Redux Toolkit

Objectives and Outcomes

In this exercise you will learn to use Redux by Redux Toolkit. You will install and configure Redux, Redux Toolkit and use it within your React application. At the end of this exercise, you will be able to:

- Install and configure Redux (using redux toolkit) within your application.
- Configure your React application to make use of Redux.

Resources

Create ListOfUsesr.js file for display the data of this exercise.

```
const UsersData = [
  {
    id: 1,
    name: "Bum Trum",
    username: "bum",
  },
  {
    id: 2,
    name: "My Mo",
    username: "my",
  },
  {
    id: 3,
    name: "Joe Nguyen",
    username: "joe",
  },
  {
    id: 4,
    name: "Kevin Khang",
    username: "kevin",
  },
  {
    id: 5,
    name: "Kane Le",
    username: "kane",
  },
  {
    id: 6,
    name: "Bin Pham",
    username: "bin",
  },

];
export default UsersData;
```

Create React App, then Install and Configure Redux, Redux Toolkit

- As a first step you will create a new React App. Then install React-Redux and Redux toolkit into your application as follows:

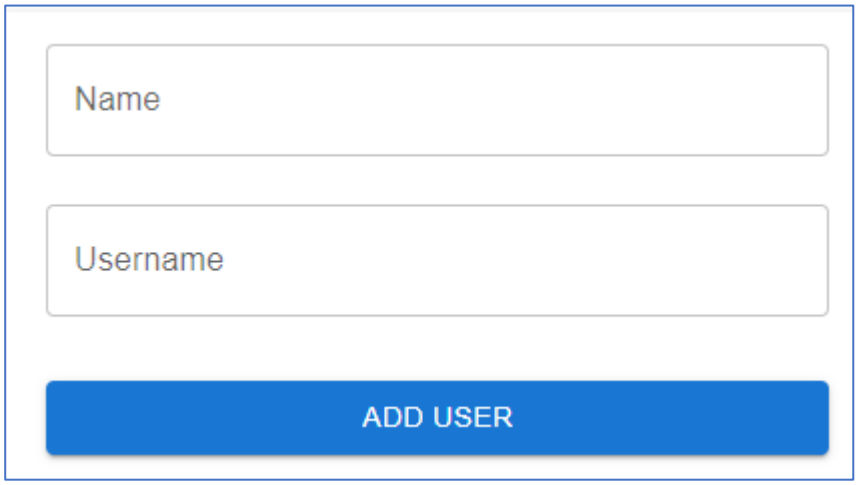
```
npx create-react-app ReduxDemo
npm install @reduxjs/toolkit react-redux
```

Your package.json file like:

