# Lesson End Project

# Creating a React Application with Redux Store

**Project agenda:** To develop a React application that that demonstrates Redux Store

**Description:** This React app showcases Redux Store integration, featuring a basic counter with increment and decrement actions. Users can interact seamlessly, highlighting the efficient state management in a React environment.

**Tools required:** Node terminal, React app, and Visual Studio Code

**Prerequisites:** Knowledge of creating a React app and an understanding of the folder structure

**Expected deliverables:** A fully functional React application demonstrating Redux Store integration, enabling users to interact with a counter through increment and decrement actions

**Steps to be followed:**

1. Create a new React app
2. Install Redux and React Redux
3. Create a new file called index.js
4. Import Provider from react-redux in App.js
5. Run the app and view it in the browser

## Step 1: Create a new React app

1.1 Open your terminal and run the **npx create-react-app redux-store-demo** command

```
shreemayeebhatt@ip-172-31-22-250:~$ "npx create-react-app redux-store-demo
```

The above command will create a new **React** app with the name **redux-store-demo**

1.2 Move to the newly created directory by running the **cd redux-store-demo** command in the terminal

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

shreemayeebhatt@ip-172-31-22-250:~$ cd redux-store-demo/
```

1.3 Open **VS Code** and navigate to the **redux-store-demo** project directory

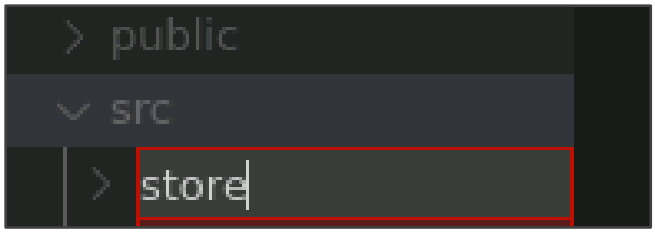
## Step 2: Install Redux and React Redux

2.1 In the terminal, make sure you are in the **redux-store-demo** directory

2.2 Run the command **npm install redux react-redux** to install the necessary dependencies

```
shreemayeebhatt@ip-172-31-22-250:~/redux-store-demo$ npm install redux react-redux
```

2.3 create a new folder named as **store** In the **src** folder




## Step 3: Create a new file called index.js

3.1 Create a new file named as **index.js**

3.2 Import the **createStore** function from **Redux**

3.3 Define the **initial state** for the **counter**

3.4 Create the **reducer** function that handles the **state** updates based on the **dispatched** actions

3.5 Use the **createStore** function to create the Redux store, passing the reducer function as an argument

3.6 Export **default** store

Then, create your **Redux store** in it. In this example, we will create a simple **counter** that can be incremented or decremented

```
import { createStore } from 'redux';
const initialState = {
count: 0
};

function reducer(state = initialState, action) {
switch (action.type) {
case 'INCREMENT':
return { count: state.count + 1 };
case 'DECREMENT':
return { count: state.count - 1 };
default:
return state;
}
}
const store = createStore(reducer);
export default store;
```

```
import { ~~createStore~~ } from 'redux';

const initialState = {
count: 0
};

function reducer(state = initialState, action) {
switch (action.type) {
case 'INCREMENT':
return { count: state.count + 1 };
case 'DECREMENT':
return { count: state.count - 1 };
default:
return state;
}
}

const store = ~~createStore~~(reducer);

export default store;
```

### Step 4: Import Provider from react-redux in App.js

4.1 Open the **App.js** file In the **src** folder

4.2 Import **React** and the necessary components from **React Redux: connect** and **Provider**

```
1   import React from 'react';
2   import { Provider } from 'react-redux';
3   import { connect } from 'react-redux';
```

4.3 Define the **App** component that receives the **count**, **increment**, and **decrement** props

```
const App = ({ count, increment, decrement }) => {
```

4.4 Render the **JSX markup** for the app's UI, displaying the count and buttons

4.5 Attach the **increment** and **decrement** functions to the respective button's **onClick** events

