

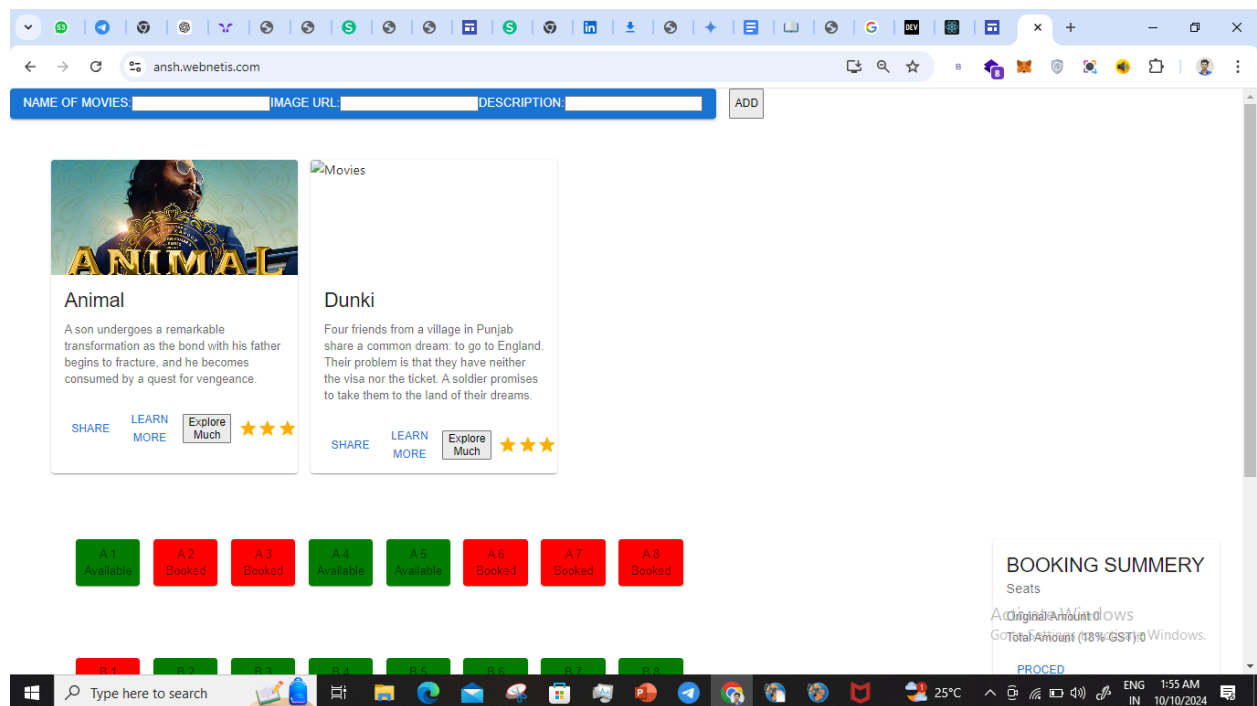
Title: Testing Report for "ansh.webnetis.com"

Prepared by: Ansh Bakshi

Date: 10-10-2024

Website 1: <https://ansh.webnetis.com/> (React-based website)

Testing Websites as a QA Engineer:



Website 1: <https://ansh.webnetis.com/> (React-based website)

Test Cases and Scenarios:

Feature	Test Case	Expected Behavior	Potential Defect
Homepage	Access the homepage	Homepage loads correctly with all content displayed	Broken images, missing sections
Navigation	Click on navigation links	User is redirected to the corresponding page	Broken links, incorrect redirection
Contact Form	Fill out the contact form with valid data	Form submission is successful, confirmation message appears	Form validation errors, unsuccessful submission
Responsiveness	View the website on different devices (desktop, mobile, tablet)	Website elements adapt and display correctly	Layout issues, content overlapping

Testing Plan for "ansh.webnetis.com"

