ASSIGNMENT 2: CLIENT-SIDE REACT APPLICATION

CAB230

Dylan Hessing
N11245310

Contents

Introduction

Purpose and Description

This web application offers users an interactive platform to explore a curated selection of movies, complete with detailed information such as plot summaries, cast and crew listings, box office performance, and aggregated ratings from IMDb, Rotten Tomatoes, and Metacritic. Users can also view filmographies and rating histories for actors, directors, and other contributors, with personalized access enabled through secure user authentication. The data is accessed through the rest API published at http://4.237.58.241:3000/.

The app is designed with a focus on modern aesthetics and smooth user experience, using Tailwind CSS for a clean, consistent interface. Large datasets are efficiently handled using AG Grid with infinite scrolling, while Axios is used for streamlined, secure API calls, including JWT-based authenticated requests for user login.

To enhance interactivity, the app features real-time search filtering, Chart.js-based visualizations of rating distributions, and intuitive navigation managed by a client-side router. These elements work together to create a responsive and user-friendly experience across the platform.

Completeness and Limitations

The developed application successfully meets all the requirements listed in the specification for a grade of 7. All listed functionality to achieve this level, and the levels below have been successfully implemented to a professional standard with solid performance. The application includes secure user authentication with refresh tokens. Data is presented to users through a clean, intuitive and responsive design. A react-router has been implemented for smooth navigation between pages. The application makes full use in exposing the capabilities of endpoints to users in many ways. For example, infinite scrolling to handle paginated search data and the use of charts to provide visual insights into rating distributions.

Whilst the application is mostly complete, there are some shortcomings around the handling of different window sizes, with smaller screens suffering. This is mostly around AG-Grids, however scrolling has been implemented to allow all information to be accessed on smaller screens. There are also some unresolved rendering issues regarding the vertical spacing of components on pages.

Use of End Points

The application makes use of all seven end points of the movie information API.

/movies/search

The movies page exposes the /movies/search API endpoint and displays results using an AG-Grid prepopulated with all available movies. It supports infinite scrolling for pagination, real-time title search, and year-based filtering. The grid has a fixed height with self-contained vertical and horizontal scrolling. Ratings are color-coded (red to green), and classifications are shown as images for improved user experience. The movie titles are clickable, linking to the /movies/data/{imdbID} endpoint

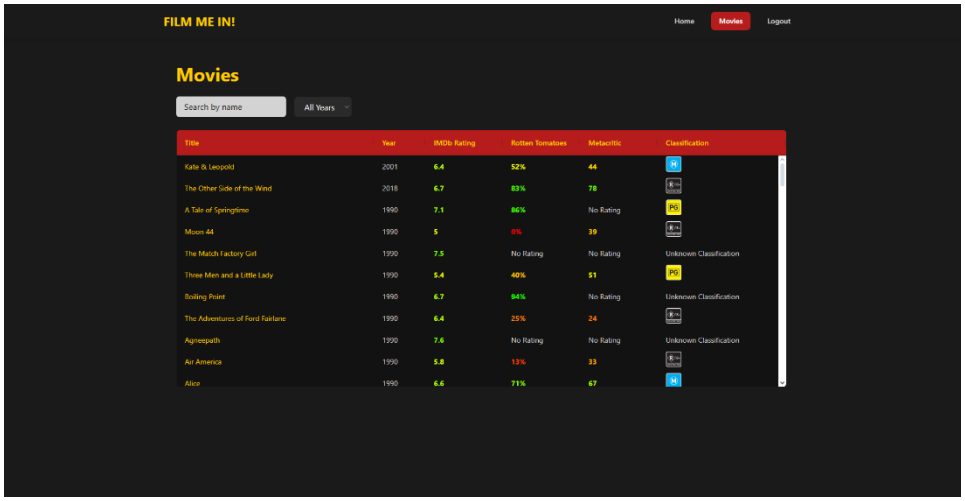


Figure 1: /movies/search endpoint

/movies/data/{imdbID}

The movie details page exposes the /movies/data/{imdbID} endpoint, displaying structured movie information alongside the film's poster at the top. Below, an AG-Grid presents the cast with sortable columns for role, name, and character (if applicable). Cast names are clickable, linking to the /people endpoint for further details.

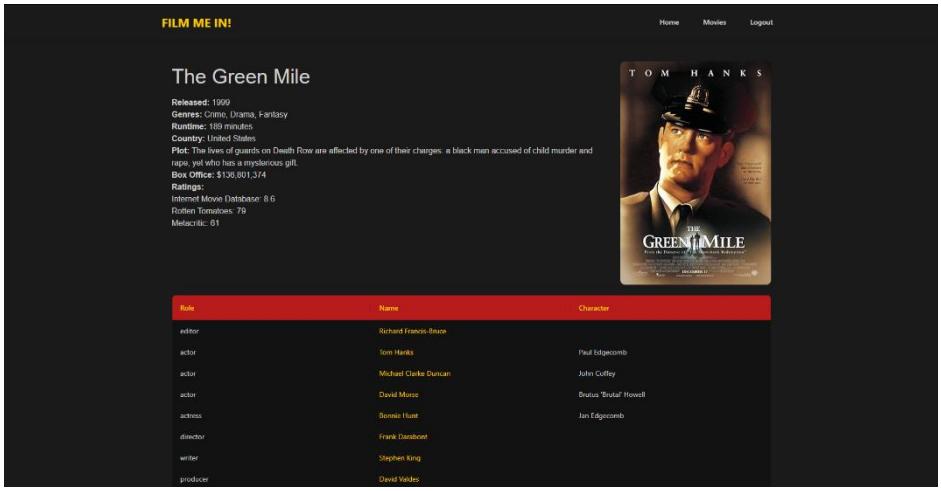


Figure 2: /movies/data/{imdbID} endpoint

/people/{id}

The person detail page exposes the /people/{id} endpoint, displaying cast member data in a sortable AG-Grid where movie titles link to their respective detail pages. A chart below visualizes the distribution of IMDb ratings across their films. When many movies are listed, hovering over bars reveals movie titles. The chart order reflects the current table sorting, ensuring consistency between the data views and creating an interactive user experience.

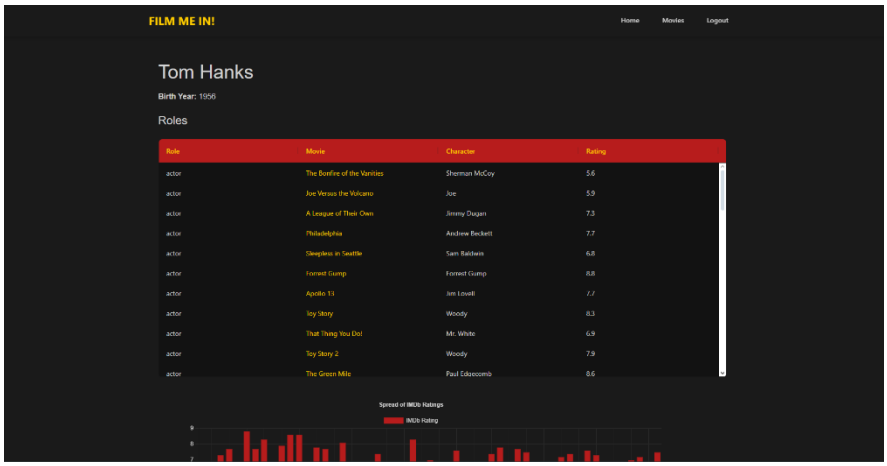


Figure 3: /people/{id} endpoint 1

Figure 4: /people/{id} endpoint 2

/user/register

The register page exposes the /user/register endpoint and is designed with a familiar, user-friendly layout. It handles input validation, duplicate registrations, and format errors, with these cases covered in the test plan.

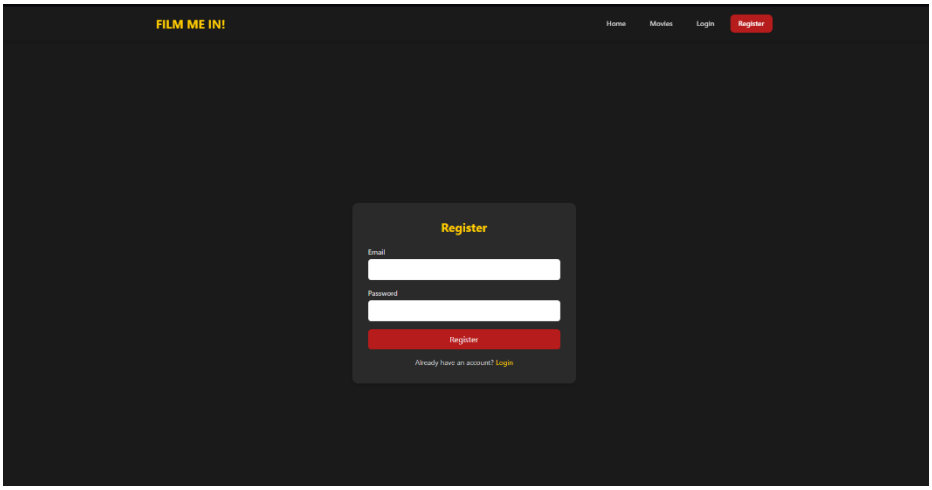


Figure 5: /user/login endpoint

/user/login

The login page exposes the /user/login endpoint and features a clear, user-friendly layout. It handles invalid input formats and incorrect credentials, with these scenarios documented in the test plan.