```
return (
  <div className="App">
    <h1>Redux Store Demo</h1>
    <p>Count: {count}</p>
    <button onClick={increment}>+</button>
    <button onClick={decrement}>-</button>
  </div>
);
```

4.6 Connect the **App** component to **Redux**

4.7 Wrap the connected component with the **Redux Provider**

4.8 Export the root component

```
//App.js
import React from 'react';
import { Provider } from 'react-redux';
import { connect } from 'react-redux';
import { createStore } from 'redux';

const initialState = {
count: 0
};

function reducer(state = initialState, action) {
switch (action.type) {
case 'INCREMENT':
return { count: state.count + 1 };
```

```
case 'DECREMENT':
return { count: state.count - 1 };

default:
return state;
}
}
const store = createStore(reducer);
const increment = () => {
return { type: 'INCREMENT' };
};
const decrement = () => {
return { type: 'DECREMENT' };
};

const App = ({ count, increment, decrement }) => {
return (
<div className="App">
<h1>Redux Store Demo</h1>
<p>Count: {count}</p>
<button onClick={increment}>+</button>
<button onClick={decrement}>-</button>
</div>

);
};

const mapStateToProps = (state) => {
return {
count: state.count,
};
};

const mapDispatchToProps = {
increment,
decrement
};
```

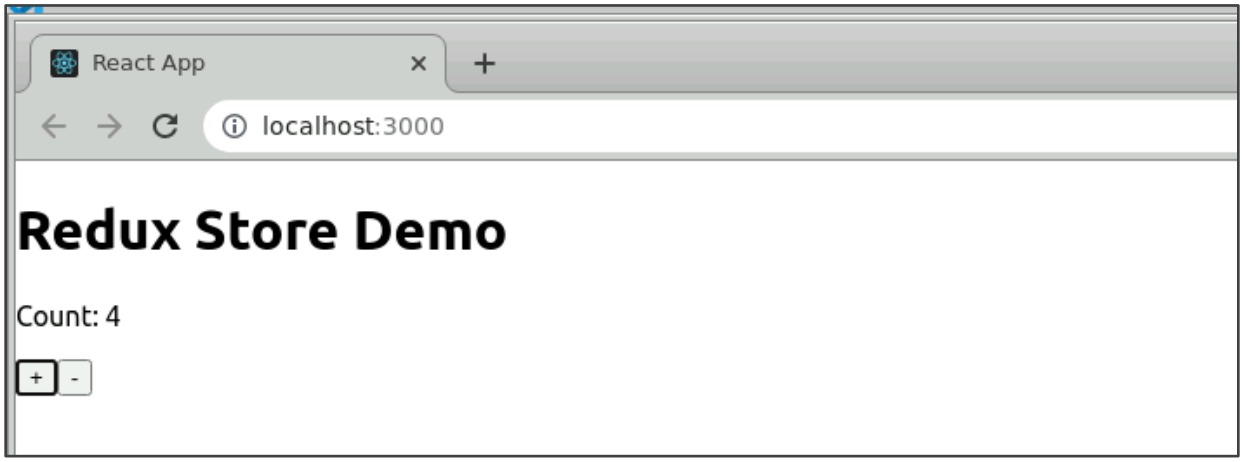
```
const ConnectedApp = connect(mapStateToProps, mapDispatchToProps)(App);
const RootApp = () => (

<Provider store={store}>
<ConnectedApp />
</Provider>
);

export default RootApp;
```

## Step 5: Run the app and view it in the browser

5.1 In the terminal, navigate to the project directory

5.2 Run the **npm start** command to start the app

5.3 Open your browser and navigate to **http://localhost:3000**




The app should be running, and you should see a simple app with a counter that can be incremented or decremented by clicking the buttons.

With this, you have successfully developed a React application demonstrating the integration of Redux Store, providing efficient state management through a dynamic counter and demonstrating seamless interaction with React components.