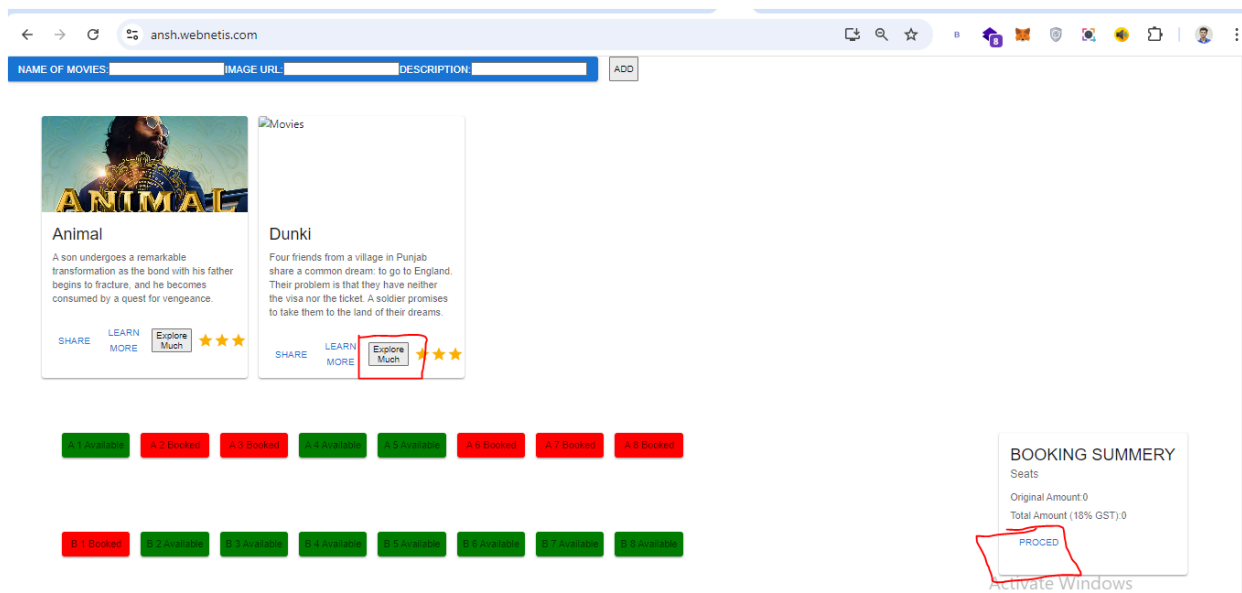
1. Black-Box Testing

- **Actions:**

- We navigated through all pages and checked if buttons and links work as expected. // But "explore much button", "Proceed button" were not working and the image was missing from the webpage"
- We filled out all forms on the site and submitted to ensure correct processing // as the database is not connected to the webpage so after every refresh the information stored gets clear.
