React-Movies-List

Ended

React

Easy

Max Marks - 18

DOWNLOAD ZIP

New Submission

Github URL *

SUBMIT

Best Score -18

Latest Submission

Description | Submissions

React-Movies-List

Submission Instructions [Please note]

Maximum Marks - 18

- The Submission should not contain spaces, for example /rct-101 folder/eval will not work
- Do not push node_modules and package_lock.json to github

```
✅ able to submit the app - 1 mark ( minimum score )
✅ On page load basic structure is present(
   1. Button should be there to toggle between the components with the correct text Content.
   2. Movie container should be present with a list of movies after fetching from json server)
   - 3 marks
✅ Able to filter movies, series, and games by making appropriate requests using query params - 3 marks
✅ Able to sort the data by selecting oldest first - 2 marks
✅ Able to sort the data by selecting the newest first - 2 marks
✅ On clicking on Add Movie button able to see the AddMovie component and the MovieList component should not exist on DOM
✅ Able to add new movies using form and data is getting updated in real-time on UI and dbjson - 4 marks
✅ Able to apply both sort and filter at a time - 2 marks
```

Installation

- Use node version(LTS) should be `v16.16.0`
- Don't change/override package.json
- please make sure you do not push package-lock.json
- Download and unzip the boilerplate file and then copy the "**contents**" of the unzipped file in the Masai Folder.
- Navigate to the Masai Folder, in VS Code.
- Run the following commands inside,
  - `npm install --engine-strict`
  - `npm start`
  - `npm run server` -> to start the json-server

- *Note*:

1. Libraries need to be installed by yourself
2. Make sure that the json-server is up and running at port `8080`
3. Create a .env file. Include `REACT_APP_JSON_SERVER_PORT=8080` in it and restart the react server
4. Use `http://localhost:${process.env.REACT_APP_JSON_SERVER_PORT}` as the json-server URL where ever you use `http://localhost:8080`

Not following the above instructions will lead your test cases to fail

Problem Statement

- You need to build a React application that fetches movie details and displays them as cards upon loading.
- The application should allow users to add new movie details by clicking on the `Add Movie` button.
- The user should be able to filter the data based on the `type`.
- The user should be able to sort the data based on the `year`.
  - Oldest first(The old movies will be visible first)
  - newest first(The new movies will be visible first)

Understanding Component Structure

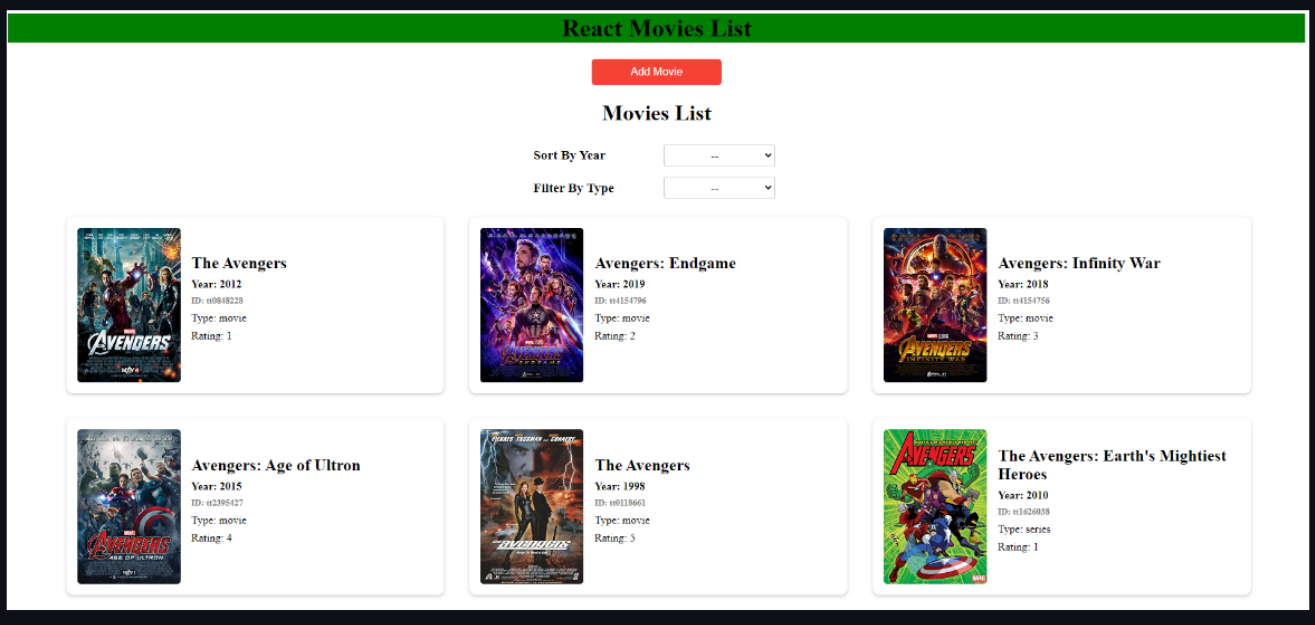
- Components
  - MovieList
  - MovieCard
  - AddMovie
- App.js

