React – Applying Redux

(1) What is Redux?

Ans: Redux is an open-source JavaScript library used for managing the state of a web application in a predictable way. It is often employed with React, a popular JavaScript library for building user interfaces, although it can be used with other frameworks or libraries as well.

(2) What is Redux Thunk used for?

Ans: Redux Thunk allows you to write action creators that return functions instead of plain objects. These functions, referred to as "thunks," can contain asynchronous logic, such as API calls, and can dispatch multiple actions during their execution. This is particularly useful for handling asynchronous operations like fetching data from a server and updating the state based on the result.

Ex:

const incrementAsync = () => {

return (dispatch) => {

setTimeout(() => {

dispatch(increment()); // Dispatching a synchronous action after a delay

}, 1000);

};

};

// Dispatching the thunk

store.dispatch(incrementAsync());

(3) What is Pure Component? When to use Pure Component over

Component?

Ans:

In React, a pure component is a component that renders the same output for the same state and props. It's designed to optimize performance by avoiding unnecessary re-renders.

Ex:

import React, { PureComponent } from 'react';

class PureExample extends PureComponent {

render() {

const { name, age } = this.props;

console.log('Rendering PureExample component.');

return (

<div>

<h2>Name: {name}</h2>

<p>Age: {age}</p>

</div>

);

}

}

export default PureExample;

* Here are some situations where using a Pure Component in React might be more advantageous than a regular Component:

1. Performance Optimization

2. Props and state immutability

3. Simple Comparisions

Ex:

import React, { Component } from 'react';

class MyComponent extends Component {

shouldComponentUpdate(nextProps, nextState) {

return nextProps.someValue !== this.props.someValue;

}

render() {

return <div>{this.props.someValue}</div>;

}

}

(4) What is the second argument that can optionally be passed

to setState and what is its purpose?

Ans: In React, the setState function can take an optional second argument, which is a callback function that will be executed once the setState operation is complete and the component has been re-rendered.

Syntax for ‘setState’ with the callback,

Ex: setState(updater , callback);

Here's an example of using the callback function:

Ex:

import React, { useState } from 'react';

function ExampleComponent() {

const [count, setCount] = useState(0);

const handleIncrement = () => {

setCount(count + 1, () => {

// This callback is executed after the state has been updated and the component has been re-rendered

console.log('Count has been updated:', count);

});

};

return (

<div>

<p>Count: {count}</p>

<button onClick={handleIncrement}>Increment</button>

</div>

);

}

export default ExampleComponent;

* The handleIncrement function uses setCount to update the count state by incrementing it.
* The second argument to setCount is a callback function that logs the updated count to the console.

(5) Create a table and seach data from table using React Js?

Code:

import React, { useState } from 'react';

const TableSearch = () => {

  const initialData = [

    { id: 1, name: 'Shirt Men', price: 250, category: 'Fashion' },

    { id: 2, name: 'Brook Shirt', price: 300, category: 'Fashion' },

    { id: 3, name: 'Lake A Shirt', price: 229, category: 'Fashion' },

    { id: 4, name: 'Bench Shirt', price: 288, category: 'Fashion' },

    { id: 5, name: 'Mesh Shirts', price: 280, category: 'Fashion' },

  ];

  const [data, setData] = useState(initialData);

  const [searchTerm, setSearchTerm] = useState('');

  const handleSearch = (e) => {

    setSearchTerm(e.target.value);

  };

  const filteredData = data.filter((item) =>

    item.name.toLowerCase().includes(searchTerm.toLowerCase())

  );

  return (

    <div className='container'>

      <div className="row mt-5">

        <div className="col-md-5 mx-auto">

          <div className="input-group">

            <input type="search" value={searchTerm} onChange={handleSearch} className="form-control border rounded-pill" placeholder='Search by name' id="example-search-input" />

          </div>

        </div>

      </div>

      <table class="table">

        <thead>

          <tr>

            <th scope="col">ID</th>

            <th scope="col">Name</th>

            <th scope="col">Price</th>

            <th scope="col">Category</th>

            <th scope="col">Actions</th>

          </tr>

        </thead>

        <tbody>

          {filteredData.map((item) => (

            <tr key={item.id}>

              <td>{item.id}</td>

              <td>{item.name}</td>

              <td>{item.price}</td>

              <td>{item.category}</td>

              <td>

                <button type="button" className="btn btn-labeled btn-info me-2"><span className="btn-label"><i className="fa fa-refresh" /></span>Edit</button>

