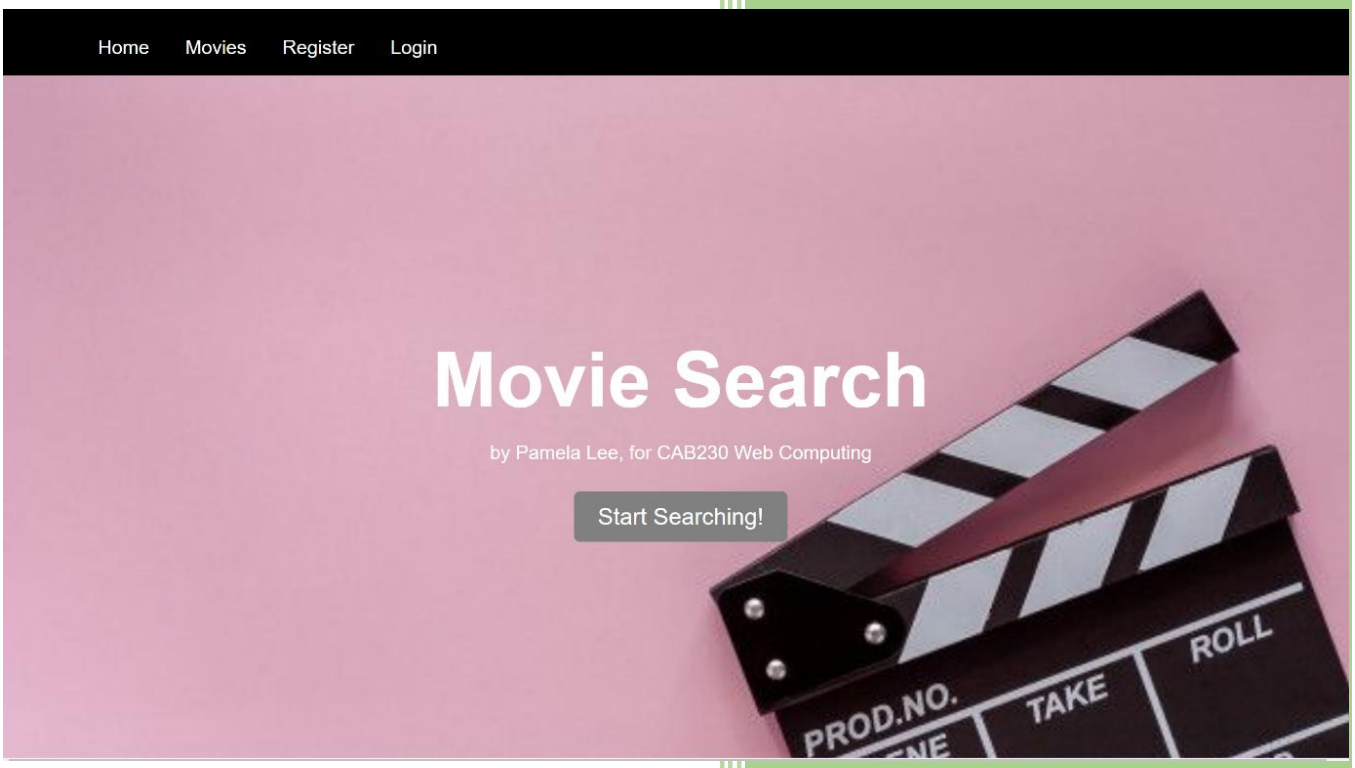
2023

CAB230 Assignment 2 Client Side



CAB230

Movies API – Client Side Application

Ying Ying Pamela Lee

N11518081

5/9/2023

Contents

Introduction

Purpose & description

I have developed a React-based web application designed to provide users with a convenient and intuitive way to view and analyse data about movies which has been exposed via a REST API. The website offers a simple yet powerful search page that allows users to quickly find the movie they are interested in. The API integration provides a vast database of movie information from 1990 – 2023, allowing the website to provide users with a wealth of information about each title, including details such as the release year, genre, cast, and plot synopsis.

The website application is designed with user-friendliness in mind. A fixed navigation bar is situated at the top viewport, ensuring that it always remains in view. This enables users to effortlessly navigate to the home page, browse through the movie list under the "Movies" section, and conveniently register or log in to their account.

In my website, I implemented simple styling using CSS, and prioritized a clean and efficient look over complex designs. To enhance the user experience, you will notice that I have used buttons that allow the user to easily process any selection/search features, as well as a search bar to allow the user to easily search through the database for their movie.

I also used special modules such as “react-router-dom”, “chart.js”, to incorporate navigation and charting to the website application. I also incorporated user testing to test the functionality of the application by getting friends to test the application. Below is a screenshot of my homepage (Fig 1), giving you a glimpse into my website’s functionality and design.

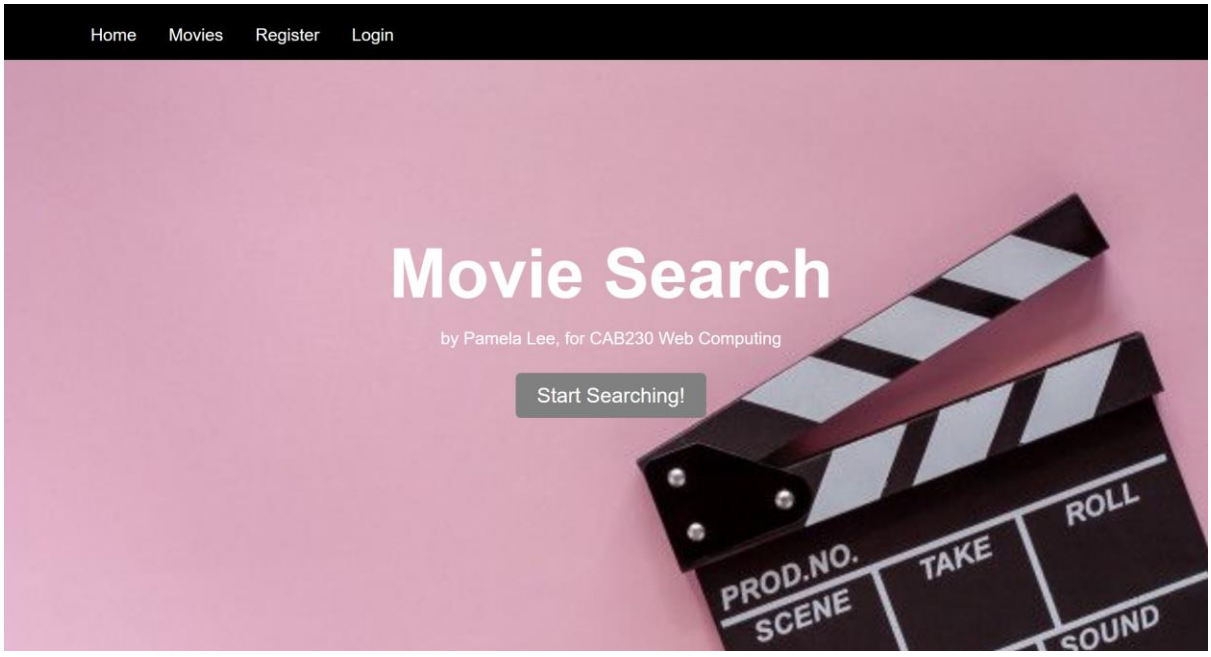


Figure 1. Home Page

Completeness and Limitations

My React application utilizes data endpoints effectively and presents the data in a clean and organized manner. The implementation of user registration and login endpoints and the authenticated data endpoint was successfully achieved. To ensure efficiency, table components were created using standard functionality provided by a component such as AG Grid and there was no excessive querying of the server.

The website design was kept simple, clean, and uncluttered. The presentation of data was also well organized. Navigation was made easy using React Router, and controlled forms were utilized for inputs. There was clear alignment between the chosen components and the data they were displaying. To enhance user experience, refresh tokens were used to ensure that users do not need to log in again every 10 minutes. A chart component was included on the 'person' page to display the IMDB scores of the films they worked on, demonstrating proficiency in utilizing chart components.

The only limitation in this website is the lack of infinite scrolling. I decided not to implement infinite scrolling as ag-grid-react does not allow filtering or sorting in infinite scrolling if an enterprise account is not used. Hence, I decided to prioritise these functions over infinite scrolling. Instead, I have included a “Load More” button (Fig 2.), which allows the user to fetch more movies if required. Using the button, this prevents excessive fetching of the API as it only fetches when needed by the user.

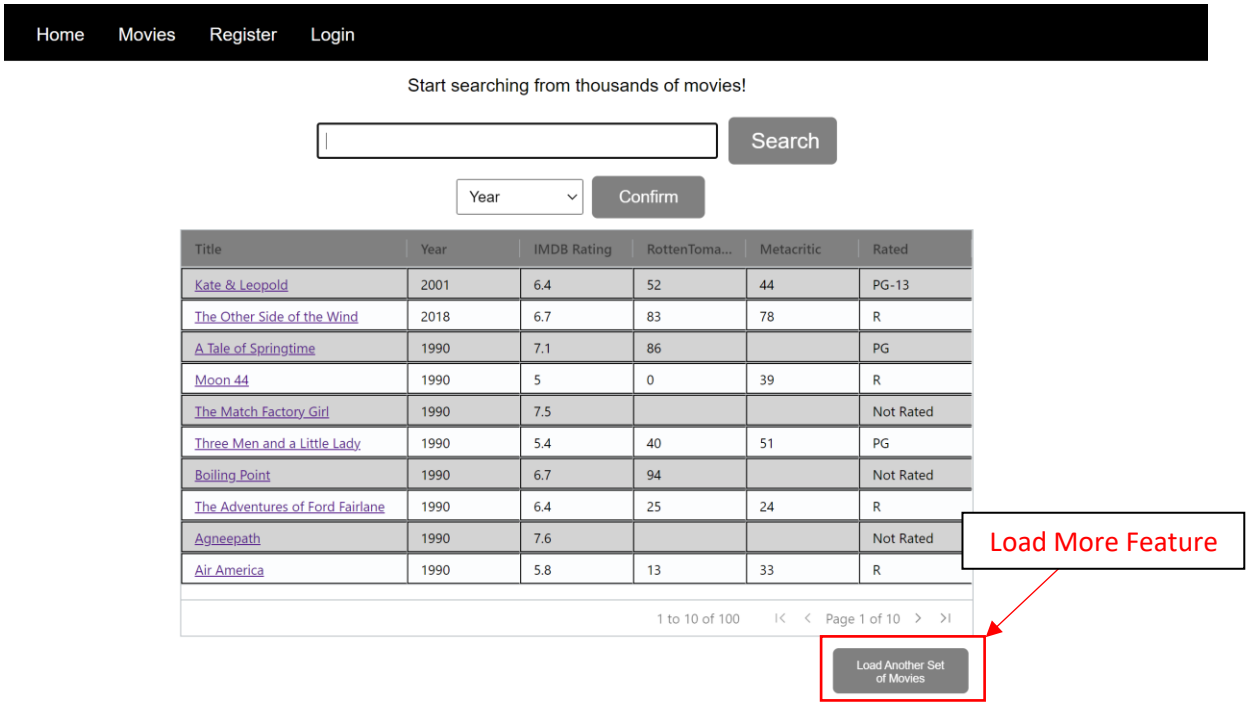


Figure 2. Movies List page with “Load More” button labelled.

Use of End Points

In this section, I will explain the facilities that I have provided in the website.

/movies/search

In the "Movies" page (Fig 3.), the user may search for movies by movie title, and by movie year. Only by clicking on the "Search" and "Confirm" button then will each filter be applied.

