DOCUMENTATION

Project Name: Movie Dashboard

Submitted By: Afroz Ansari

1. Overview

The Movie Dashboard is a React application designed to provide users with an overview of movies. The app also allows users to search for movies, gives detailed information about a movie and provides users with the Top performers movies list.

Features:

- **Top Performers List:** Displays top performing movies.
- **Movie Details Card: C**ards showing detailed information about movies.
- **Search Function:** A panel to filter movies by title.

2. Approach and Key Design Decisions

2.1. Component-Based Architecture

A modular component-based architecture is opted, which is a core feature of React. Each part of the UI (e.g., the search filter, movie details, and top performing list) was separated into individual components. This design decision:

- Encourages reusability and maintainability.
- Allows for easier testing and debugging of individual components.
- Provides flexibility to extend or modify individual components without affecting others.

2.2. State Management Using Redux

Redux is used for state management. A single store is used for managing state of this application. Also used useContext() hook for search feature.

2.3. Search Mechanism

The **search functionality** works by filtering the movie titles based on the Movie Name. When the user types in the search box:

- This approach allows for real-time filtering without needing to reload the page.
- This method is simple and efficient for handling search functionality with small datasets (like the Json file used, moviesandseries.json).

2.4. Movie Card

- Movie Card displays detailed information about a movie, such as its title, director, genre, and Oscar wins/nominations.
- By displaying the movie details in a separate card, we allow for easy adjustments and enhancements in the future (e.g., adding a trailer link or casting information).

2.5. Leader Board Displaying Top 5 Movies

This feature displays the top 5 movies at the time depending on rating obtained from the data.

2.6. Data Handling

Static data is used here for easier calculations which is stored in Json format. The Data is stored in the moviesandseries.json file in the public folder.

2.7. Responsive Design

The application is developed facilitating a Responsive Design.

---

3. Trade-offs

3.1. Simplicity vs. Scalability

Static data vs. dynamic API calls

- We chose to use static mock data (moviesandseries.json) for simplicity, as it allows us to focus on the UI. However, this limits the app's scalability.
- In a production environment, we would replace the static file with API calls to fetch movie data from a server, which would allow the app to work with a larger, dynamic dataset.

3.2. Filtering Data on the Frontend vs. Backend

Filtering of movie data is done entirely on the frontend using JavaScript. This works well for small datasets but could lead to performance issues with large datasets, especially if the number of movies grows significantly.

3.3. Custom Styling for the UI

Custom styling has been done using in the css file separately given under the Styles folder for unique styles and design.

---

4. Future Improvements and Enhancements

1. Add error handling for scenarios where data fetching fails or the user enters an invalid search query.
2. Database needed for movie list.
3. User rating can be calculated by taking rating from multiple users and the average can be calculated rather than the static rating value.
4. We can include the Summary and more details of a movie when the specific movie card is selected.
5. Reviews can also be added for movie info.
6. Convert loading state to skeleton loading.

5. Conclusion

This Movie Dashboard application demonstrates a clean, simple React app that provides an interface for exploring movie data and Leaderboard of top performers. The design and architecture allow for future scalability and enhancements, such as integrating an external API, adding dynamic movie selection, and displaying charts for better data visualization.