#### Key Features and Design Decisions:

Getting Data from API & Loading:

* Retrieved the movie data from the provided JSON endpoint.
* Used useEffect to simulate data fetching with a 1-second delay. In a real-world scenario, this would involve fetching data from an API.
* Utilized useState to manage the loading state and display a spinner while data is being fetched.
* The fetched data to convert it into a structured format suitable for consumption by the React components.

Component Structure:

* Developed a component-based architecture for building the dashboard.
* Created separate React components for each widget.
  + OscarOverview: Displays Oscar-related statistics.
  + TopPerformers: Shows a list of top-performing movies.
  + SearchFilter: Allows users to filter movies based on various criteria.
  + LanguageInsights: Provides insights into movie languages.
  + CountryInsights: Provides insights into movie countries.
  + MovieDetailsCard: Displays detailed information about a selected movie.
* Used a layout component to structure the overall dashboard layout and arrange the widgets in a visually appealing manner.

State Management & Filtering:

* Used useState to manage the filtered data and the selected movie.
* handleFilterUpdate is a useCallback function to efficiently update the filtered data and manage side effects.
* Passed filtered data to child components for rendering.
* Components can access and update the state using appropriate hooks (e.g., useState for local state).

Data Visualization Libraries:

* Incorporated charting libraries like React Chartjs 2 to create visually compelling charts and graphs for widgets like OscarOverview, LanguageInsights and CountryInsights.

Search and Filter Functionality:

* Designed a SearchFilter component that allows users to filter movies based on various attributes like genre, year, imdb rating.
* Updated the application state based on user selections in the filter panel.
* Re-render relevant components (e.g., OscarOverview, SearchFilter, TopPerformers, LanguageInsights, CountryInsights, MovieDetailsCard) to reflect the filtered data.

Responsive Design:

* Used a responsive UI library like Bootstrap to ensure the dashboard adapts seamlessly to different screen sizes and devices.
* Includes a basic UI with a search filter, dashboard sections, and a footer.
* Displayed a simple user profile with an image and name in the top right corner.

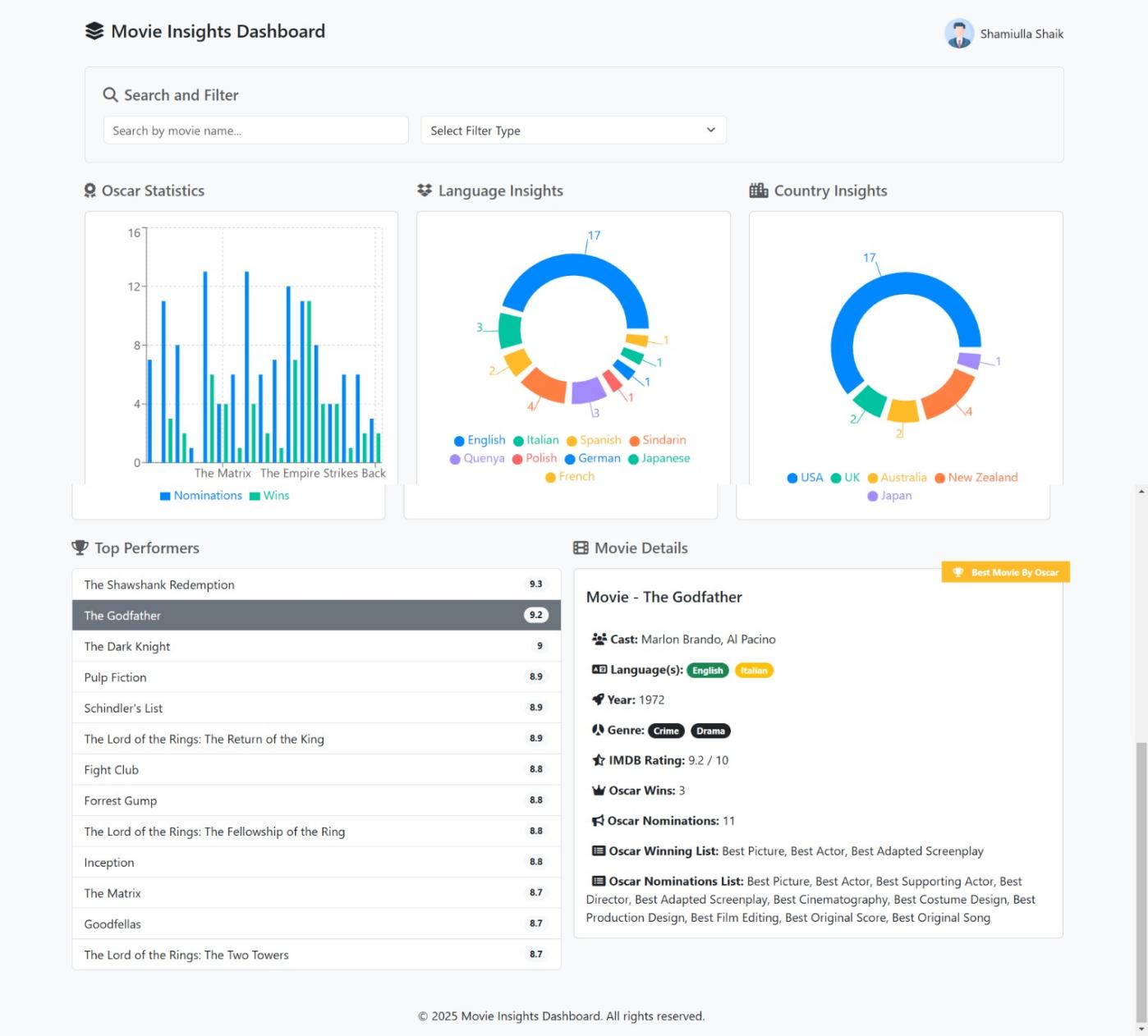
Testing:

