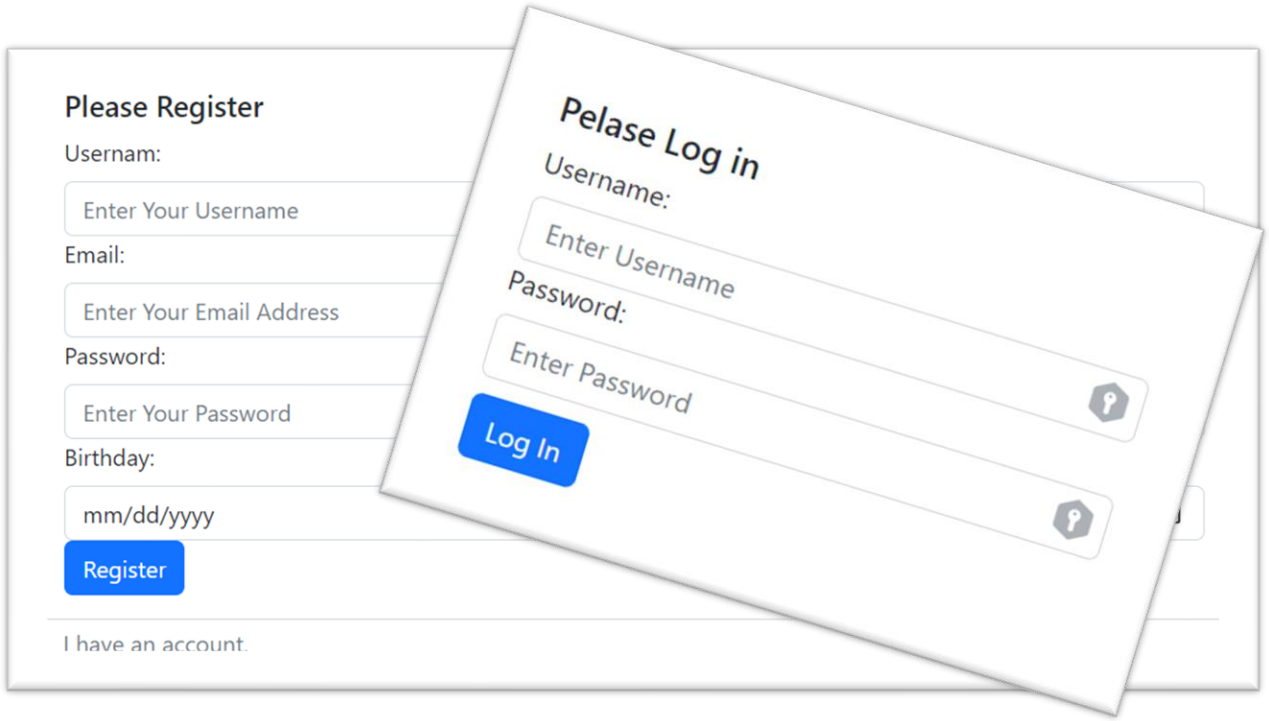
TOOLS, SKILLS, METHODOLOGIES

The project required proficiency in JavaScript, Node.js, Express, MongoDB for the server-side development. On the client-side, skills in React, React Bootstrap, React Redux, and Parcel were utilized. The development process followed an Agile methodology, enabling iterative development and continuous feedback.