Note - `Make sure you use only the given components and don't create new files and folders. Changing the component name, and structures might result in giving you zero marks.`

Understanding Data Structure

- db.json (go through db.json file to understand the data structure)

Note - `Make sure you use only the given data and don't create new data. Changing data might result in giving you zero marks`



App

- This component will be displaying either the `AddMovie` or `MovieList` component at a time. (By default we should see

`MovieList` only)

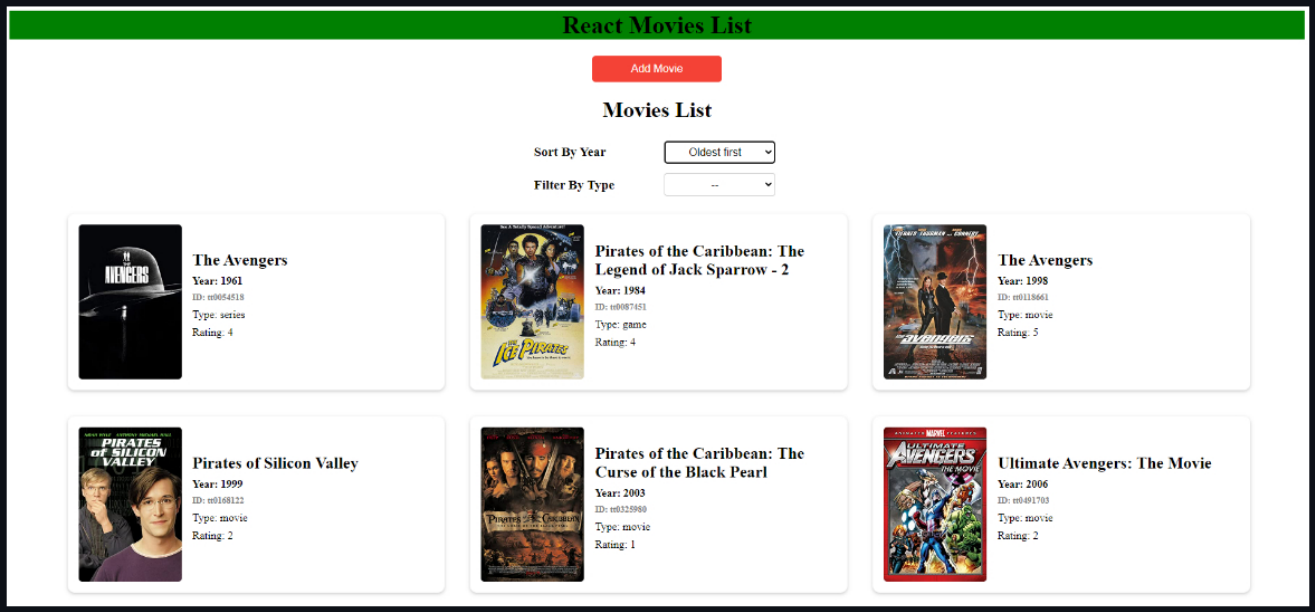
- There should be a button whose text content will toggle between `Add Movie` or `Show Movies`
- By default, the button should have textContent `Add Movie`. (Now the `MovieList` component should exist on DOM)
- When we click on `Add Movie` the textContent of the button should be changed to `Show Movies`.(Now it should display only a form(`AddMovie` component) and the `MovieList` component should not exist on DOM)