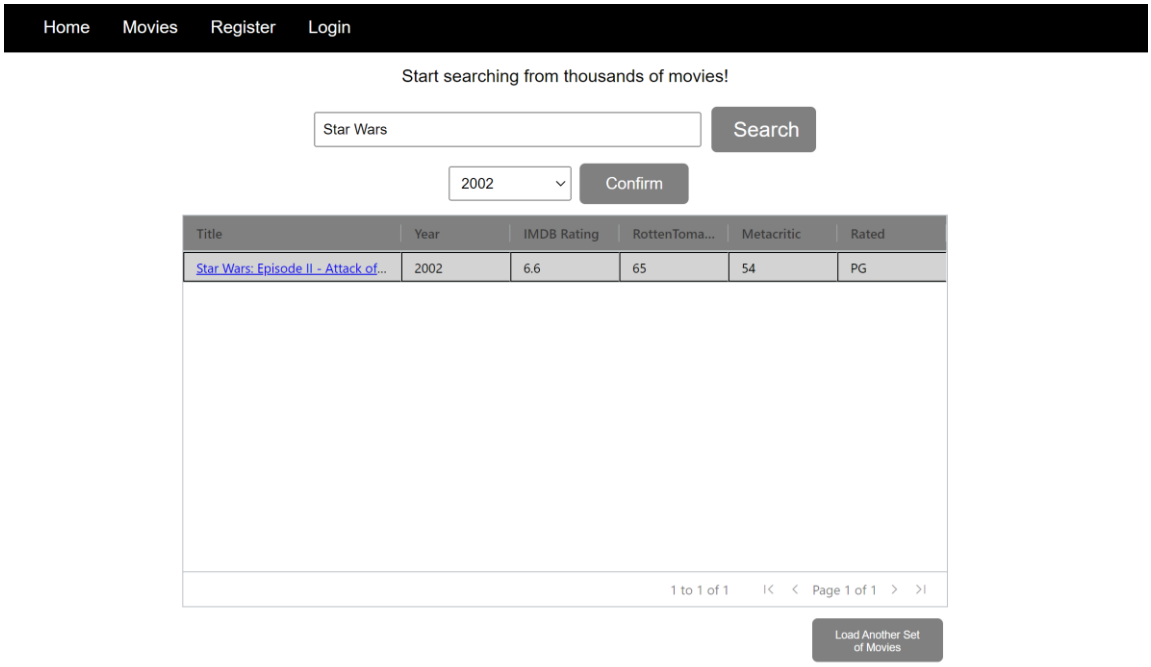


Figure 3. Movies List Page.

/movies/data/{imdbID}

Clicking on a title in the "Movies" table will lead you to the movie's details page (Fig 4.), as shown below.

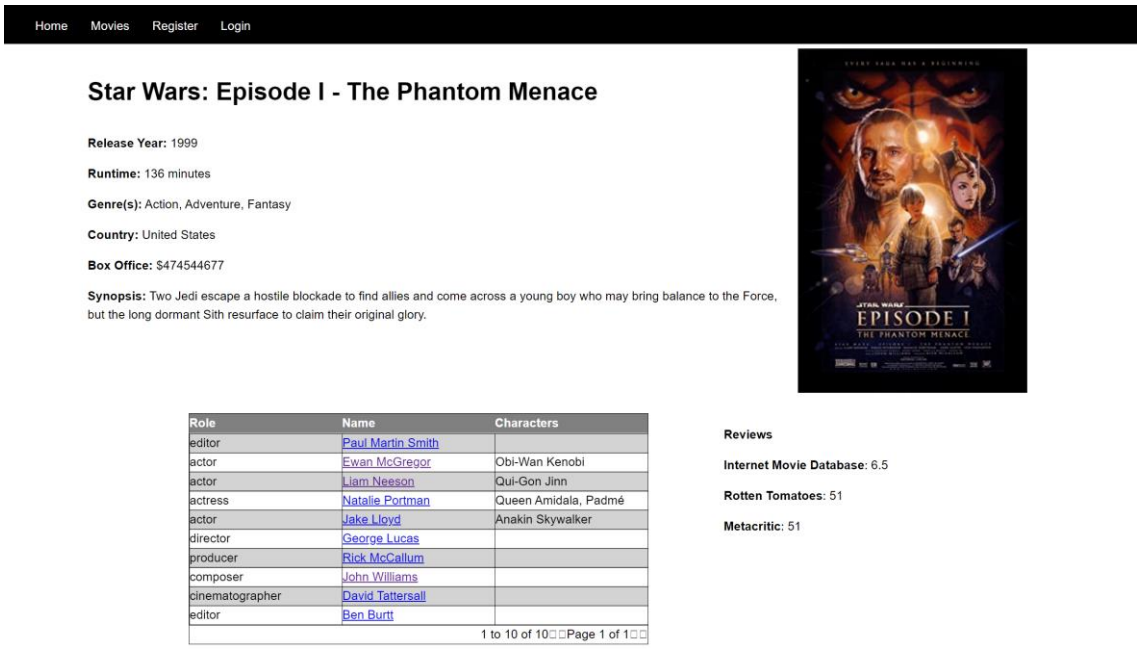


Figure 4. Movie Details Page.

/people/{id}

Clicking on a principal in the principal list will lead the user to this page (Fig 5) if they are not logged in. It will prompt them to login or register (if they do not have an account).

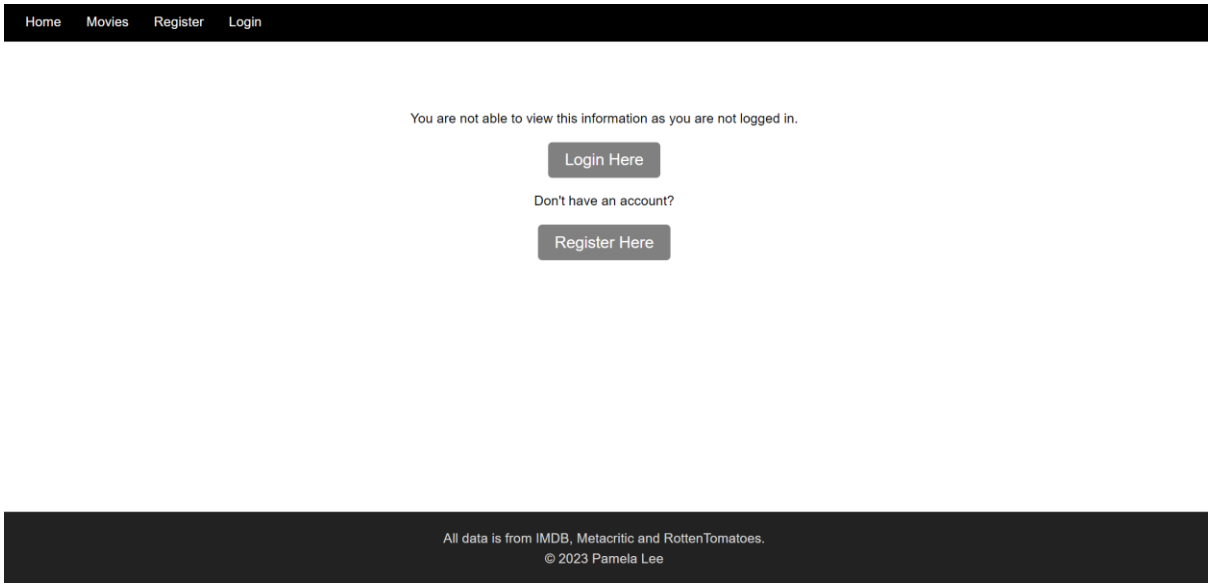


Figure 5. Login/Register Prompt Page.

Otherwise, if the user is logged in, they will be able to view the full details (Fig 6). In the example below, we are viewing Ewan McGregor's details. Not only are all of his roles available for viewing, but there is also a chart showing his IMDb ratings at the bottom of the screen.

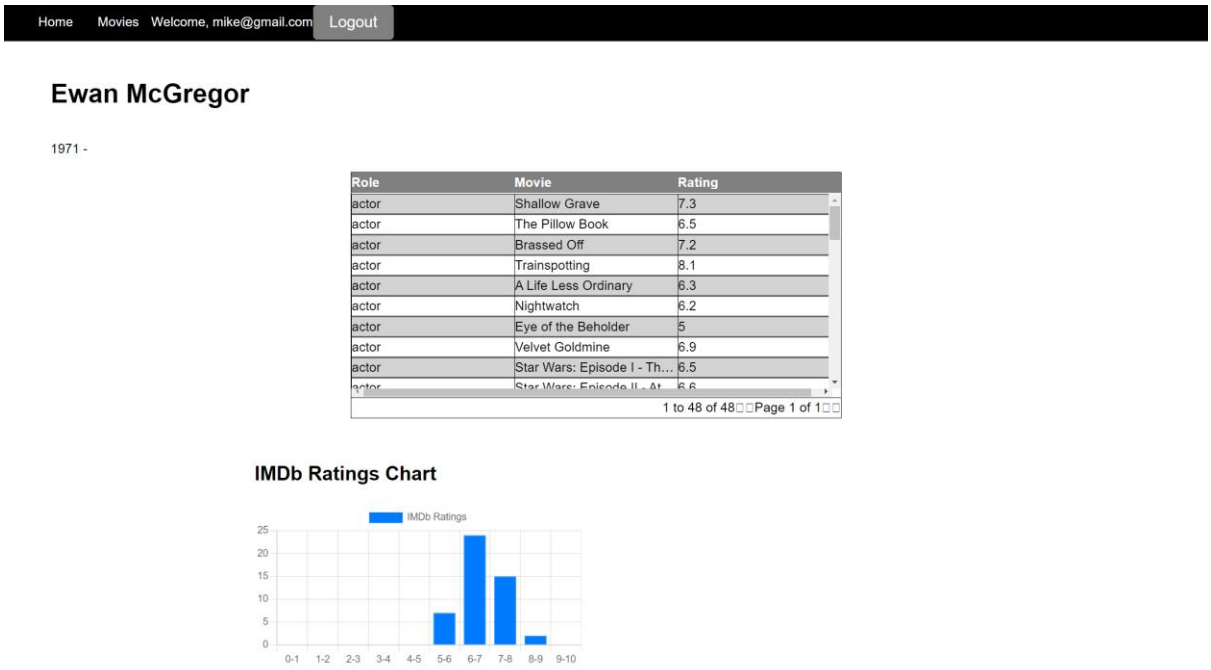


Figure 6. Principal's Details Page.

/user/register

The registration page (Fig 7) allows the user to create a new account. This form will submit successfully and send a post request to the API if passwords match, else, it will prompt the user to key in the password again.