- By Checking the navigation menu, including any dropdowns or sub-menus.// the seat selection buttons were working properly ,
- By Viewing the site on multiple devices and checking responsiveness.// the

Website layout could not adapt to mobile devices.

Defects





2. Functional Testing

- **Actions:**

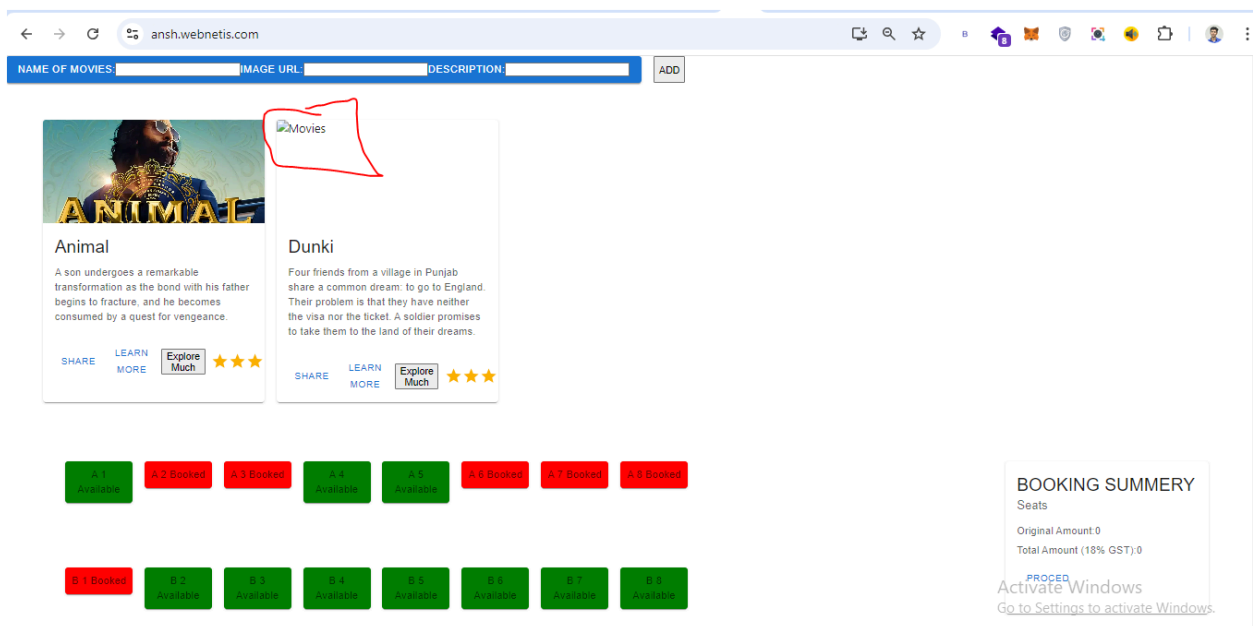
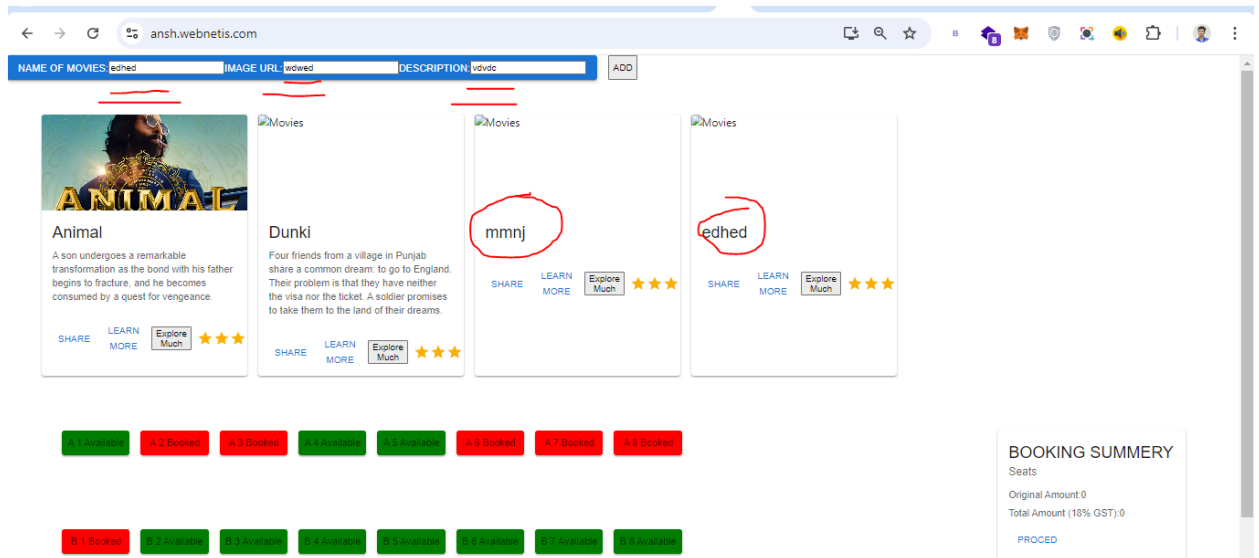
We tested all forms and user inputs, ensuring data validation and submission.//
when a user puts in some irrelevant information it still takes it