* Wrote unit tests for components to ensure their functionality and catch regressions during development.

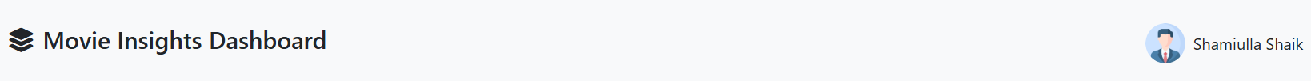
Deployment:

* Considered deployment options like hosting the React application on a platform like Firebase for demoing purpose.
* Demo can be found here - <https://glance-care-dashboard.firebaseapp.com/>

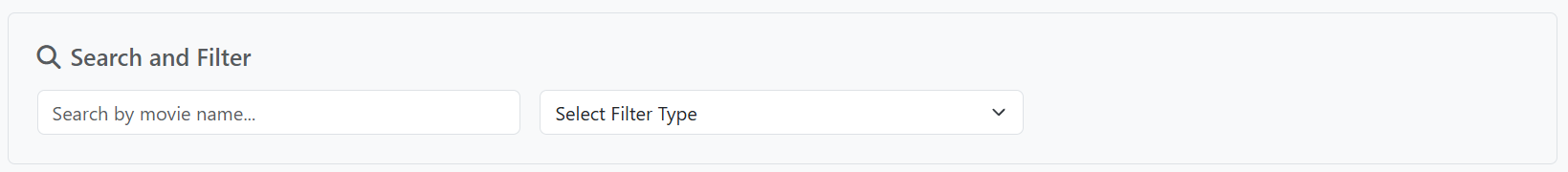
**The Application View**



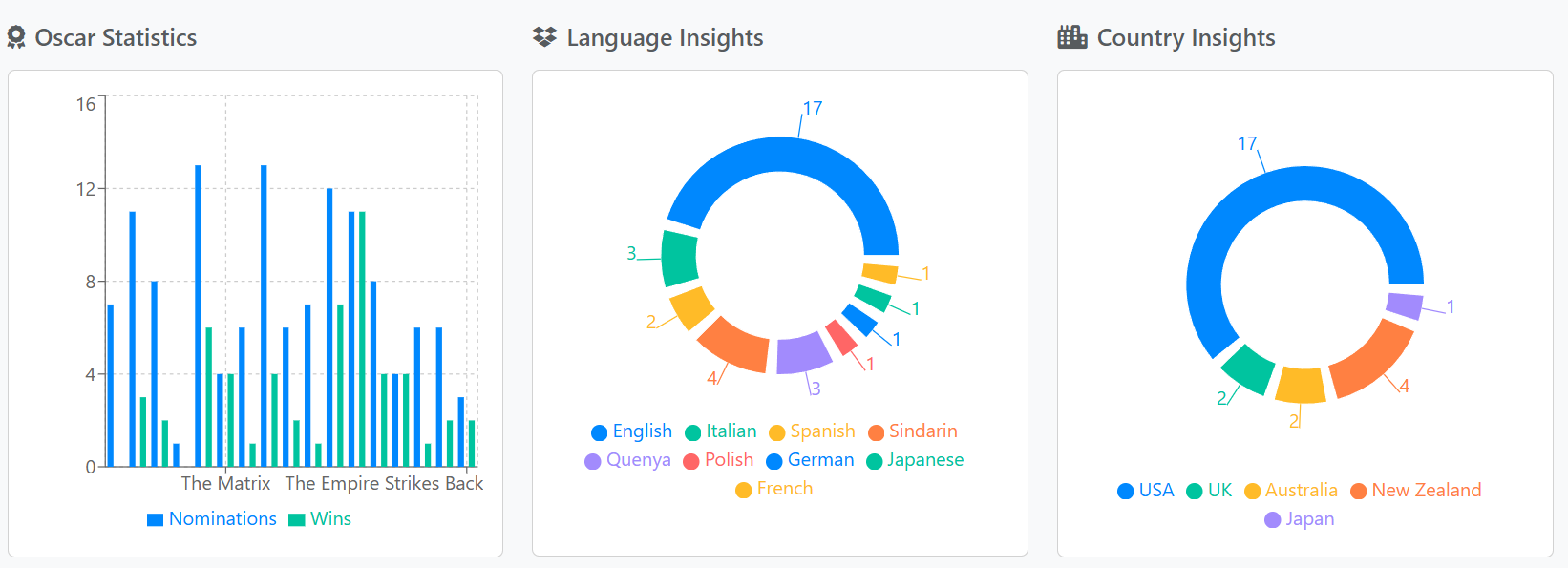
The Header Menu



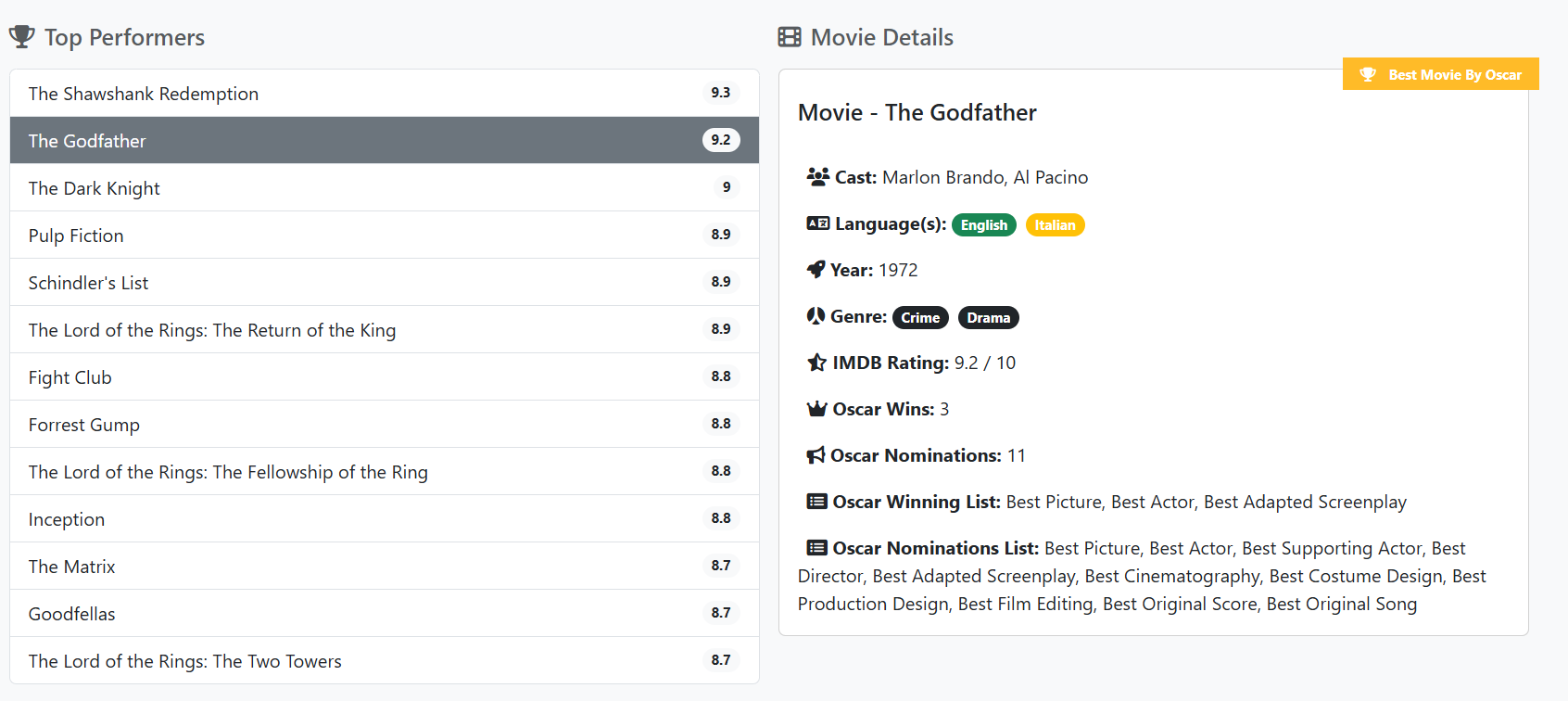
The Search & Filter Panel