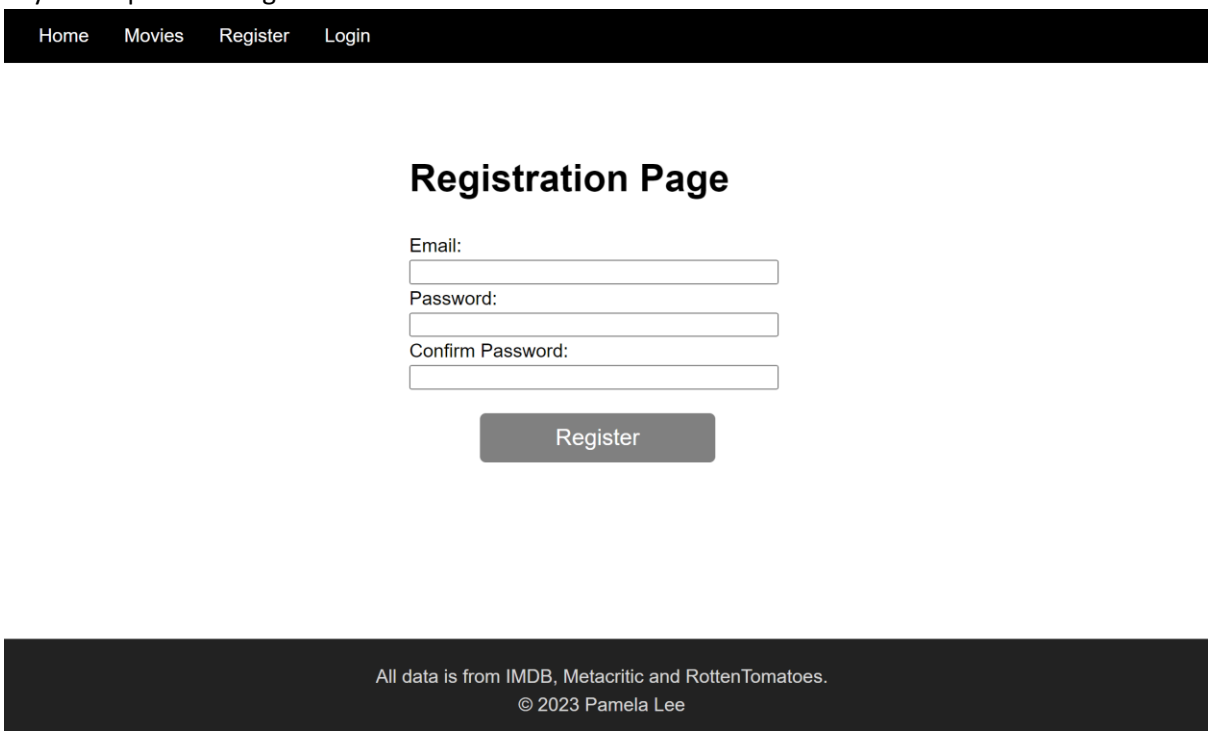


Figure 7. Registration Page.

/user/login

The login page (Fig 8.) allows the user to login to their existing account. This form will submit successfully and send a post request to the API if the account details are correct.

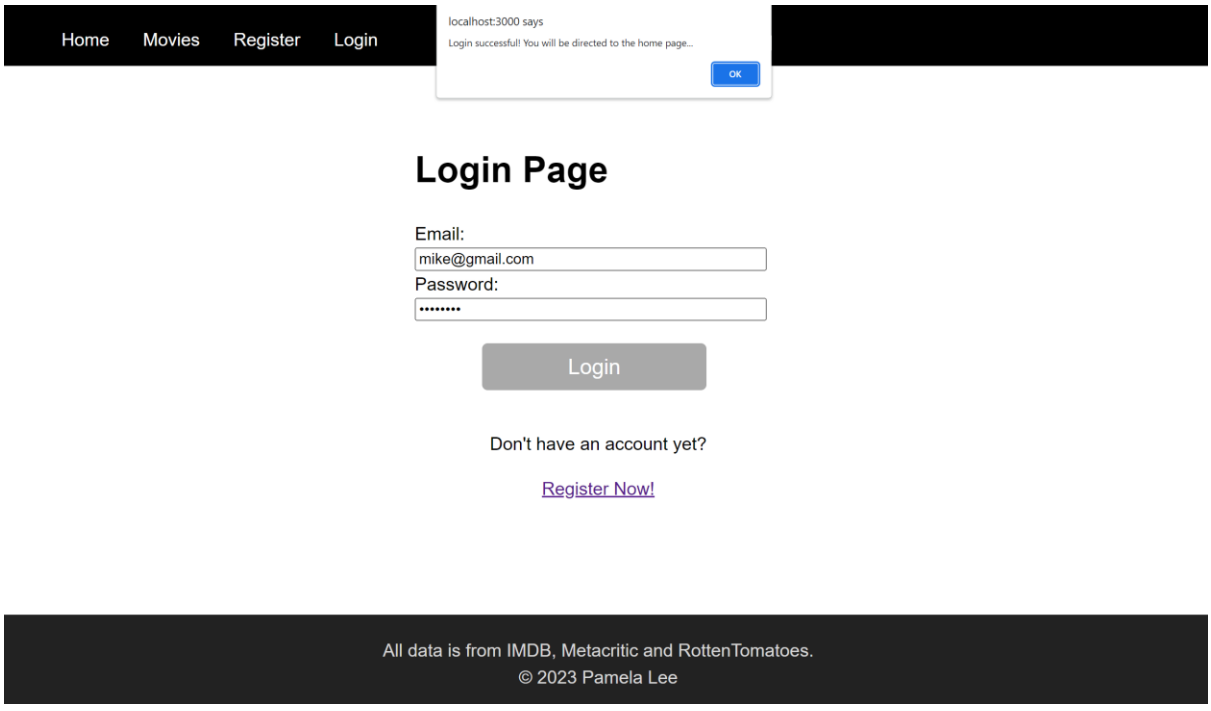


Figure 8. Login Page.

/user/refresh

The token will be sent to refresh every 9 minutes, before the token expires.

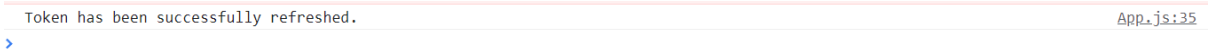


Figure 9. Successful token refresh is printed in console.

/user/logout

Clicking on the log out button at the top of the screen will send a POST request to the API, which will then invalidate the refresh token. The screenshot below (Fig 10) shows a successful logout attempt.

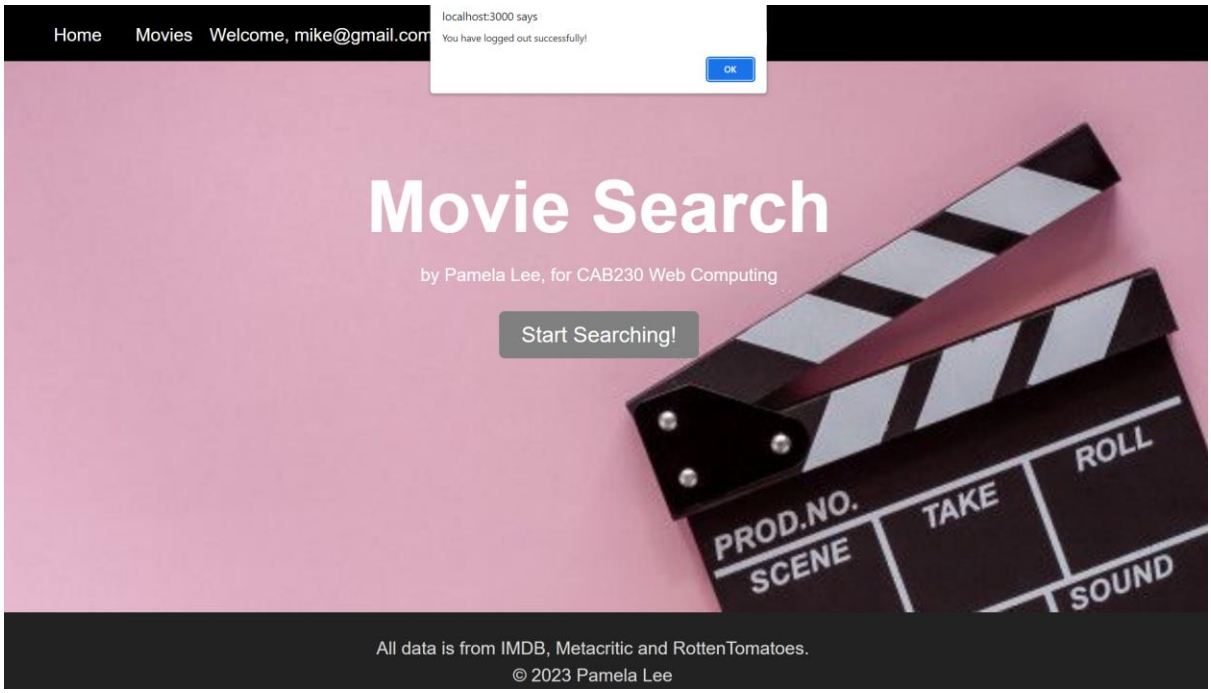


Figure 10. Successful logout attempt

Modules Used

This is just a list of the external modules that I have used. You need not specify core React modules.

Ag-grid-react

Module to provide fully featured table components, including infinite scrolling.

https://www.ag-grid.com/react-grid/

react-router-dom

Module that provides routing capabilities for web applications built with React. It allows developers to create dynamic and interactive user interfaces by enabling navigation between different components or pages based on the URL path.

https://reactrouter.com/en/main/start/overview

Chart.js

Simple yet flexible JavaScript charting library for the modern web.

https://www.chartjs.org/

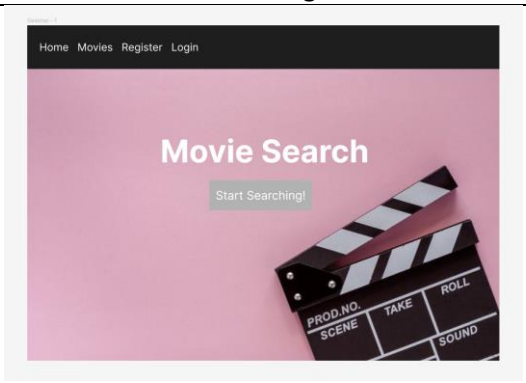
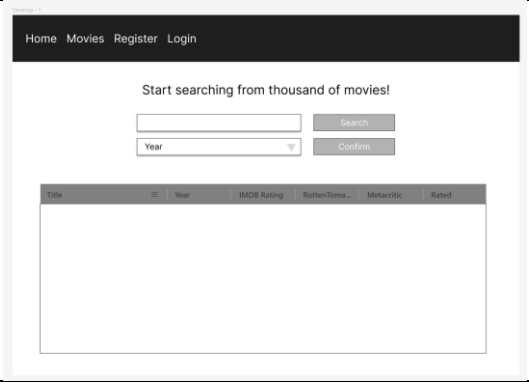
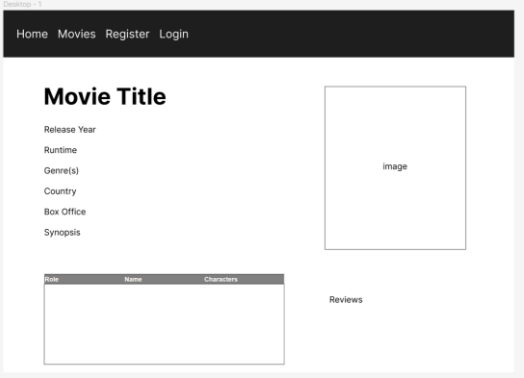
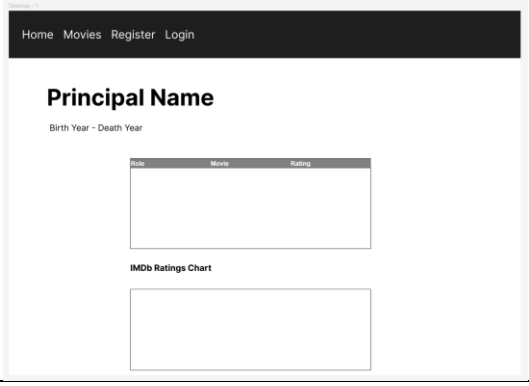
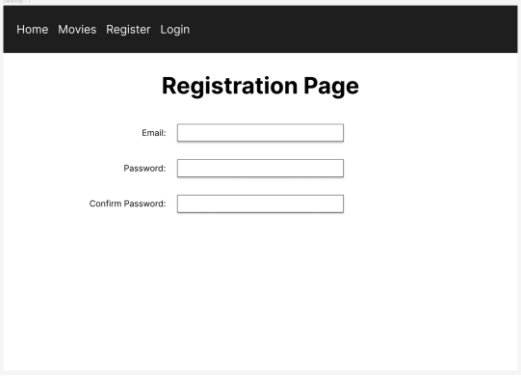
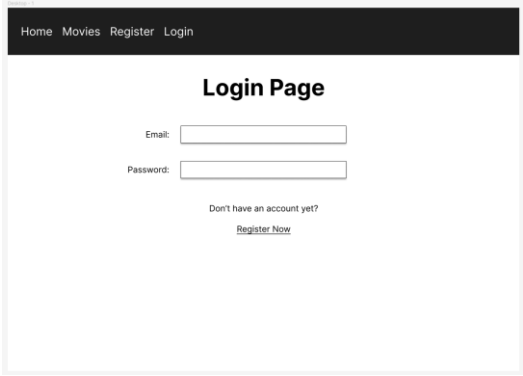
React components for Chart.js, the most popular charting library.

https://react-chartjs-2.js.org/

Application Design

Navigation and Layout

I did my design mock-ups on Figma, which are displayed in the table below. My application design strictly adhered to the mock-ups. I prioritized a clean and minimalist design over flowery aesthetics.