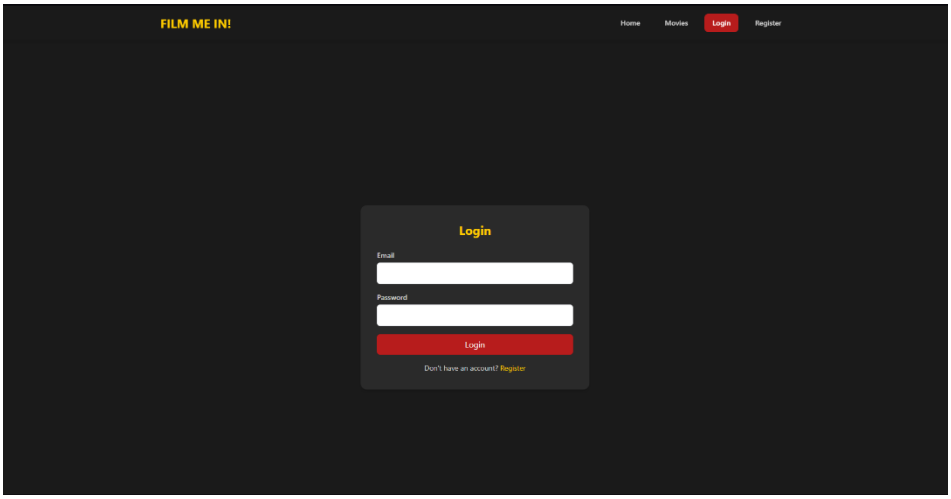


Figure 6: /user/register endpoint

/user/refresh

The /user/refresh endpoint is not exposed to users directly. It is used internally to refresh JWT tokens via Axios by intercepting 401 responses and retrying failed requests with a new token.

/user/logout

The /user/logout endpoint is accessed by logged-in users via the Logout button. It logs the user out by invalidating and removing stored tokens.

Figure 7: /user/logout endpoint

Modules Used

The following external modules were used in the implementation of the application:

- **ag-grid-react**
  A powerful data grid component for React applications, used here to efficiently render large datasets with features like infinite scrolling and filtering.
  https://www.npmjs.com/package/ag-grid-react
- **axios**
  A promise-based HTTP client for making API requests. It simplifies sending authenticated requests and handling responses and errors.
  https://www.npmjs.com/package/axios
- **chart.js**
  A flexible JavaScript charting library used to create visual representations of movie rating distributions in the app.
  https://www.npmjs.com/package/chart.js
- **react-chartjs-2**
  A React wrapper for Chart.js that allows Chart.js components to be used easily within a React project.
  https://www.npmjs.com/package/react-chartjs-2
- **react-router-dom**
  Provides declarative routing for React apps. It enables page navigation and maintains a clean URL structure throughout the app.
  https://www.npmjs.com/package/react-router-dom
- **tailwindcss**
  A utility-first CSS framework used to design a clean, modern, and responsive user interface.
  https://www.npmjs.com/package/tailwindcss

Application Design

Navigation and Layout

During the design process, I focused on creating a user-friendly and intuitive interface that supports smooth navigation and logical interaction patterns. I began with low-fidelity sketches (shown below) of the main endpoints to map out the core user journey which helped map out the flow between pages. Whilst these were helpful in the initial phase, many changes were made to these designs due to the nature of the data and technical limitations. One example is the handling of ratings on the person details page. While the original plan was to group ratings by number in the chart, it was later decided to display all ratings to encourage user interaction through table sorting. Other pages such as the login and register page took inspiration from existing implementations to provide users with familiar interactions.

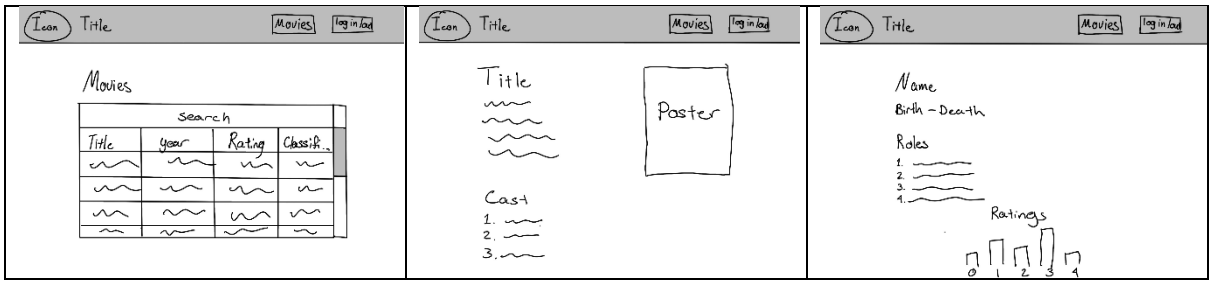


Figure 8: Low-Fidelity Wireframes

Most of the navigation is handled through a Navbar that remains consistent throughout the application, allowing users to access the core pages. This design choice was made to give the site a familiar structure and ensure users can always return to a known location, regardless of where they are in the app. Beyond this, contextual navigation is incorporated throughout the application. For example, movie titles are clickable across tables and charts taking users directly to the movie details page. Within each movie detail page, actor and director names are also clickable, routing users to their individual profile pages. These interconnected links encourage exploration and reflect the relational nature of movie data. Logical page transitions are also present, such as linking the login and register page, and automatic redirection to the last visited protected page after a user logs in, in an effort to achieve the smoothest user experience possible.

The application flow is visualised below:

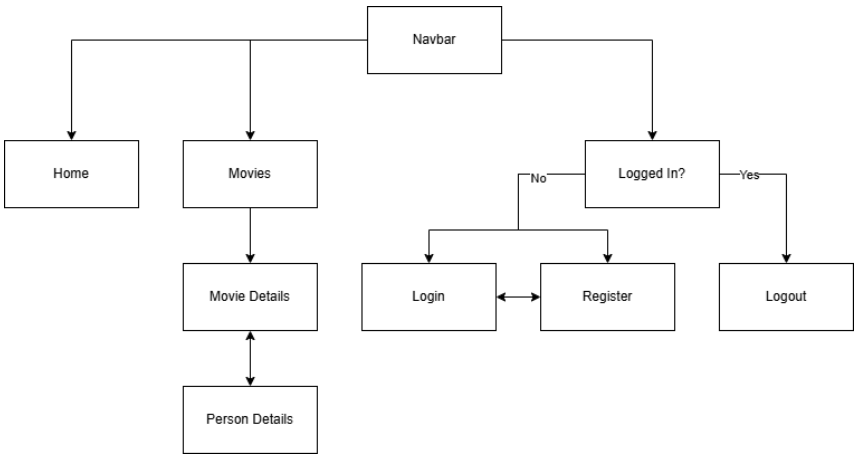


Figure 9: Application Flow

Usability and Quality of Design

Overall, the application demonstrates strong usability, supported by a clean, modern layout and intuitive navigation. Tailwind CSS enforces consistent styling across fonts, sizes, and colours, contributing to a polished and familiar user experience aligned with common web standards. Interactive elements such as buttons and links are clearly distinguishable, with hover effects and appropriate sizing that enhance discoverability.

Navigation is streamlined through a fixed navbar and contextual links (e.g., clickable movie titles and actor names), enabling smooth content exploration. The simple layout and restrained content density promote ease of use across user groups.

Affordance and learnability are well addressed—buttons change colour on hover to signal interactivity, and the landing page provides clear guidance to key areas like movies and registration. These design choices help first-time users navigate with minimal effort.

AG Grid components are visually integrated with the app's theme, and form feedback during login and registration is informative and clearly styled. However, usability compromises exist—for example, enabling horizontal scrolling improves mobile accessibility but introduces clutter on larger screens. There are some rendering issues present with the AG-Grid, such as the bottom borders not rounding when the height of the grid is less than the preset height of the div containing it and the grid taking up the space of the preset height even when the content does not require it to.

One area needing improvement is loading state handling. For example, while loading data in AG-Grid components, entries filled with 'Invalid' are displayed which may confuse users. Realtime searching, although good for responsiveness, causes these entries to appear briefly on every keystroke. Implementing skeleton loaders or styled spinners would enhance perceived performance and maintain engagement.

Accessibility remains limited. The lack of theme customization may affect colour-blind users, and colour-coded indicators like rating-based text can be hard to interpret for users with vision impairments. Broader accessibility support—such as improved colour contrast, keyboard navigation, and screen reader compatibility—should be considered.

