

CAB230  
ASSIGNMENT 2  
– CLIENT SIDE  
MOVIES API

Darcie Pearson  
n11046244

# TABLE OF CONTENTS

Introduction.....	3
Purpose & description .....	3
Completeness and Limitations.....	3
Use of End Points.....	4
/movies/search .....	4
/movies/data/{imdbID}.....	4
/people/{id} .....	5
/user/register.....	5
/user/login .....	5
/user/refresh .....	5
/user/logout.....	5
Modules Used.....	6
Ag-grid-react.....	6
Chart-JS.....	6
React-Strap .....	6
React-Strap .....	6
Material UI.....	6
Radix UI.....	6
YearPicker .....	6
Application Design.....	7
Navigation and Layout .....	7
Usability and Quality of Design .....	8-9
Accessibility .....	10
Technical Description .....	11
Architecture.....	11
Test plan .....	12-14
Difficulties / Exclusions / unresolved & persistent errors .....	15
Extensions (Optional) .....	15
User guide.....	15
References .....	15

# INTRODUCTION

## PURPOSE & DESCRIPTION

This application is a React-based web application that allows users to view, filter and analyse data about movies from IMDb that have been exposed via a REST API. This application's design and layout aim to present all information in a stylised and intuitive manner, allowing users to search for movie titles, view individual movie details, and authenticate to view restricted content about individuals that were involved in production.

Style and usability were at the forefront when designing and building this application, reflected in the UI design, application feedback and proposed user interactions. The use of pre-built WCAG standard approved colour schemes, contrasts and components ensures that this application meets visual accessibility standards while maintaining strong style and efficiency. The application's colour scheme, typography and layout adhere to a fantasy-inspired design, aiming to modernize vintage fairy-tale imagery, enticing the user to explore the application's features in a playful and engaging manner.

## COMPLETENESS AND LIMITATIONS

[Grade 6] This application makes use of all provided data endpoints, including authenticated data endpoints and token refresh. Navigation is handled using the React Router and routes are rendered correctly in an intuitive manner. Authentication forms are controlled. Both Ag Grid and ChartJS are utilized to display the queried data to the user in a clean and consistent format. All components render in a uniform manner, adhering to the overall style of the application.