Home Page	Movies List Page
	
Movie Details	**Principal Details**
	
Register Page	**Login Page**
	

I will describe the different pages in detail with screenshots of the actual website instead of the mock-ups. The descriptions include the flow between the screens. You will notice that the navigation bar is consistent over the different pages and will remain at the top of the user's viewport when scrolling down. This allows the user to navigate between different pages even if they are not currently at the top of the page. I did not apply this to the footer, as I felt that that was unnecessary. The footer will only be visible once the user scrolls down to the bottom of the page.

1. Home Page

The home page (Fig. 11) has a simple design – a navigation bar, an eye-catching backdrop, as well as a header simply telling the user that this is a movie search website. The user is guided to click on "Start Searching" to get started, or if they like, they may navigate to other options through the navigation bar. It is designed to be straightforward and user-friendly.

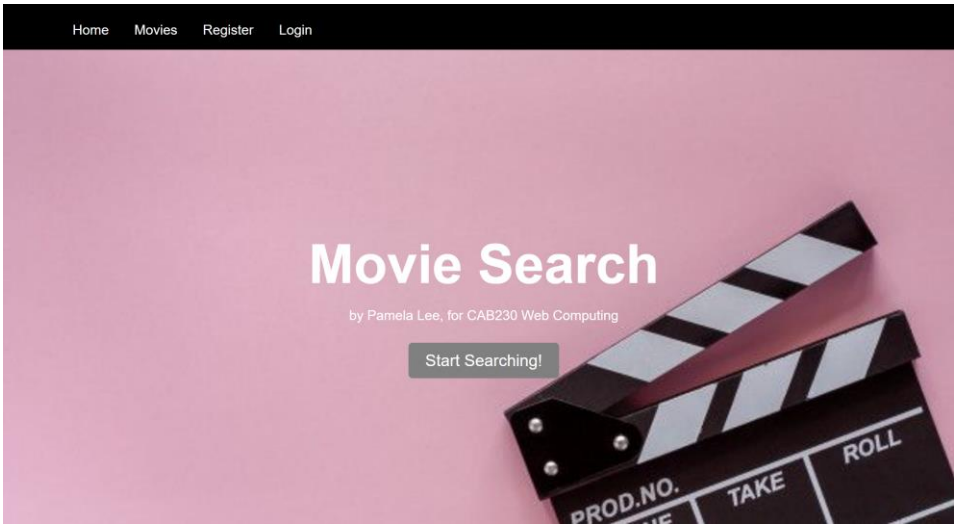


Figure 11. Home Page

2. Movies List Page

This page (Fig. 12) is accessed by either clicking on "Movies" in the navigation bar or "Start Searching" on the home page, the user will be directed here. The user can freely browse through the list of movies or use the search bar / year selection filter to look for the movie of their choice. The table only loads 100 movies at once, so the user may click on "Load More Movies" to load another set of 100 movies from the database.

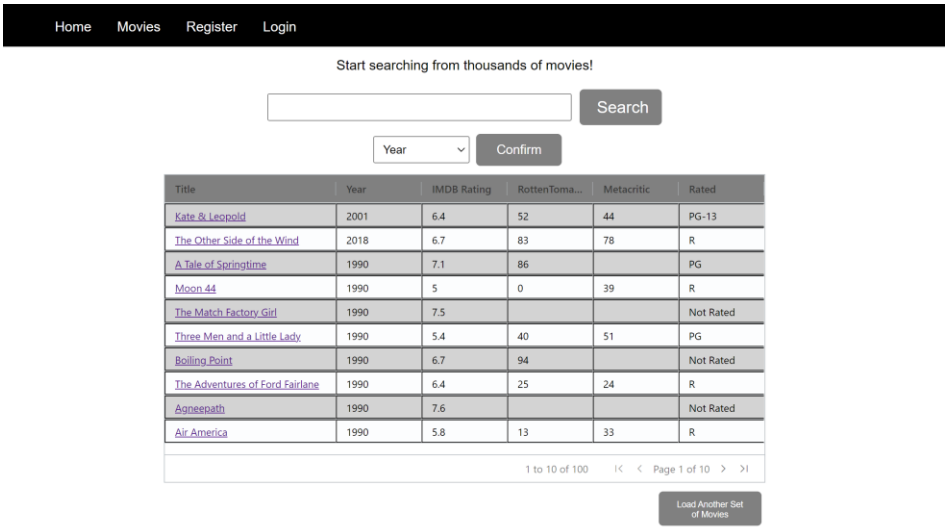


Figure 12. Movies Listing Page

3. Movie Details Page

This can be accessed via clicking a movie title in the table in the “Movies” page (Point 3). Details of the movie are released here (Fig. 13).

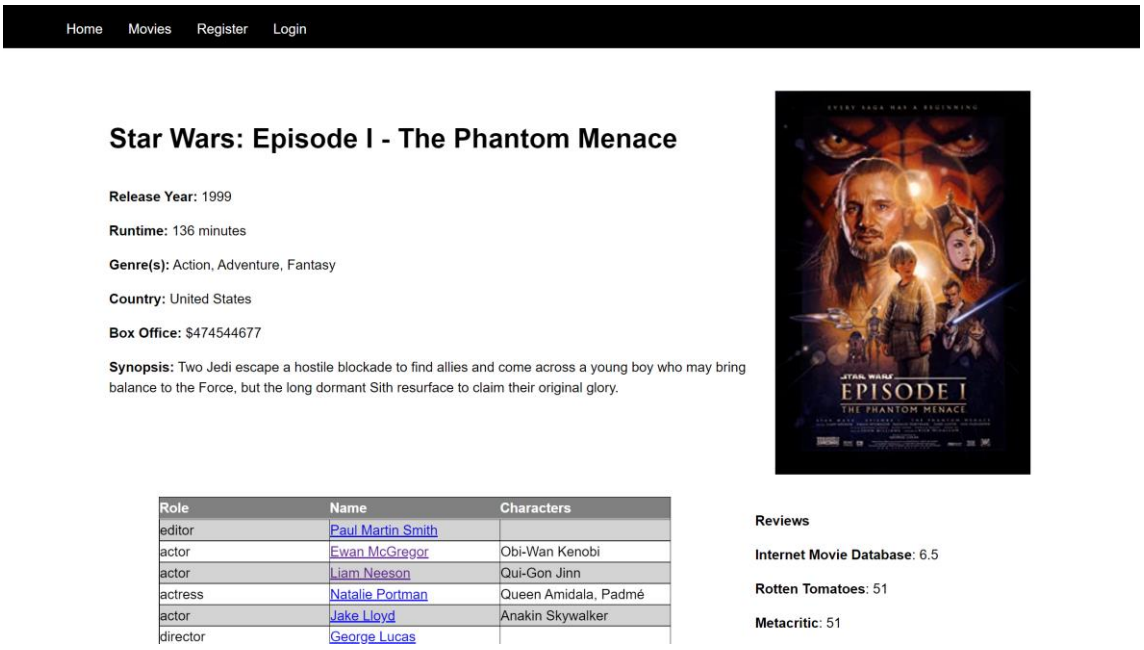


Figure 13. Movie Details Page

4. Principals Page (Authenticated)

This can be accessed via clicking the principal’s name in the table in the “Movies” page (Fig. 14). Details of the principal are available here. It is important to not that if the user is not logged in, they will not be able to view the details at all, and instead, it will lead the user to the Unauthenticated Page (Point 5).

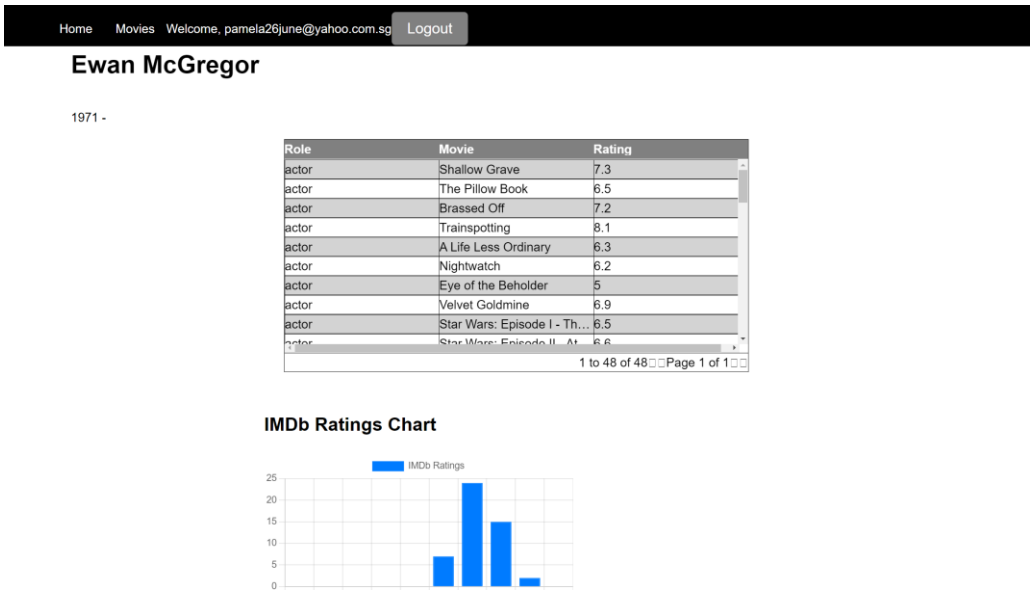


Figure 14. Principal Details Page (with Chart)

5. User is Unauthenticated

Users will be directed to this page (Fig. 15) if they attempt to view a principal's detail while not logged in. The login button here will lead them to the login page (Point 6), and the register button will lead them to the registration page (Point 7).