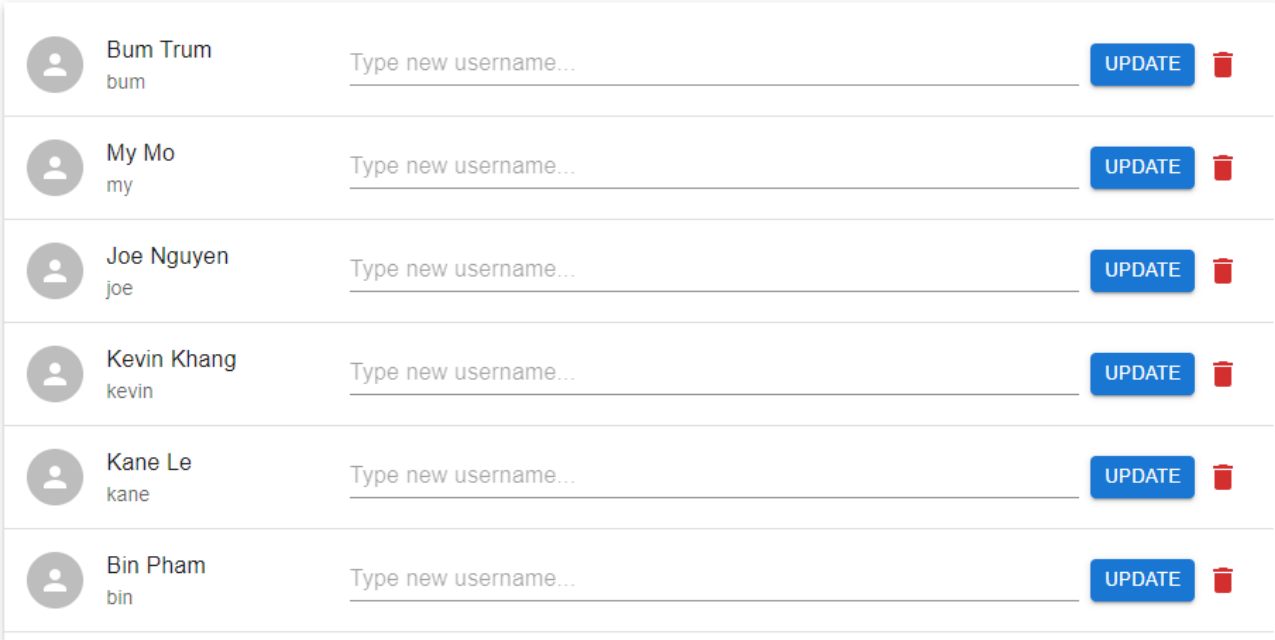
```
"react-redux": "^8.0.4",
"@reduxjs/toolkit": "^1.8.6",
```

Create two components for display the view.

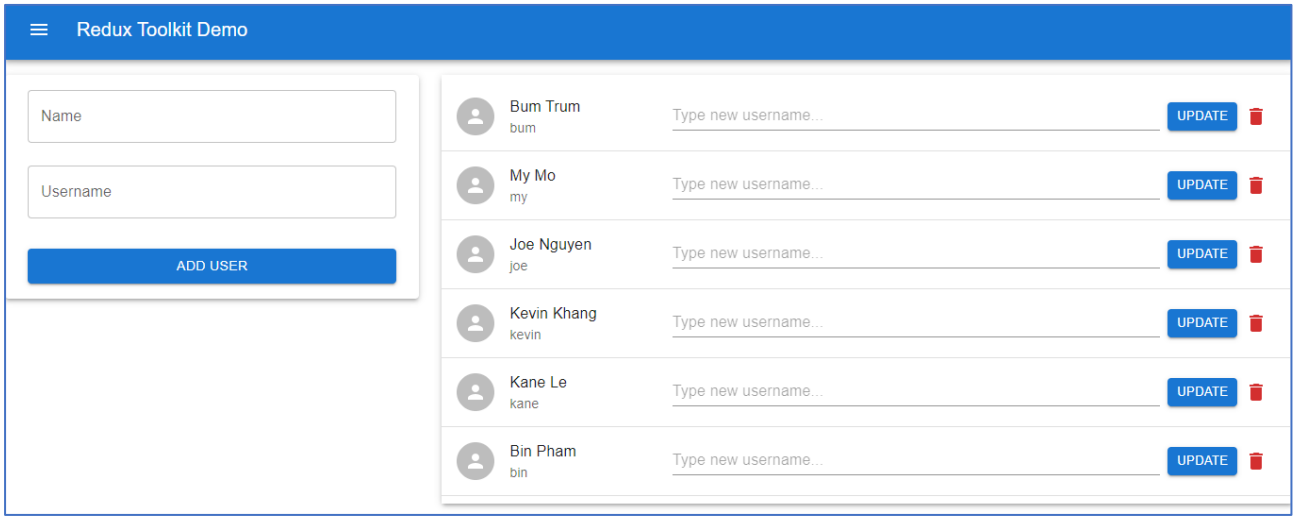
- Create the AddUser component to add the new user by yourself (use Materialize, Material UI or some libraries which support Material design), the screen display as follow:



- Create the User component to show the list of users, update the username and delete user by yourself, the screen display as follow:



Import two components to App.js file



Create a Redux Store

- Open the Index.js file, import configureStore and write codes as follow:

```
import { configureStore } from '@reduxjs/toolkit'
export const store = configureStore({
  reducer: {},
});
```

- Import the Redux store we just created, put a <Provider> around your <App>, and pass the store as a prop:

```
<Provider store={store}>
    <App />
</Provider>
```

Create a Redux State Slice

- Add a new file named src/features/Users.js. In that file, import the createSlice API from Redux Toolkit.

```
import { createSlice } from "@reduxjs/toolkit";
import UserData from '../ListOfUsers';
export const userSlice = createSlice({
    name: "users",
    initialState: {value: UserData},
    reducers: {
        addUser: (state, action)=>{ // Write code for addUser function
                },
        deleteUser: (state, action)=>{ // Write code for deleteUser fuction
                },
```

```
      updateUsername: (state, action)=>{ // Write code for updateUsername function
            }
   },
});
export default userSlice.reducer;
export const {addUser, deleteUser, updateUsername} = userSlice.actions;
```

Add Slice Reducers to the Store

- Next, we need to import the reducer function from the User slice and add it to our store. By defining a field inside the reducer parameter, we tell the store to use this slice reducer function to handle all updates to that state.
- Open Index.js file and update codes as follow:

```
....
import userReducer from './features/Users';
...
const store = configureStore({
  reducer: {
    users: userReducer,
  }
});
```

Use Redux State and Actions in React Components

- Now we can use the React-Redux hooks to let React components interact with the Redux store. We can read data from the store with useSelector, and dispatch actions using useDispatch.
- Open the AddUser component and update codes as follow:

```
...

import { useState } from 'react';

import { addUser } from '../features/Users';

import { useDispatch } from 'react-redux';

...

const dispatch = useDispatch();
const [name, setName]=useState('');
const [username,setUsername]=useState('');
```

- Update some codes for two Input get the states and set the states with onChange event:

```
        <TextField
              label="Name"
              name="name"
              value={name}
              onChange={(event) => {setName(event.target.value);}}
            />
            <TextField
             name="username"
              label="Username"
              value={username}
              onChange={(event) => {setUsername(event.target.value);}}
            />
        <Button onClick={()=> {
              dispatch(addUser({id: 0, name: name, username: username}));
              }}
              >
             Add user
            </Button>
```

- Update the addUser function in features/User.js:

```
…
reducer:{
addUser: (state, action)=>{
 state.value.push(action.payload);
  },
```

- Open the User component and update codes as follow:

```
…
import { useState } from 'react';
import { useSelector } from "react-redux";
import { useDispatch } from 'react-redux';
import { deleteUser, updateUsername } from '../features/Users';
…
const dispatch = useDispatch();
const userList = useSelector((state)=> state.users.value);
const [newUsername,setNewUsername]=useState('');
```

- Update some codes for getting the states and set the states with delete and update function:

```
{userList.map((user)=>{
    return(
      <>
<ListItem key={user.id} >
    <ListItemText primary={user.name} secondary={user.username} />
    <TextField
  placeholder='Type new username...'
    onChange={(e)=>setNewUsername(e.target.value)}
  />
  <Button
  onClick={()=>{dispatch(updateUsername({id: user.id, username: newUsername}));
    }}>
      Update
    </Button>
    <IconButton aria-label="delete" color="error"
    onClick={()=> {
      dispatch(deleteUser({id: user.id}));
      }}>
    <DeleteIcon />
  </IconButton>
  </ListItem>

</>
```

- Update the deleteUser and updateUsername functions in features/User.js:

```
…
reducer:{

addUser: (state, action)=>{

 …
  },

deleteUser: (state, action)=>{
   state.value = state.value.filter((user)=> user.id !== action.payload.id);
  },
updateUsername: (state, action)=>{
  state.value.map((user)=>{if(user.id === action.payload.id){
     user.username = action.payload.username;}
  });
},
```

Run your React app and see the result.

Conclusions

In this exercise you learnt to use React with Redux and Redux Toolkit.