.

If any modals or pop-ups appear, ensure they display and close correctly.

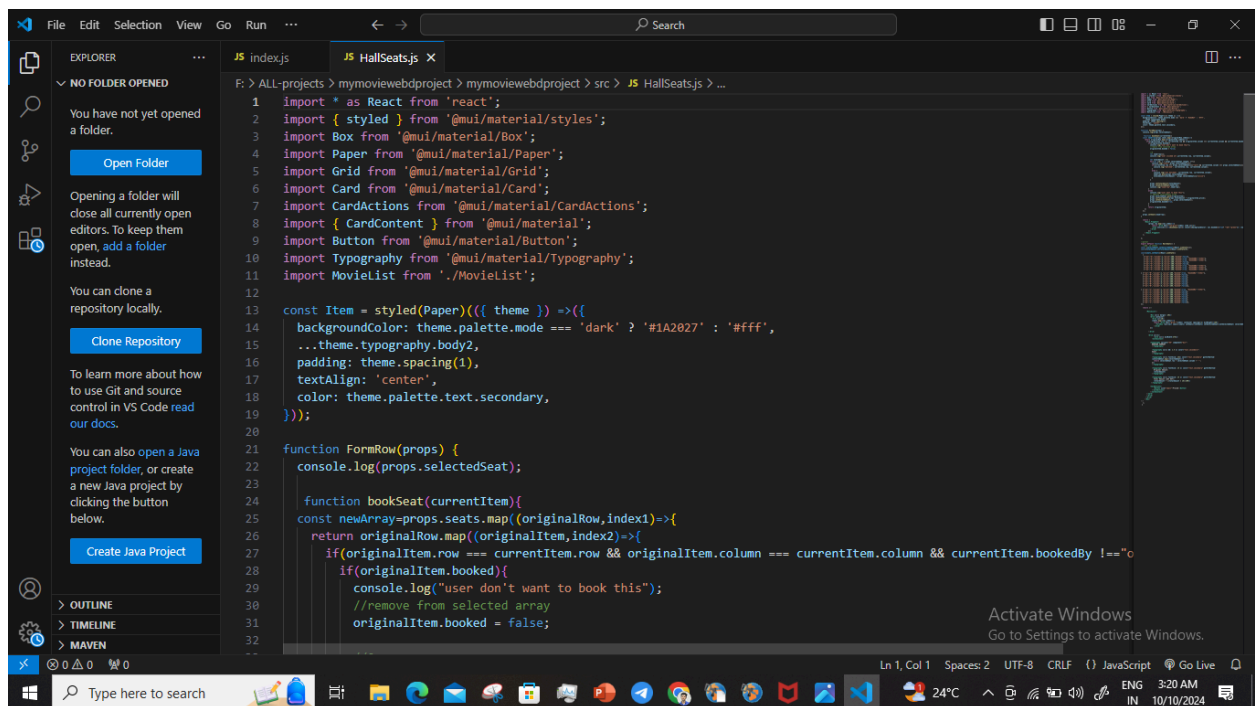
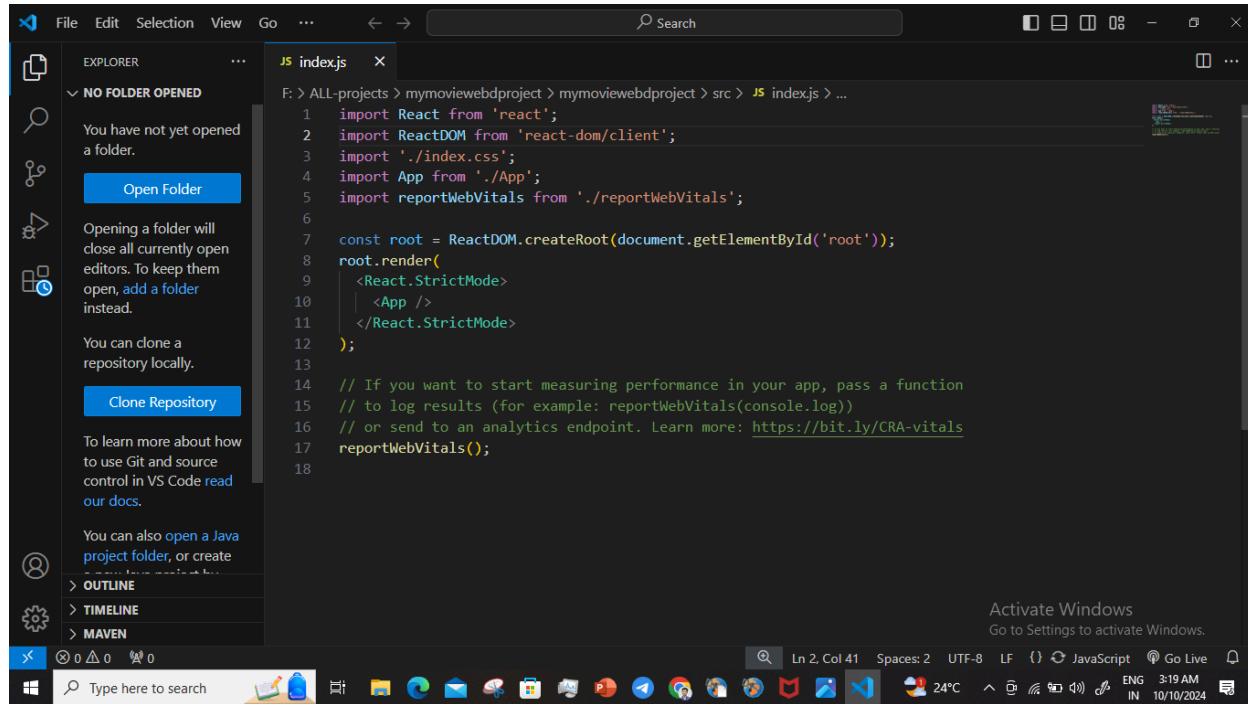
We tested API calls to ensure data retrieval, display, and any CRUD (Create, Read, Update, Delete) operations are possible or not // But it failed to retrieve
image from the link of code