In conclusion, the application is visually consistent, easy to use, and thoughtfully structured, but would benefit from improved loading feedback, enhanced accessibility features, and refinement of layout responsiveness to ensure a more inclusive and seamless experience.

Accessibility

The website was analysed and rated against the Priority 1 accessibility requirements provided by the W3C (1999).

Requirement	Achievement	Reasoning
Provide a text equivalent for every non-text element – alternatives to images, symbols, scripts, graphical buttons, sounds, audio and video files and so on.	Pass	Text equivalents for images are available through captions.
Ensure that all information conveyed with colo**u**r is also available without colo**u**r, for example from context or markup.	Pass	All coloured information is accessible through mark up.
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.	Pass	Semantic elements have been used to ensure that even with no styles loaded, the content is still structured and readable.
Ensure that text equivalents are updated when dynamic content changes.	Pass	Text equivalents are updated when dynamic content changes.
Avoid causing the screen to flicker.	Marginal Fail	Screen flickering is mostly not present. Slight flickering present when entering titles into search bar on movies page)
Use the clearest and simplest language appropriate for a site's content.	Pass	Language is simple and clear, appropriate for the sights content.

For tables, identify row and column headers – clearly differentiated from the data.	Pass	Through theming the header row is clearly differentiated from the data in tables.

Although the application meets many Priority 1 requirements, there are several Priority 1 and 2 accessibility requirements that were not addressed. This is expected, as the primary focus of development was on functionality and usability rather than accessibility.

Technical Description

Architecture

The application follows a standard React architecture using Vite as the build tool. The source code is organized within the src directory according to modular design principles, promoting clarity, maintainability, and scalability. Currently contained in the src directory is:

- **layouts/**: Contains reusable layout components for the application. These components are typically used to wrap other pages and define a consistent structure (e.g., headers, footers, side navigation) across various pages.
- **pages/**: Includes top-level components corresponding to application routes (e.g., Home, Login, Movies). Each file in this folder represents a user-facing page.
- **api/**: Provides utility modules for handling API interactions, such as user authentication and data fetching. These modules encapsulate external communication logic.
- **assets/**: Stores static files including images, icons, and stylesheets used throughout the application.
- **styles/**: Contains CSS/SCSS files or styled components used for the application's overall styling.
- **App.jsx and main.jsx**: Serve as the application’s entry points. main.jsx initializes the React root, while App.jsx configures routing and wraps the app in global providers.

This structure was chosen to clearly separate concerns, simplify development workflows, and support future scalability. The below images capture the main and src directories respectively

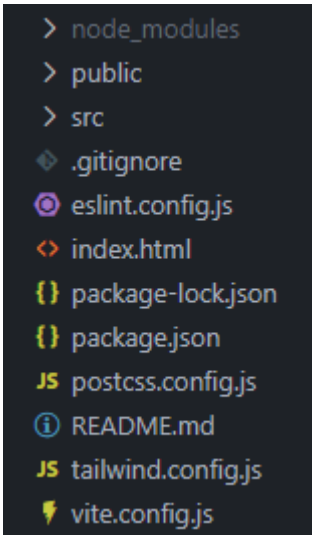


Figure 10: Main Directory

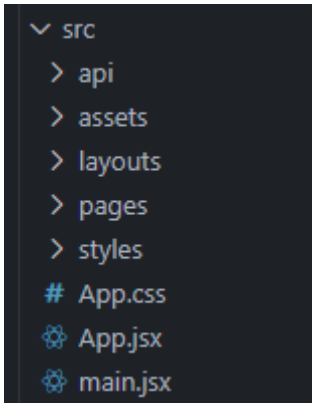


Figure 11: Src Directory

Test Plan

Task	Expected Outcome	Result	Screenshot (appendix A)
Register User	User registers with valid details; account is created and user is automatically logged in.	Pass	n/a
Log in as User	User logs in with correct credentials; they are redirected to the home page. Logout is shown in navbar. Login and register buttons not present on landing page.	Pass	1
Search for movie	Entering a movie title displays matching movies in the search results.	Pass	2
Infinite scroll on movie list	As the user scrolls down, more movies load automatically (infinite scroll) without errors.	Pass	n/a
Click movie in movie list (movie page)	Clicking a movie item in a list navigates to that movie's detail page.	Pass	n/a
Click cast member from cast list (logged in)	On a movie page, clicking a cast member's name (while logged in) opens that person's detail page.	pass	n/a
Click movie in filmography on person's page	On a person's page, clicking a listed movie opens that movie's detail page.	pass	n/a
Access people page without logging in	Accessing the protected People page redirects to an access denied message.	Pass	3

Login after redirect from protected page	After being redirected to login for a protected page, submitting valid credentials returns user to that page.	Pass	n/a
Open login page from a non-landing page	User can navigate to the login page from any page in the app (via navbar or link) and is returned to previous page upon login.	Pass	n/a
Open login page via registration page	The user is redirected to the login page.	Pass	n/a
Open registration page via login page	The user is redirected to the registration page.	Pass	n/a
Navigate: Login → Register → Login	When the user logs in they should not redirected to the home page.	Pass	n/a
Click app title (navbar)	Clicking the web app title in the navbar navigates to the home/landing page.	Pass	n/a
Click Home button	Clicking the “Home” button navigates to the main landing/home page.	Pass	n/a
Click Movies button	Clicking the “Movies” button navigates to the movies listing page.	Pass	n/a
Click Register button	Clicking the “Register” button opens the user registration page.	Pass	n/a
Click Log in button	Clicking the “Log in” button opens the login page.	Pass	n/a
Click Log out button	Clicking “Log out” signs the user out and returns to the landing or login page.	Pass	n/a
Invalid email format on register	Entering an incorrectly formatted email shows a validation error and registration is blocked.	Pass	4
Invalid password format on register	Entering a password that doesn’t meet criteria shows a validation error (e.g. too short) and blocks registration.	Pass	5
Incorrect login credentials	Entering wrong email/password shows an error message and the user remains logged out.	Pass	6
Non-existent path (404)	Navigating to an invalid URL shows a 404 error page or redirects to a safe page.	Pass	7
Sort table by category (movie details)	Clicking table column headers sorts the data; rows reorder correctly according to the selected category.	Pass	8

Sort table by category (person details)	Clicking table column headers sorts the data; rows reorder correctly according to the selected category. Chart is consistent with ordered table.	Pass	9
Filter movies by year	Applying a year filter in search limits results to movies from that year or range.	Pass	10
Search for non-existent movie	Searching for a non-existent title shows “End of Results” .	Pass	11
JWT token is refreshed on expiration with valid Refresh Token	New and valid JWT token is stored in local storage.	Pass	n/a
Access protected route with expired/invalid JWT	Using an expired or invalid token to access a protected page redirects to login or shows a session expired message.	Pass	n/a
Submit form with empty fields	Attempting to submit forms with all required fields empty triggers validation errors and prevents submission.	Pass	12
Enter whitespace-only input	Input fields containing only spaces/tabs are treated as empty; validation errors appear as with blank fields.	Pass	13
Refresh on protected page	Refreshing a protected page while logged in should keep the session; if not logged in, it should redirect to login.	Pass	n/a
Attempt action with incomplete data	Trying to perform actions (e.g. submit forms) with missing required data should trigger validation messages and not break functionality.	Pass	14
Register with an existing email	Trying to register with an email already in use shows an "email already exists" error and prevents duplicate account.	Pass	15
Simulate API timeout or no response	If the backend API times out or fails, the app should display a user-friendly error (e.g. "Server unavailable") without crashing.	Pass	16
Press browser Back button after logout	After logging out, using the browser back button should not return to protected content (should redirect to login).	Pass	17

Difficulties/Exclusions/Unresolved & persistent errors

One of the main major roadblocks I faced was implementing refresh tokens. This was due to incorrect use of the endpoint and a lack of thorough testing upon initial

implementation. This caused a persistent bug, where on a failed refresh call, it would enter an infinite loop of sending API calls which would exceed the rate limit and block requests. It was difficult to locate the cause of this bug as its occurrence was low due to the time it took for the JWT token to expire. Once the issue was tracked down it was a simple fix.

Styling and customizing AG Grid to match the app's aesthetic and feel intuitive for users also proved time-consuming. The library is powerful but not inherently user-friendly when it comes to styling, especially with Tailwind CSS. I resolved this by using custom CSS overrides in combination with Tailwind classes to maintain visual consistency and improve usability, particularly for mobile and smaller screen sizes. There are also persistent issues with sizing vertically, where pages that should require no scrolling exceed the window and cause overflow. The AG-Grid also presented an unresolved issue of visualizing data loading from the API. This is currently not handled and the grid displays invalid data whilst it is loading.

Another persistent styling issue encountered is regarding vertical spacing; where pages that do not require much vertical space overflow and require scrolling to see the bottom of the screen.

