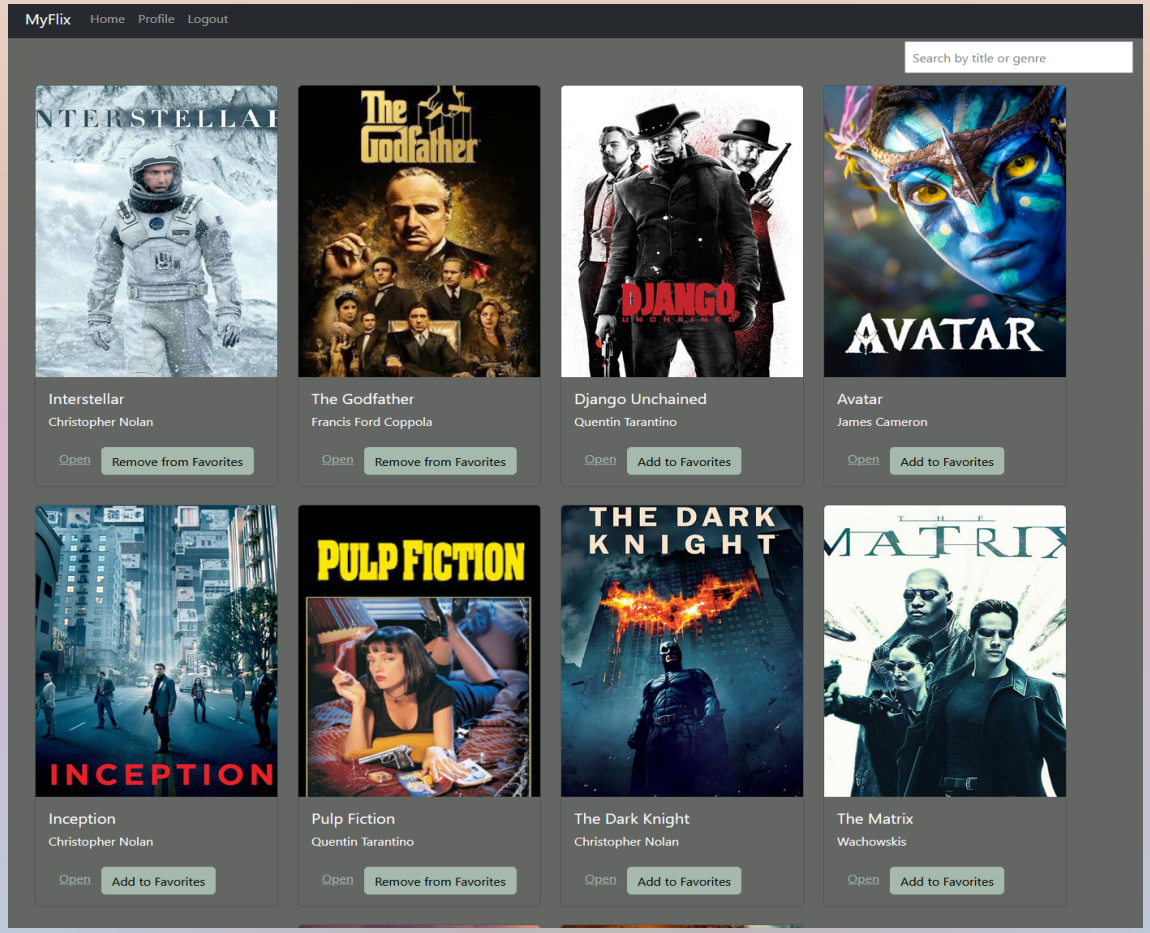
MyFlix

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 accound 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).

```
import { MovieCard } from "../movie-card/movie-card";
import { MovieView } from "../movie-view/movie-view";
import { LoginView } from "../login-view/login-view";
import { SignupView } from "../signup-view/signup-view";
import { NavigationBar } from "../navigation-bar/navigation-bar";
import { UserProfile } from "../profile-view/profile-view";
import Container from 'react-bootstrap/Container';
import Row from "react-bootstrap/Row";
import Col from 'react-bootstrap/Col';
import { BrowserRouter, Routes, Route, Navigate } from "react-router-dom

export const MainView = () => {
    const storedUser = JSON.parse(localStorage.getItem("user"));
    const storedToken = localStorage.getItem("token");
    const [movies, setMovies] = useState([]);
    const [user, setUser] = useState(storedUser ? storedUser : null);
    const [token, setToken] = useState(storedToken ? storedToken : null)
    const [triggerMe, setTriggerMe] = useState([]);
    const [searchQuery, setSearchQuery] = useState("");

    useEffect(() => {
        setTriggerMe(false);
        if (!token) return; // Don't fetch if there's no user
        fetch(`https://movie-flix19-efb939257bd3.herokuapp.com/users`, {
            method: "GET",
            headers: {
                Authorization: `Bearer ${token}`,
                "Content-Type": "application/json",
            }
        }).then((response) => {