The Various Insights (Data Visualization Charts)



Other Widgets



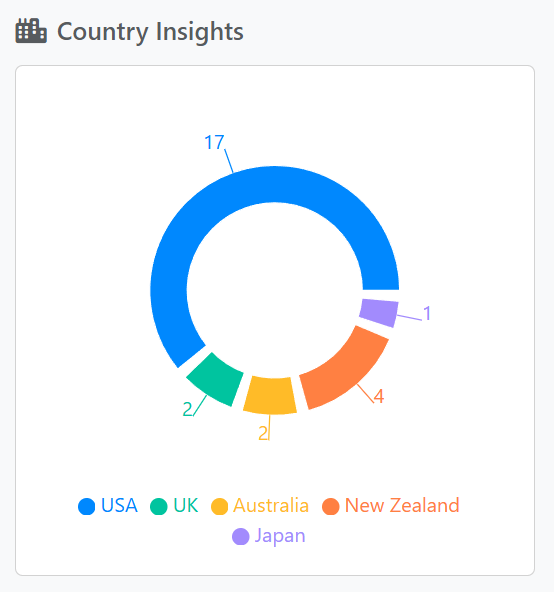
**Functionality of Each Component:**

1. **Country Insights:**

CountryInsights component effectively processes movie data and displays a pie chart visualizing the distribution of countries.

* The countryData object is created using reduce to accumulate the number of movies for each country.
* The countryChartData array is then constructed by converting the object into an array suitable for the PieChart component.