Extensions

There are many extensions that could be made to the application to further enhance it. This includes but is not limited to:

1. Adding Theme customization (Light/Dark Mode, High Contrast)
    a. Enhances accessibility
    b. Makes the app feel more modern and inclusive
2. Implement Proper loading states
    a. Use skeleton loaders or spinners to give feedback during data fetching (especially for AG-Grid)
    b. Improves perceived performance and reduces confusion.
3. Improve Mobile Responsiveness
    a. Adapt AG Grid and layouts better for small screens (e.g., collapse less important columns).
    b. Ensures good usability across all devices.
4. Integrate Tooltips and Button Labels
    a. Improves clarity of icons and actions, especially helpful for new users or visually impaired users.
5. Enhance Accessibility Features
    a. Add ARIA labels, keyboard navigation support, and sufficient contrast.
    b. Broadens the app's usability and meets inclusivity standards.
6. Allow Favourites, Custom Lists, and Watchlists
    a. Users can create their own personalized movie lists, add favourites, and maintain a watchlist.
    b. Encourages user interaction, engagement, and long-term retention.
7. Allow User Reviews and Comments
    a. Users can leave reviews or comments on movies, adding social proof and a sense of community.
    b. Promotes user-generated content and enhances interactivity.
8. Allow Search by Actor, Genre, etc.
    a. Increases usability of search function

User guide

Navbar

The navbar is used for the core navigation throughout the app. Four pages can be accessed through the navbar:

1. Home - via the website name or the home button
2. Movies - via the movies button
3. Login - via the login button (only visible if not logged in)
4. Register - via the register button (only visible if not logged in)

Additionally, logged in users will be presented with a log out button instead of Login and Register buttons. Clicking this will automatically log the user out and redirect them to the login page.

Figure 12:Navbar (logged out)

Figure 13: Navbar (logged int)

Home Page

The home/landing page is the initial page users will be presented with when accessing the site. Users are greeted with a welcome message and provided three navigation options:

1. Start Browsing – redirects to the movies page
2. Login – redirects to the login page
3. Register – redirects to the register page

Logged in users will only see option 1.

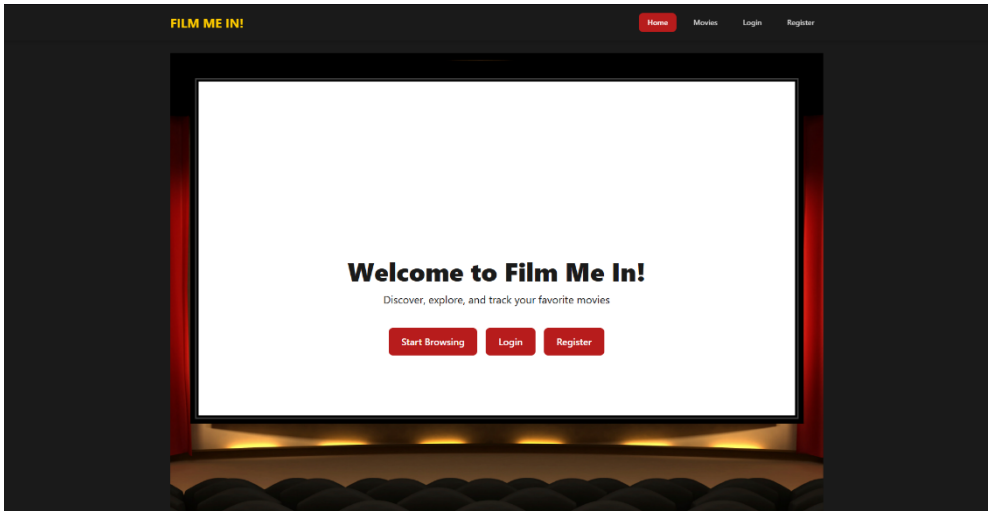


Figure 14: Landing Page (logged out)

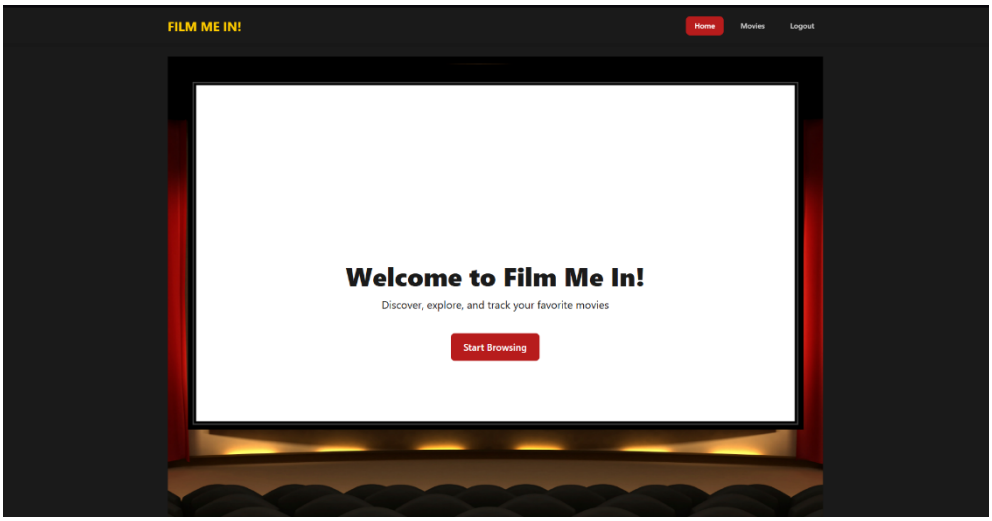


Figure 15: Landing page (logged in)

Movies Page

The movies page is where users can browse movies available on the API. Users can utilize the provided search bar to search for movies by title. The year drop down can be used to filter the results by a year. This table has no sorting options. The title of any movie can be clicked, which will navigate to the movie details page for the given movie. Horizontal and vertical scroll bars are available on the bottom and right of the grid to view all the data available.

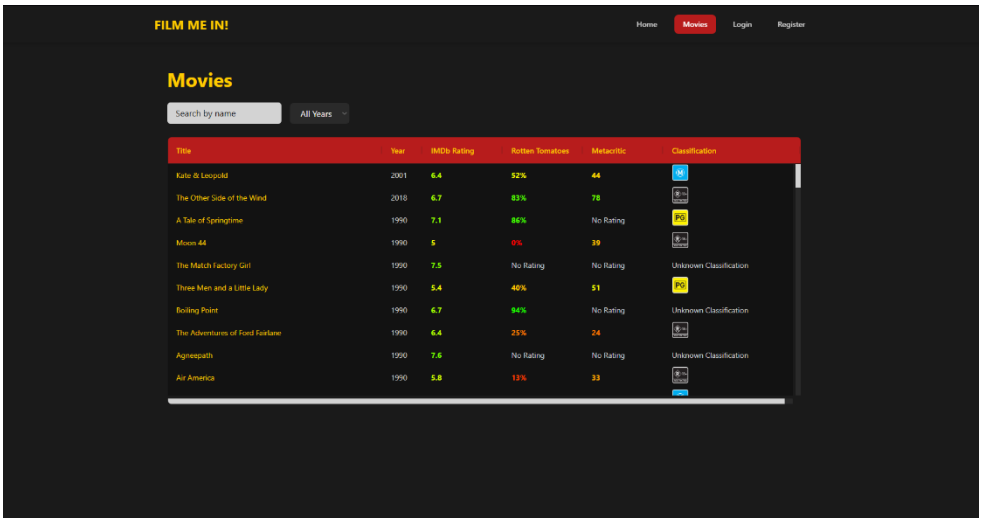


Figure 16: Movies Page

Login Page

Existing users can enter an email in the email field and a password in the password field and click the login button to login. If the credentials are correct, the user will be logged in. If the credentials are incorrect the user will be informed of the error and prompted to retry. The user can also navigate to the register page by clicking the yellow word ‘Register’.

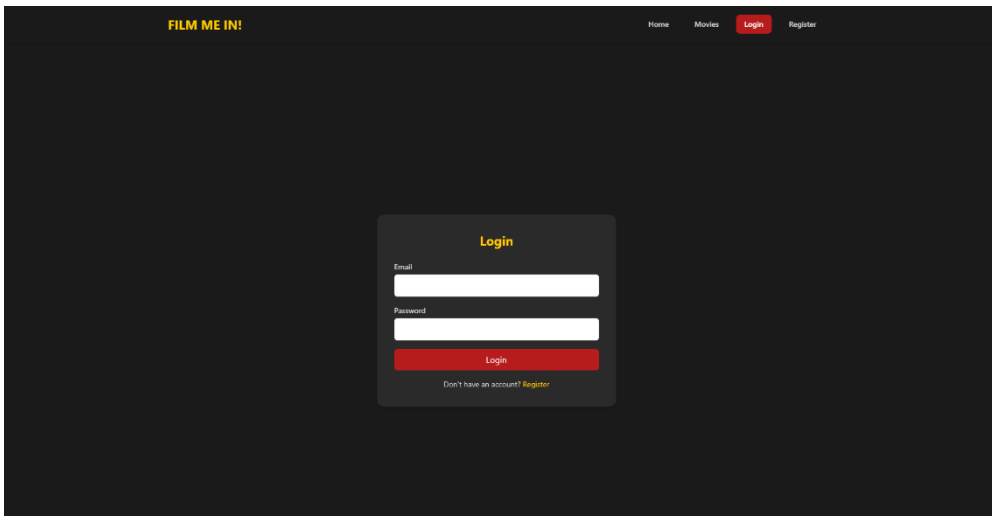


Figure 17: Login Page

Register Page

New users can enter an email in the email field and password in the password field and click the Register to register. If the email already exists, users will be presented with a user already exists message. If the email is incorrectly formatted, the user will be informed. If the password does not meet the password requirements the user will be informed. The user can also access the login page via the yellow word ‘Login’.