            if (!response.ok) {
                throw new Error('Failed to fetch users');
            }
            return response.json();
        })
            .then((users) => {
                // Find and return the user object matching the username
                const loggedInUser = users.find(u => u.username === user
                setUser(loggedInUser);
                localStorage.setItem("user", JSON.stringify(loggedInUser
                if (!loggedInUser) {
                    throw new Error('User not found');
                }
            })
            .catch((error) => {
                console.error('Error fetching user data:', error);
                throw error;
```

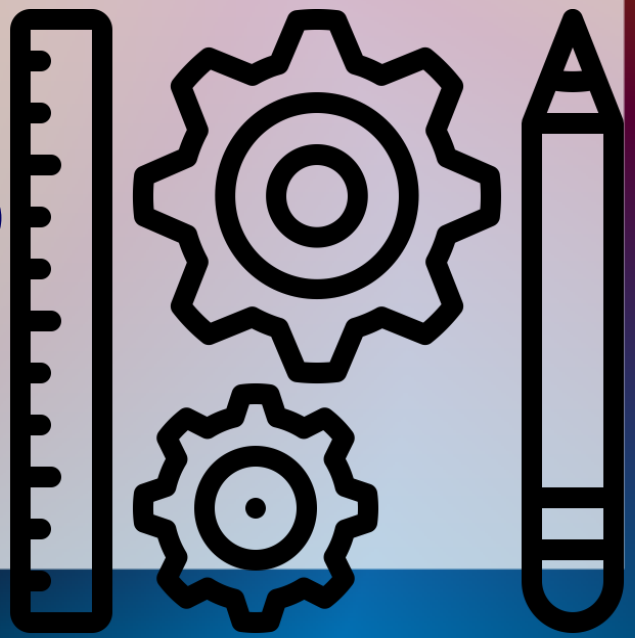
Duration

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 emplementating authentication, on the other hand I found myself enjoing the styling and understanding how React components work together with boostrap although it consumed a lot of time.

Created by Sam Park
from Noun Project

Methodologiest and Tools

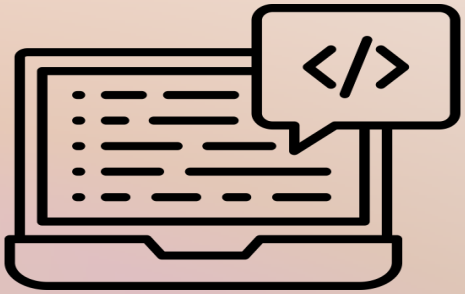
- Node.js - JavaScript runtime environment.
- MongoDB - NoSQL database for storing data.
- Mongoose - ODM for MongoDB.
- Postman – for testing a d working with API
- Heroku- used for hosting
- Passport - Middleware for authentication.
- JWT - JSON Web Tokens for secure access.
- CORS - Cross-Origin Resource 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 Delopment 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 beetwen frontend and backend.



