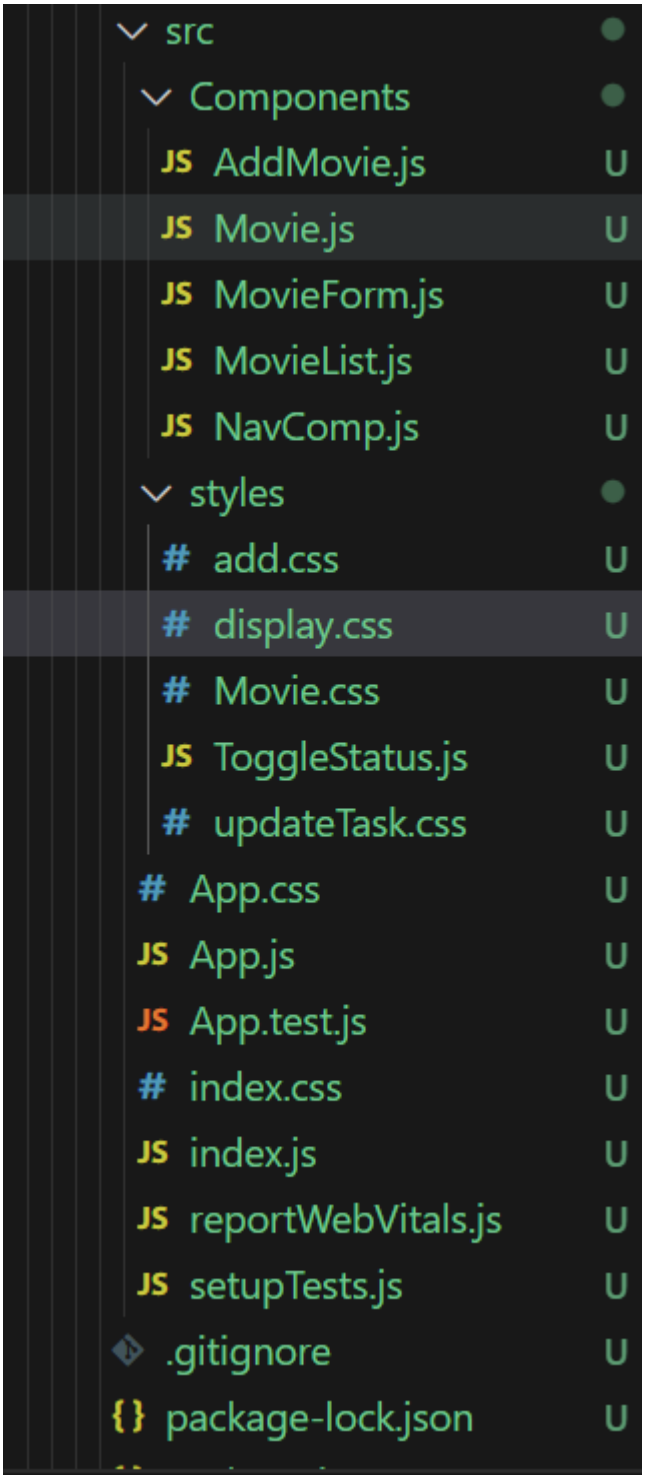
Project setup:

- Create the new directory for the project .

1. **Backend:**
   i. Create a folder for backed inside the main directory of the project.
   ii. Create the index.js file for writing the server-side code for backend, which deals with api calls and database.
   iii. Create a file for schema that validates the movies while fetching the movies from the database and uploading the new movies without any missing fields.
   iv. Install the required packages in the Express file (i.e., backend folder) using the "npm install" command
   v. Make sure that you have the package.json and package-lock.json files in the backend folder.
   vi. If you have installed any of the node packages in your server file, then you will have the node modules folder

2. **Frontend**
   i. In the front-end we are using react so we use "npx create-react-app appname" for creating the project setup folder.
   ii. Once the react app is created we have to install the packages that we used in the front-end app using "npm install" command.
   iii. In the front we have to start with App.js folder , which will reflect the the child components in it to render the complete application.