                <button type="button" className="btn btn-labeled btn-danger"><span className="btn-label"><i className="fa fa-trash" /></span>Delete</button>

              </td>

            </tr>

          ))}

        </tbody>

      </table>

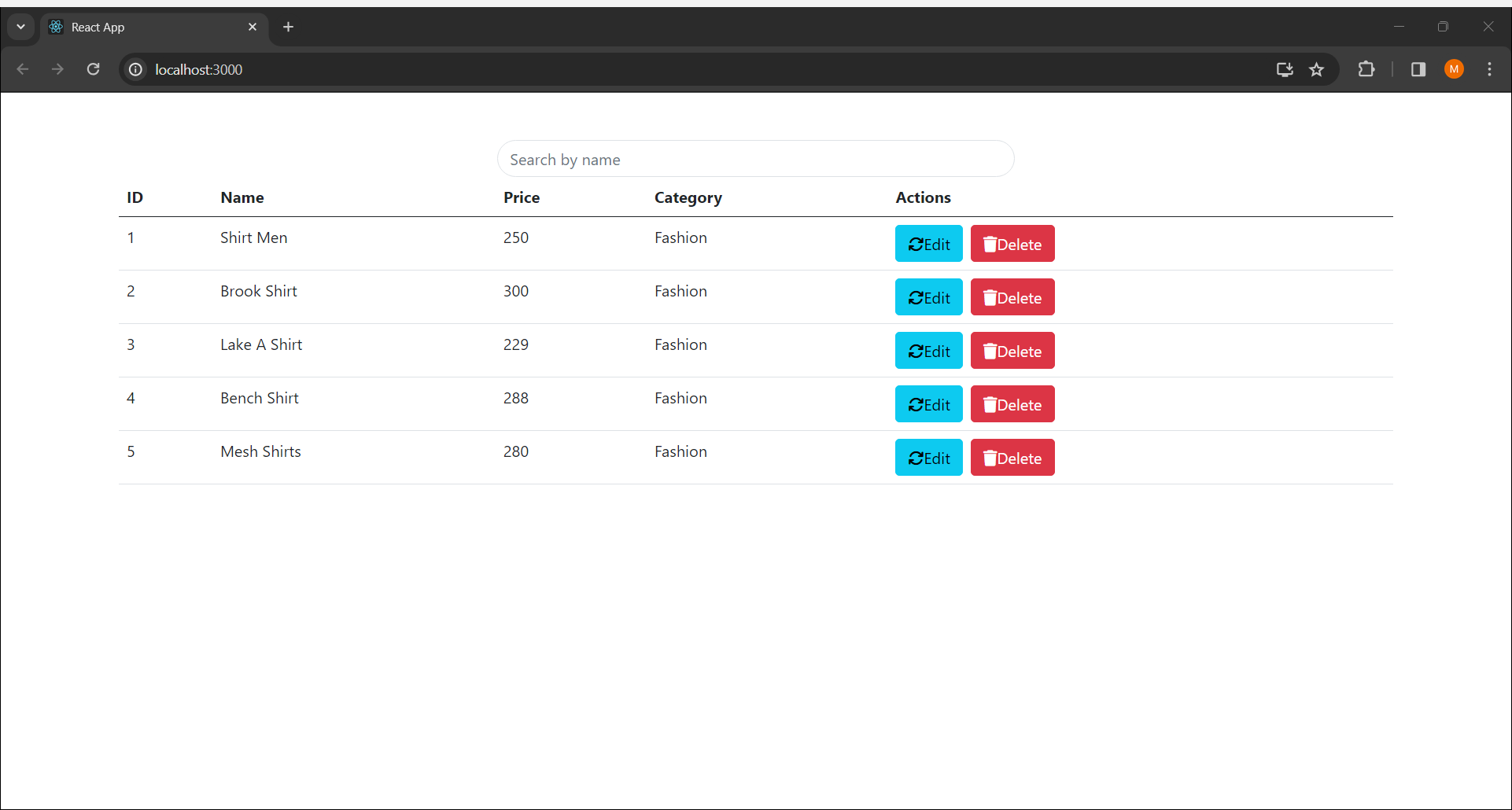
    </div>

  );