SERVER SIDE & CLIENT SIDE

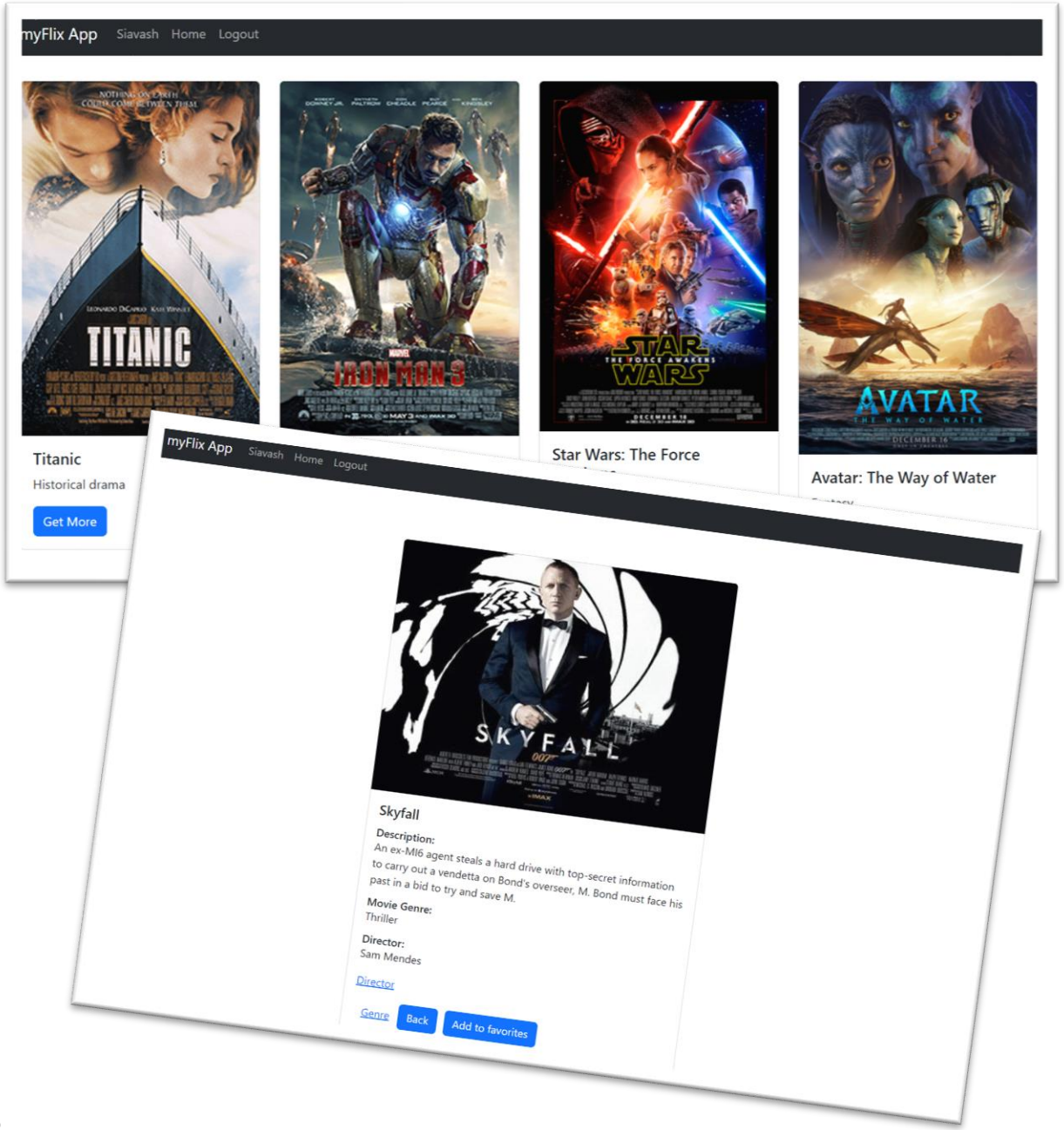
The development of the server-side and client-side was crucial to ensure a functional and cohesive application. The server-side provides the necessary APIs to handle data retrieval, user management, and movie interactions, while the client-side delivers an engaging user experience.

The goal of this part was to implement the required features for both the server-side and client-side. The server-side development impacted the client-side as it needed to consume the provided APIs to display and manage movie data.



The server-side was developed using JavaScript, Node.js, Express, and MongoDB. The client-side utilized JavaScript, React, React Bootstrap, React Redux, and Parcel as the key technologies. Various development tools like code editors, version control systems, and testing frameworks were used.

