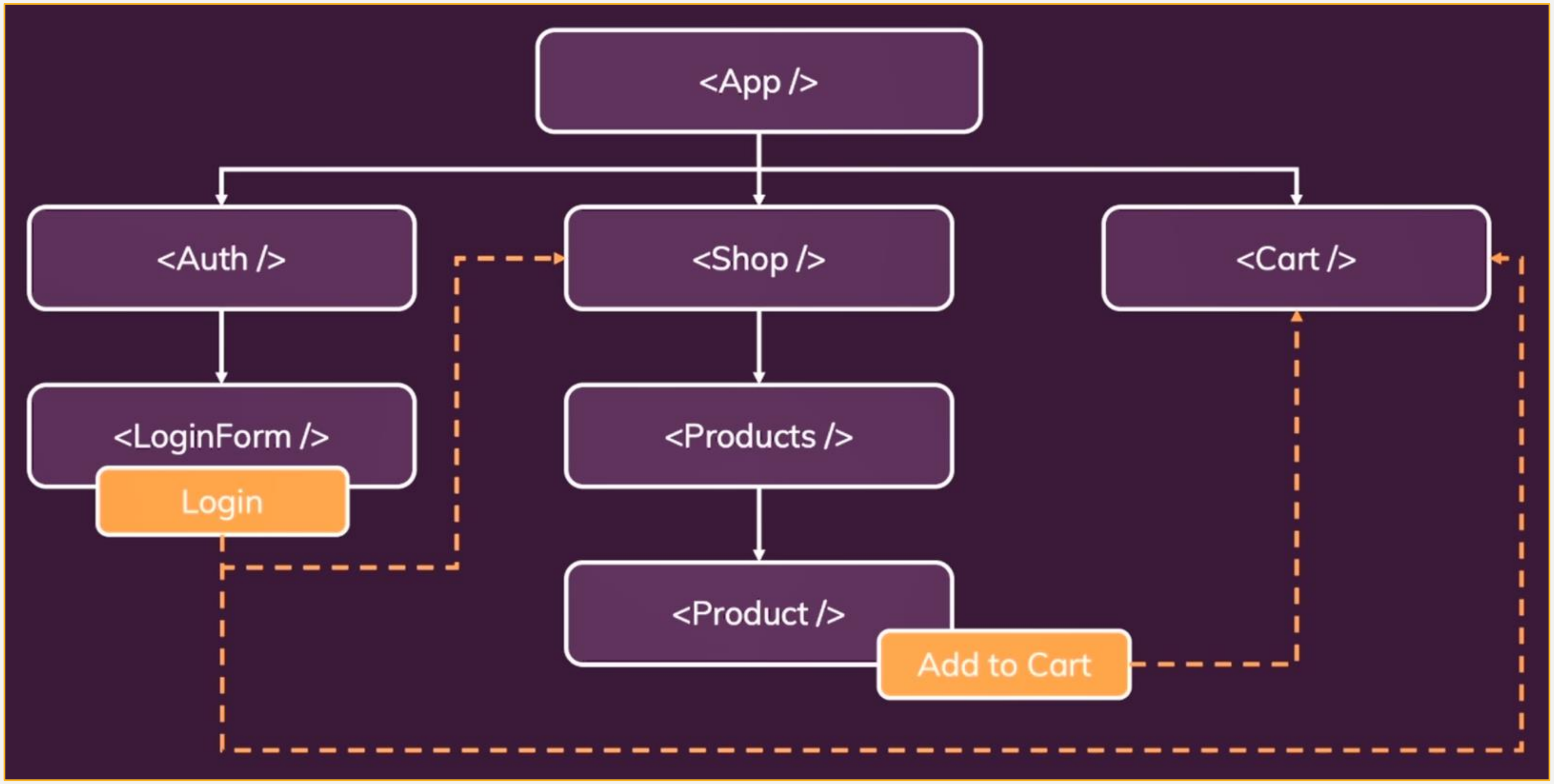
React

Redux

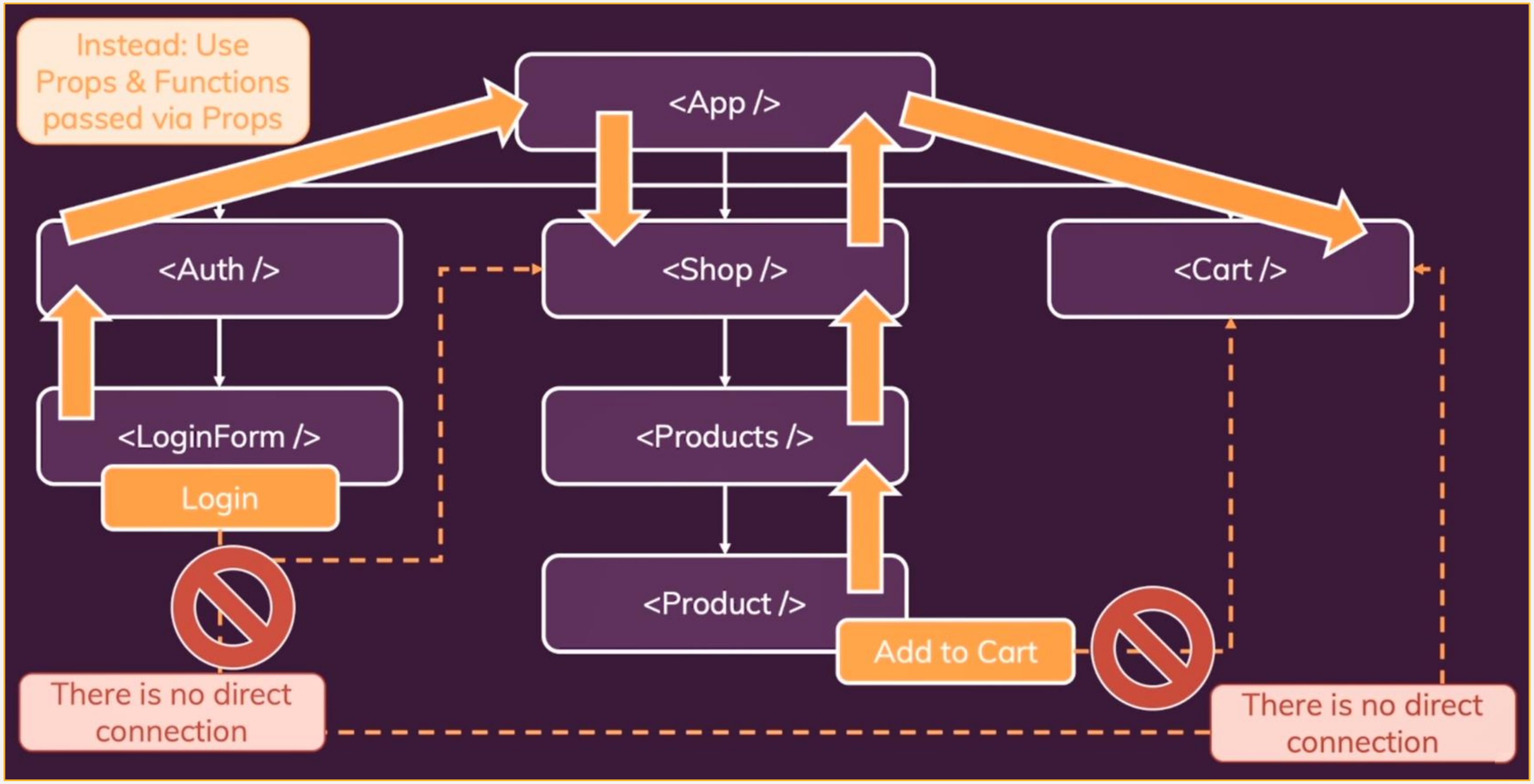
What is Redux?