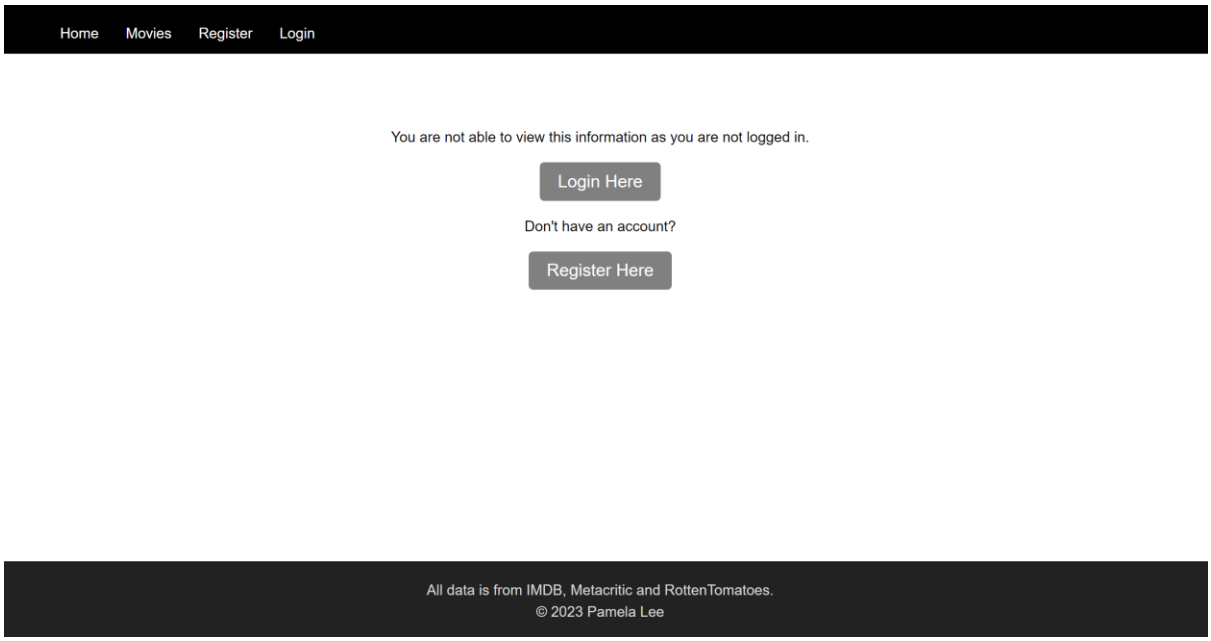


Figure 15. Login/Register Prompt page

6. Login Page

User will need to key in their registered email address and password (Fig. 16). Upon successful log in, they will be directed to the home page (Point 1).

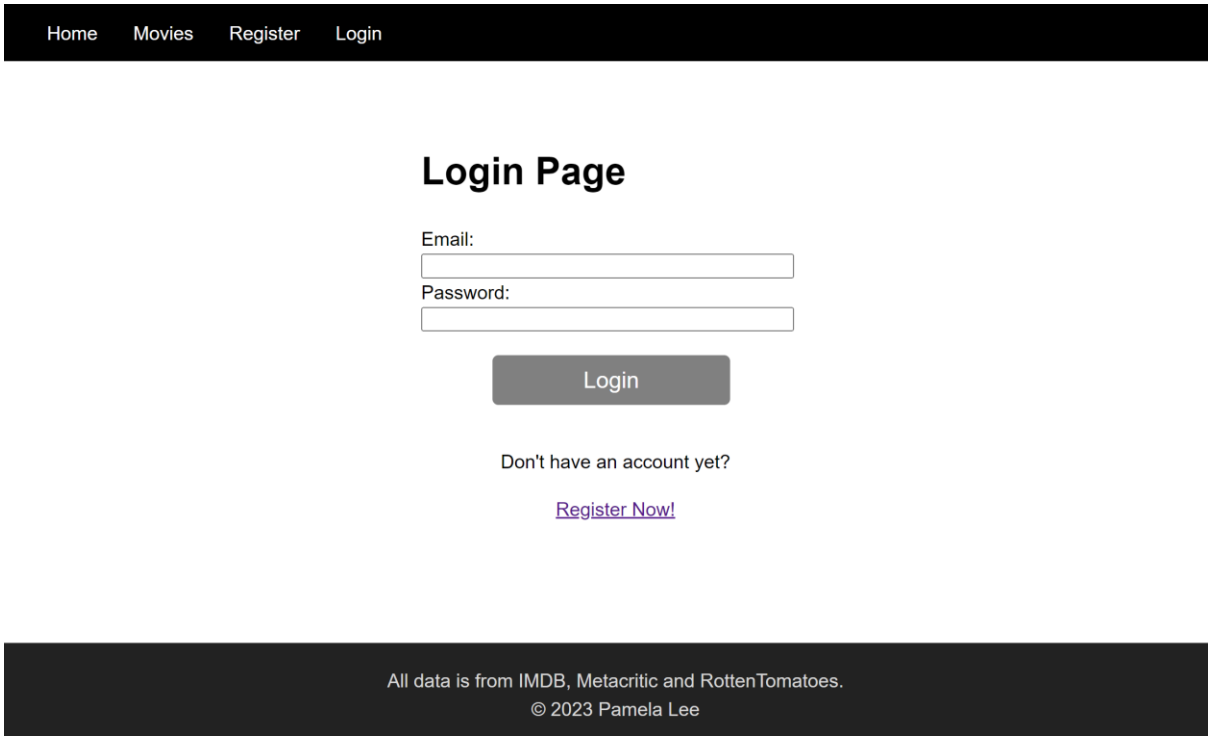


Figure 16. Login page

7. Registration Page

The registration page (Fig. 17) will allow the user to create a new account. Upon successful creation, the user will be required to log in before being able to access authenticated information.

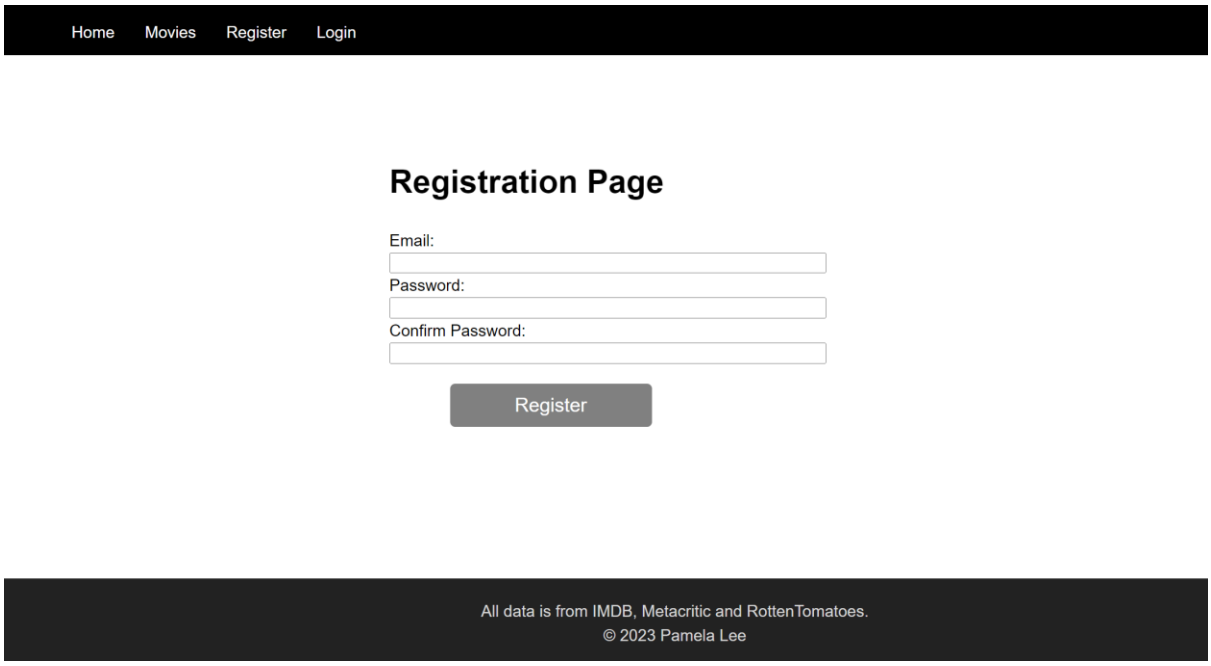


Figure 17. Registration page

Usability and Quality of Design

In this section, I will be critically assessing the quality of my own design work by highlighting the good and the bad aspects of my design.

I focused on designing the website for "lazy people" – prioritizing a straightforward design and getting to the point. It is easy to navigate, and I ensured that the user will be able to see the main points on each screen easily by placing them in the center or increasing the size of important points. The application is made to be consistent with user expectations from other apps – a navigation bar, a home page etc.

Navigation

I have designed the website with the user's expectations in mind – a navigation bar at the top of the viewport. The navigation bar allows easy access to breadth and depth of the site. However, I believe that more could be done to inform the user of where they are within the site. This could be done by highlighting the user's current location in the navigation bar (Fig. 18) and allow them to know where to locate the information if they wish to access it again.

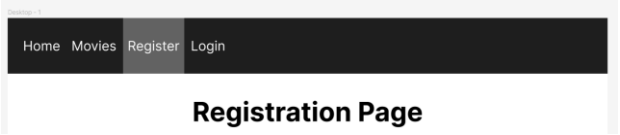


Figure 18. Registration page with "Register" highlighted in the Navigation bar

The current navigation bar should be easy to learn and is consistent throughout the site with no scrolling. If I were to do things differently, I would have put the register/login "Welcome user" and logout buttons on the right side of the screen, separate from the home page and movies. This could create a sort of separation between the user authentication process and movie search process (Fig. 19).

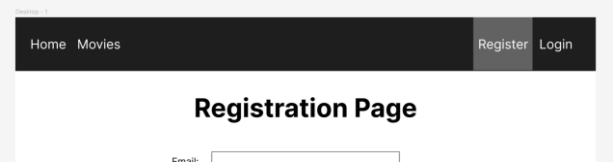


Figure 19. Navigation bar with a clear distinction between user authentication and movie search functionality

I believe visual design is consistent across screens. I made sure that the navigation bar and footer are consistent, as well as the typography and colour. I only used one font and keep to a set of monotone colours across buttons, backdrop, and navigation bar. Only the home page was slightly more varied in colour to entice the user and leave a good first impression but kept the clean look without being too distracting. However, I did feel like the minimal colour and fonts might have been a bit too boring. I could have added a bit more visual content and such as images or even nicer fonts to make it look better. I tried to keep the layout as standard as possible as well. However, I realized that zooming in and out of the page affects the layouts in certain ways, which could be a resulting of using marginTop, marginLeft etc. Keeping the layout as consistent as possible would have been a huge improvement to usability across different platforms and devices.

Accessibility

In this section I will analyse my site from the perspective of accessibility, relying on our lecture slides and video, and the checklist from the W3C.

1. **Provide a text equivalent for every non-text element – alternatives to images, symbols, scripts, graphical buttons, sounds, audio, and video files and so on.**

Yes. The images on this website (i.e. Poster) have a text equivalent provided (Fig 20.).

```
<div style={{ marginLeft: '20px' }}>
    <img src={movie.poster} alt="Poster" />
</div>
```

Figure 20. Movie Poster with alternative text.

2. **Ensure that all information conveyed with color is also available without color, for example from context or markup.**

My website is lacking in this feature. If a button changes color when it is hovered over, the same information is should also be conveyed through other means such as a tooltip or text label. When I mouse over a poster, I should be able to see a label as well.

3. **Organize documents so they may be read without style sheets. For example, when an HTML document is rendered without associated style sheets, it must still be possible to read the document.**

I did feel like I used too many div tags or relied too heavily on CSS for layout and made the content difficult to read or understand without the style sheets.

4. **Ensure that text equivalents are updated when dynamic content changes.**

Yes, I believe that this has been achieved. Since my content is largely text, when the page changes in the table, the text changes along with it.

5. **Avoid causing the screen to flicker.**

I did not see this happening while navigating from screen to screen.

6. **Use the clearest and simplest language appropriate for a site's content.**

I believe that this point has been achieved as I aimed to keep the site minimalistic and avoided using too many texts or complex language.

7. **For tables, identify row and column headers – clearly differentiated from the data.**

Yes, this has been done by filling the row and column header cells in a much darker colour.

Technical Description

Architecture

My application source code is organized like this (Fig. 22):

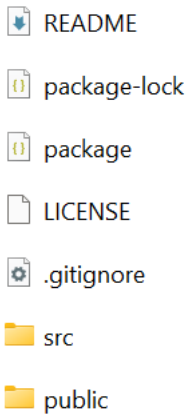


Figure 22. / Directory

/src

I only have App, and index files and folders (Fig. 22) The rest of the files have been stored in a folder under components and pages. Components refer to the Header, Footer files while Pages are Register, Login etc.