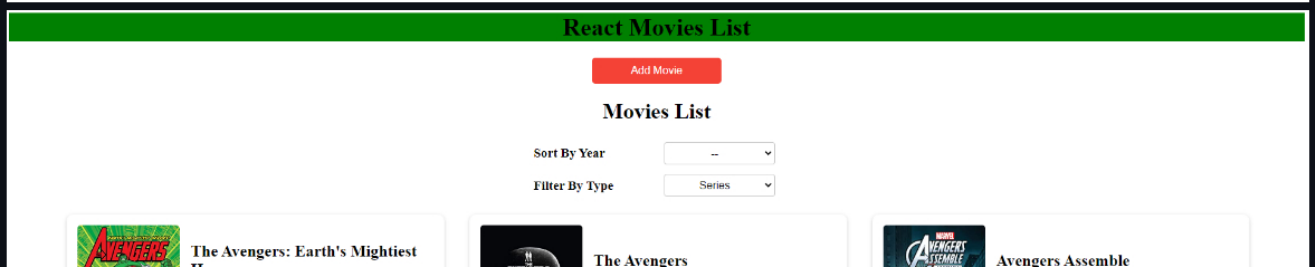
MovieList

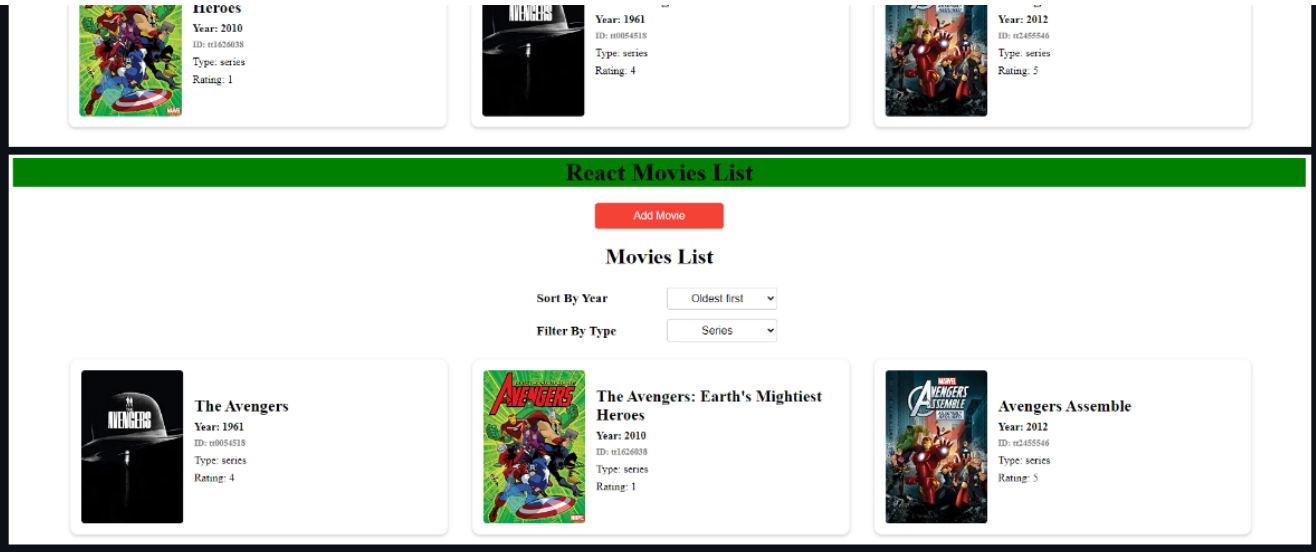
- This component will handle fetching movie data and displaying them using MovieCard.
- Here we will also be having the option to sort the Movies based on the `year` and filter the movies based on the `type`.(There are 2 select tags present to use the feature, and data should be updated on DOM without any extra button)
- For sorting the movies based on the `year`. There should be two options.
  - Oldest first(The old movies will be visible first)
  - Newest first(The new movies will be visible first)
- The combination of sorting and filtering should also work
- By default, there will be no sorting or filtering, the order of movies will be based on the fetched data(similar to the data order in db.json).

Note 1:-

1. The select tag for sorting should have the following options (`data-testid="sort"` provided in boilerplate)
   - there should be a default option with the text Content `--` and value should be "default".
   - there should be an option with the text Content `Oldest first` and value should be "oldest-first" for displaying the oldest movies first
   - there should be an option with the text Content `Newest first` and value should be "newest-first" for displaying the latest movies first
2. The select tag for filter should have the following options (`data-testid="filter"` provided in boilerplate)
   - there should be a default option with the text Content `--` and value should be "default"
   - there should be options to select movies of a specific type with the following options `Movie`, `Series`, `Game` and values should be "movie", "series", "game" respectively.