- **Redux** is an open-source JavaScript library for managing and centralizing application state.
- It is most commonly used with libraries such as React or Angular for building user interfaces. Similar to (and inspired by) Facebook's Flux architecture
- It was created by Dan Abramov and Andrew Clark.
- Redux is a small library with a simple, limited API designed to be a predictable container for application state.
- Redux can be used with any UI layer, it was originally designed and intended for use with React
- React Redux is the official React UI bindings layer for Redux.

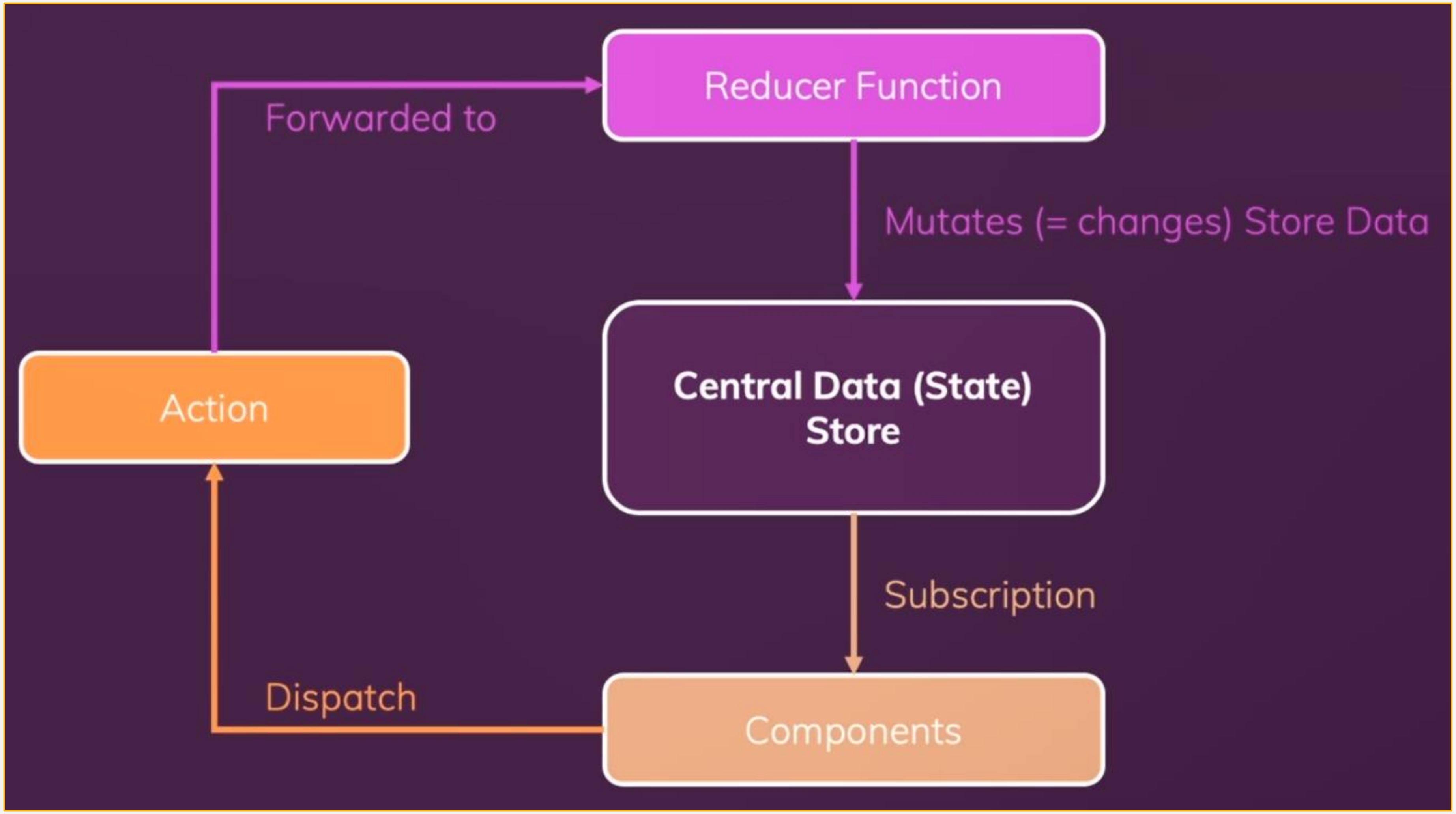
Example with Redux



Example without Redux


Instead: Use Props & Functions passed via Props
<App />
<Auth />
<Shop />
<Cart />
<LoginForm />
Login
<Products />
<Product />
Add to Cart
There is no direct connection
There is no direct connection

Core Redux Concepts



Installing packages & Preparation

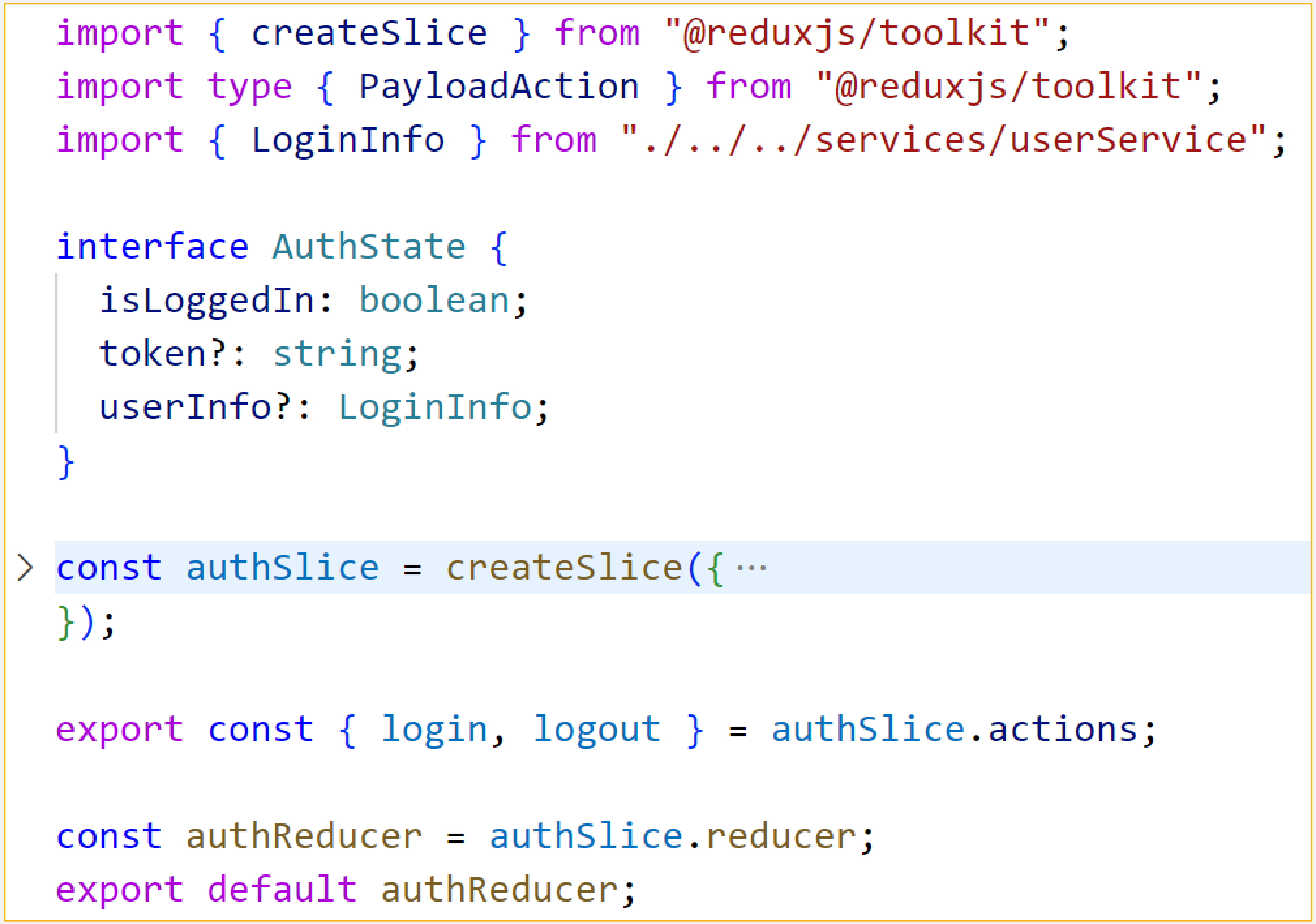
- In terminal, type:

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

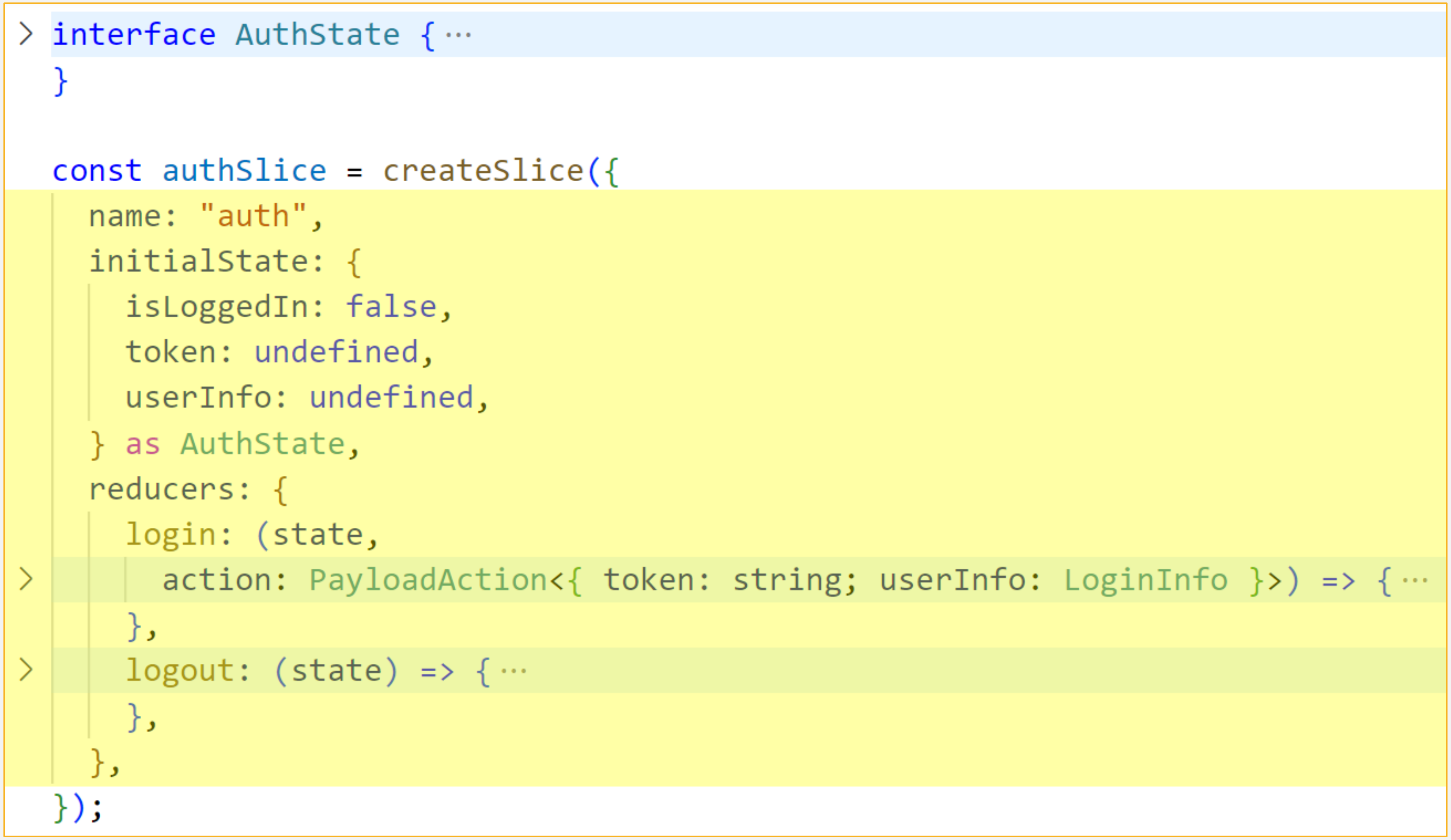
- In “src” folder, create a new folder named “store” to manage the related redux files.
- In “store” folder, create a new folder named “reducers” to retrieve specific actions and update states.

Creating Redux Reducers & Actions

- In folder “reducers”, create a new file named “auth.ts”

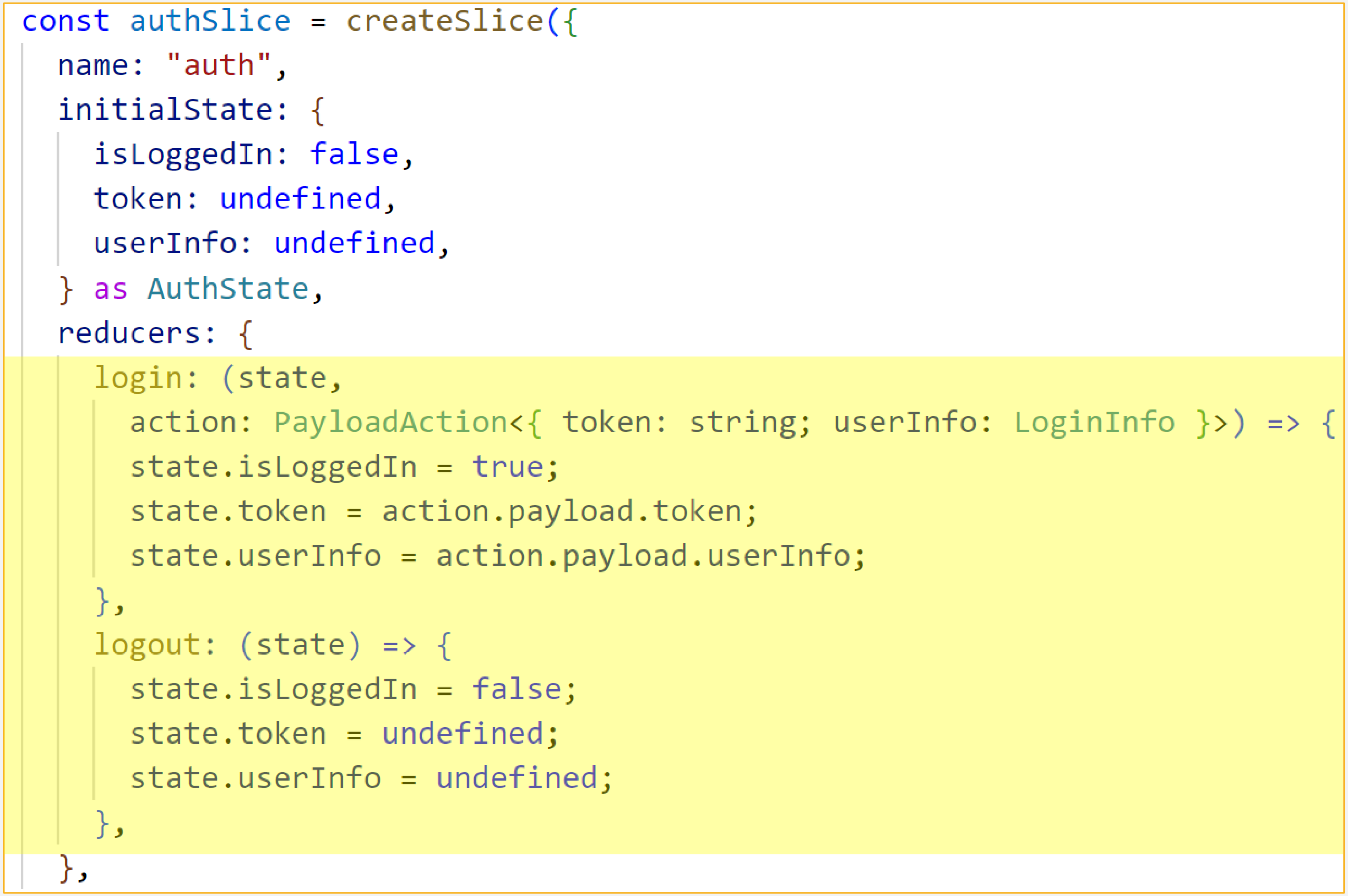


```
import { createSlice } from "@reduxjs/toolkit";
import type { PayloadAction } from "@reduxjs/toolkit";
import { LoginInfo } from "./../../services/userService";

interface AuthState {
  isLoggedIn: boolean;
  token?: string;
  userInfo?: LoginInfo;
}

const authSlice = createSlice({ ···
});

export const { login, logout } = authSlice.actions;

const authReducer = authSlice.reducer;
export default authReducer;
```



```
> interface AuthState {⋯
  }

  const authSlice = createSlice({
    name: "auth",
    initialState: {
      isLoggedIn: false,
      token: undefined,
      userInfo: undefined,
    } as AuthState,
    reducers: {
      login: (state,
>       action: PayloadAction<{ token: string; userInfo: LoginInfo }>) => {⋯
      },
>     logout: (state) => {⋯
      },
    },
  });
```