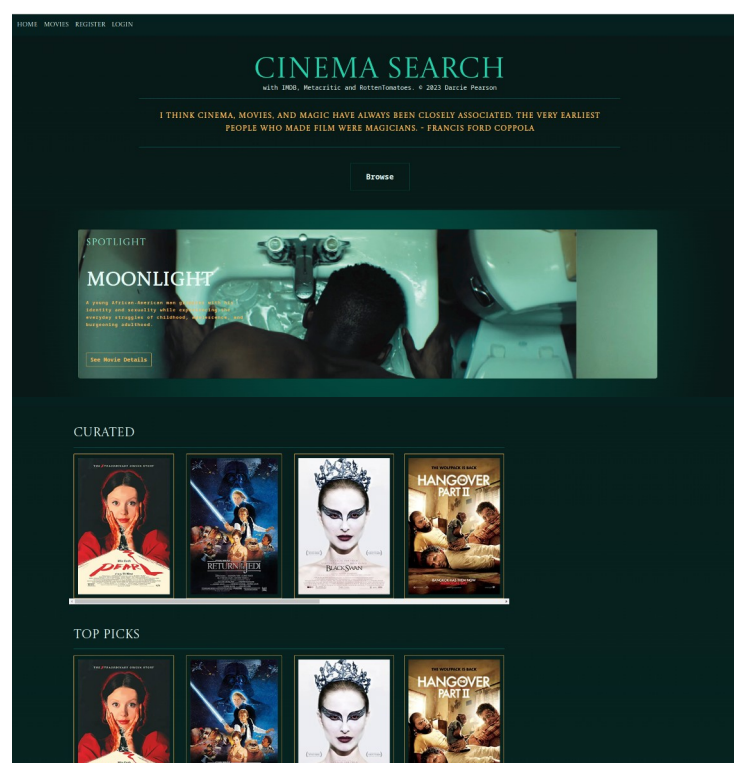


Figure 1. Application Homepage

# USE OF END POINTS

</movies/search/>

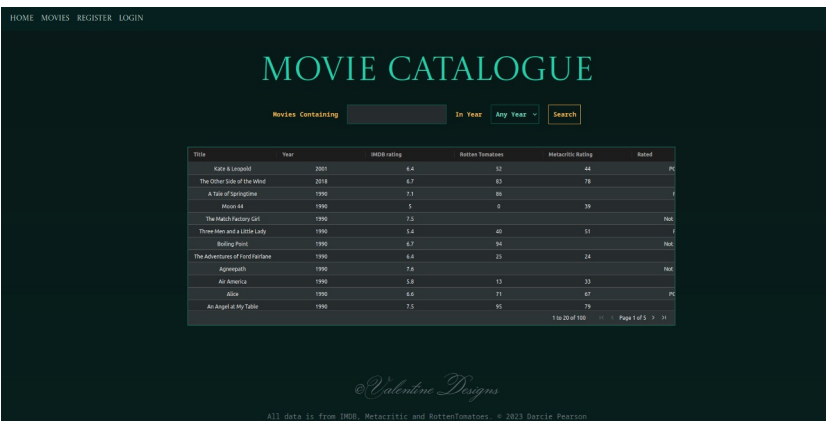


Figure 2. Movie Page

</movies/data/{imdbID}>

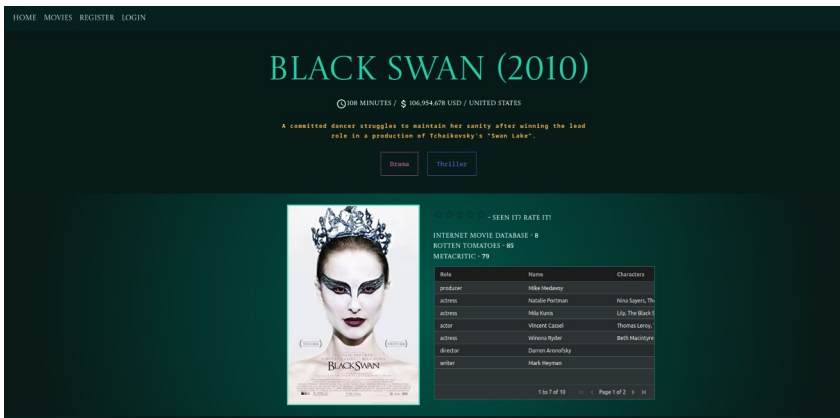


Figure 3. Individual Movie Page

</people/{id}>

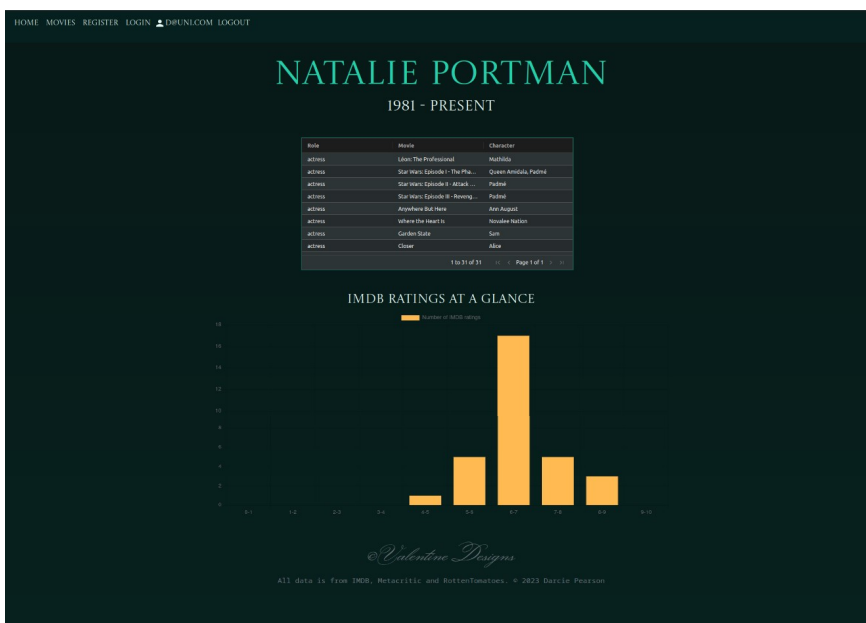


Figure 4. Principal Page

Figure 2. </movies/search/> returns a page with a table of movies and a fully functioning search bar. This search bar allows the user to input search queries and view the returned data via the table. On click of a movie, the user is taken to the corresponding page associated with the IMDb-ID at [movies/{id}](/movies/{id}).

Figure 3. </movies/data/{imdbID}> returns a page displaying details of movie that is identified by the IMDb ID. These details include the name, year, runtime, box-office sales, countries, plot, genres, poster, ratings and principals. On click of a principal, the user is taken to the corresponding page associated with the ID of that person.

Figure 4. </people/{id}> returns a page with details on the associated person, including their name, birth/death year and a graph of their IMDb ratings.

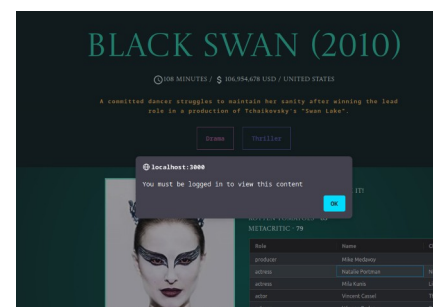


Figure 5. Blocked content message

Figure 5. If the user tries to access this endpoint while not logged in they will get a window alert that then redirects them to the login page.

## /user/register/

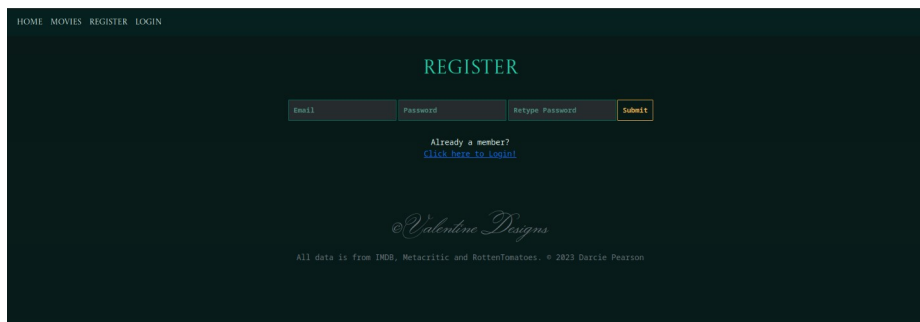


Figure 6. Register Page

Figure 6. `user/register/` returns a page with a form that prompts the user to input an email and password. On submit an account is made for the user and they are redirected to the login page.

