# Redux Toolkit - Practice

## 1. Install Redux Packages

```
npm i react-redux @reduxjs/toolkit
```

## 2. Create Redux Store

In the folder **src**, create a new folder: **Store**. Inside this folder, create a file named **store.js**.

```
import {configureStore} from "@reduxjs/toolkit"

export const store = configureStore({
   reducer:{ }
})
```

## 3. Sharing Redux store using Provider

If you want to share the store data around all the components, it should wrap the App component, as it is the main component. Wrap the App component by Provider that provides react store to the React app.

```
import { Provider } from "react-redux";
import { store } from "./Store/store";

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <Provider store={store}>
    <React.StrictMode>
      <App />
    </React.StrictMode>
  </Provider>
);
```

## 4. Create a state Slice

In `src` folder, create a new folder: `Features`. The Features folder will contain the reducers of your application.

- Create a new file named **CustomerSlice.js** inside the **Features.**
- Set the name property and initial value.
- add one or more reducer functions to define how the state can be updated.
- Export all the reducers so it can be accessed outside the file.

```
import { createSlice } from "@reduxjs/toolkit";
import CustomerData from "../CustomerData";

const initialState = CustomerData;
const customerSlice = createSlice({
  name: "customers",

  initialState,

  reducers: {},
});

export default customerSlice.reducer;
```

a **Note: Every aspect of the application that you want manage state, you can create a reducer for each.**

## 5. Add Slice Reducers to the Store

In **src/Store/store.js**, import the customerReducer from the **Features/customerSlice.js** file.

```
import { configureStore } from "@reduxjs/toolkit";
import customerReducer from "../Features/CustomerSlice"

export const store = configureStore({
    reducer: {
        customers: customerReducer,
    }
})
```

## 6. Getting data from the store

The use of the **useSelector** hook is accessing Redux State and to extract data from the Redux store state.

```
import React from "react";
import { useSelector } from "react-redux";
function CustomerList() {
//Retrieve the current value of the state and assign it to a variable.

  const customers = useSelector((state) => state.customers);

  return (
    <div>
      <h3>Customer List</h3>

      <table className="table table-striped table-warning">
        <tbody>
          {customers.map((customer, index) => (
            <tr key={index}>
              <td>{customer.id}</td>
              <td>{customer.name}</td>
              <td>{customer.email}</td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
}
export default CustomerList
```

# Basic CRUD Implementation using React and Redux Toolkit

## 7. Create a Reducer function in the customerSlice.js to add customer data

Here, implementing Create(add) operation of CRUD operations. In the **src/Features/CustomerSlice.js**, write code to create the reducer **addCustomer**. The **addCustomer** reducer will add the value of the customer state by pushing the new value to the state.

**State** is the current value of the state,
**Action** is triggered outside the reducer and provides a value as payload.
**Payload** is the value coming from the component that will be used to update the value of the state.

```
import { createSlice } from "@reduxjs/toolkit";
import CustomerData from "../CustomerData";

const initialState = CustomerData;
const customerSlice = createSlice({
  name: "customers",
  initialState,
  reducers: {
        addCustomer(state,action){
         state.push(action.payload)
       }

},
});
export default customerSlice.reducer;
export const {addCustomer} = customerSlice.actions
```

## 8. Dispatch add action to send payload to add customer data to the store

We will send customer data to the store via the code (**CustomerAdd.js**). Here, we will use the reducer(**addCustomer**) which is in the **customerSlice**. The reducer also has an action creator. Using the action creator, we will send customer data to the store.

```
import React,{useState} from 'react'
import { addCustomer } from '../Features/CustomerSlice'
import {useDispatch} from 'react-redux'

function CustomerAdd() {
   //declare the state variable to handle the form values

    const dispatch = useDispatch()
    Function addhandler(){
      if(id && name && password && email )
    dispatch(addCustomer({ id: id, name: name, email: email, password: password }))

  }
  return (
    <div>
      <h3>Add New Customer</h3>
      ..................
      ..................
      ..................
      <button type="button" onClick={addhandler}>
```

```
          Register Customer
        </button>

        </div>
          )
    }
    export default CustomerAdd
```

## 9. Dispatch delete action to delete data from the store

- Implement delete operation of CRUD operations. Create Reducer for Delete action. Open the **CustomerSlice** and create **deleteCustomer** function.

```
    deleteCustomer(state, action) {
        return state.filter((customer) => customer.id !== action.payload)
    },

export const {addCustomer,deleteCustomer} = customerSlice.actions
```