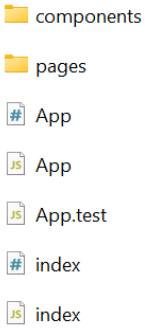


Figure 22. /src Directory

/src/components

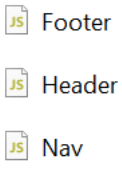


Figure 22. /src/components Directory

/src/pages

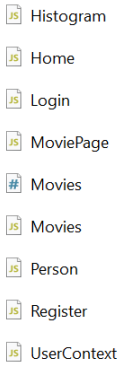


Figure 23. /src/pages Directory

Test plan

Task	Expected Outcome	Result	Screenshots Appendix A Figure No.
Search For Movies	Results displayed in table	PASS	1
Filter Movies by Year	Results displayed in table	PASS	2
Load Next Set of Movies	Results displayed in table	PASS	3
Selecting a movie to view details	Leads user to Movie Details Page	PASS	4
Viewing principal details when not authenticated	Leads to Login/Register Prompt Page	PASS	5
Login: Incorrect Password/Email	Prompts user with alert; does not authenticate user.	PASS	6
Login: Correct Password/Email	Alerts user and redirects them to homepage.	PASS	7
Viewing principal details when authenticated	Displays principal's details.	PASS	8
Register: Creating an already existing account	Alerts user and redirects them to the login page.	PASS	9
Register: Creating a new account	Alerts user and redirects them to the login page.	PASS	10

Difficulties / Exclusions / unresolved & persistent errors

Topics to include here could be:

- What were your major roadblocks / how did you resolve them?
- Any functionality you didn't or couldn't finish and the technical issues encountered
- Are there any outstanding bugs?

In the process of developing the movie search application, I faced a few roadblocks, which I managed to resolve through research and collaboration. One major roadblock I faced was with the integration of the authentication and authorization features using JWT tokens. It took me some time to understand the flow of information between the frontend and backend, but with the help of documentation and online forums and using console.log, I was able to successfully implement this feature. I also had a lot of problems with the refresh token, but eventually managed to fix it by testing it with shorter refresh times and using console.log to verify its implementation and eventually implemented it successfully.

While I was able to implement most of the desired functionalities, there were a few that I couldn't complete due to technical issues. For instance, I could not implement infinite scrolling in ag-grid-react. In addition, there seems to be some errors in my chart (Fig. 24), although it displays nicely on the website. I ignored this error as the chart seems to be displaying fine. The code for the chart can be found under "/pages/Histogram.js"

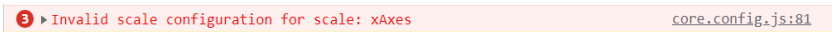


Figure 24. Chart error

Another bug that I have had is that the search function / filter function no longer works the moment the user uses the "Load More" button on the bottom right of the screen (Fig. 25).

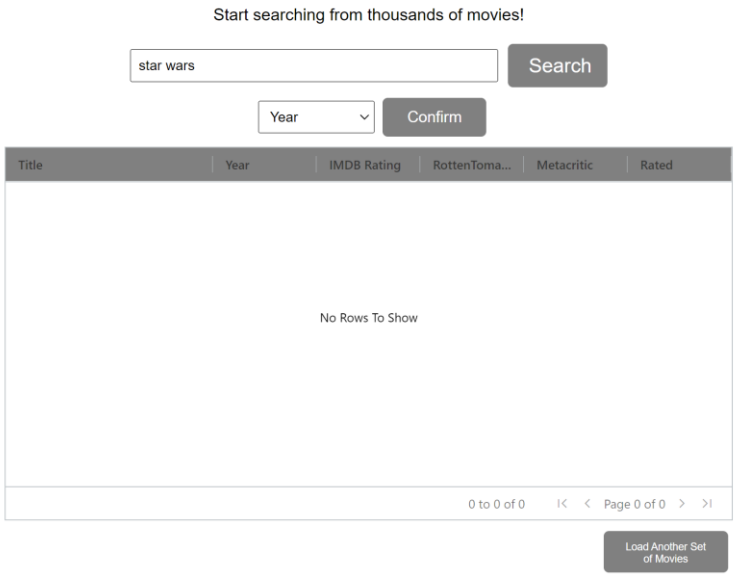


Figure 25. Load Another Set Error

There are no more other outstanding bugs in the website application.

Extensions (Optional)

There are several potential future extensions and improvements that could be made to my movie search app. Here are a few ideas:

1. User Reviews: Allow users to leave their own ratings and reviews for movies, in addition to displaying the existing ratings and reviews from external sources.
2. Watchlist: Implement a feature that allows users to add movies to a watchlist, which they can refer to later. This could be useful for users who come across interesting movies but do not have the time to watch them immediately.
3. Recommendations: Utilize machine learning algorithms to generate personalized movie recommendations based on a user's viewing history, ratings, and watchlist.
4. Social Sharing: Allow users to share their favorite movies or movie recommendations on social media platforms like Twitter, Facebook, and Instagram.
5. Advanced Search: Implement an advanced search feature that allows users to filter movies by genre, year of release, language, and more.
6. Localization: Add support for multiple languages, allowing users to search for movies and view content in their preferred language.

These are just a few potential extensions and improvements that could be made to my movie search app. With continued development and enhancements, the app could become a more comprehensive and personalized movie-watching experience for users.

User guide

Here is a guide on how to use my application. It is simple – the landing page will be the home page (Fig. 26), where you can simply click on the "Start Searching" button to get started.

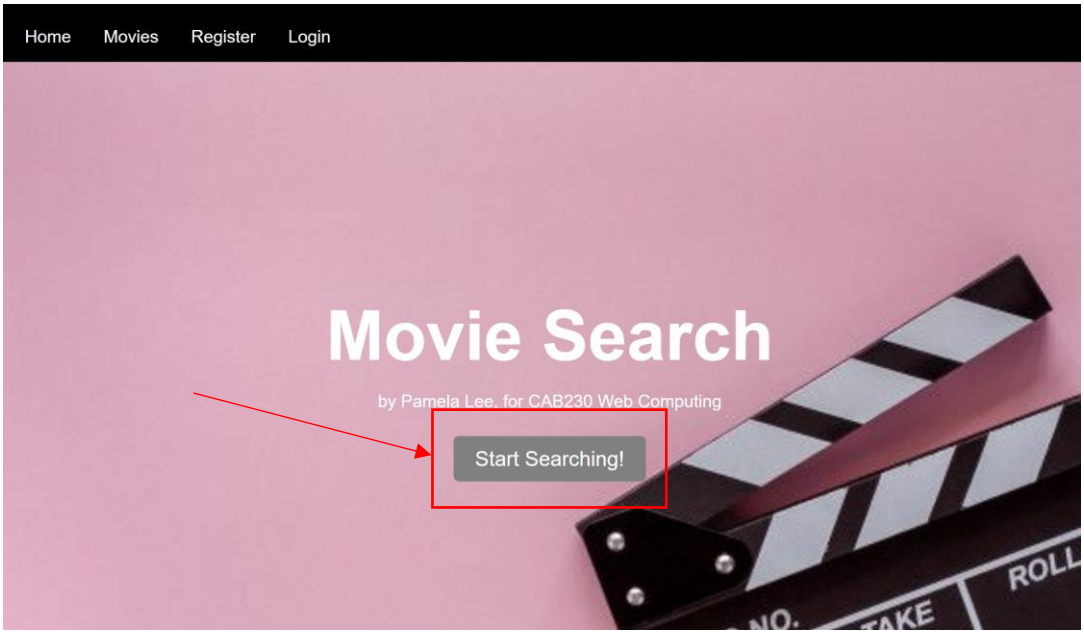


Figure 26. Landing Page

Look through the table below, or just search for your favourite movie using the search bar! Click on the movie to select it.

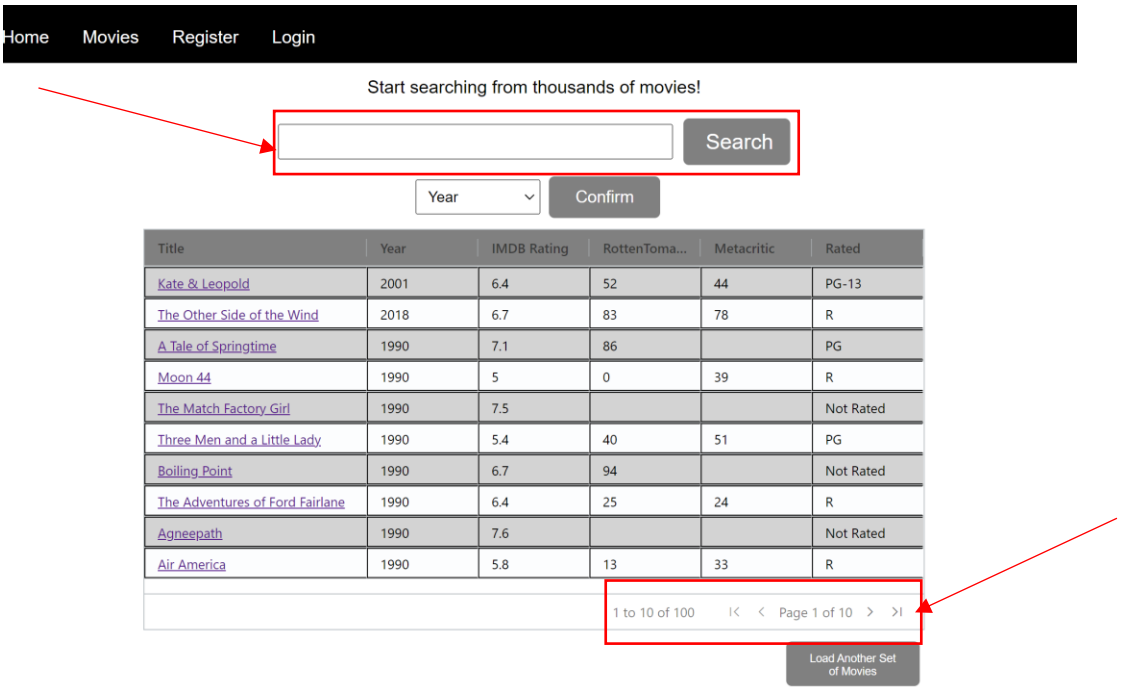


Figure 27. Movie Search

Browse through the movie details! Click on the principal to view more details about them.