Created by Orange Cat
from Noun Project

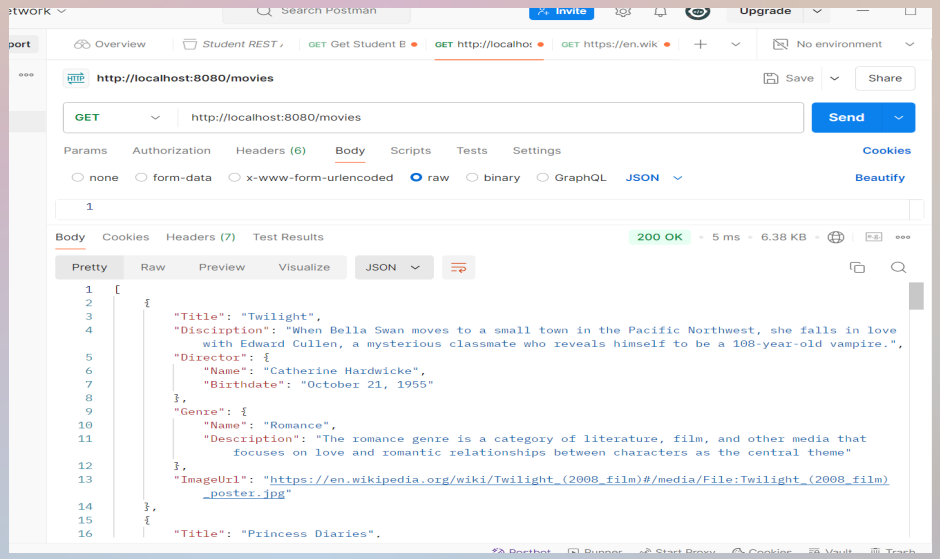
Step 2- Connecting and managing the data base by integrating MongoDB with the backend using Mongoose. Then I handled data opearaton and ensured accurate storage and retrival.