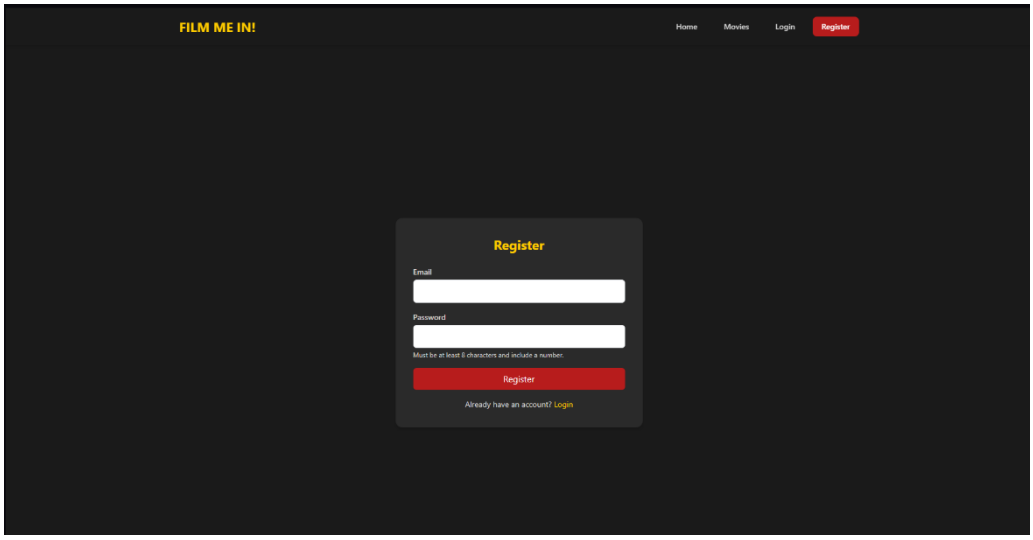


Figure 18: Register Page

Movie Details Page

The movie details page presents key data for a given movie. Users can scroll through the cast of the movie and click on names to be redirected to the person details page for the given person. The table can be sorted alphabetically or numerically by clicking on the column headers. A vertical scroll bar is present on the right of the grid if the entries overflow.

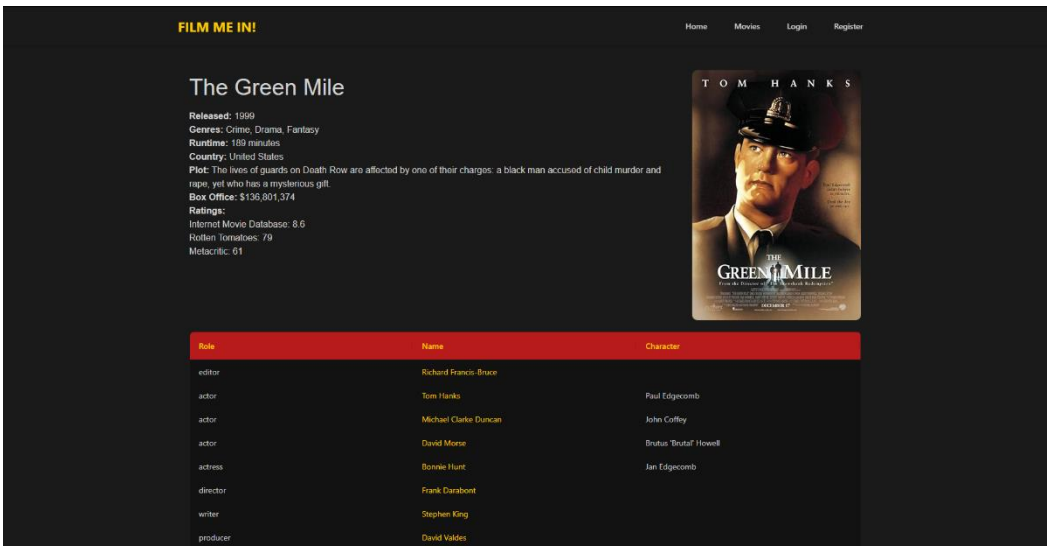


Figure 19: Movie Details Page

Person Details Page

The person details page presents key data for a given person. This page is only accessible to logged in users. If a user attempts to access the page prior to logging in, they will be prompted to log in. Logged in users will be presented with the filmography of the given person. They can scroll through the filmography and see the distribution of ratings in the chart. The user can sort the table in alphabetic or numeric order by clicking on the column headers. When sorting by header, the chart will rearrange itself in that order. The movie titles are clickable and redirect to the movie details page for that movie. A vertical scroll bar is available to the right of the grid if the entries overflow.

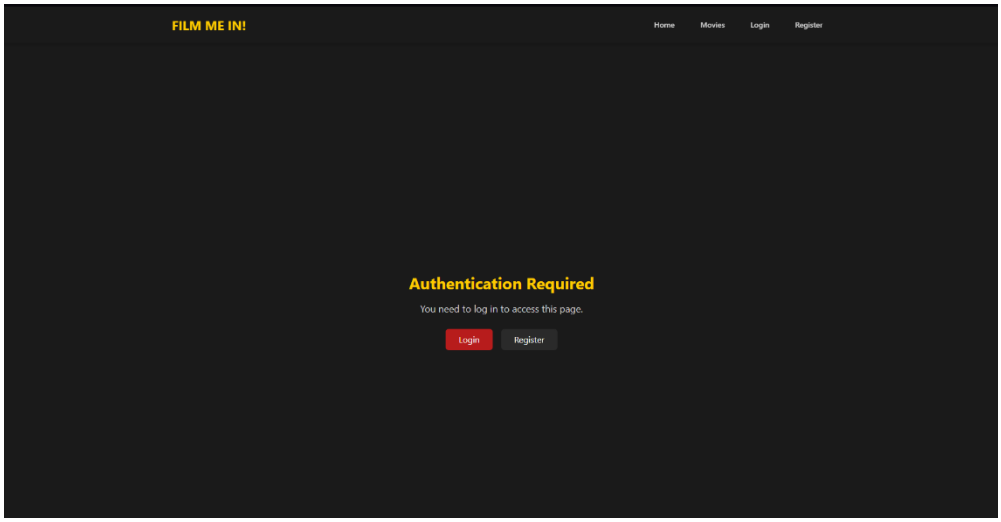


Figure 20: Person Details Page (logged out)