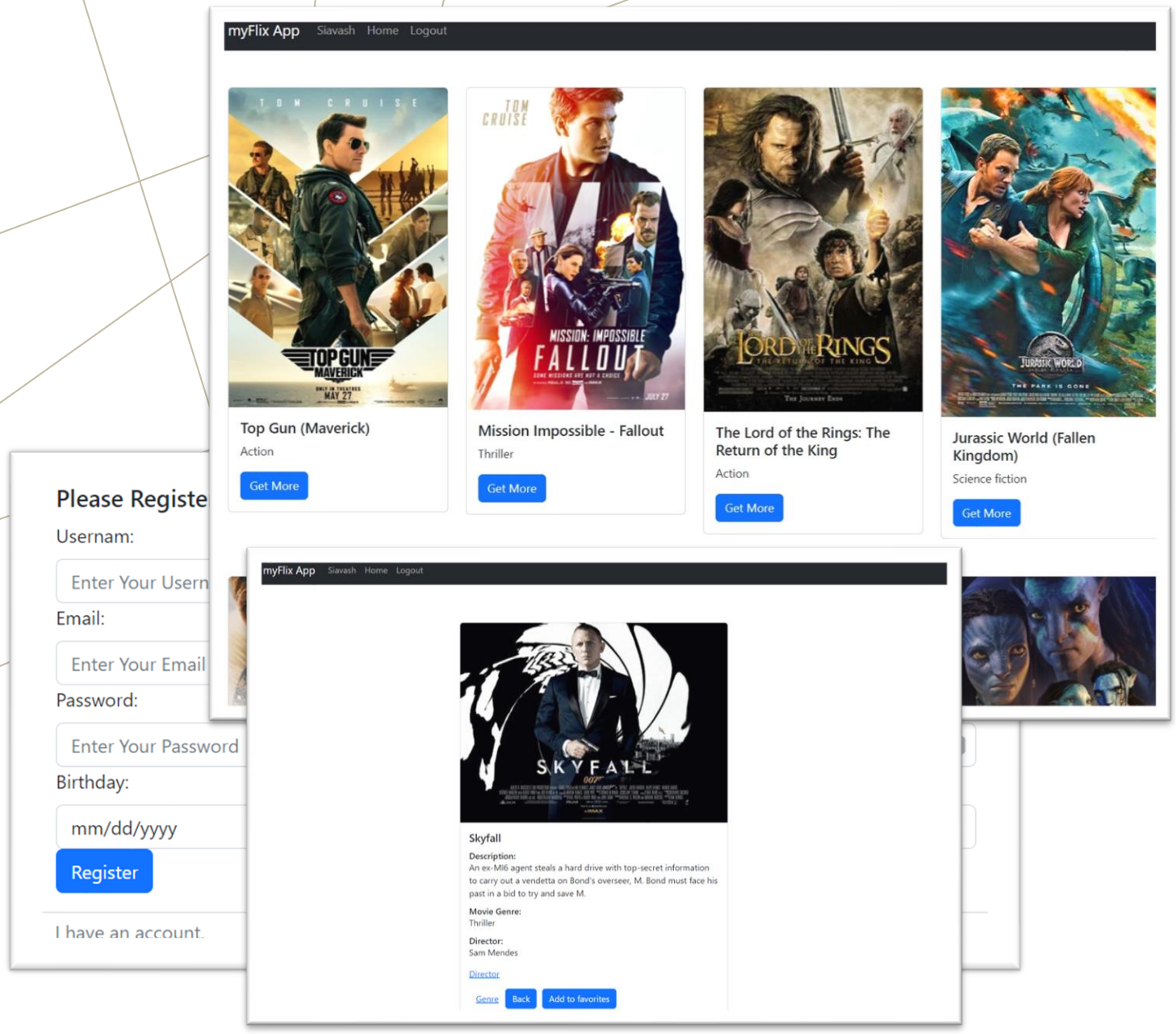
The server-side development progressed smoothly, and the REST API was successfully implemented, providing the necessary endpoints for data retrieval and user management. The client-side development also went well, with the main view, single movie view, login view, registration view, genre view, and director view being implemented as planned. However, due to time constraints same advanced features and additional testing coverage were not fully realized.



One challenge encountered was handling authentication and authorization on the server-side. To overcome this, JSON Web Tokens (JWT) were utilized for user authentication, and access control checks were implemented for protected routes. Additionally, thorough testing and validation processes were employed to ensure data integrity and security.

Several decisions were made during the development stage, such as choosing Node.js and Express for server-side development to leverage JavaScript expertise and react for client-side development due to its component-based architecture and reusability. The decision to use React Bootstrap and React Redux was made to expedite UI development and manage state effectively.

7



The original objective was to create a user-friendly movie application with essential features, and the final results aligned with this objective. The **myFlix** application successfully allows users to browse and manage movies, register, and log in, add movies to favorites, and explore genres and directors. The server-side REST API and the client-side interface achieved the desired functionality and user experience.

FINAL THOUGHTS

The most challenging part of the project was implementing the authentication and authorization.