iv.

v. This is the “src” folder structure I used for completing the assignment.

vi. Inside the App.js component I used NavComp.js component which acts as my Navbar in the application. (This has a Title and a button that enables the form to add new movie)

vii. I create a component called AddMovie.js for the add movie form when user clicks on the add movie button in the Navbar.

viii. To display list of movies in the app I used MovieList.js Component, which fetches the movies list from the database using an API call and renders then using the Movie.js, this component takes the each movie’s data as props for displaying it.

ix. Once we displayed the details of movies, we must add the buttons to edit and delete the movies from the list in database, so I Added a component that takes the pop-up from for editing the movie details (i.e, is MovieForm.js which edits the movie).

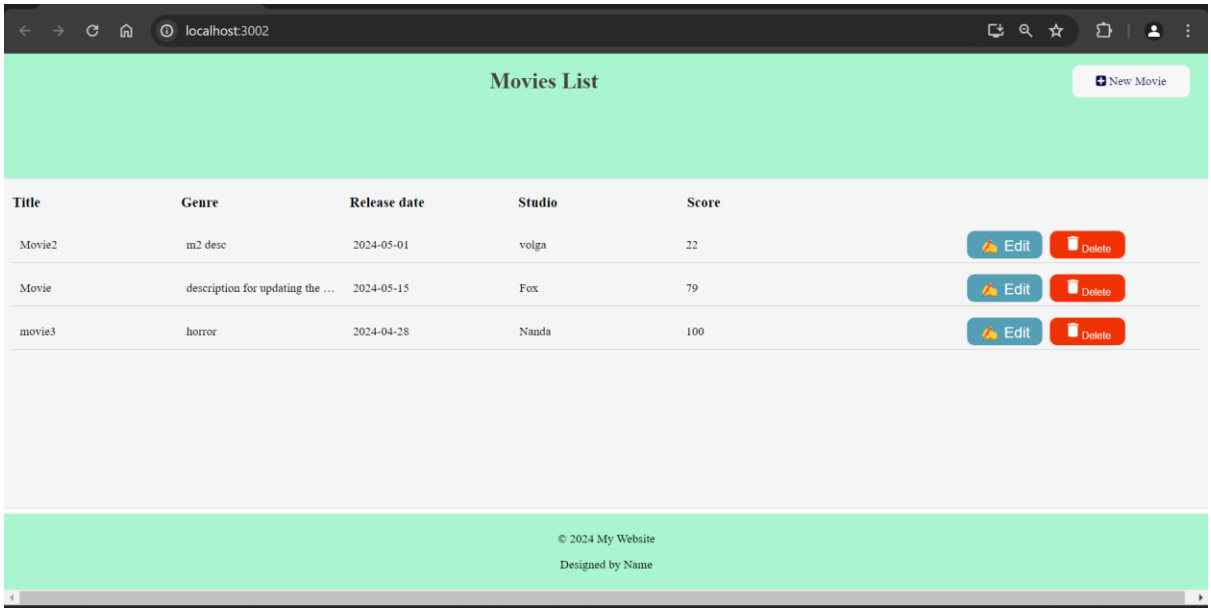
x. Finally, a button that deletes the movie from the database and removes it from the list in UI.

Conclusion: With these files I created for making the Favourite movie Listing App with CRUD operation from database using the Express JS and react app is Implemented. For better understanding, please go through the code files.