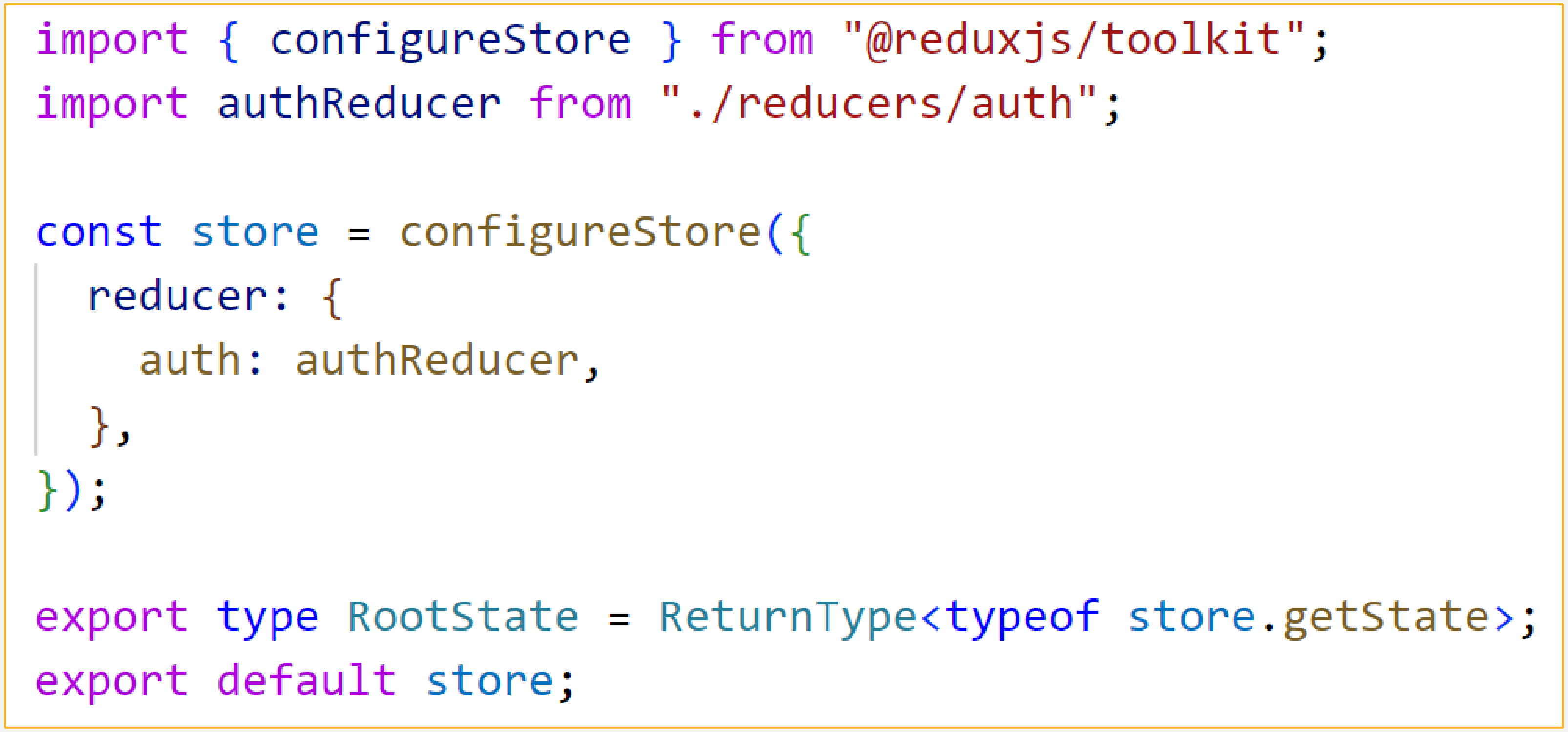
Reducer Action Implementation



```
const authSlice = createSlice({
  name: "auth",
  initialState: {
    isLoggedIn: false,
    token: undefined,
    userInfo: undefined,
  } as AuthState,
  reducers: {
    login: (state,
      action: PayloadAction<{ token: string; userInfo: LoginInfo }>) => {
      state.isLoggedIn = true;
      state.token = action.payload.token;
      state.userInfo = action.payload.userInfo;
    },
    logout: (state) => {
      state.isLoggedIn = false;
      state.token = undefined;
      state.userInfo = undefined;
    },
  },
```