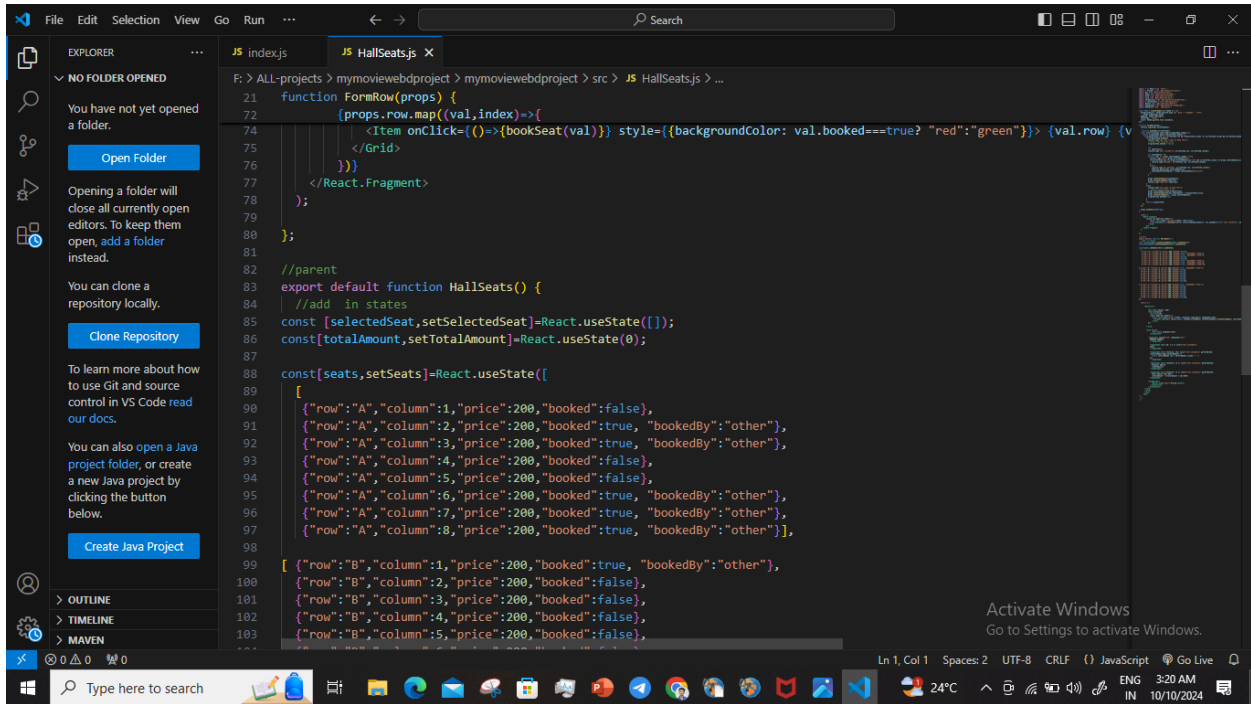
- **Defects**



3. White-Box Testing

- **Actions:**
 - We review the website's code and check for logical errors, unnecessary loops, and data flow issues//but the code was correct .





4. Unit Testing

- **Actions:**
 - We tested individual components, such as React components and JavaScript functions, in isolation. // they performed well
 - Check each CSS class to ensure consistent and expected styling. //the css styling need to made effective white background could be enhanced

Unit Testing for different source-code files

1. HallSeats.js

Unit Testing

- **Component Functionality:**
 - **Selected Seat Logic:** The `bookSeat` function handles booking and unbooking seats. This logic works correctly, as it updates the state based on user interactions.
 - **Seat Availability Display:** The conditional rendering of seat availability (red for booked, green for available) functions as expected.
- **CSS Styling:**

- **Enhancement Suggestion:** Implement a consistent styling approach for seat states. Consider adding hover effects to improve user experience. Additionally, a white background for the overall layout could enhance visibility.

2. Home.js

Unit Testing

- **Rating Display:**
 - The `Rating` component from Material UI is correctly set up to display a static rating. The implementation appears functional.

3. Layout.js

Unit Testing

- **Outlet Functionality:**
 - The `Outlet` component effectively allows for nested routes, which is a standard practice in React Route

4. MovieList.js

Unit Testing

- **State Management:**
 - The `addMovie` function and the input handling for movie details work as expected. However, consider validating user inputs to prevent incorrect data entry.

5. reportWebVitals.js

Unit Testing

- **Functionality:**
 - The function correctly imports and utilizes the `web-vitals` library to log performance metrics.