- Open the `CustomerList.js` component file and add the delete button to activate the `deleteHandler` function.

```
<tbody>
    {customers.map((customer, index) => (
      <tr key={index}>
        <td>{customer.id}</td>
        <td>{customer.name}</td>
        <td>{customer.email}</td>
         <td>
       <button onClick={() => deleteHandler(customer.id)}>Delete</button>
       </td>

    </tr>
    ))}
  </tbody>
```

Create a function called **deleteHandler**. The function will accept the parameter index that will be sent as payload to the **deleteCustomer** function in the reducer.

```
import { useSelector, useDispatch } from "react-redux";
import { deleteCustomer } from "../Features/CustomerSlice";

.....
const deleteHandler = (id) => {

dispatch(deleteCustomer(id));
 };
```

## 10. Create a Reducer function to update customer data.

Here, implementing Update operation of CRUD operations.

- Before creating Reducer function, Open the **CustomerList.js** component file and add the update link to open the **CustomerUpdate** component. This component is to modify the customer data. Send the customer id through the link to the **CustomerUpdate** component.

```
<tbody>
     {customers.map((customer, index) => (
       <tr key={index}>
         <td>{customer.id}</td>
         <td>{customer.name}</td>
         <td>{customer.email}</td>
         <td>
        <button onClick={()  =>deleteHandler(customer.id)}>Delete</button>
         </td>
            <td>
               <Link to={`/update/${customer.id}`} className="btn btn-info">
                    Update
               </Link>
            </td>
  </tr>

     ))}
   </tbody>
```

- Add the Route for the **CustomerUpdate** component in **App.js**

```
<Route path="/update/:id" element={<CustomerUpdate />} />
```

- In **CustomerUpdate** component, use **useParams** hook to retrieve the customer id from URL.

```
const { id } = useParams();
```

- In the same **CustomerUpdate** component, use **useSelector** hook to retrieve all the customers data from the store.

```
const customers = useSelector((state) => state.customers);
```

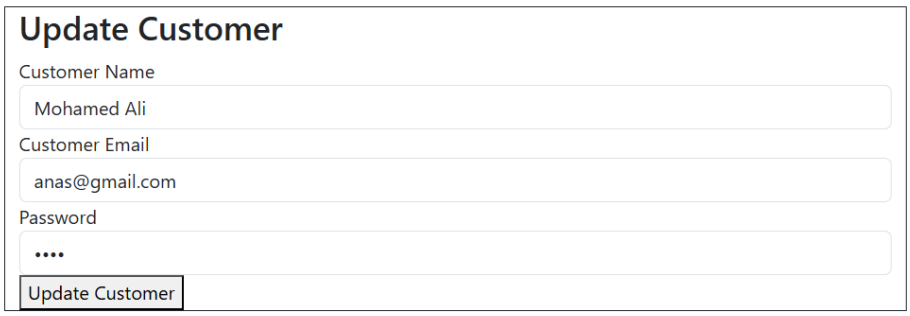
- In the same **CustomerUpdate** component, Find the required customer that is to be modified based on the id. The id you get through the URL from CustomerList component

```
const customer = customers.find(c => c.id === id);
```

- Assign the customer data to the state variable in order to place values on the UI for editing.

```
const [name, setName] = useState(customer.name);
const [email, setEmail] = useState(customer.email);
const [password, setPassword] = useState(customer.password);
```

- Design the CustomerUpdate component as follows:




- Call the updateHandler function in the on click event of the button.

```
<button type="button" onClick={updateHandler}>
      Update Customer
    </button>
```

- Write the **updateHandler** funtion that will be used to dispatch the updateCustomer reducer to Update values in the store

```
function updateHandler() {
       if (name && password && email)
         dispatch(
           updateCustomer(
              { id: id, name: name, email: email, password: password })
         );
        navigate("/")
  }
```

- In the `src/Features/CustomerSlice.js`, write code to create the reducer `Customer`. The `updateCustomer` reducer will update the value of the customer state by customer index.

```
      updateCustomer(state, action){
            const customer = state.find( cust => cust.id === action.payload.id)
            if (customer) {
              customer.name = action.payload.name;
              customer.email = action.payload.email;
              customer.password = action.payload.password;
            }
          }


export const { addCustomer,deleteCustomer,updateCustomer } = customerSlice.actions;
```