Pie Chart Configuration:



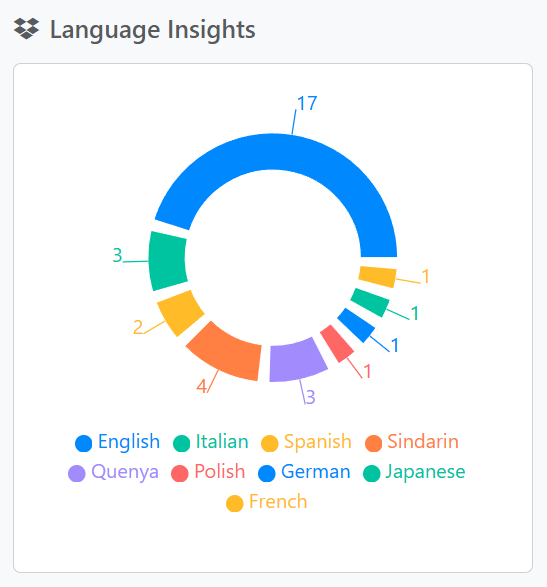
* A PieChart component from recharts is used to render the pie chart.
* The Pie component defines the data, data key, animation, and other visual properties.
* Cell components are used to define the color of each pie slice.
* Tooltip and Legend components enhance user interaction by providing hover information and labels for data slices.

1. **Language Insights:**

LanguageInsights component effectively processes movie data and displays a pie chart visualizing the distribution of languages.

* The languageData object is created using reduce to accumulate the number of movies for each language.
* The languageChartData array is then constructed by converting the object into an array suitable for the PieChart component.

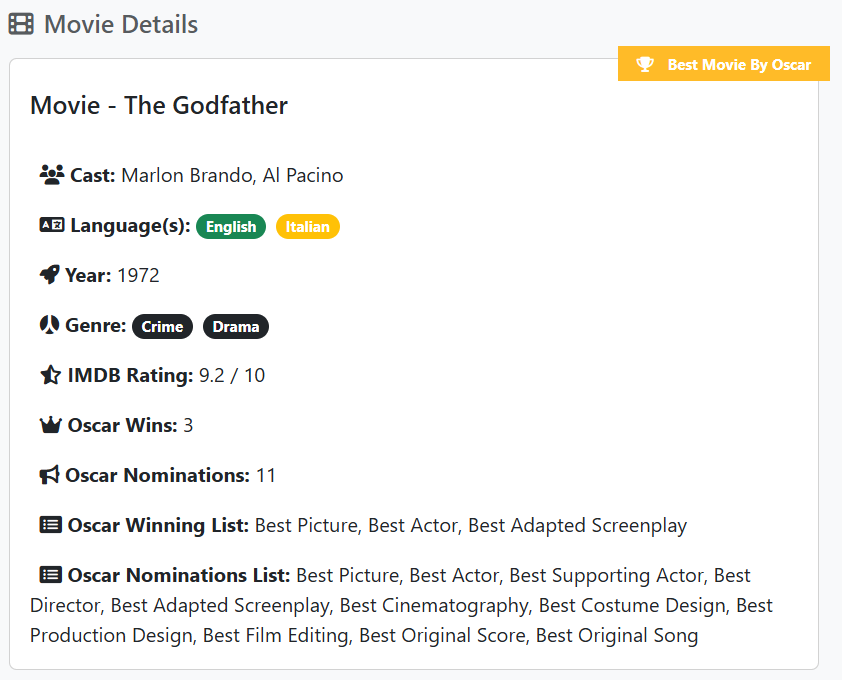
Pie Chart Configuration:



* A PieChart component from recharts is used to render the pie chart.
* The Pie component defines the data, data key, animation, and other visual properties.
* Cell components are used to define the color of each pie slice.
* Tooltip and Legend components enhance user interaction by providing hover information and labels for data slices.