Creating Store

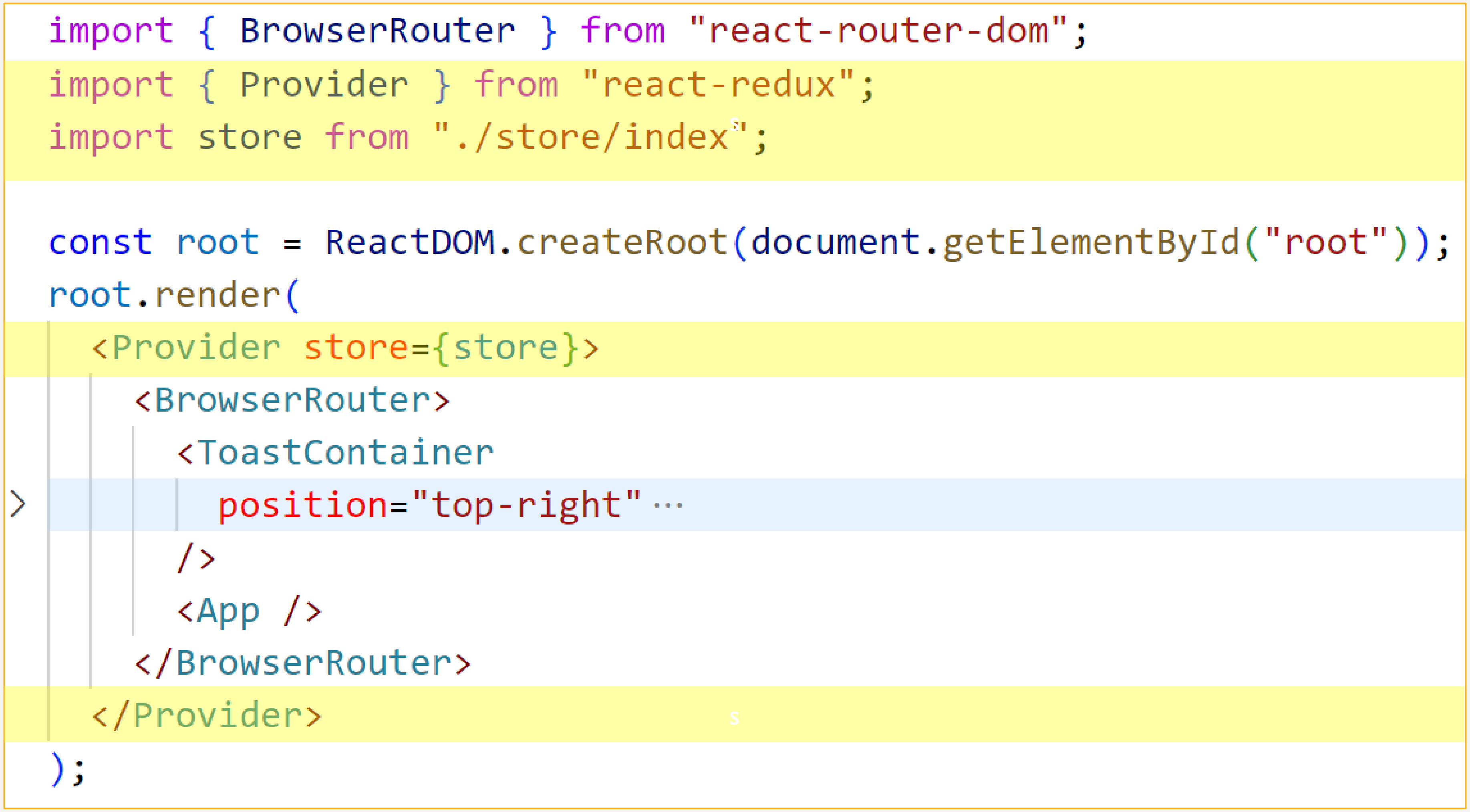
- In “store” folder, create a new file named “index.ts”



```ts
import { configureStore } from "@reduxjs/toolkit";
import authReducer from "./reducers/auth";

const store = configureStore({
  reducer: {
    auth: authReducer,
  },
});

export type RootState = ReturnType<typeof store.getState>;
export default store;
```