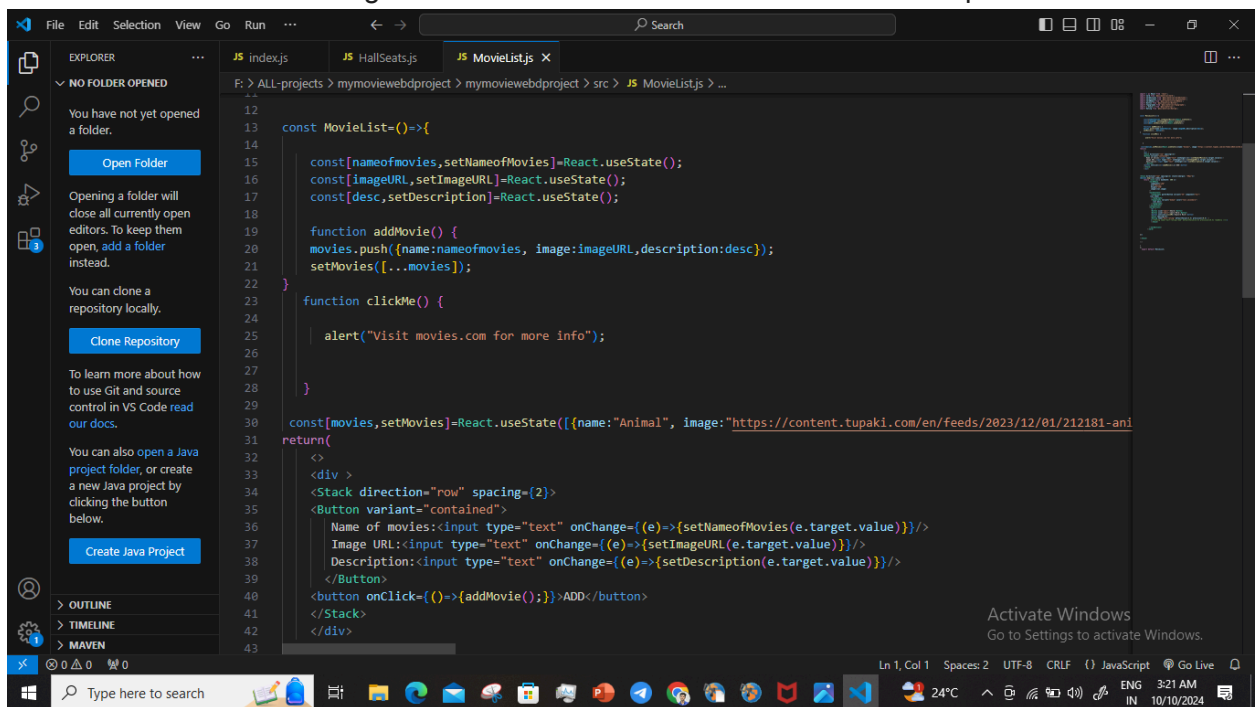
6. setupTests.js

Unit Testing

- **Configuration:**
 - The setup correctly configures Jest to use `@testing-library/jest-dom`.

5. Integration Testing

- **Actions:**
 - Test interactions between components to ensure smooth data flow and functionality. //inside code there was an very easy encapsulation and code ran successfully
 - We verified API integration with the UI and ensured accurate data presentation.



```

12
13 const MovieList=()=>{
14
15   const[nameofmovies,setNameofMovies]=React.useState();
16   const[imageURL,setImageUrl]=React.useState();
17   const[desc,setDescription]=React.useState();
18
19   function addMovie() {
20     movies.push({name:nameofmovies, image:imageURL,description:desc});
21     setMovies([...movies]);
22   }
23   function clickMe() {
24
25     alert("Visit movies.com for more info");
26
27   }
28
29   const[movies,setMovies]=React.useState([{name:"Animal", image:"https://content.tupaki.com/en/feeds/2023/12/01/212181-anl
30   return(
31
32     <div>
33       <Stack direction="row" spacing={2}>
34         <Button variant="contained">
35           Name of movies:<input type="text" onChange={(e)=>{setNameofMovies(e.target.value)}}/>
36           Image URL:<input type="text" onChange={(e)=>{setImageUrl(e.target.value)}}/>
37           Description:<input type="text" onChange={(e)=>{setDescription(e.target.value)}}/>
38         </Button>
39         <button onClick={()=>{addMovie()}}>ADD</button>
40       </Stack>
41     </div>
42
43

```

Integration Testing for different source files

1. HallSeats.js

- **Component Interaction:**
 - The **FormRow** component interacts seamlessly with the **HallSeats** component, managing seat selections effectively.

- **State Management:**
 - The use of `useState` to manage seat bookings and the total amount works well, ensuring data flows smoothly between the components.

2. Home.js

Integration Testing

- **Component Interaction:**
 - Since this component primarily displays static content, there are no interactions to test. However, ensure it integrates well within the overall application layout.

3. Layout.js

Integration Testing

- **Routing:**
 - Check that the layout integrates smoothly with the rest of your application, especially ensuring that nested routes render correctly within the layout.

4. MovieList.js

Integration Testing

- **Component Interaction:**
 - The interaction between the input fields and the state management for movie lists functions smoothly. The `Rating` component is well integrated and appears within the card layout.
- **API Integration:**
 - Ensure that if you later add API functionality, the state reflects the fetched movie data accurately.

5. reportWebVitals.js

Integration Testing

- **Performance Monitoring:**
 - Make sure this file correctly hooks into your app's lifecycle to capture metrics.