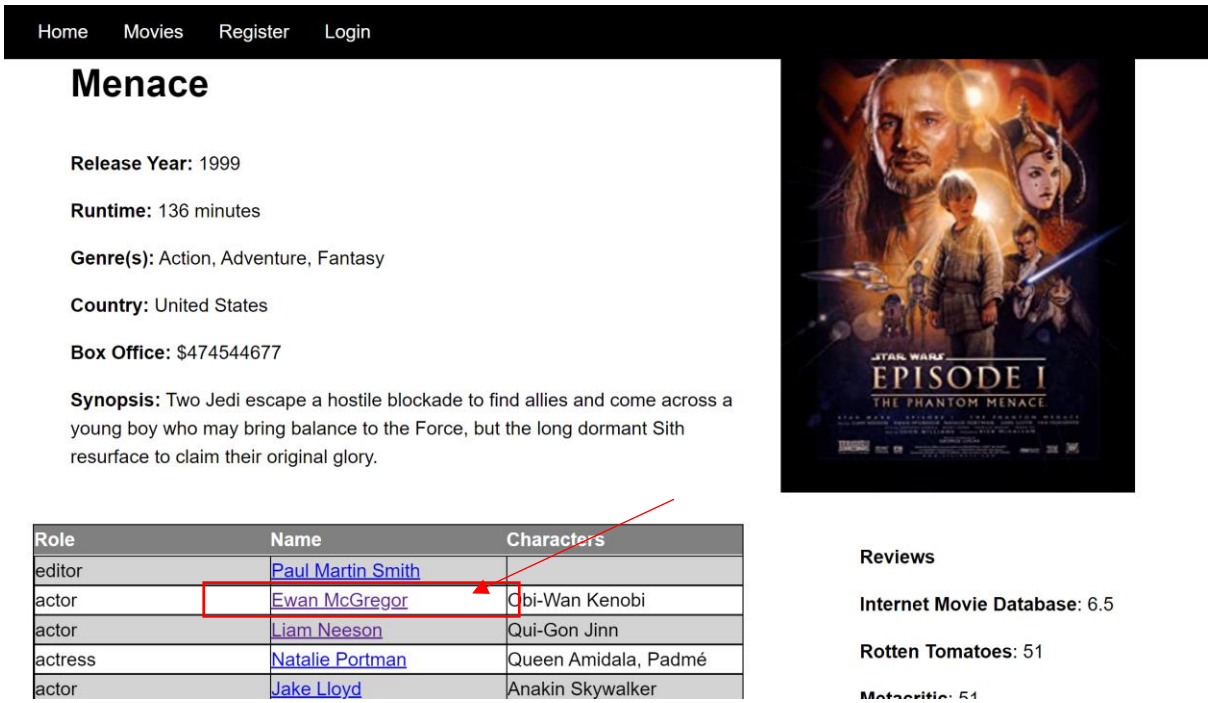


Figure 28. Movie Details

Don’t have an account? Create one:

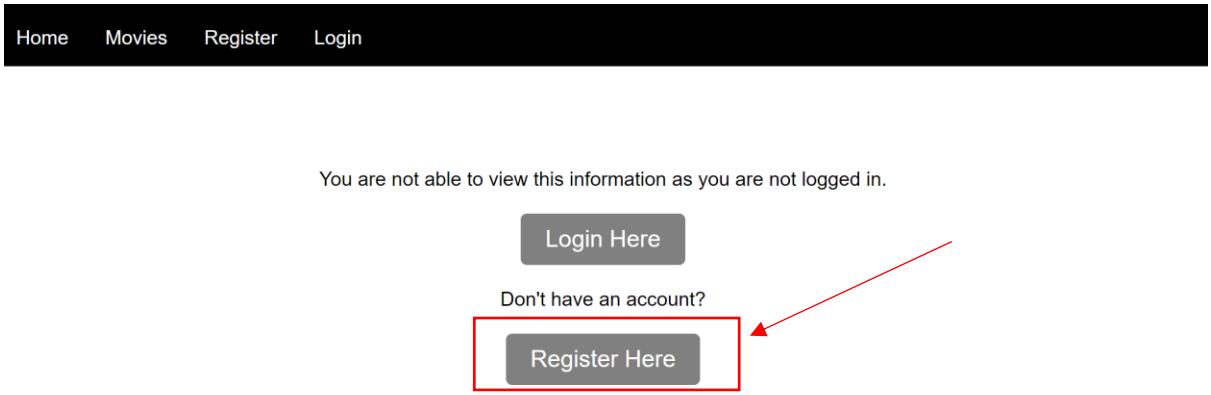


Figure 29. Register

Upon successful account creation, log in!

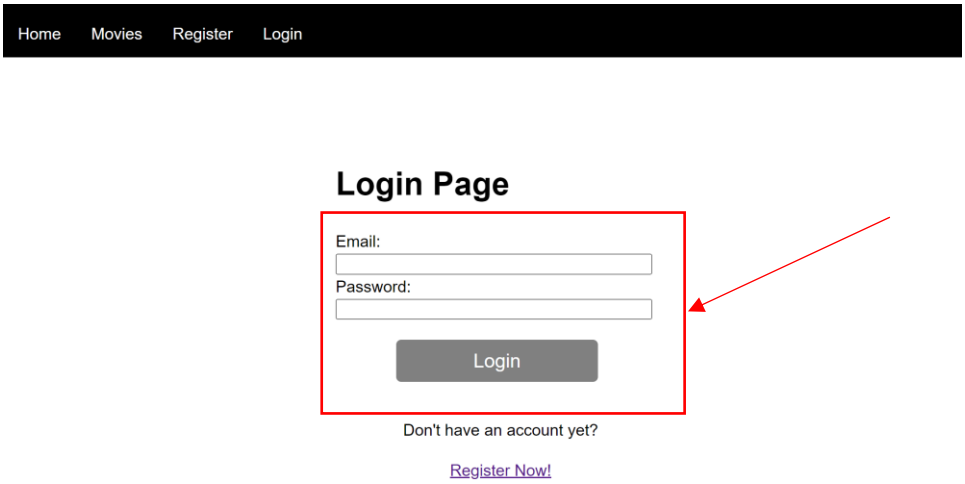


Figure 30. Login

Following the steps from Fig 26, 27 and 28, you will be able to view principal details now! Click on any of his movies to look at the movie details.

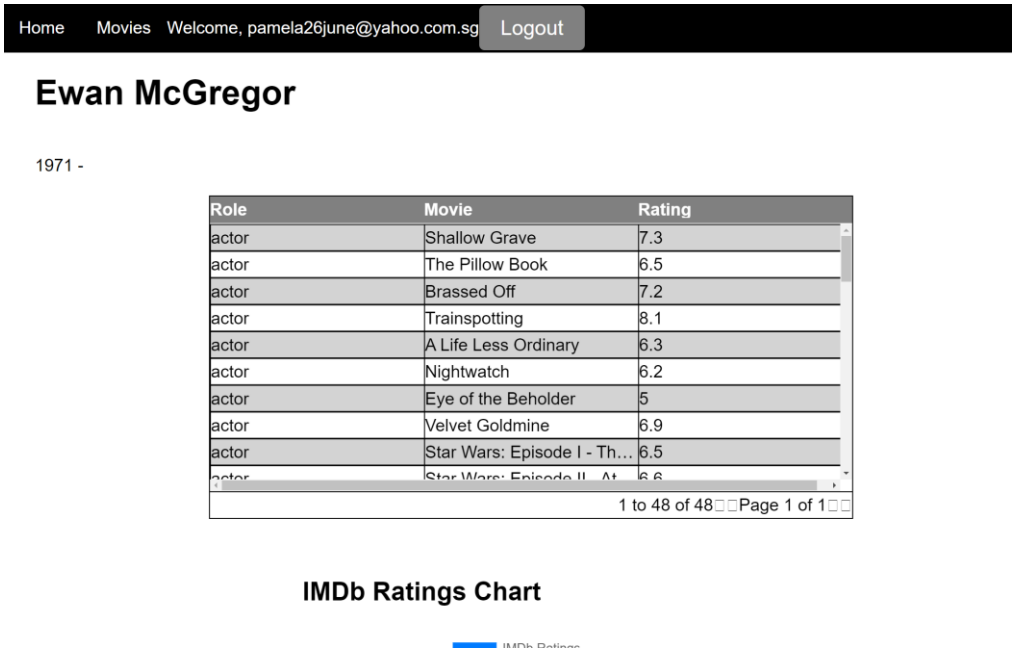


Figure 31. Principal Details

Click on the logout button anytime to log out of your account.

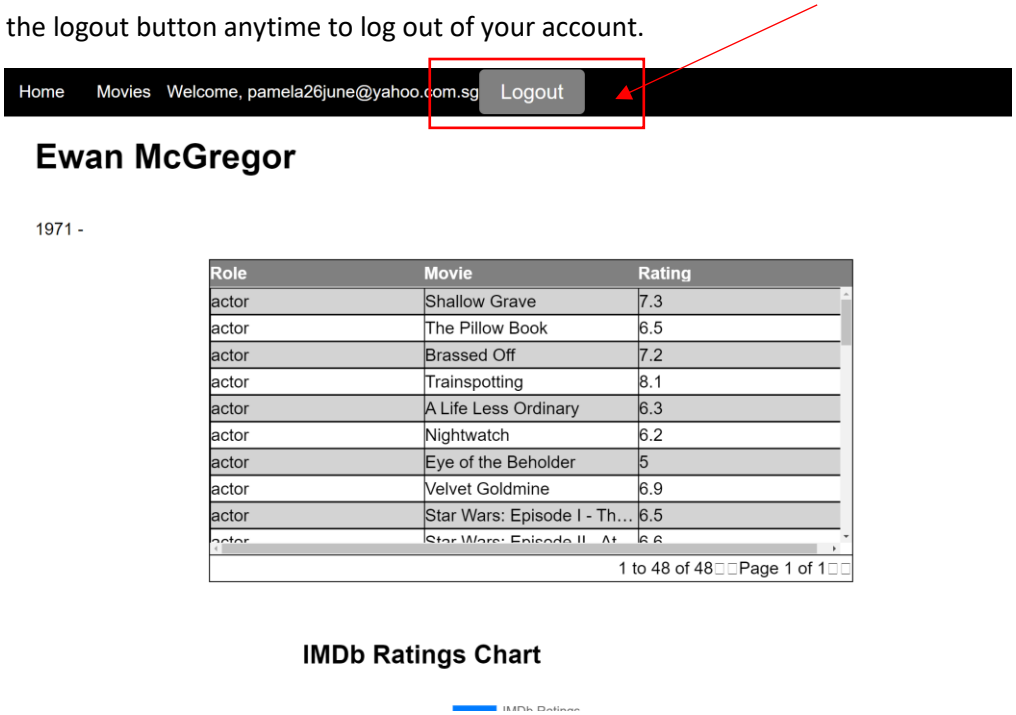
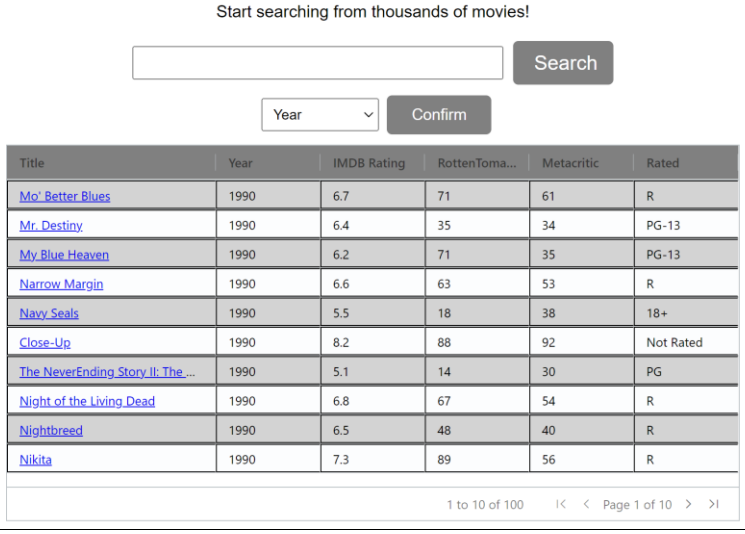


Figure 32. Logout

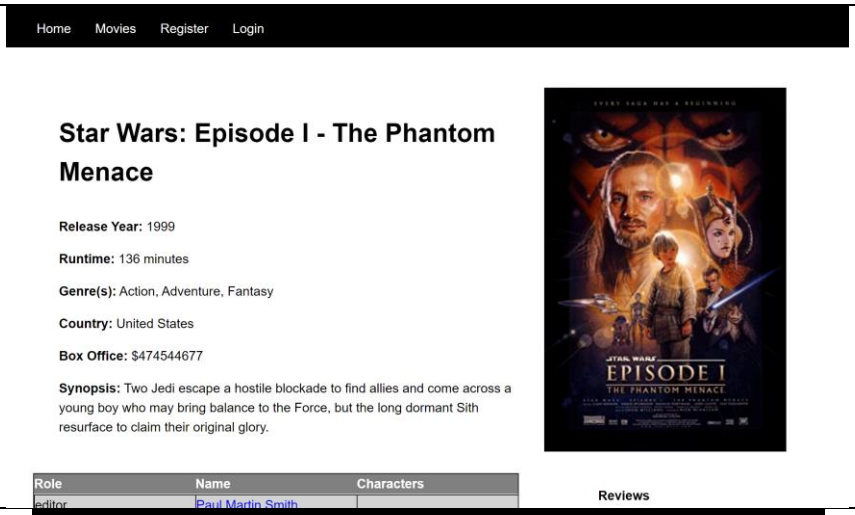
References

1. AG Grid. (n.d.). React Grid: The Enterprise Datagrid for React. Retrieved May 11, 2023, from https://www.ag-grid.com/react-grid/
2. Chart.js. (n.d.). Simple, clean and engaging charts for designers and developers. Retrieved May 11, 2023, from https://www.chartjs.org/
3. Figma. (n.d.). Retrieved Retrieved May 11, 2023, from https://www.figma.com/
4. React Training. (n.d.). React Router: Declarative Routing for React.js. Retrieved May 11, 2023, from https://reactrouter.com/en/main/start/overview
5. React-Chartjs-2. (n.d.). React Wrapper for Chart.js 2 Open-Source Library. Retrieved May 11, 2023, from https://react-chartjs-2.js.org/