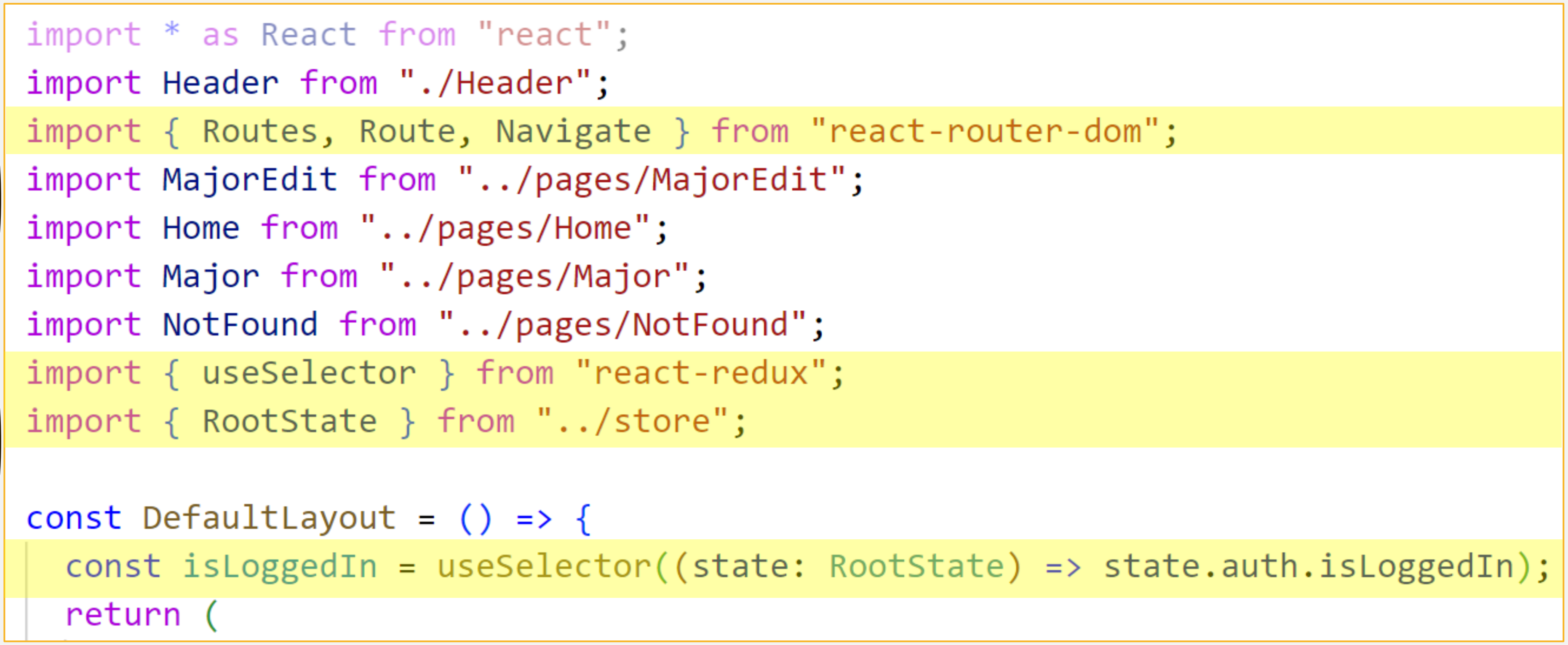
Provide the Redux Store to React

- In the “src/index.tsx” file adding Provider

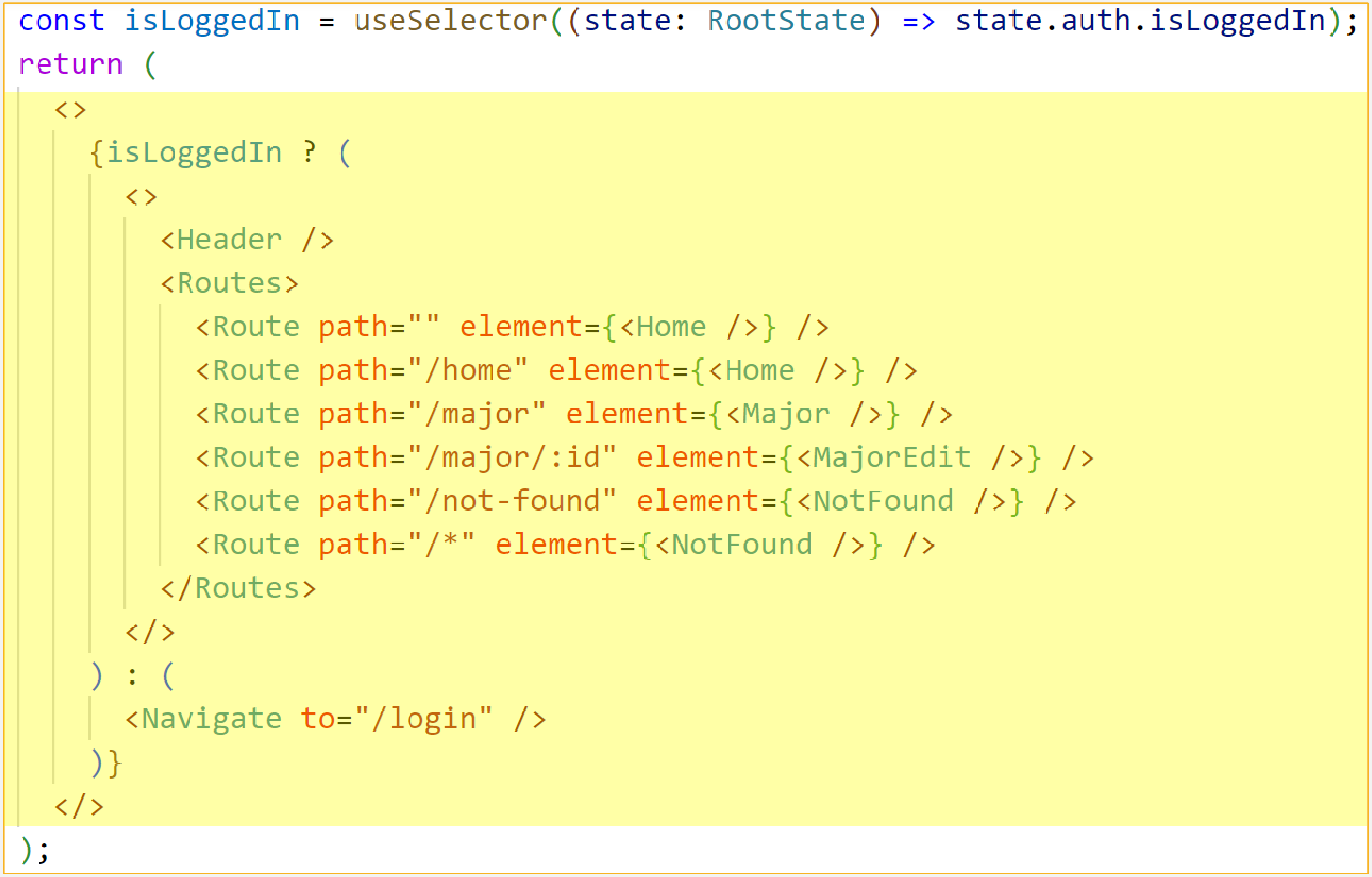


```
import { BrowserRouter } from "react-router-dom";
import { Provider } from "react-redux";
import store from "./store/index";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <Provider store={store}>
    <BrowserRouter>
      <ToastContainer
        position="top-right" ···
      />
      <App />
    </BrowserRouter>
  </Provider>
);
```

Subscribing Value in DefaultLayout



```
import * as React from "react";
import Header from "./Header";
import { Routes, Route, Navigate } from "react-router-dom";
import MajorEdit from "../pages/MajorEdit";
import Home from "../pages/Home";
import Major from "../pages/Major";
import NotFound from "../pages/NotFound";
import { useSelector } from "react-redux";
import { RootState } from "../store";

const DefaultLayout = () => {
  const isLoggedIn = useSelector((state: RootState) => state.auth.isLoggedIn);
  return (
```



```
const isLoggedIn = useSelector((state: RootState) => state.auth.isLoggedIn);
return (
  <>
    {isLoggedIn ? (
      <>
        <Header />
        <Routes>
          <Route path="" element={<Home />} />
          <Route path="/home" element={<Home />} />
          <Route path="/major" element={<Major />} />
          <Route path="/major/:id" element={<MajorEdit />} />
          <Route path="/not-found" element={<NotFound />} />
          <Route path="/*" element={<NotFound />} />
        </Routes>
      </>
    ) : (
      <Navigate to="/login" />
    )}
  </>
);
```

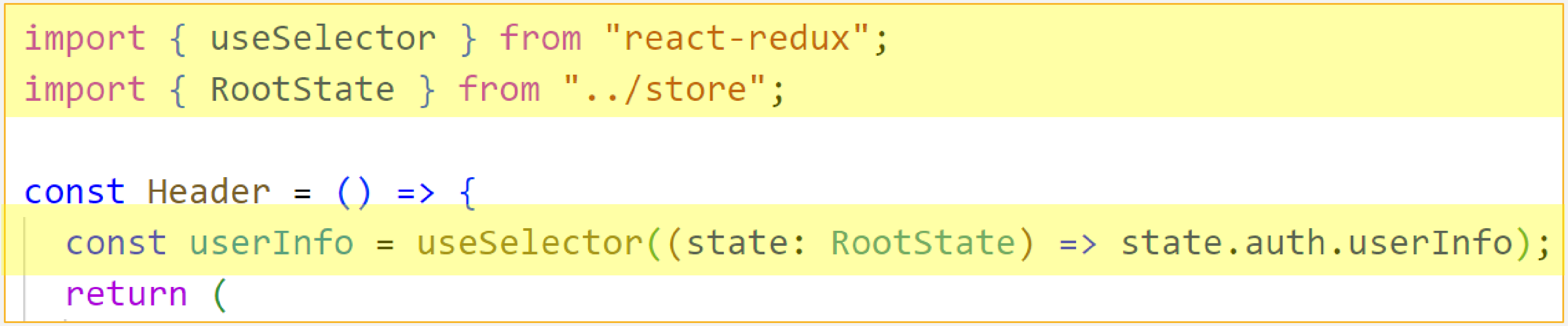
Dispatching in Login

```
import { useDispatch } from "react-redux";
import { login } from "../store/reducers/auth";
```

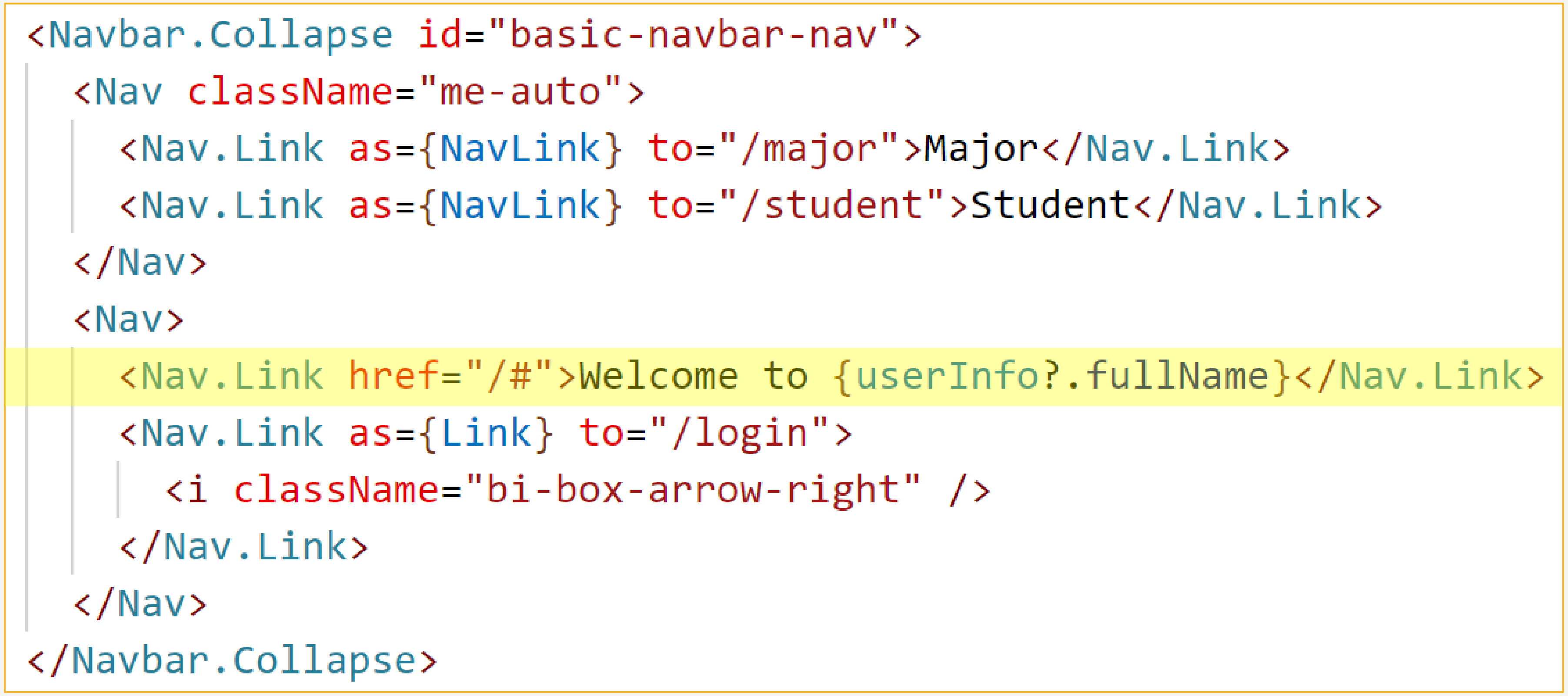
```
const Login = () => {
  const dispatch = useDispatch();
```

```
userService.login(username, password).then((res) => {
  if (res.errorCode === 0) {
    setMessage("");
    dispatch(login({
      token: res.data.accessToken,
      userInfo: res.data
    }));
    navigate("/home");
  } else {
    setMessage(res.message);
  }
});
```