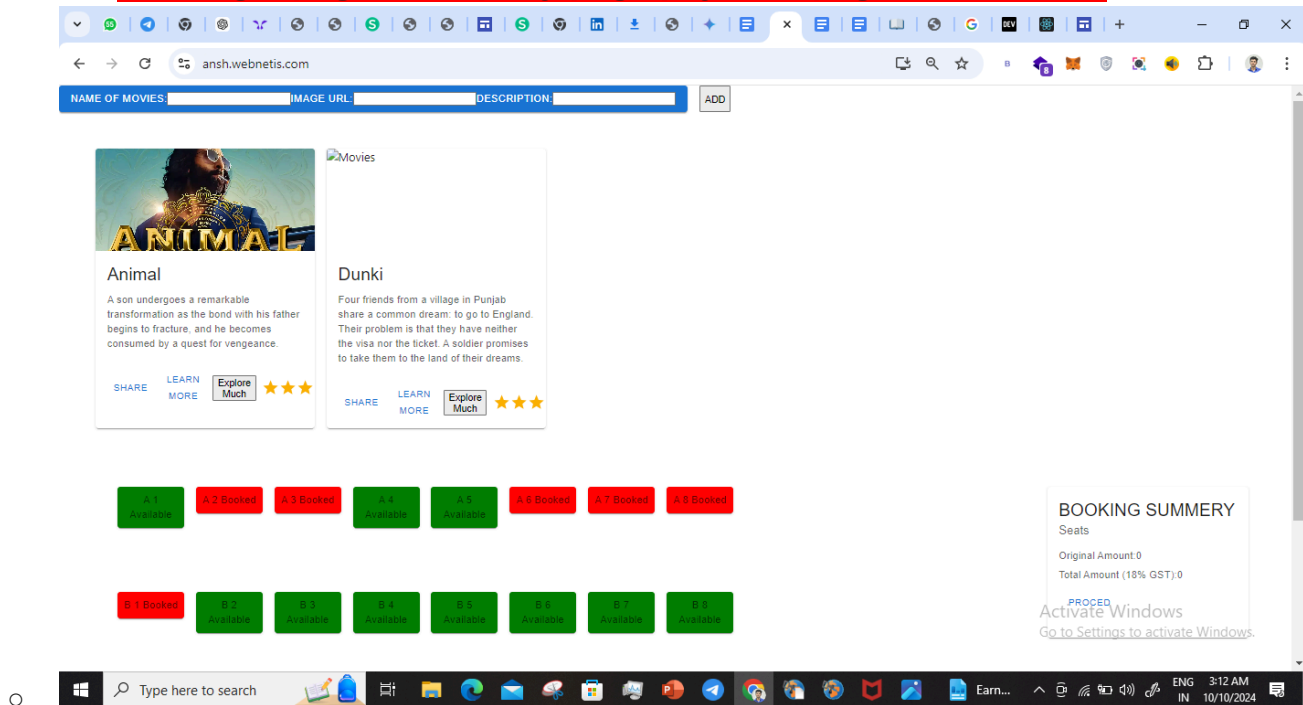
6. setupTests.js

Integration Testing

- **Testing Library Integration:**
 - Ensure that the custom matchers work effectively with your tests.

6. System Testing

- **Actions:**
 - We simulated real-world user flows such as registration, login, and form submission.// No facility available for user to put query,or register,or proceed for payment integrated
 -
 - Test different user roles, if applicable, and validate access control.// the website was accessible by any person,but there should be access control policy
 -
- **Defects:-** user login page,user query page,payment pageNot available



7. Performance Testing

- **Actions:**

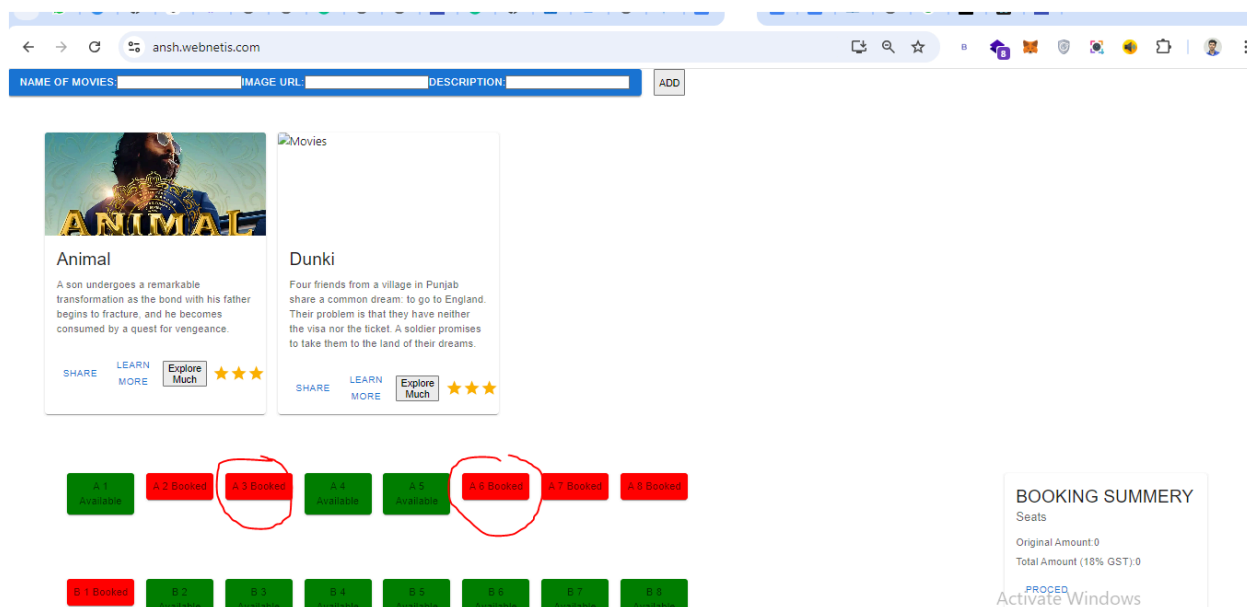
- We Measured page load times across devices and network speeds. // Average 5G (50-100 Mbps): Between 1-2 seconds for a full page load.