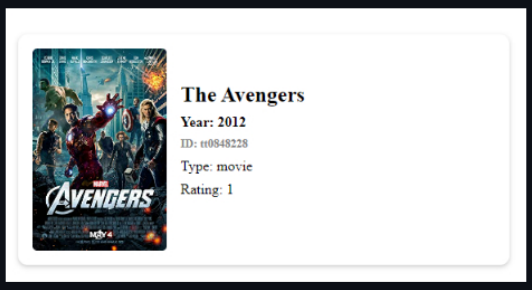




Note 2

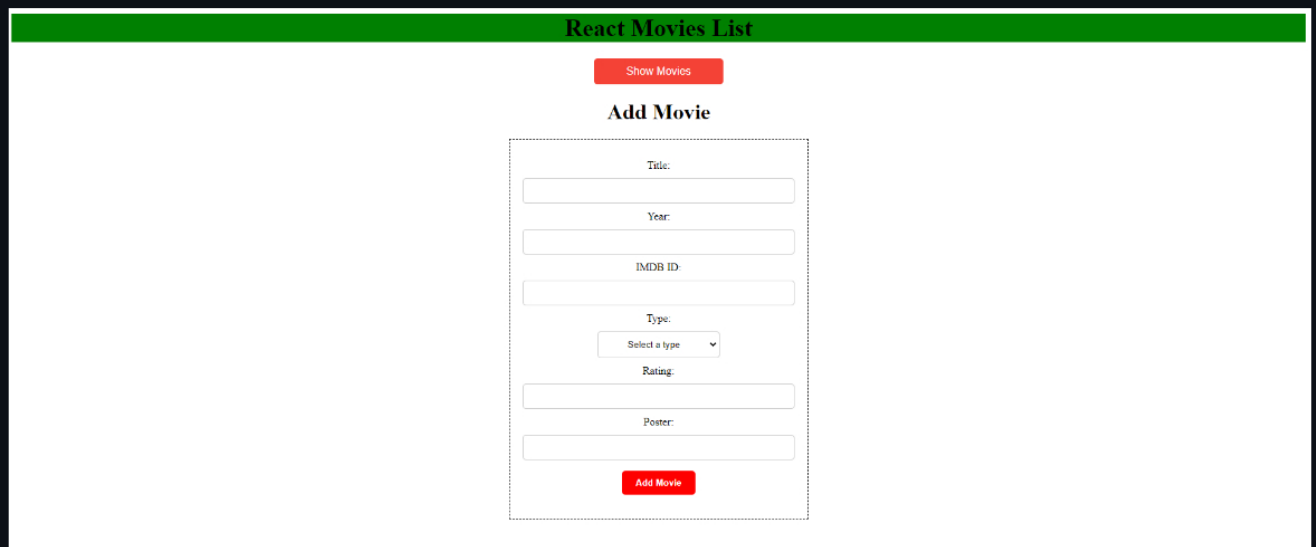
- You need to use `fetch`, to get the movie details.
- To sort and filter the movies, you need to use query parameters and make a request to the JSON server and get the required data. It's mandatory not to perform sorting or filtering in-app and instead, use query parameters to sort, and filter the data on the server side. refer to json server documentation for query params:- https://www.npmjs.com/package/json-server

MovieCard



- This component will display the movie details in detail.
- The div with `data-testid="movie-card"` should have the following HTML elements - `img` tag should have `src` of the url the movie(`poster`) - `h2` tag should have text content `title` of the movie - `h4` tag should have the text content `year` as `Year: {year}` - `h6` tag should have text content `imdbID` as `ID: {imdbID}` - `p` tag should have the `type` as `Type: {type}` - `p` tag should have the `rating` as `Rating: {rating}` - The order of the HTML element should be the same as above, and go through the component `image` for better understanding.