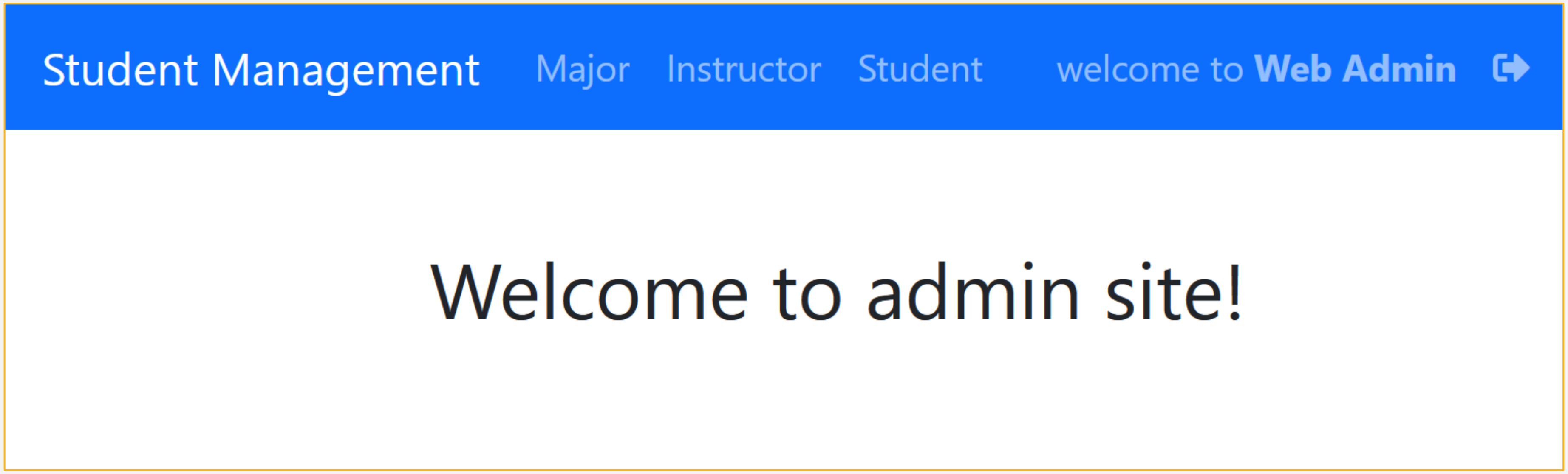
Subscribing Value in Header



```
import { useSelector } from "react-redux";
import { RootState } from "../store";

const Header = () => {
  const userInfo = useSelector((state: RootState) => state.auth.userInfo);
  return (
```



```
<Navbar.Collapse id="basic-navbar-nav">
  <Nav className="me-auto">
    <Nav.Link as={NavLink} to="/major">Major</Nav.Link>
    <Nav.Link as={NavLink} to="/student">Student</Nav.Link>
  </Nav>
  <Nav>
    <Nav.Link href="/#">Welcome to {userInfo?.fullName}</Nav.Link>
    <Nav.Link as={Link} to="/login">
      <i className="bi-box-arrow-right" />
    </Nav.Link>
  </Nav>
</Navbar.Collapse>
```



Welcome to admin site!

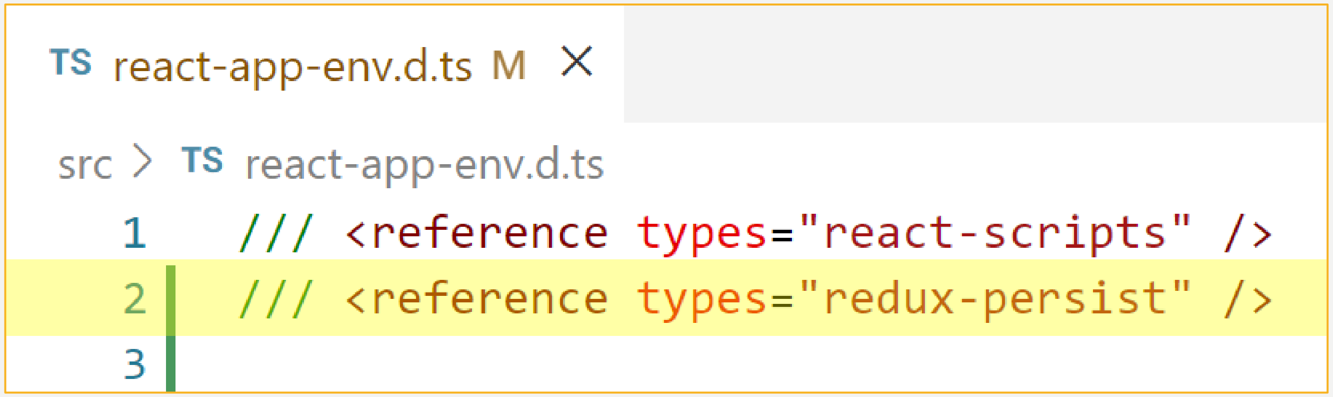
Dispatching in Header

```
import { useSelector, useDispatch } from "react-redux";
import { RootState } from "../store";
import { logout } from "../store/reducers/auth";

const Header = () => {
  const dispatch = useDispatch();
  const userInfo = useSelector((state: RootState) => state.auth.userInfo);
  return (
```

```
<Nav>
  <Nav.Link href="/#">Welcome to {userInfo?.fullName}</Nav.Link>
  <Nav.Link onClick={() => dispatch(logout())}>
    <i className="bi-box-arrow-right" />
  </Nav.Link>
</Nav>
```

Declare redux-persist package along compilation

- In “react-app-env.d.ts” file, add a new row for redux-persist package



```
/// <reference types="react-scripts" />
/// <reference types="redux-persist" />