## /user/login/

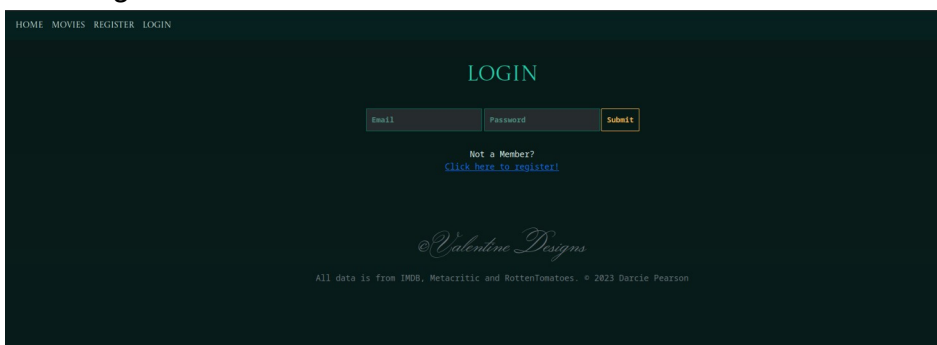


Figure 7. Login Page

Figure 7. `user/login/` returns a page with a form for the user to input their account details. On submit the user is logged in and they are redirected to the page they were previously on.

## /user/logout/

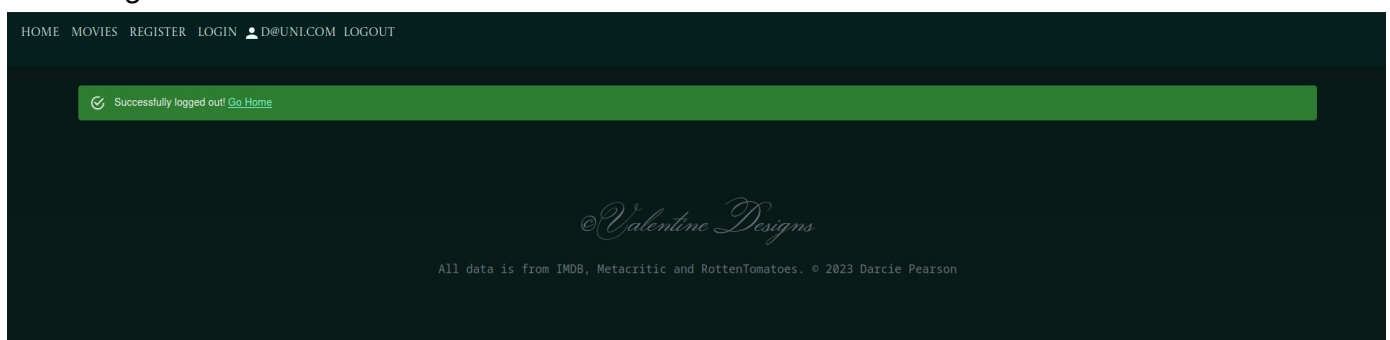


Figure 8. Logout Endpoint

Figure 8. `user/logout/` route returns an alert and the current user is logged out. Additionally, I have chosen to add a logout button to the navigation bar that logs the user out on click.

## /user/refresh/

`user/refresh/` route refreshes the users tokens. The visual feedback is the same as the log out route (Figure 8), as this function is intended to be automatic, when the user attempts to access a restricted resource with an expired token.

# MODULES USED

## Ag-grid-react

Module to provide fully-featured table components, including sorting and filtering.

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

## Chart-JS

Module to provide fully-featured charting components.

<https://github.com/chartjs/Chart.js>

## React-strap

Module to provide pre-styled components, themes and layouts.

<https://reactstrap.github.io/?path=%2Fdocs%2Fhome-github--page>

## Material UI

Module to provide pre-styled components, themes and icons.

<https://mui.com/>

## Radix UI

Module that provides pre-styled and accessible components, themes and colour schemes.

<https://www.radix-ui.com/>

## YearPicker

Module that provides a drop-down year picker component that can then be altered to fit the applications needs.

<https://www.npmjs.com/package/react-dropdown-date>

# APPLICATION DESIGN

## NAVIGATION AND LAYOUT

The design process for this application was iterative and feedback-driven. I started by implementing the main components such as the navigation bar, the data display components and the authentication forms, then gradually implemented the less important and more stylistic components such as the cards, icons and buttons. To begin the design process I created application wireframes, which I then mapped out an intended user flow diagram (*Figure 9*). I designed the navigation and layout of this application to attempt to guide the user through all the available endpoints in a natural succession.

The flow of this application begins on the homepage, where the user is prompted to the movies search page via the browse button. The user is then encouraged to browse and select an individual movie, taking them to the movie's details page. From there, they can navigate to the principals' page. To access the principals' page, the user is additionally guided through registration and authentication. If the user continues browsing they will also be taken through the refresh endpoint unknowingly as their authentication token expires. This flow ends when the user decides to log out of the application.