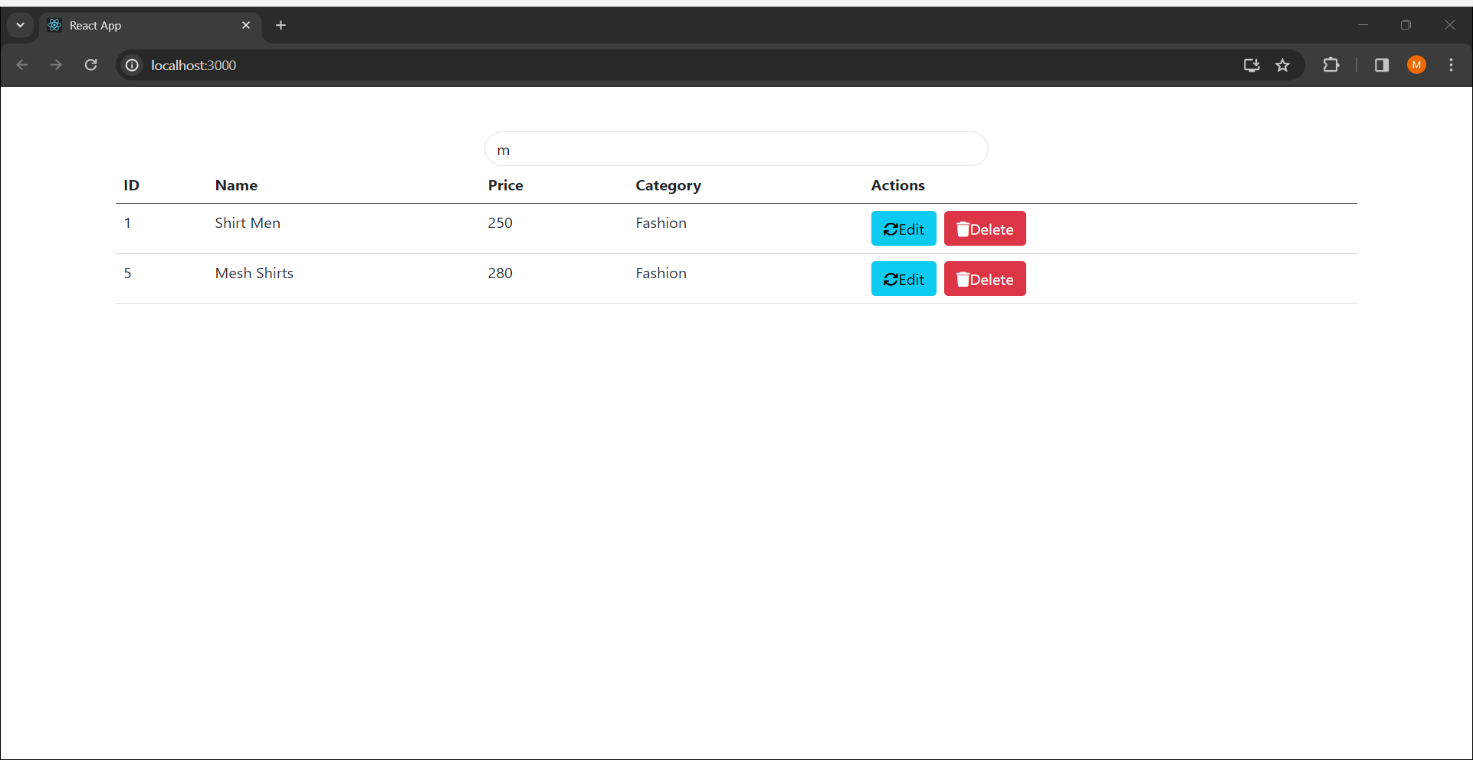
};

export default TableSearch;

Output:



Workable:



(6) Create Login registration with CRUD Application using API (Redux)