AddMovie



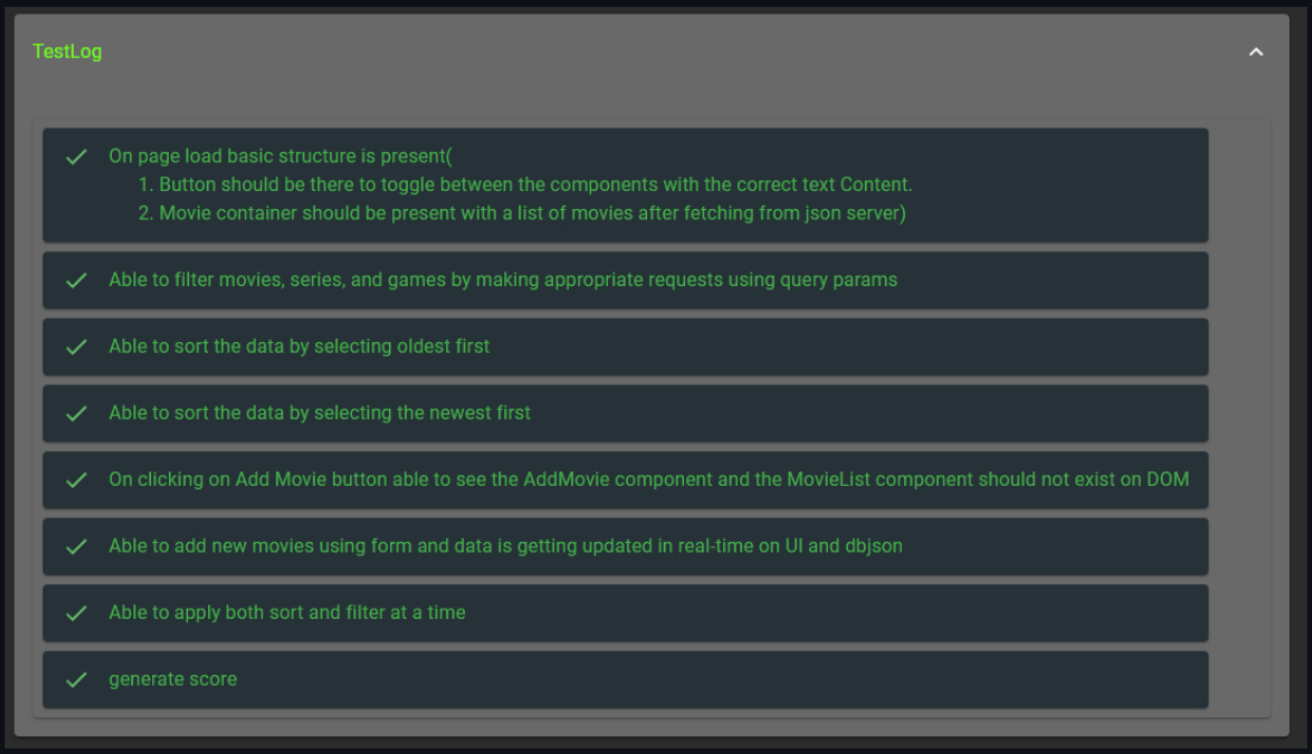
- This is just a `form` where we can enter the details of the movie we want to add.

- After submitting the form, make a post request to the required endpoint and update db.json.
- When you click on the `Show Movies` button it should display `MovieList` and which in turn should display the list of all the movies along with the added movie(this component should make a fetch request whenever it gets rendered on the DOM so that it will display the updated Movies data).

Note:- The input tags inside the form should have the following attributes.

1. input for the title should `name="title"`
2. input for the year should have `name="year"`
3. input for imdbID should have `name="imdbID"`
4. select tag for selecting movie type should have `name="type"`
   - options should be there for `Movie`, `Series`, and `Game` with the same text content, and the default option should have textContent `Select a type`
5. input for rating should be having `name="rating"`
6. input for the poster should be having `name="poster"`
7. You are free to create the label tags.

The problem is tested on CP and all testing cases are working as per the expectation



General Instructions (*IMPORTANT*)

1. Do not use Global CSS, instead use `<componentName>.module.css` convention for CSS in that file.
2. Do Not Remove `data-cy="xxxx"` from anywhere, this is used by testing tools to test your code, and removal of this will lead to a low score.
3. Make sure you use only the given components and don't create new files and folders as changing the component name, or structures might result in giving you zero marks
4. Make sure you use only the given data and don't create new data, as changing data might result in giving you zero marks.

General guidelines

- The system on cp.masaischool.com may take between 1-20 minutes for responding,
- so we request you to read the problem carefully and debug it before itself
- we also request you not just submit it last minute
- try to keep one submission at a time