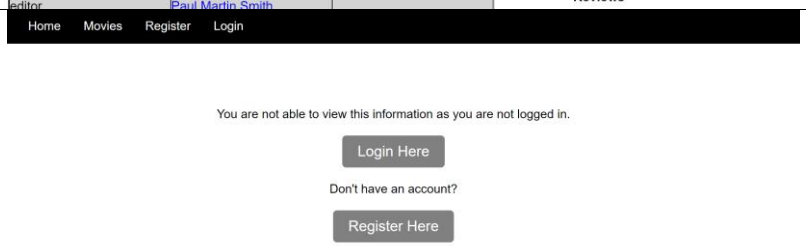
Appendix A

Figure No.	
1	
2	
3	

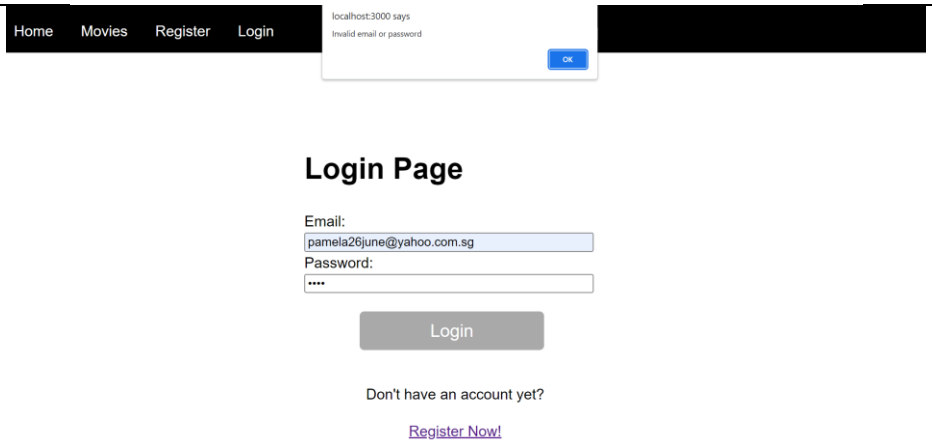
4


Home Movies Register Login
Star Wars: Episode I - The Phantom Menace
Release Year: 1999
Runtime: 136 minutes
Genre(s): Action, Adventure, Fantasy
Country: United States
Box Office: $474544677
Synopsis: Two Jedi escape a hostile blockade to find allies and come across a young boy who may bring balance to the Force, but the long dormant Sith resurface to claim their original glory.
EPISODE I
THE PHANTOM MENACE
Role
Name
Characters
editor
Paul Martin Smith
Reviews

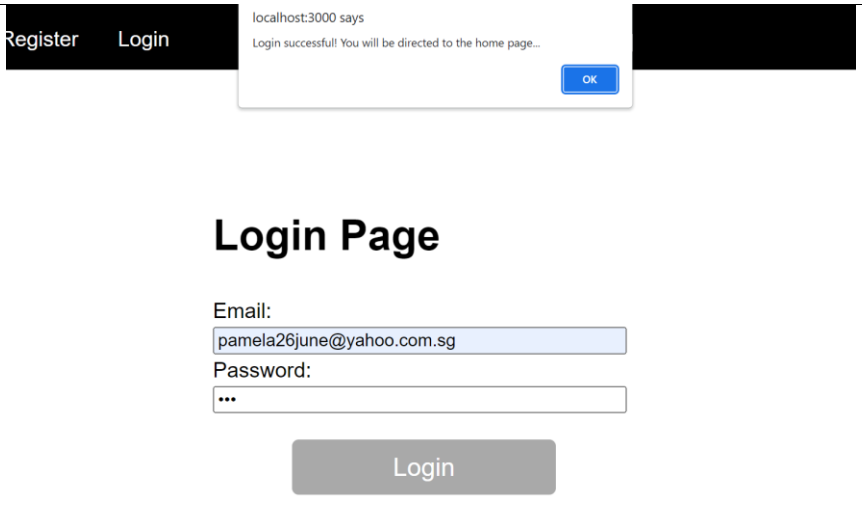
5

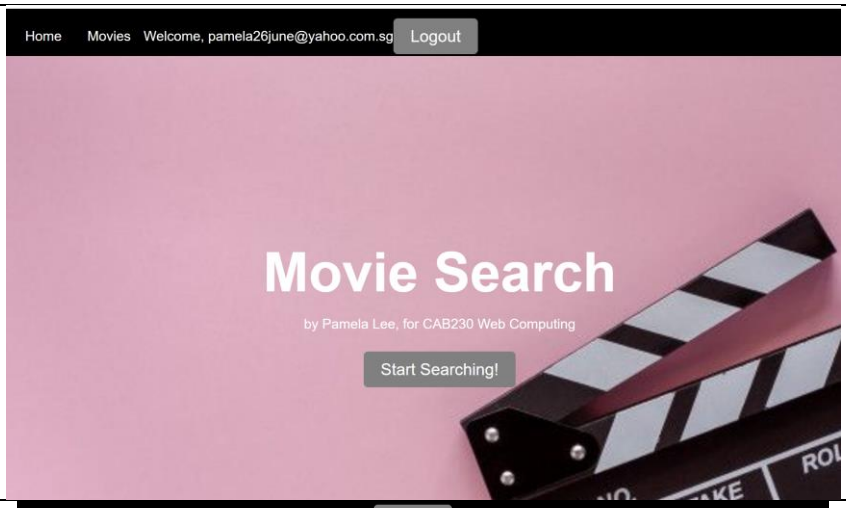

Home Movies Register Login
You are not able to view this information as you are not logged in.
Login Here
Don't have an account?
Register Here

6

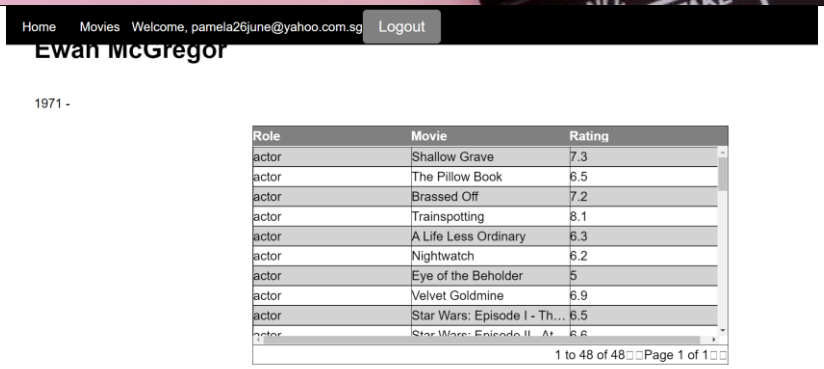

Home Movies Register Login
localhost:3000 says
Invalid email or password
OK
Login Page
Email:
pamela26june@yahoo.com.sg
Password:
Login
Don't have an account yet?
Register Now!

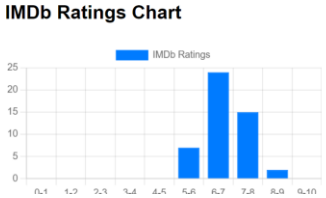
7


Register Login
localhost:3000 says
Login successful! You will be directed to the home page...
OK
Login Page
Email:
pamela26june@yahoo.com.sg
Password:
Login

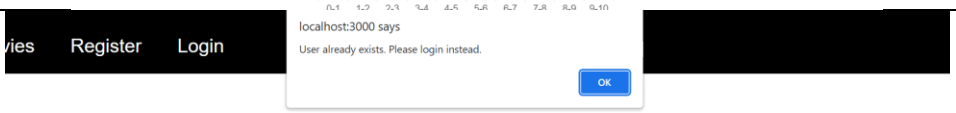

Home
Movies
Welcome, pamela26june@yahoo.com.sg
Logout
Movie Search
by Pamela Lee, for CAB230 Web Computing
Start Searching!

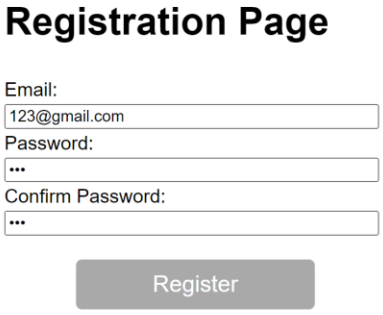
8

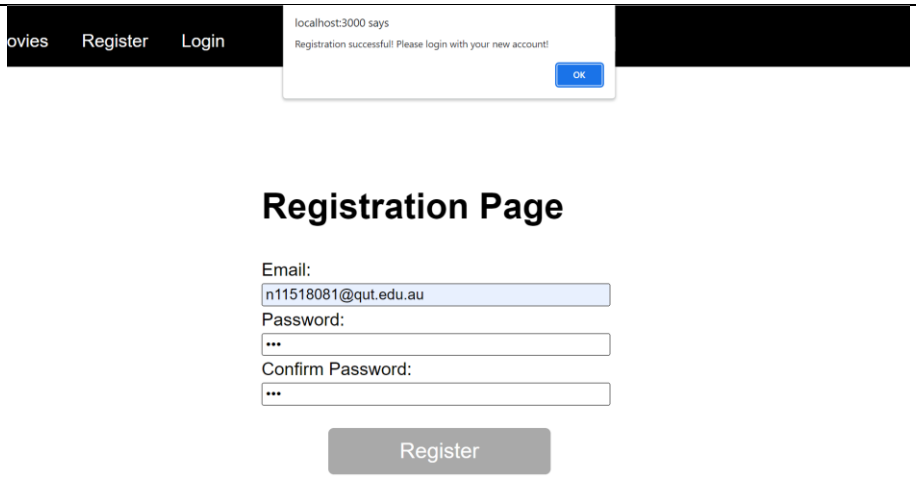

Home
Movies
Welcome, pamela26june@yahoo.com.sg
Logout
Ewan McGregor
1971 -
Role
Movie
Rating
actor
Shallow Grave
7.3
actor
The Pillow Book
6.5
actor
Brassed Off
7.2
actor
Trainspotting
8.1
actor
A Life Less Ordinary
6.3
actor
Nightwatch
6.2
actor
Eye of the Beholder
5
actor
Velvet Goldmine
6.9
actor
Star Wars: Episode I - Th...
6.5
1 to 48 of 48
Page 1 of 1


IMDb Ratings Chart
IMDb Ratings
25
20
15
10
5
0

9


ies
Register
Login
localhost:3000 says
User already exists. Please login instead.
OK


Registration Page
Email:
123@gmail.com
Password:
Confirm Password:
Register

10	 ovies Register Login localhost:3000 says Registration successful! Please login with your new account! OK Registration Page Email: n11518081@qut.edu.au Password: ••• Confirm Password: ••• Register