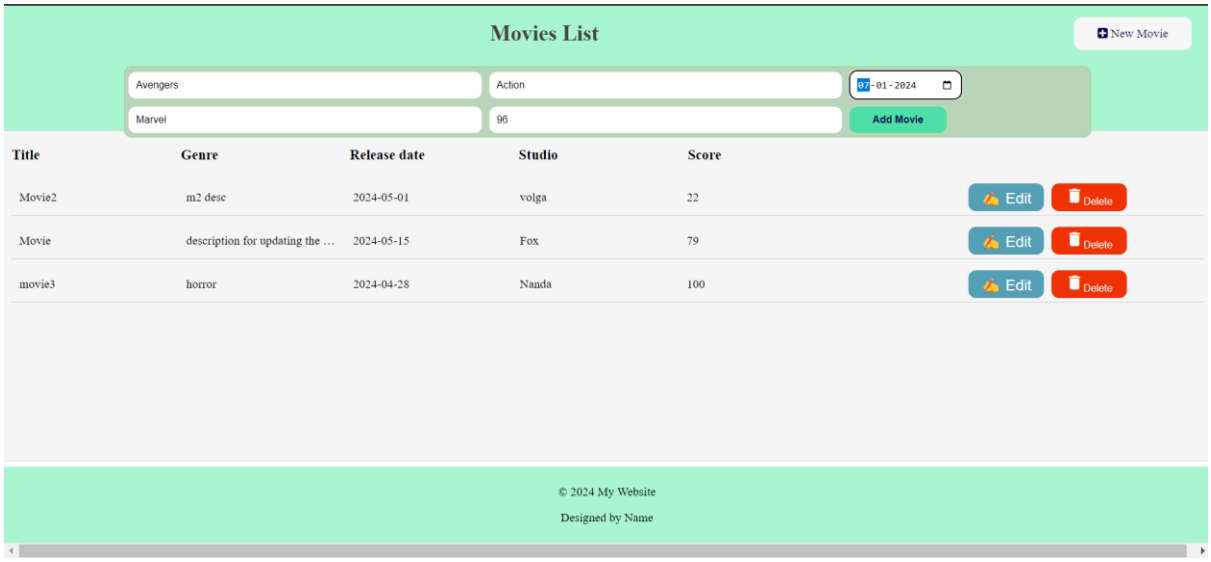
I have added all the files that changed or created in the react app and in the backend server as well.

Output Screenshots:

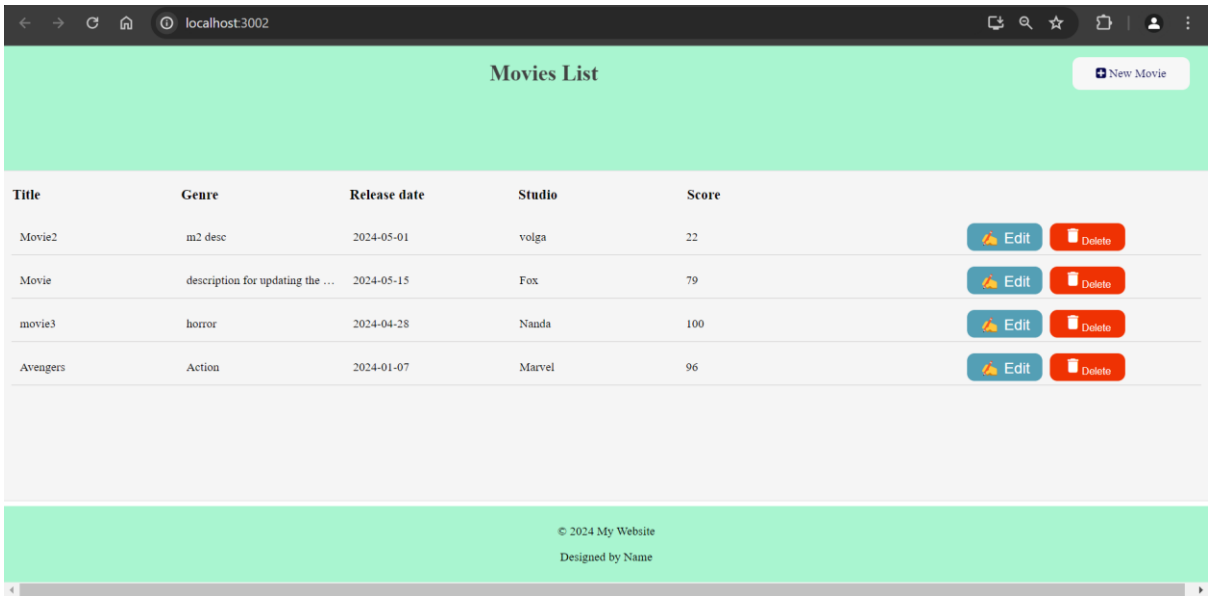
1. Home page showing the movies list fetched from the database.