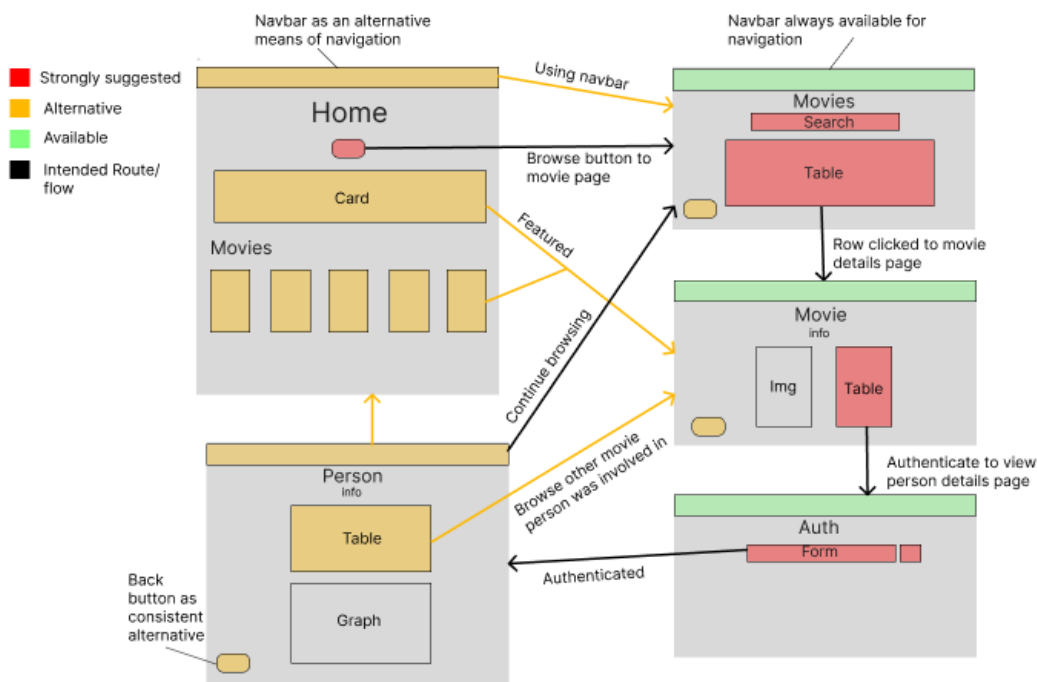


Figure 9. User flow diagram

The above diagram displays the intended usage of this application. The key highlights the components that facilitate the flow between screens. In my design, I have included several components that act as strong indicators of flow, which are seen highlighted in red. These components aim to guide the user through the application on the intended route to reach all the data endpoints. The components highlighted yellow to act as alternative routes that are available to the user depending on their goals when using the application. This allows for a wider range of interactions and to fulfil different user goals. The components highlighted in green aims to provide the user with navigation at any time, ensuring they always have available routes to take to meet their individual needs.

# USABILITY AND QUALITY OF DESIGN

## DISPLAY AND LAYOUT

The display and layout of my application are clearer on some pages than on others. I prefer pages with their content in the centre with large margins on the sides, this is reflected in the layout choices made during the designing of the home page and movie page. Although I have tried to keep the number of different components to a minimum, these pages have significantly more than others, most of which are concentrated in the centre. This layout could be perceived as cluttered or clumsy, potentially invoking concerns regarding a lack of spacial awareness. This could be improved by increasing white space between components and utilising a larger portion of the page width.

The increased number of components on the movie page meant that designing an efficient and effective layout was challenging. Although generally frowned upon, I made the design decision to use icons and badges to indicate information to add style and cut down on text (*Figure 10*), doing this, provokes a myriad of problems concerning not only accessibility, but also cultural literacy, as I assume that users would be able to discern what information they are looking at (Andrews, 2017). E.g. the Dollar sign icon means box office sales. This could be improved by adding in-page text descriptors to all icons and badges and in turn increasing the range of users that can use or comprehend my site.



Figure 10. Use of icons and badges in displaying information

## NAVIGATION

A navigation bar is consistently present in the top left of each page to allow the user to navigate between pages. Although all pages are available to the user via URL routing, the navigation bar changes its status depending on whether the user is logged in or not. displaying the login and register pages when the user is not logged in, and the current user and a logout option when they are.

The intended navigation of my application is largely handled by components that prompt the user to follow the flow of pages corresponding to the data endpoints. These components include tables whereby clicking a row navigates to the next page. This navigation could be unclear to some users, as there is no indication that they will be navigated to the corresponding data on click of a row other than row highlighting on hover. This could be improved by adding a note or short alert that informs them the table rows act as links to new pages.



## USER FAMILIARITY

I have attempted to model my application on similar movie searching/streaming websites such as IMDb and Netflix (Figure 12). I aim to keep my design consistent with user expectations by implementing familiar components such as cards (Figure 11), a static navigation bar and simple authentication forms and buttons. Overall my application could be improved by increasing stylised components that are consistent with other movie sites.