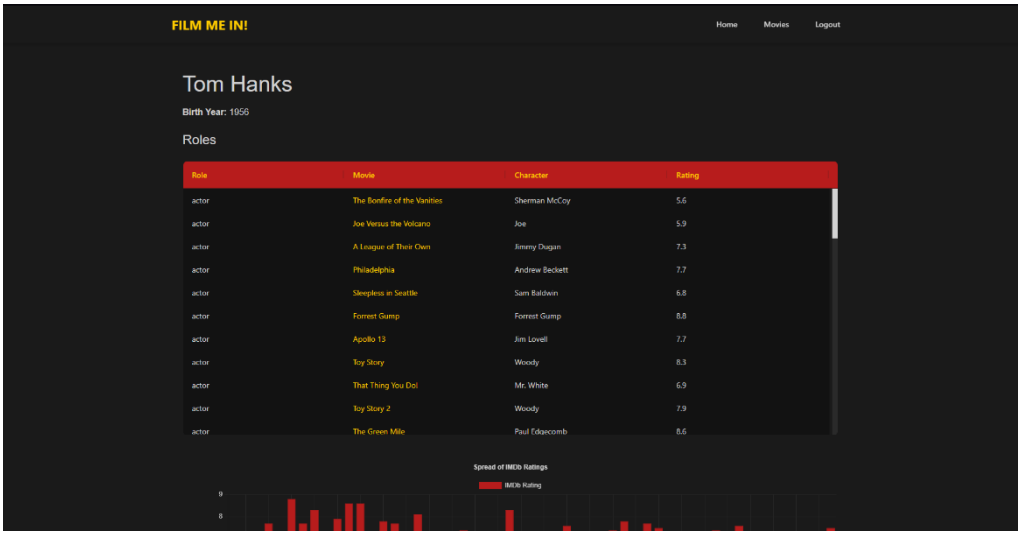


Figure 21: Person Details Page (Logged in) 1

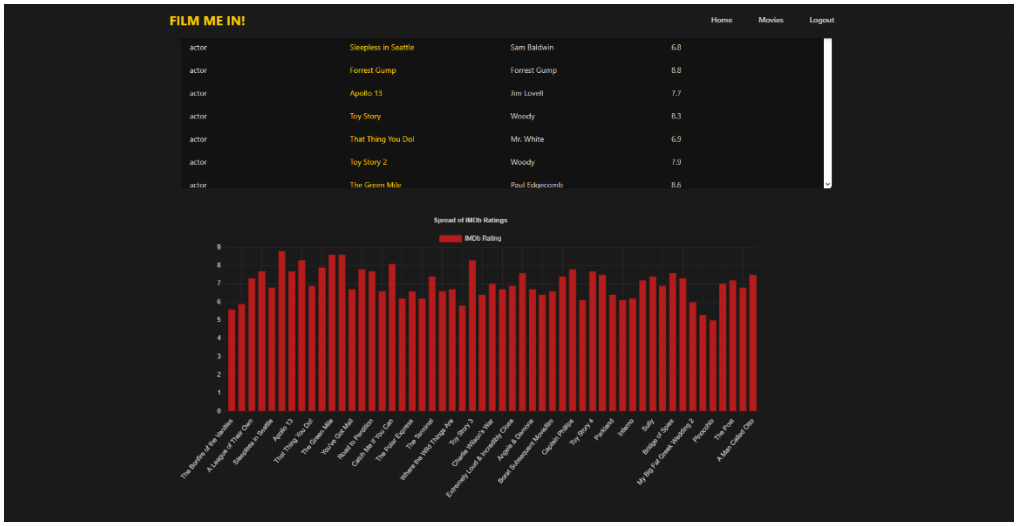


Figure 22: Person Details Page (Logged in) 2

References

Ag Grid Ltd. (2025). Quick Start. https://www.ag-grid.com/react-data-grid/getting-started/

Axios (2025) Getting Started. https://axios-http.com/docs/intro

Chart.js (2025). Chart.js. https://www.chartjs.org/docs/latest/

Tailwindcss (2025). Get started with Tailwind CSS. https://tailwindcss.com/docs/installation/using-vite

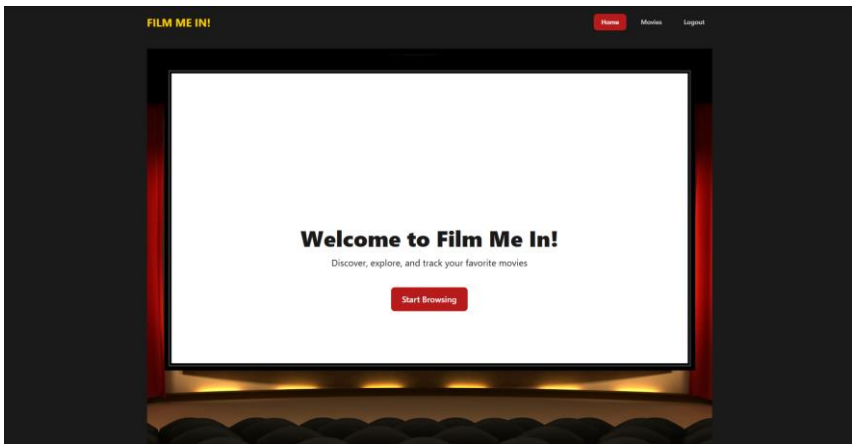
React Router (2025). React Router Home. https://reactrouter.com/home

W3C. (1999) Checklist of Checkpoints for Web Content Accessibility Guidelines 1.0. https://www.w3.org/TR/WAI-WEBCONTENT/full-checklist

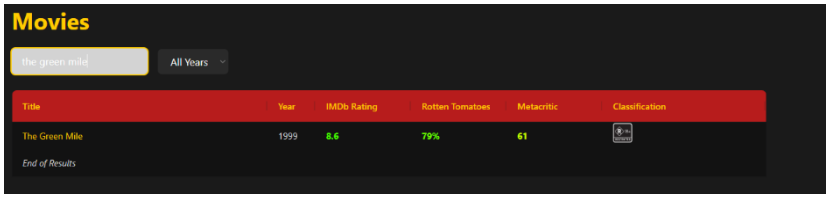
Appendix

Appendix 1. Test Screenshots

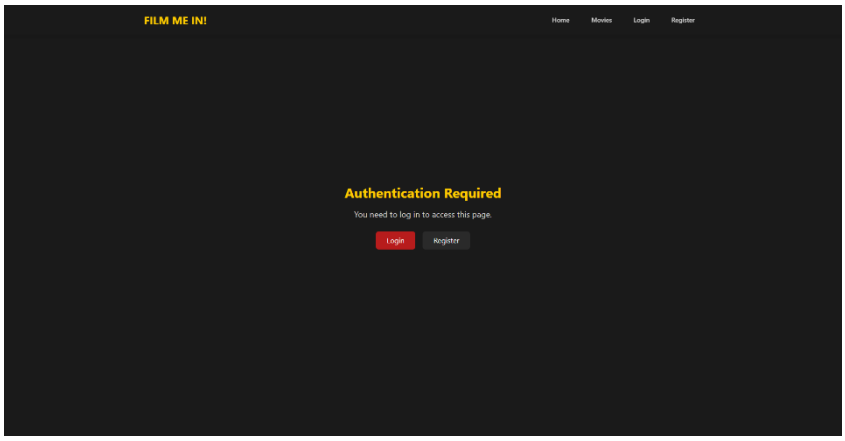
1.



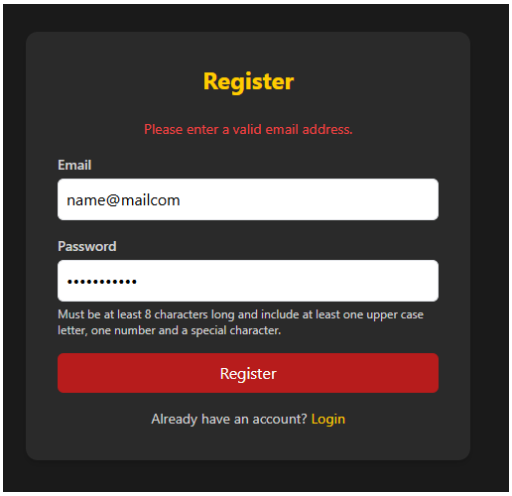
2.



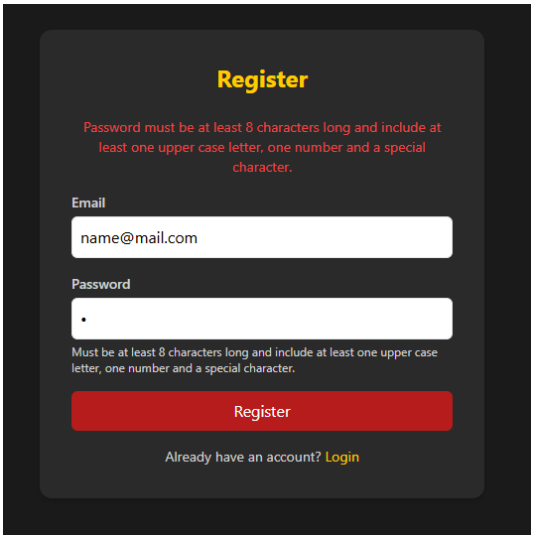
3.



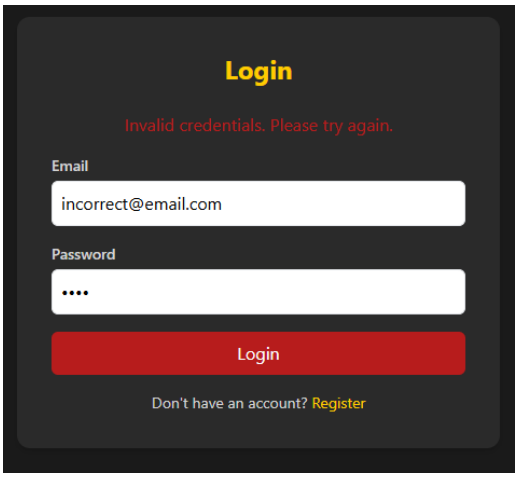
4.