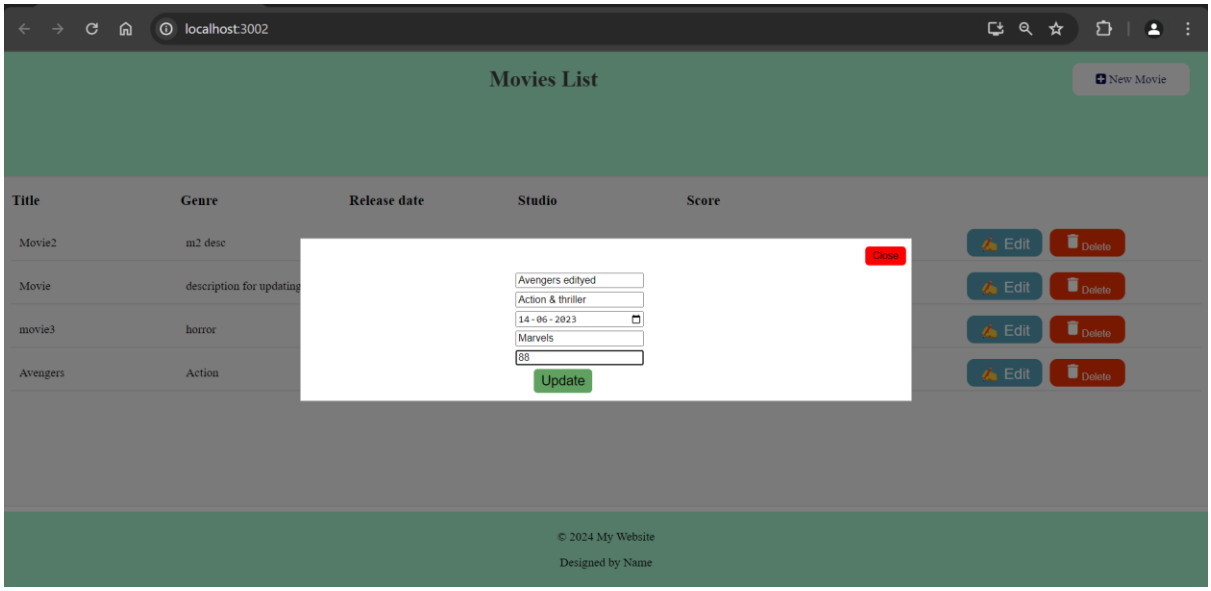
2. Add Movie button clicked enables the form to add the new movie to the database



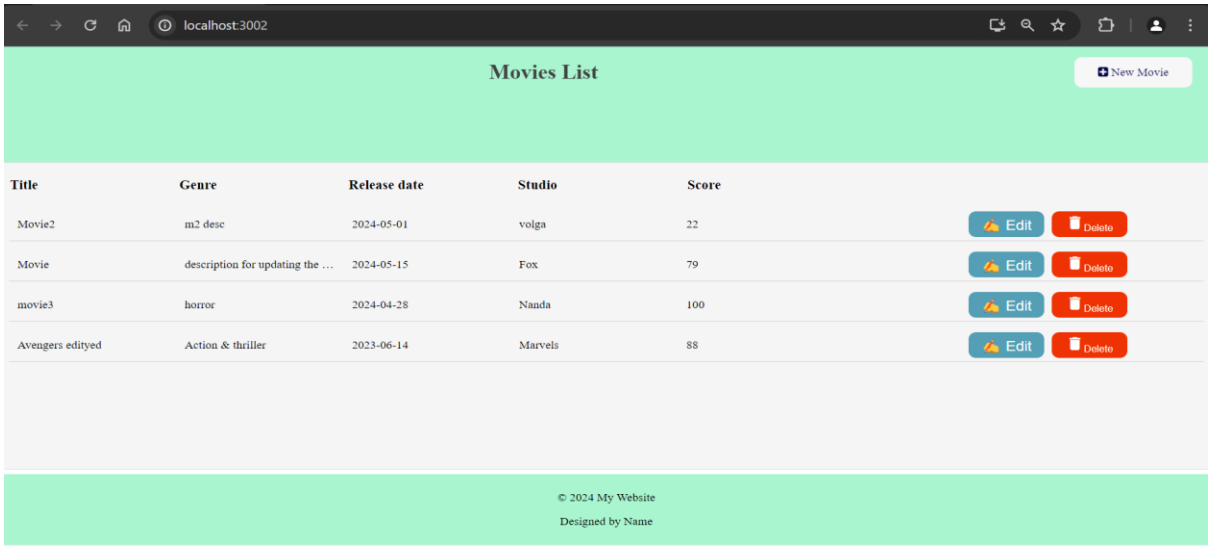
3. Update movie list with the newly added movie