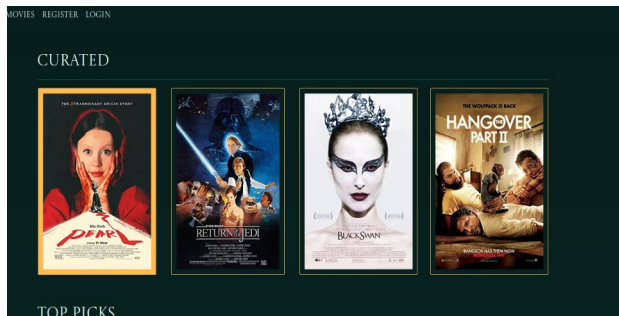


Figure 11. Movie cards on application homepage

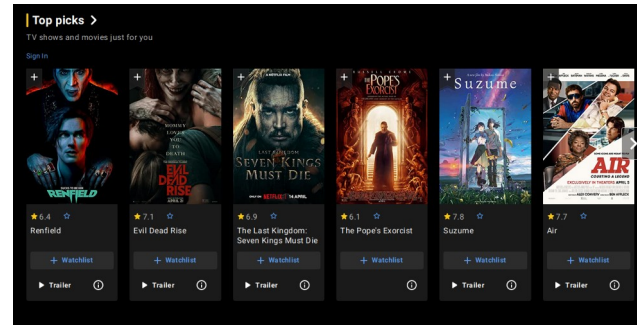


Figure 12. Movie cards on IMDb website

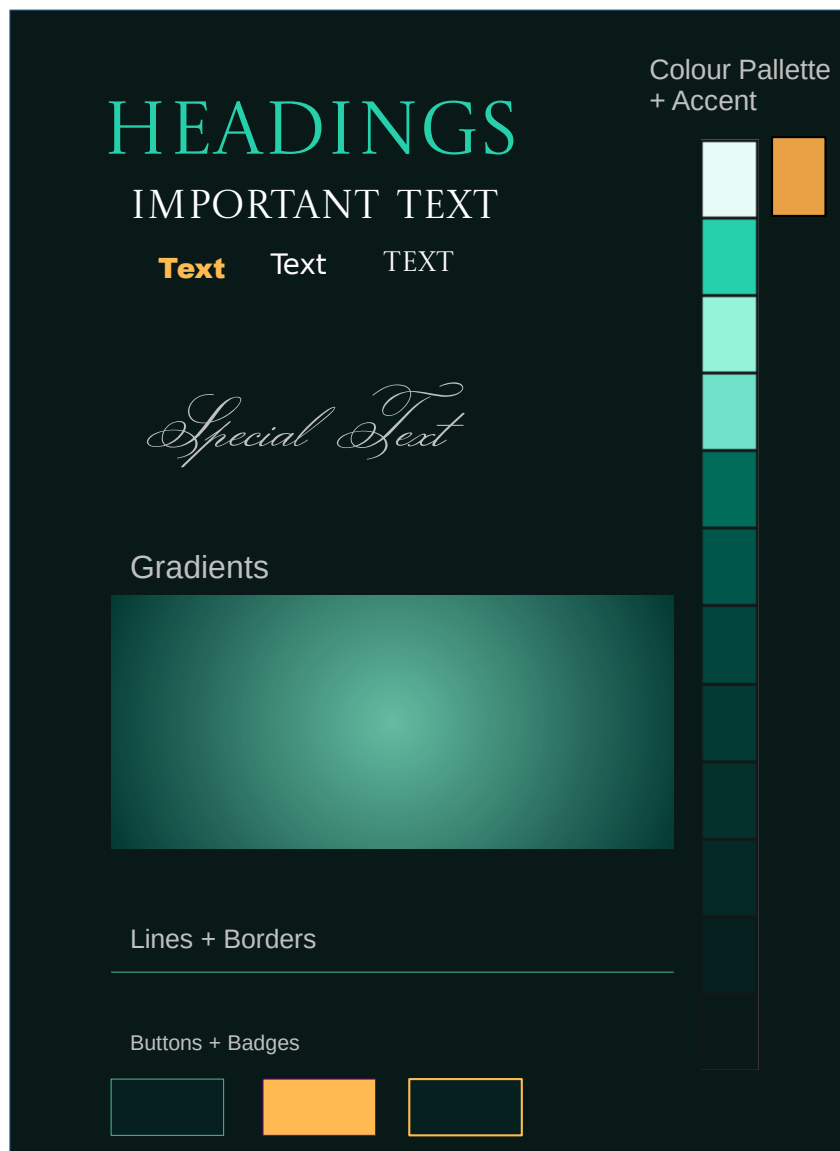


Figure 13. Application style guide

## VISUAL CONSISTENCY

I have attempted to keep my fonts, text size and colour schemes consistent across all pages of my application. I have done this by maintaining a style guide (Figure 13).

While my UI may appear conformative, it runs the risk of being perceived as dull due to the lack of different colours. Some individuals may also not respond well to my use of an unconventional colour palette and accent. Additionally, my font choices are equally subjective, and some viewers may struggle with reading cursive or find uppercase lettering unappealing (Dutta, 2020). These factors could pose a challenge in appealing to a broad audience.

# ACCESSIBILITY

Requirement	Check	Explanation
Provide a text equivalent for every non-text element	No	My site attempts to provide a text equivalent for every non text element. Although, due to the implementation of the movie cards and the retrieving and displaying of the individual movie poster, my site does not display an alt tag referring to the specific movie the poster is displaying, but a broader "Movie poster". Although the movie is indicated by the title of the page, the alt tag should still be consistent with the poster it is associated with. A large exception to the subsequent analysis is the tables and graphs, which is discussed on the next page.
Ensure that all information conveyed with color is also available without color.	Yes	My site allows for information to be retrieved without colour from the associated markup documents.
Organize documents so they may be read without style sheets.	Partially	My site attempts to format the associated markup documents appropriately. Although, on validating my markup several errors have been presented. On some pages I have been careless with the HTML syntax hierarchy, for example on the homepage, there are 3 distinct heading 2's. This creates significant difficulty for information comprehension without the associated style sheet. The information can still be read, but the information importance may be convoluted.
Ensure that text equivalents are updated when dynamic content changes.	Yes	My sites dynamic content is displayed via Javascript variables in HTML tags. Therefore, these variables are updated in text when rendered in the browser. When the content changes so does the associated HTML.
Avoid causing the screen to flicker.	No	My site attempts to ensure all components do not flash or flicker by limiting CSS animations. Although, the homepage has an animated card that could be a problem for those sensitive to moving content. Additionally, due to my application development, some screens implement a window reload between rendering. In this, the screen flashes a bright white. Although only one flash, repeated exposure could cause significant harm to those sensitive to flashing.
Use the clearest and simplest language appropriate for a site's content.	Yes	My site uses clear and simple text that is appropriate for an english speaking user.
For tables, identify row and column headers.	Yes	All tables in my site have distinct headers and columns Although this is not consisting in the associated markup documents.

## ACCESSIBILITY CONCERNS

When considering the overall accessibility of this application, I am quite disappointed with the analysed outcome. Normally inclusivity is a top priority of the design and development of a website. As this is my first application in React, I found it difficult to maintain and translate normal HTML and CSS accessibility standards to JSX. For example, having a large amount of components made it hard to maintain the strict information hierarchy that I am used to with HTML. Additionally, the usage of external modules made it difficult to understand how the information I provided to the components was displayed and subsequently how to make this information accessible in a non stylised format. As discussed, the majority of data is retrievable and comprehensible via the associated markup tags. A large exception to this is the tables and graphs. When analysing the accessibility of this application, I had a hard time finding the text representation of the Ag-grid-React rows in the browser-rendered source code. This information not being easily accessible poses significant accessibility concerns and decreases the overall accessibility rating of this application dramatically. The tables and graphs being a large part of the navigation and functioning of this site means that individuals using a screen reader or equivalent mark-up reliant tools are excluded from the proper function of this application.