Register
Please enter a valid email address.
Email
name@mailcom
Password
Must be at least 8 characters long and include at least one upper case letter, one number and a special character.
Register
Already have an account? Login

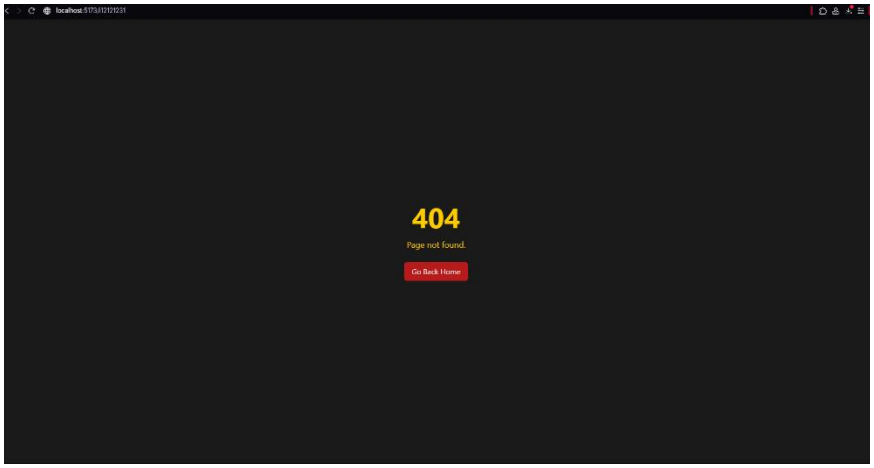
5.


Register
Password must be at least 8 characters long and include at least one upper case letter, one number and a special character.
Email
name@mail.com
Password
Must be at least 8 characters long and include at least one upper case letter, one number and a special character.
Register
Already have an account? Login

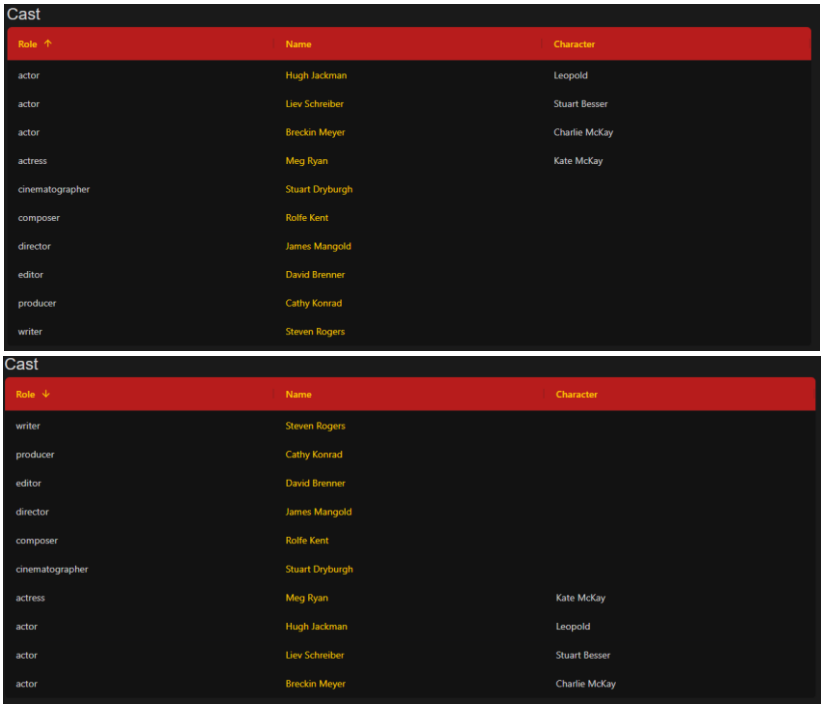
6.


Login
Invalid credentials. Please try again.
Email
incorrect@email.com
Password
Login
Don't have an account? Register

7.


404
Page not found.
Go Back Home

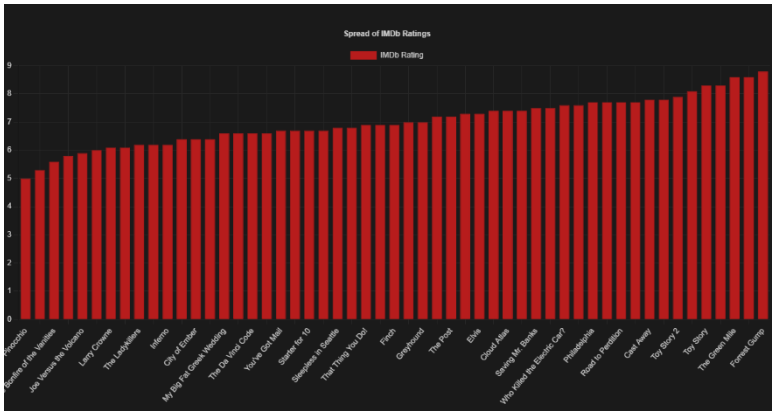
8.


Cast
Role
Name
Character
actor
Hugh Jackman
Leopold
actor
Liev Schreiber
Stuart Besser
actor
Breckin Meyer
Charlie McKay
actress
Meg Ryan
Kate McKay
cinematographer
Stuart Dryburgh
composer
Rolfe Kent
director
James Mangold
editor
David Brenner
producer
Cathy Konrad
writer
Steven Rogers
Cast
Role
Name
Character
writer
Steven Rogers
producer
Cathy Konrad
editor
David Brenner
director
James Mangold
composer
Rolfe Kent
cinematographer
Stuart Dryburgh
actress
Meg Ryan
Kate McKay
actor
Hugh Jackman
Leopold
actor
Liev Schreiber
Stuart Besser
actor
Breckin Meyer
Charlie McKay

9.

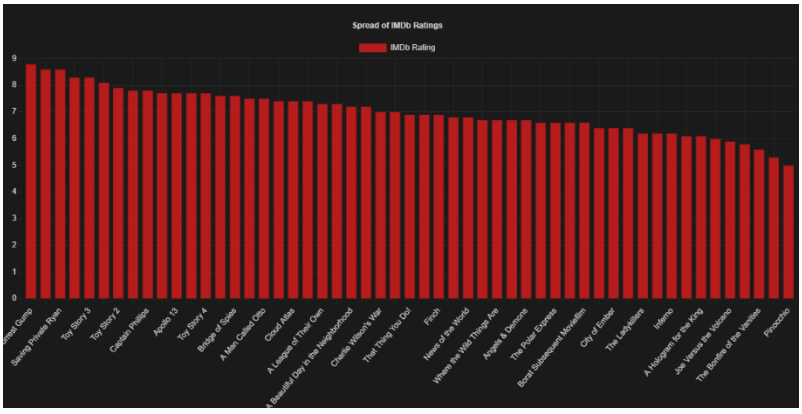
Roles

Role	Movie	Character	Rating ↑
actor	Pinocchio	Geppetto	5
actor	The Circle	Bailey	5.3
actor	The Bonfire of the Vanities	Sherman McCoy	5.6
producer	The Ant Bully		5.8
actor	Joe Versus the Volcano	Joe	5.9
producer	My Big Fat Greek Wedding 2		6
actor	Larry Crowne	Larry Crowne	6.1
actor	A Hologram for the King	Alan	6.1
actor	The Ladykillers	Professor G.H. Dorr	6.2
producer	Connie and Carla		6.2
actor	Inferno	Robert Langdon	6.2

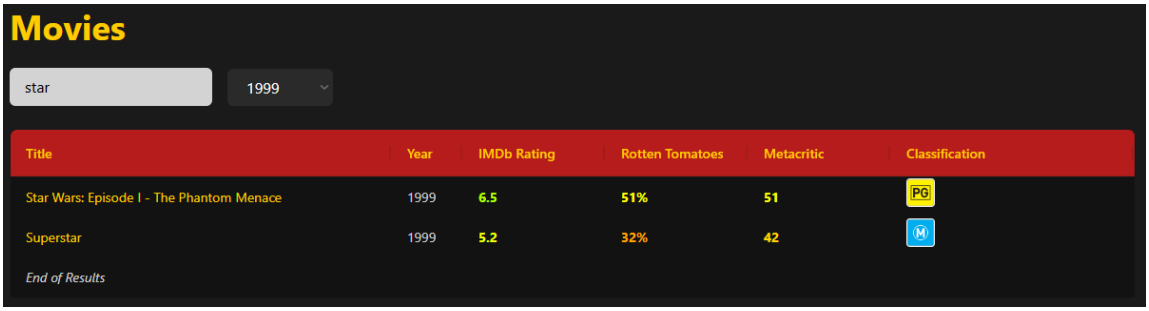


Roles

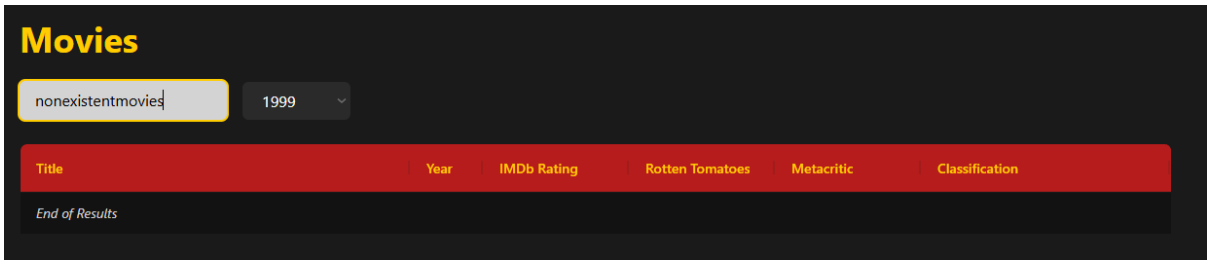
Role	Movie	Character	Rating ↓
actor	Forrest Gump	Forrest Gump	8.8
actor	The Green Mile	Paul Edgecomb	8.6
actor	Saving Private Ryan	Captain Miller	8.6
actor	Toy Story	Woody	8.3
actor	Toy Story 3	Woody	8.3
actor	Catch Me If You Can	Carl Hanratty	8.1
actor	Toy Story 2	Woody	7.9
actor	Cast Away	Chuck Noland	7.8
actor	Captain Phillips	Captain Richard Phillips	7.8
actor	Philadelphia	Andrew Beckett	7.7
actor	Apollo 13	Jim Lovell	7.7