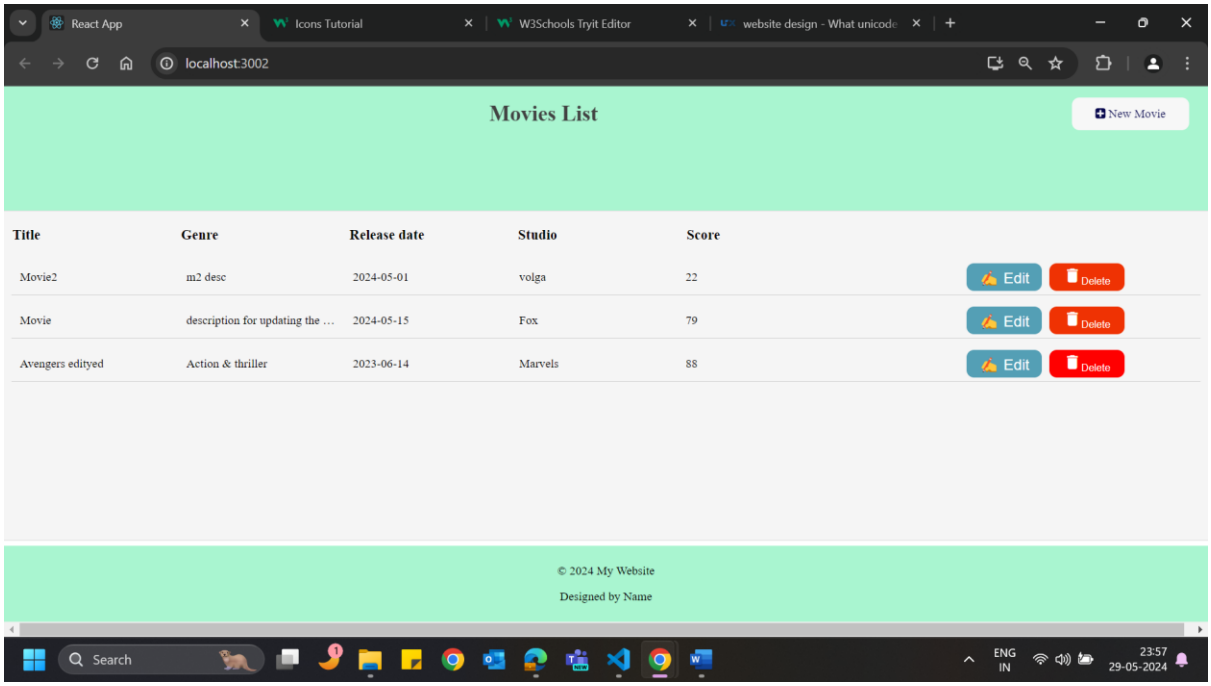
4. Editing the movie to update the details of the movie in the list