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

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

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-relationa way. The API provides movie information in JSON format. To test the API, I used Postman.




```
152
153  //READ
154  app.get('/users', (req, res) =>{
155      res.status(200).json(users);
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 CLIEND-SIDE

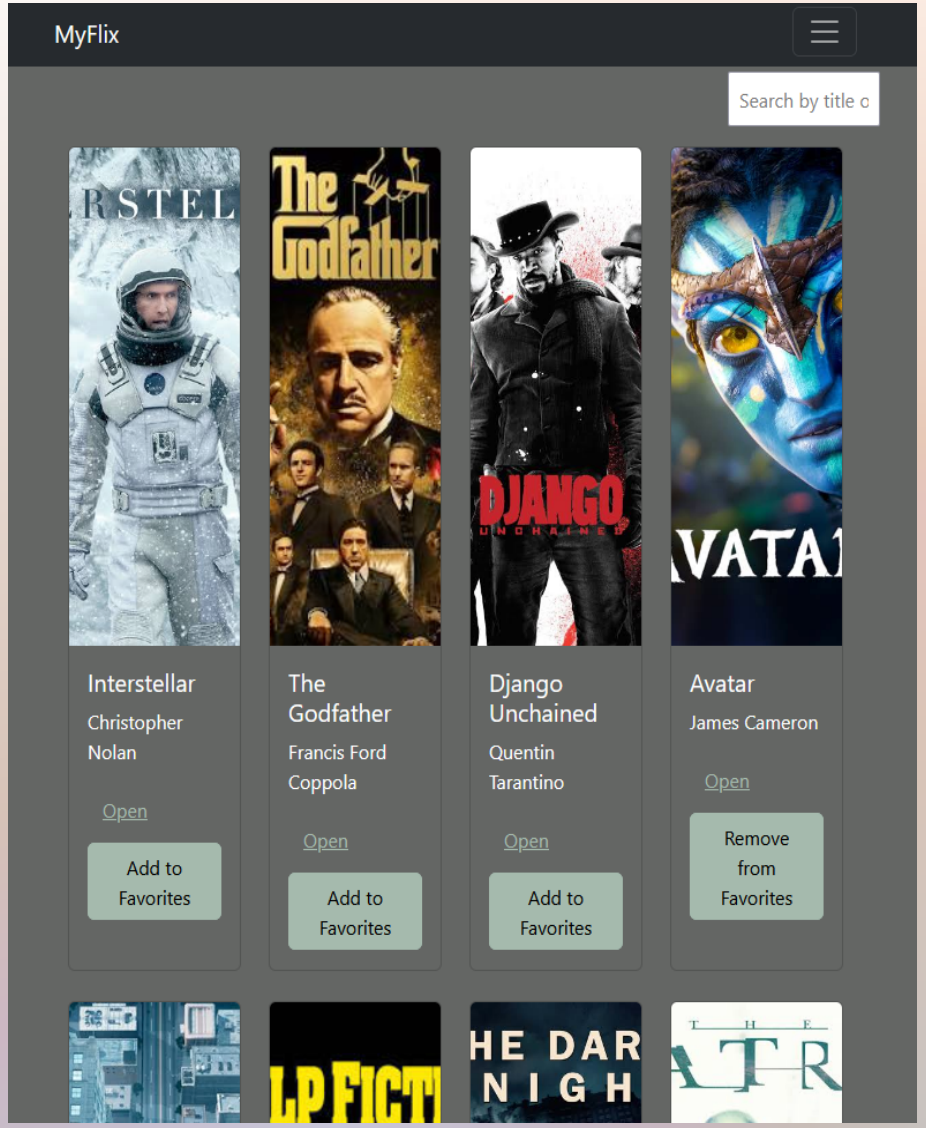