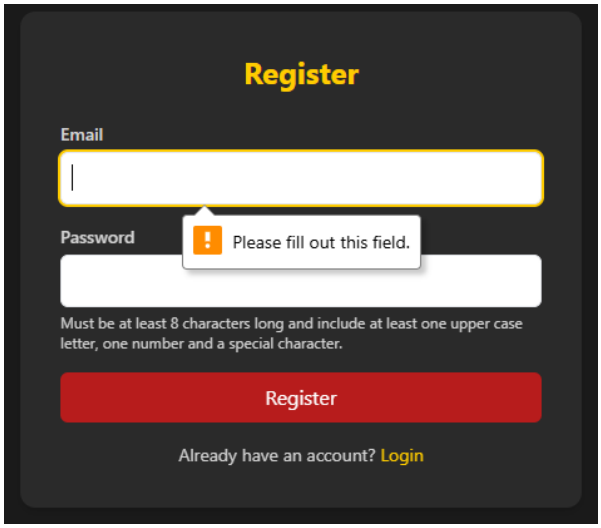
10.


Movies
star
1999
Title
Year
IMDb Rating
Rotten Tomatoes
Metacritic
Classification
Star Wars: Episode I - The Phantom Menace
1999
6.5
51%
51
PG
Superstar
1999
5.2
32%
42
End of Results

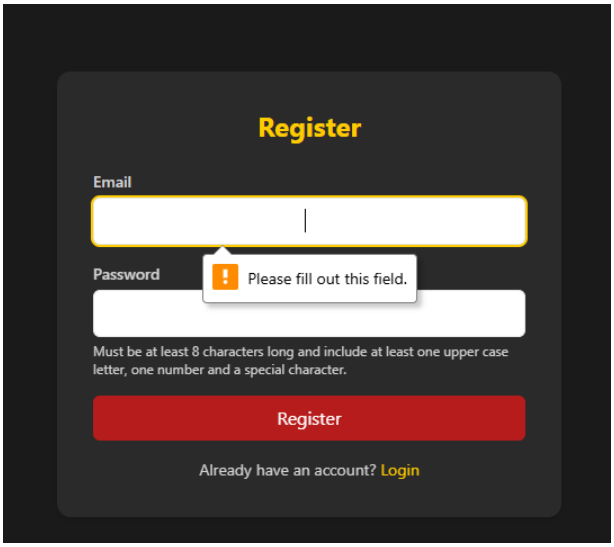
11.


Movies
nonexistentmovies
1999
Title
Year
IMDb Rating
Rotten Tomatoes
Metacritic
Classification
End of Results

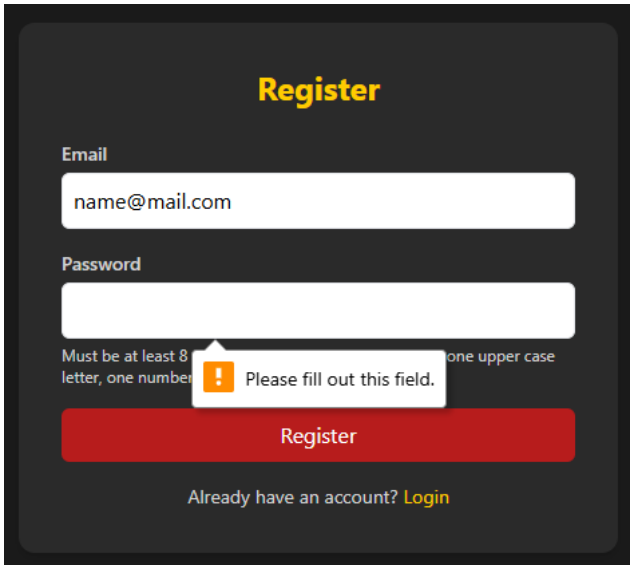
12.


Register
Email
Password
Please fill out this field.
Must be at least 8 characters long and include at least one upper case letter, one number and a special character.
Register
Already have an account? Login

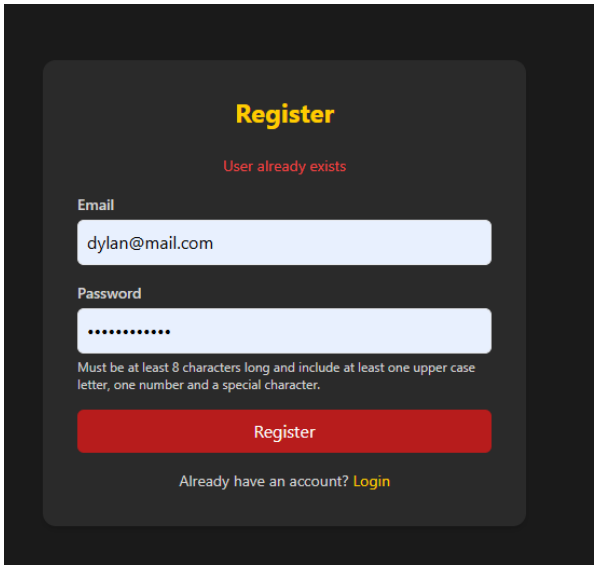
13.


Register
Email
Password
Please fill out this field.
Must be at least 8 characters long and include at least one upper case letter, one number and a special character.
Register
Already have an account? Login

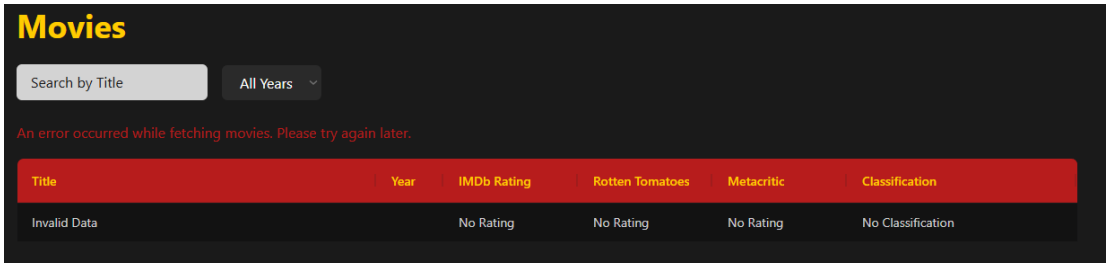
14.


Register
Email
name@mail.com
Password
Must be at least 8
one upper case
letter, one numbe
Please fill out this field.
Register
Already have an account? Login

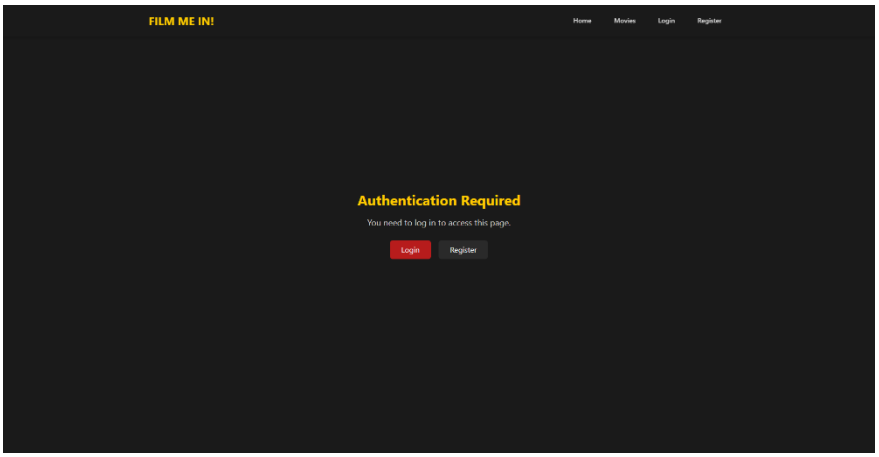
15.


Register
User already exists
Email
dylan@mail.com
Password
Must be at least 8 characters long and include at least one upper case letter, one number and a special character.
Register
Already have an account? Login

16.


Movies
Search by Title
All Years
An error occurred while fetching movies. Please try again later.
Title
Year
IMDb Rating
Rotten Tomatoes
Metacritic
Classification
Invalid Data
No Rating
No Rating
No Rating
No Classification

17.


FILM ME IN!
Home
Movies
Login
Register
Authentication Required
You need to log in to access this page.
Login
Register