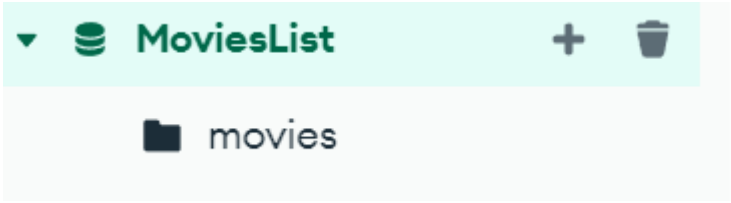
5. After updating the movie details screenshot with the edited movie



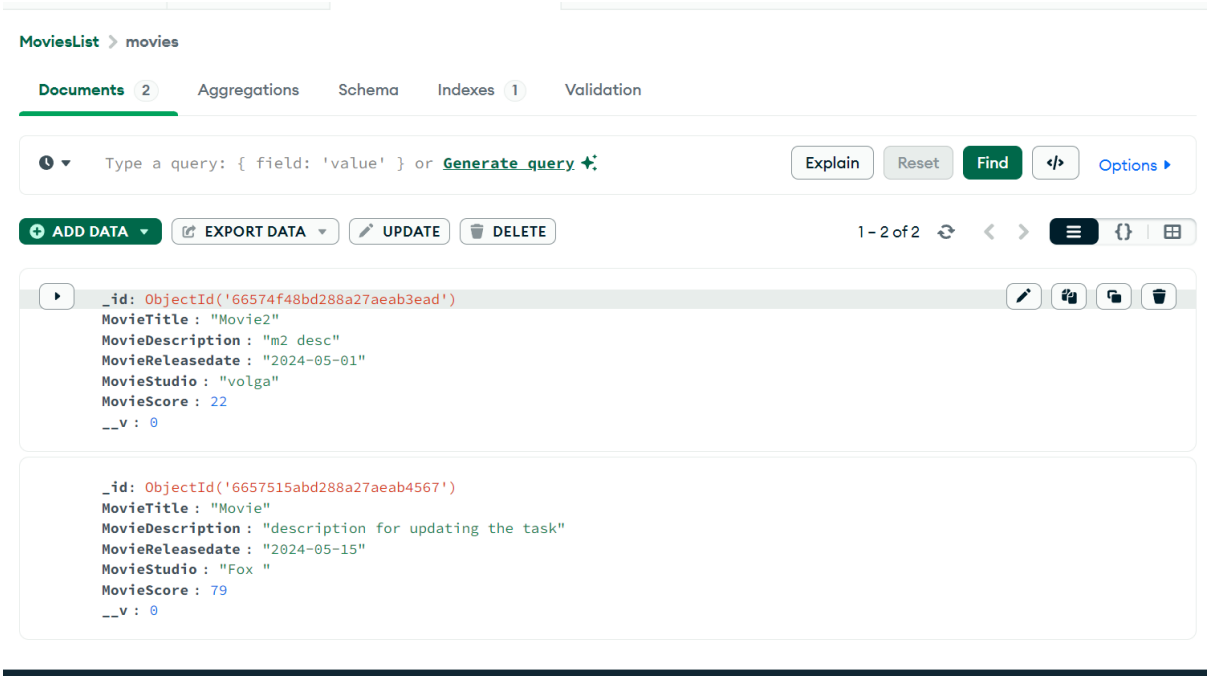
6. Deletion of movie is successful if the delete button is clicked



7. This is the database and collections I used in mongo DB, MoviesList is the database name and Movies is the collection used for storing movies.



8. The movies list I have after the delete operations is


MoviesList > movies
Documents 2
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
1 - 2 of 2
_id: ObjectId('66574f48bd288a27aeab3ead')
MovieTitle : "Movie2"
MovieDescription : "m2 desc"
MovieReleasedate : "2024-05-01"
MovieStudio : "volga"
MovieScore : 22
__v : 0
_id: ObjectId('6657515abd288a27aeab4567')
MovieTitle : "Movie"
MovieDescription : "description for updating the task"
MovieReleasedate : "2024-05-15"
MovieStudio : "Fox "
MovieScore : 79
__v : 0