I built the user interface using React.js and Bootstrap as a UI library for styling and responsiveness. The app is a single-page application, it provides several interface views including a movie view, a registration view, log in view, and a profile view (where users can update their data and list of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

User Stories

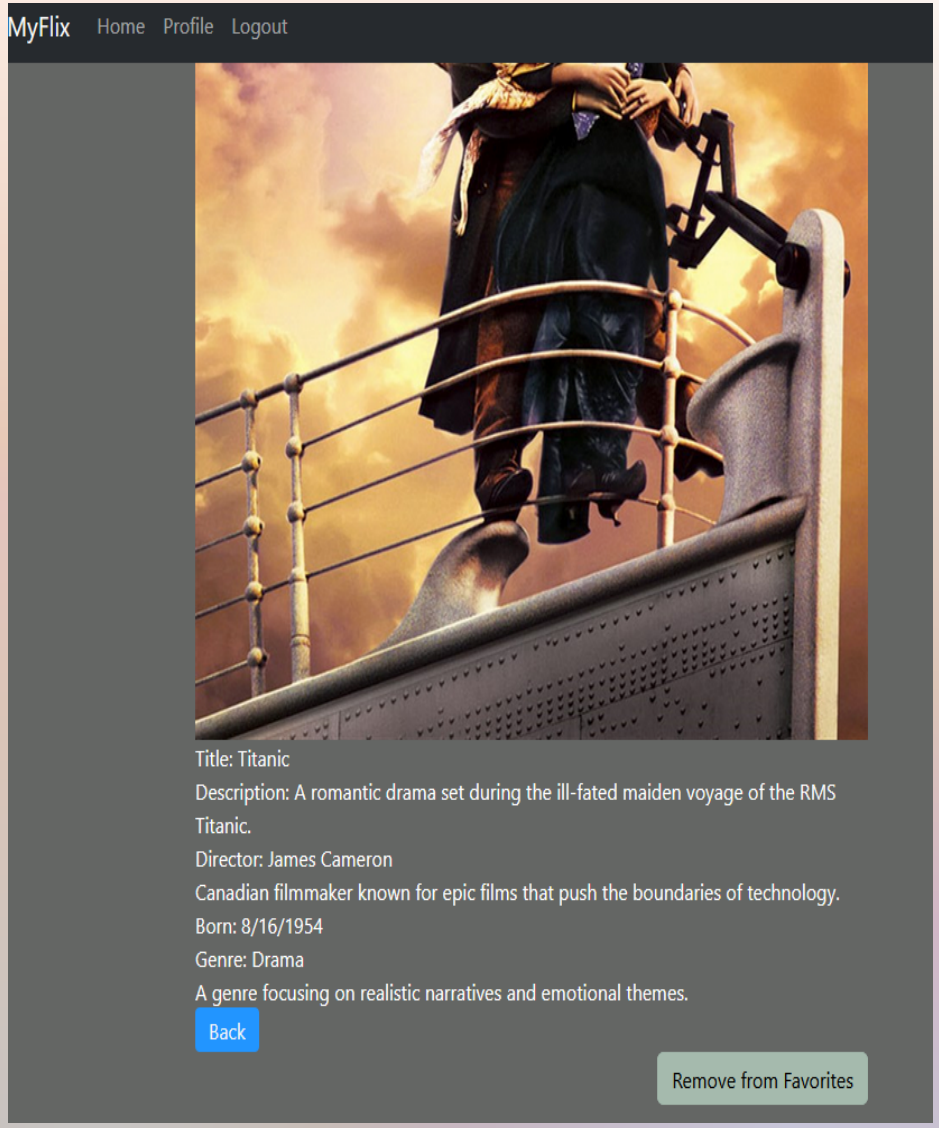