# TECHNICAL DESCRIPTION

## APPLICATION ARCHITECTURE

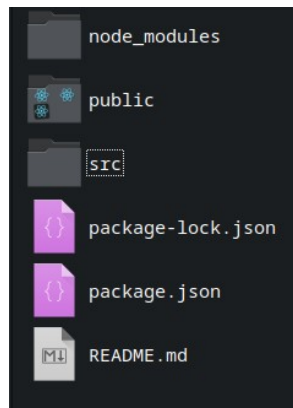


Figure 14. Application directory

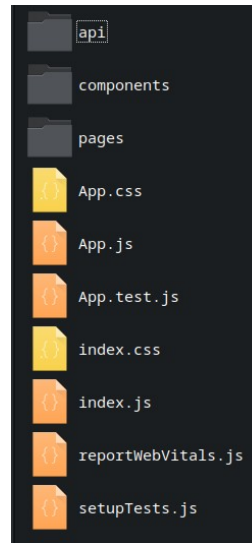


Figure 15. src folder

This application is controlled using components that contain the state of data, which are then passed through to the pages. The src directory (Figure 14) contains the three folders: api, components and pages. The api folder contains files that interface with the API to provide data to components. The components folder contains components that are reused throughout the application. The pages folder includes the pages that are available to the user where this data and corresponding components are displayed.

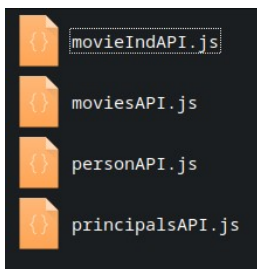


Figure 16. api folder

The api folder contains the React Hooks that supply data to the components.

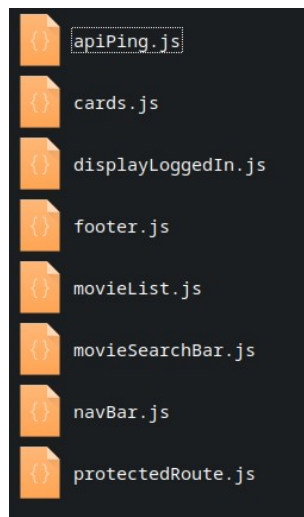


Figure 17. components folder

The components folder contains structures that interface with the supplied data.



Figure 18. pages folder

The pages folder displays these components and allows the user to query and alter the state of the data.

# TEST PLAN

Task	Expected Outcome	Result	Screenshot/ appendix B
Click on home in navigation bar	Home page is displayed	PASS	Figure 1
Click on movies in navigation bar	Movies page is displayed	PASS	Figure 2
Click on login in navigation bar	Login page is displayed	PASS	Figure 7
Click on register in navigation bar	Register page is displayed	PASS	Figure 6
Click on browse on home page	Movies page is displayed	PASS	Figure 1
Click on a movie card on the home page	The details page for that movie is displayed	PASS	B. Figure 1
Clicking the back button	The previous page is displayed	PASS	B. Figure 1
Click on movie pagination navigators	Different results are displayed per page in the table	PASS	B. Figure 2
Entering search text and clicking search	Only results matching the search text are displayed in the table	PASS	B. Figure 3
Clicking on the year picker component	Years from 1991 to 2023 are displayed	PASS	B. Figure 4
Clicking search with a date set	Only results matching the year selected are displayed in the table	PASS	B. Figure 4
Clicking search with text in search field and year in year picker	Only results matching the search text and year selected are displayed in the table	PASS	B. Figure 4
Click on individual movie row	Movie page with associated movie details is displayed	PASS	B. Figure 5
Click on individual principal row	Depending on authentication status, the corresponding page is displayed	PASS	B. Figure 6,7
Click on principals row as a non logged in user	If a user is redirected to the login page	PASS	B. Figure 6
Click on principals row as a logged in user	Principals page is displayed	PASS	B. Figure 7
Valid user credentials entered in the login page	The user is logged in	PASS	B. Figure 8
Invalid user credentials are entered in login form	Corresponding error message is displayed	PASS	B. Figure 10
Valid user credentials entered in the register page	An account is created for that user to which they can login to	PASS	B. Figure 12,16,17
Invalid user credentials are entered in register form	Corresponding error message is displayed	PASS	B. Figure 11
Click on logout in navigation bar	User credentials are cleared and cannot view authenticated content	PASS	B. Figure 13
Token is refreshed	If a users bearer token is expired their refresh token is sent to re authenticate	PASS	B. Figure 14
Loading feedback	If a resource is loading a loading dialogue appears until the data is presented.	PASS	B. Figure 15

*Note: Tasks without corresponding screenshots in appendix B have had their functionality displayed in previous screenshots*