Fast 5G (100+ Mbps): Often under 1 second, particularly for optimized pages.

8. Usability Testing

- **Actions:**

- We navigated the website from the user perspective to assess intuitiveness. // the background could be enhanced, more of more features could be added , **could be made to work compatible with cell phone**
-
- // **The seat no's could be made bold for proper visibility**
-

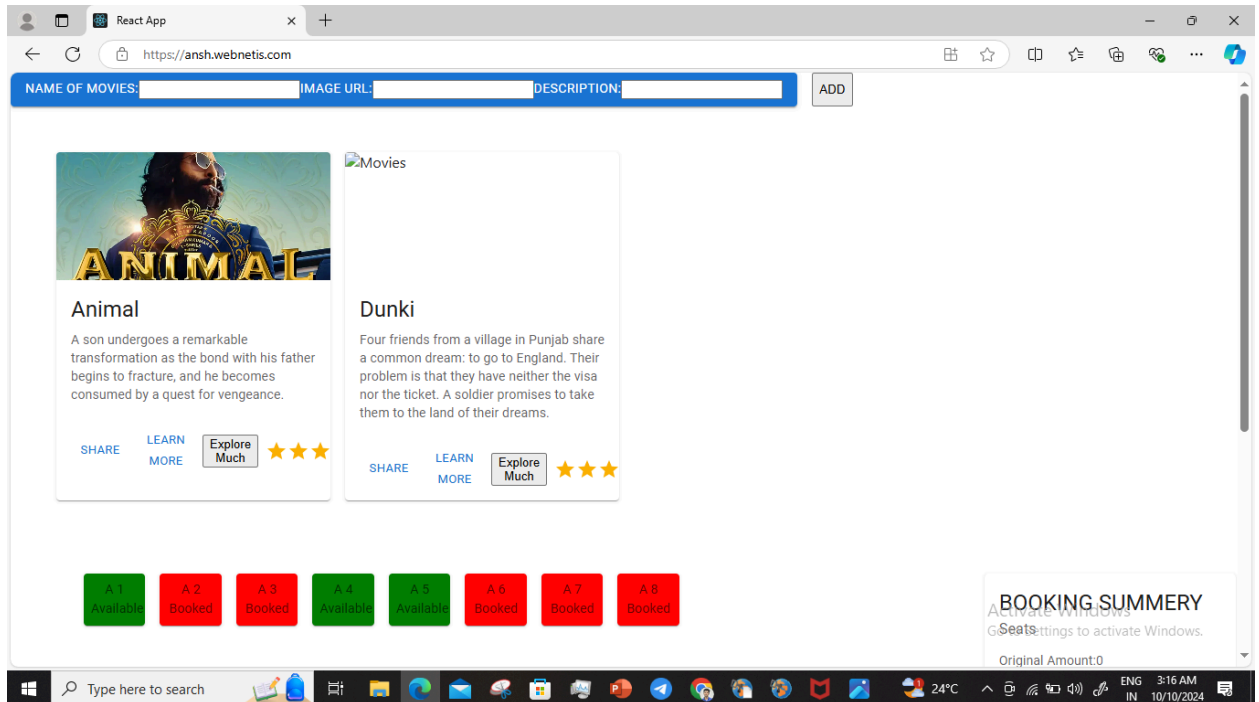


9. Compatibility Testing

- **Actions:**

- We test the website on different browsers (Chrome, Firefox, Safari) and devices.// and the webpage was working seamlessly on every browser

- Check layout and functionality on different operating systems.// it gave same page on different Browser



10. Load Testing

- **Actions:**
 - We simulated high traffic conditions to assess how the server handles concurrent requests.// but it was able to handle multiple request and was not lagging the speed to adapt user input

11. Stress Testing

- **Actions:**
 - We increased traffic beyond typical levels to test stability under extreme conditions.// but the webpage worked efficiently
 - We evaluated system recovery time after high traffic spikes.//and was able to perform well in high traffic also

12. Scalability Testing

- **Actions:**
 - We gradually increased user load to assess how the system scales. //it was scalable,consistent, durable

Additional Considerations:

- **Performance Testing:** We Measured website loading times and responsiveness on different internet speeds it was satisfactory

- **Security Testing:** We Ensured that the website uses HTTPS and has secure data handling practices.

Source Code link :-

<https://github.com/iamanshbakshi/mymoviebdproject/commit/abc206e0a952a47e0fef9422450afe1641f6a830>