- As a user, I want to be able to access information about movies so that I can learn more about movies I've watched or am interested in.
- As a user, I want to be able to create a profile so I can save data about my favorite movies.

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



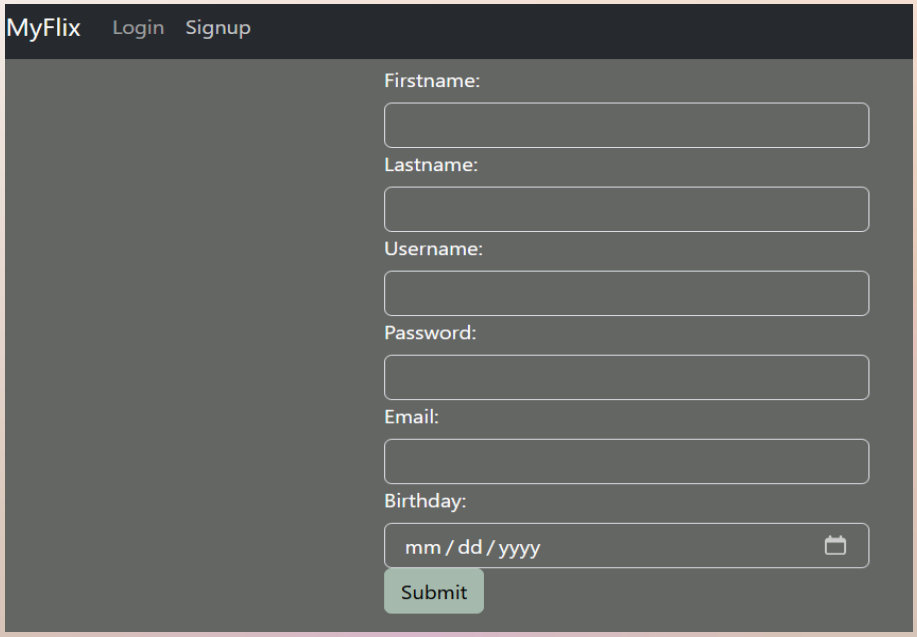
Main View

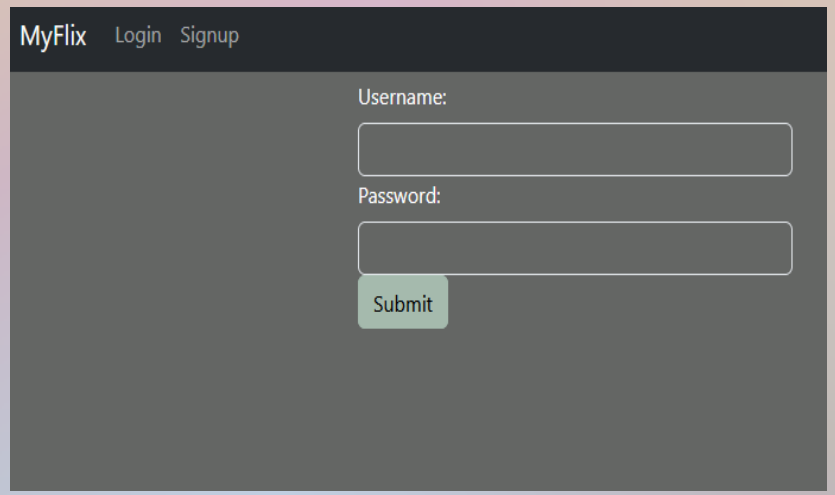


Movie view and Movie details

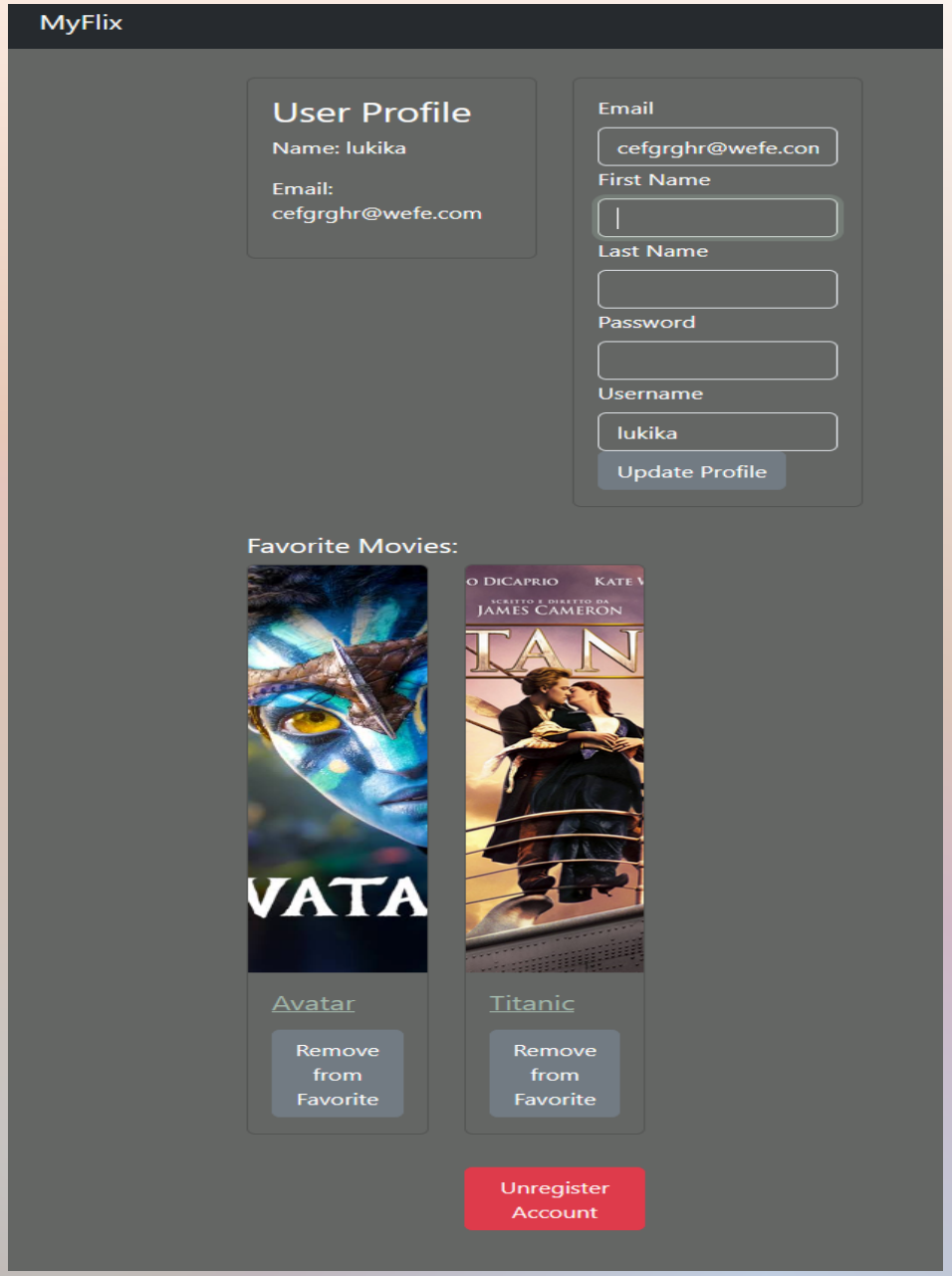
Page Views

PAGE VIEWS





Login and Signup View



Profile View