## APPENDIX B

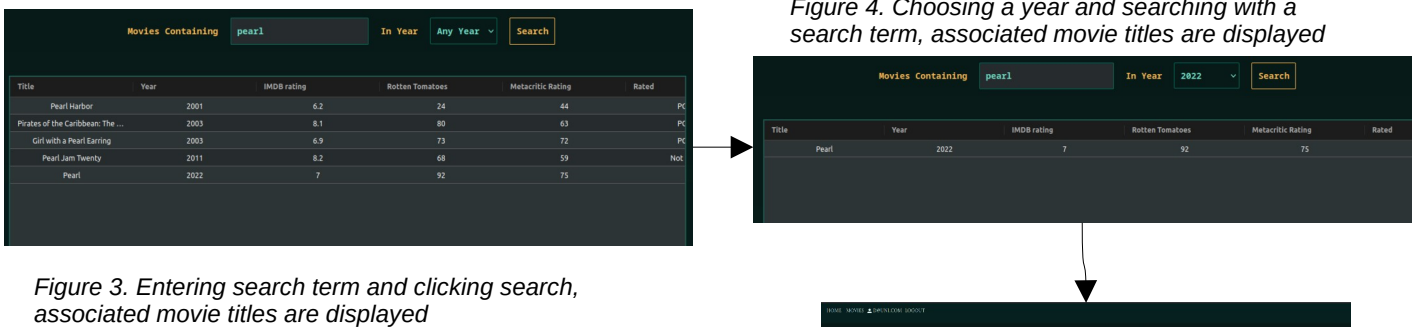
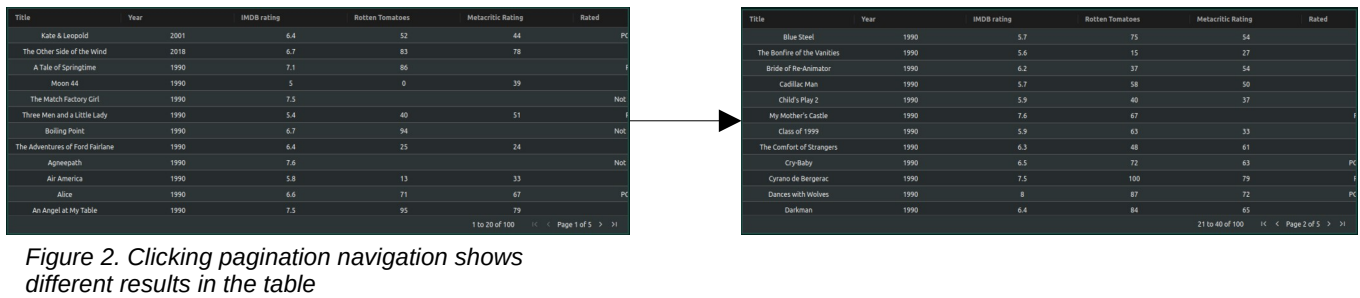
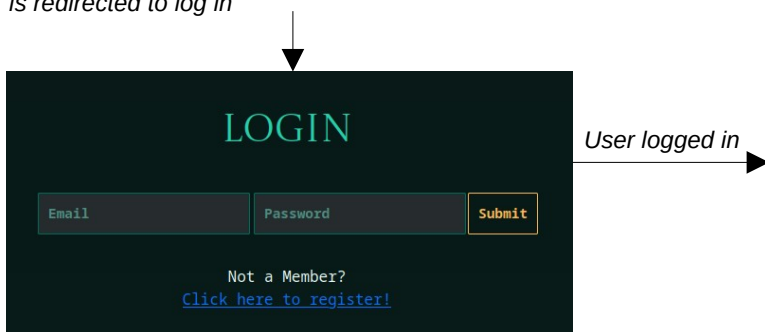
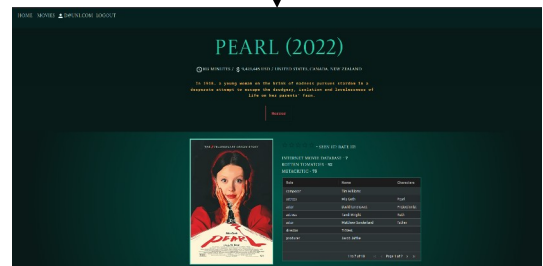
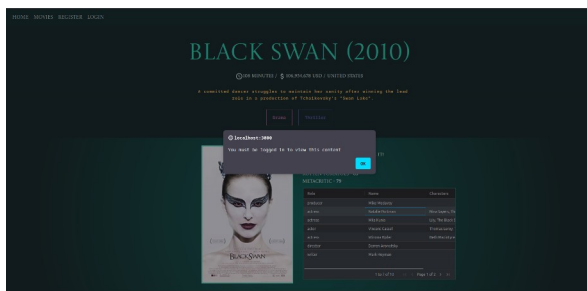
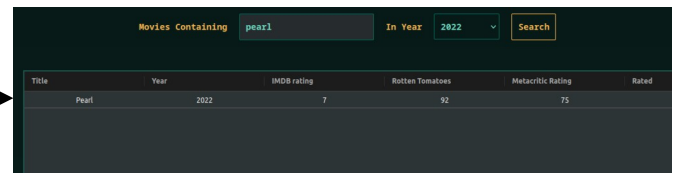


Figure 4. Choosing a year and searching with a search term, associated movie titles are displayed



## APPENDIX B CONT.



Figure 8. Successful login

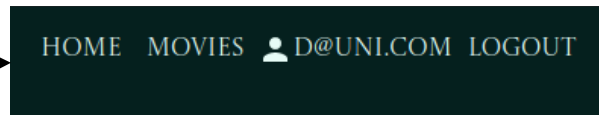


Figure 9. Navigation bar updated with user account

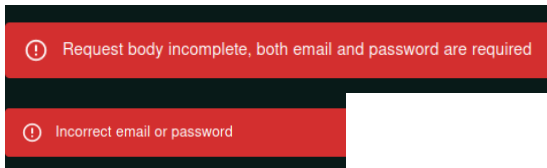
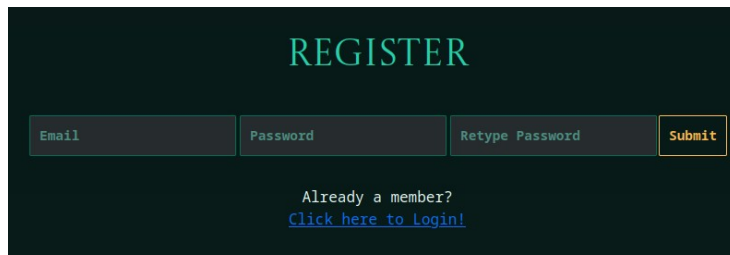