1. **Movie Details Card:**

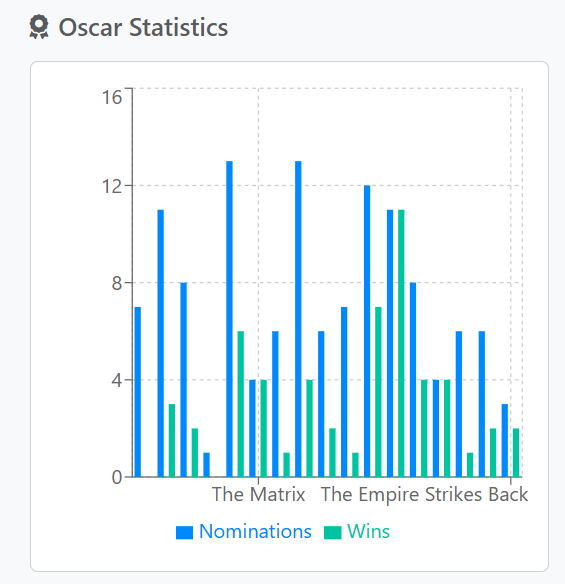
MovieDetailsCard component effectively displays movie details in a visually appealing card format.



* Takes a movie object as a prop containing movie details.
* Displays movie title, cast, languages, year, genre, IMDB rating, Oscar wins, nominations, and winning/nominated categories.
* Conditionally renders a "Best Picture" ribbon if the movie won the award.
* Uses Bootstrap classes for styling and layout.

**Oscar Overview:**

OscarOverview component effectively creates a bar chart visualizing Oscar nominations and wins for each movie in the data set.



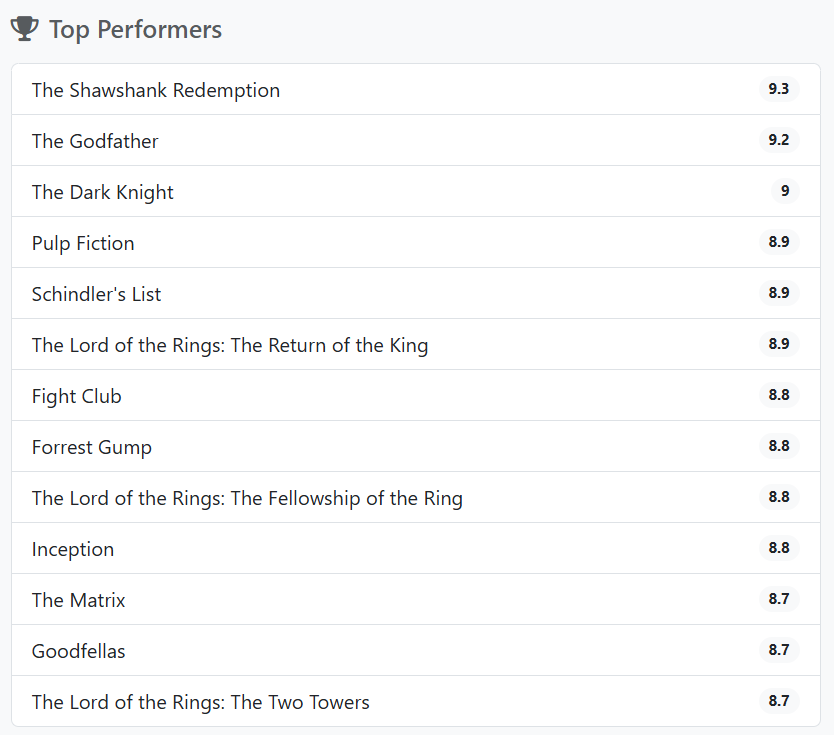
* Processes movie data to create chartData with movie titles, nominations, and wins.
* Renders a BarChart using recharts components.
* Configures the chart with gridlines, axes, tooltips, legend, and colored bars for nominations and wins.

**Search Filter:** SearchFilter component effectively creates a search and filter bar for a movie database application.



* Provides search and filter functionality for movie data.
* Offers options to filter by year, genre, and IMDB rating.
* Updates filtered data based on user input and selections.

**Top Performers:** TopPerformers component effectively displays a list of top-rated movies with the following key features:



* Sorting: Sorts the input data by imdb\_rating in descending order to find the top-rated movies.
* Display: Displays a list of the top movies with their titles and ratings.
* Highlighting: Highlights the currently selected movie by changing its background color and text color.
* Click Handling: Handles click events on each movie title to update the selected movie state and pass it to the parent component.