Figure 10. Unsuccessful login error messages



Figure 13. On click of log out button the user is logged out and the navigation bar is reset



Successful registration

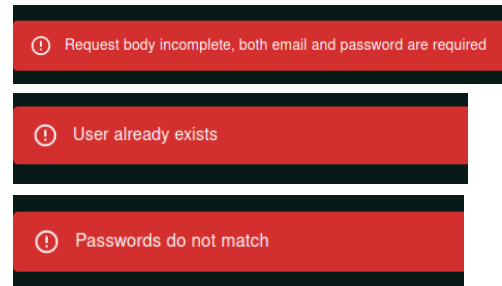


Figure 11. Unsuccessful registration error messages

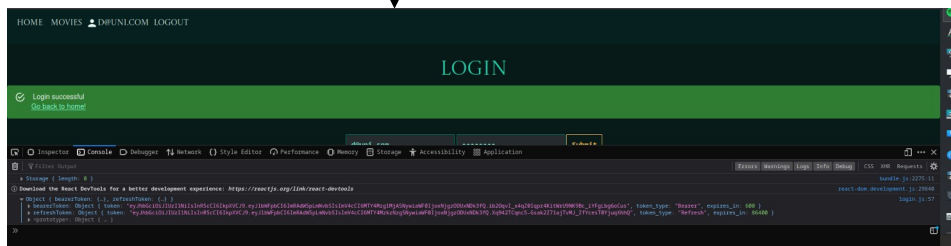


Figure 12. Successful registration redirected to login, where if successful, tokens are set

User continues browsing past bearer token expiry, tokens are refreshed without user interaction

```
bearerToken: Object { token:
  "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWVpbC6ImRAdW5pLmNvbSIsImV4cCI6MTY4Mzg1Mjc5NCwiaWF0IjoxNjgzODUyMTk0fQ.wA-Yq8kHHfyQcUNTkmwX0sCHvFQbW3hDy0TjQ10xU0", token_type: "Bearer", expires_in: 600 }

refreshToken: Object { token:
  "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWVpbC6ImRAdW5pLmNvbSIsImV4cCI6MTY4Mzg1Mjc5NCwiaWF0IjoxNjgzODUyMTk0fQ.84MflEpmhX5L248lNa8GWSGfK995v06SEXYEpGvJ6K0", token_type: "Refresh", expires_in: 86400 }
```

Figure 14. Tokens are refreshed

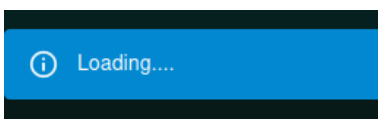


Figure 15. Loading message that appears whenever a resource or page is loading

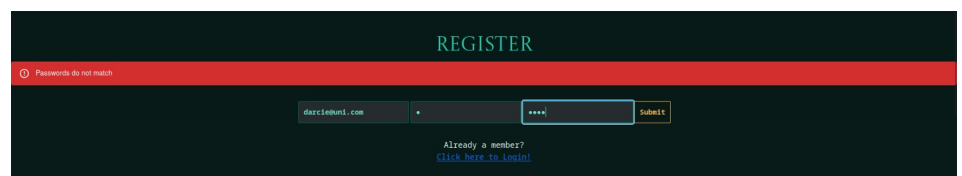


Figure 16. controlled forms give feedback to the user if their password no not match before they press submit

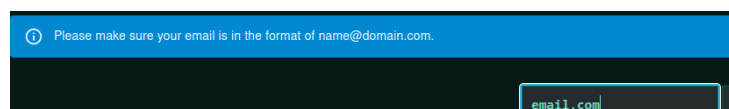


Figure 17. controlled forms give feedback to the user if their email is not in the correct format.



## DIFFICULTIES / EXCLUSION / UNRESOLVED & PERSISTENT ERRORS

During development of this application I came across numerous errors that brought up feelings of frustration and self doubt. These errors occurred specifically when handling the authentication tokens during user login, token refresh and when allowing the user access to the authenticated route. These difficulties put a major roadblock in development and I felt as if they would never get working. After separating the problem into smaller manageable parts, I was able to slowly re-build a working authentication system. Other difficulties I faced were associated with the handling of JSON data returned from the REST API. As my first time working with JSON formatting and a REST API, I found there to be a steep learning curve when attempting the format and display the queried data. This was coupled by the unfamiliarity of the swagger docs. After asking a tutor for clarification during my practical, I was able to slowly understand what was required and how to accomplish these requirements.

A persistent error I came across was the positioning of the page when the user was redirected using `useNavigate`. For example, if the user had scrolled down on the homepage and pressed a card, the associated page would display at that same position they had scrolled down to, making it look like there was nothing on the page or it didn't load correctly. I found that using window refresh on landing solved this issue but concluded that would be unnecessary to add these in as it would add more strain to the application. Therefore, on some redirection the user may have to slightly scroll up to see the content.

## EXTENSIONS

Extensions for this application could include improving the homepage to utilise non static data when displaying details on the card components. Another extension could include developing the user account to utilise a user profile, allowing the user to favourite or create a list of movies and people they have searched. This could also include saving movies and people the user has viewed in a browsing history page.

## USER GUIDE

To retain the length of this report, it is assumed the user guide can be inferred from the user flow diagram (*Figure 9*) and the describing paragraph on page 7. Additionally, Appendix B steps through the use of the application in detailed screenshots.

## REFERENCES

Dan Andrews. (2017). The impact of culture on iconography. Medium.

<https://design-nation.icons8.com/the-impact-of-culture-on-iconography-df880f83e7cb>

Sandipan Dutta. (2020). Typography in UI design and accessibility factors. Medium.

<https://medium.com/@sndpn1997/typography-in-ui-design-and-accessibility-factors-90f53702d042>