```

Persisting State with Redux Toolkit

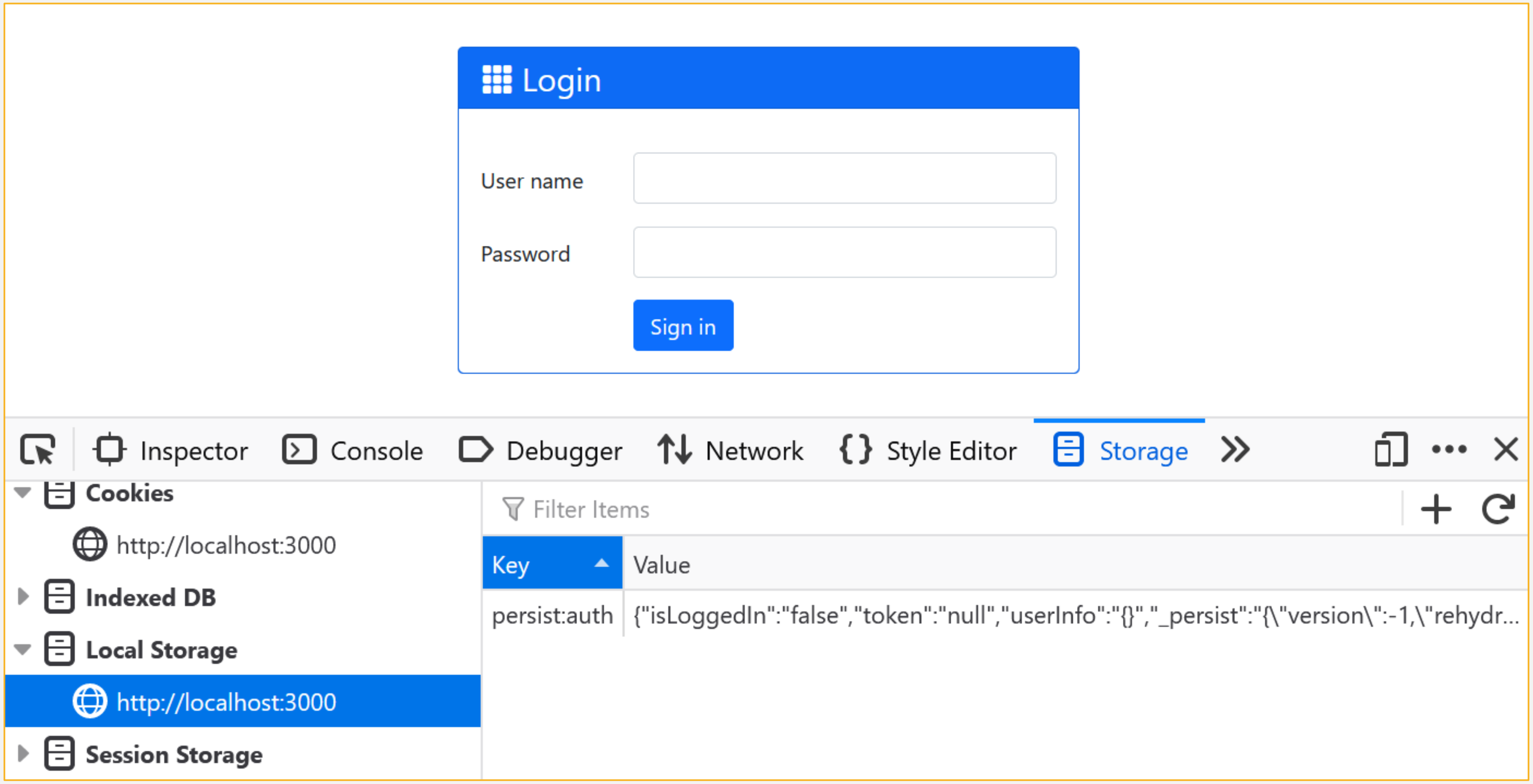
- Modify “index.ts” file in “store” folder

```
import { configureStore, combineReducers } from "@reduxjs/toolkit";
import authReducer from "./reducers/auth";
import storage from "redux-persist/lib/storage";
import { persistReducer, persistStore } from "redux-persist";
import thunk from "redux-thunk";

const autPersistConfig = { key: "auth", storage };
const rootReducer = combineReducers({
  auth: persistReducer(autPersistConfig, authReducer),
});
const store = configureStore({
  reducer: rootReducer,
  middleware: [thunk],
});

export type RootState = ReturnType<typeof store.getState>;
export default store;
export const persistor = persistStore(store);
```

Configure Redux Store in index.tsx

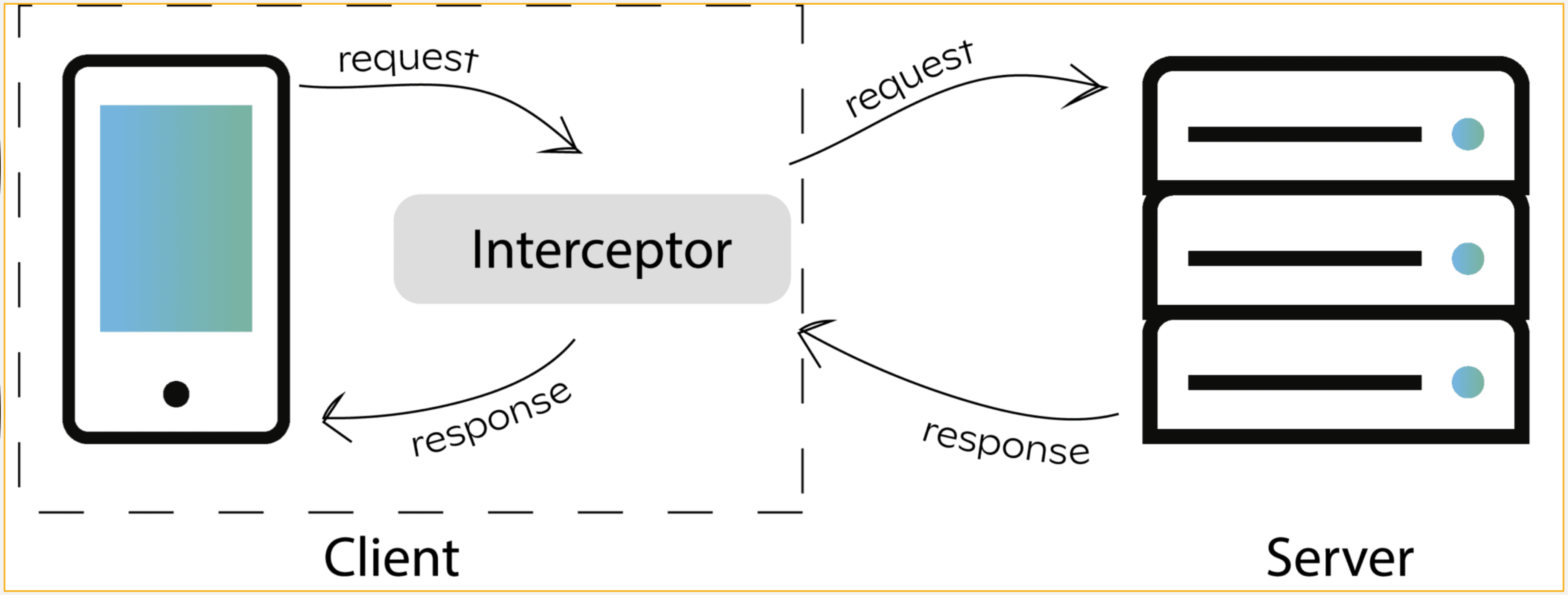
```
import { BrowserRouter } from "react-router-dom";
import { ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import { Provider } from "react-redux";
import store, { persistor } from "./store/index";
import { PersistGate } from "redux-persist/integration/react";

const root =
  ReactDOM.createRoot(document.getElementById("root") as HTMLElement);
root.render(
  <Provider store={store}>
    <PersistGate loading={null} persistor={persistor}>
      <BrowserRouter>
        <ToastContainer theme="colored" />
        <App />
      </BrowserRouter>
    </PersistGate>
  </Provider>
);
```

Local Storage Data

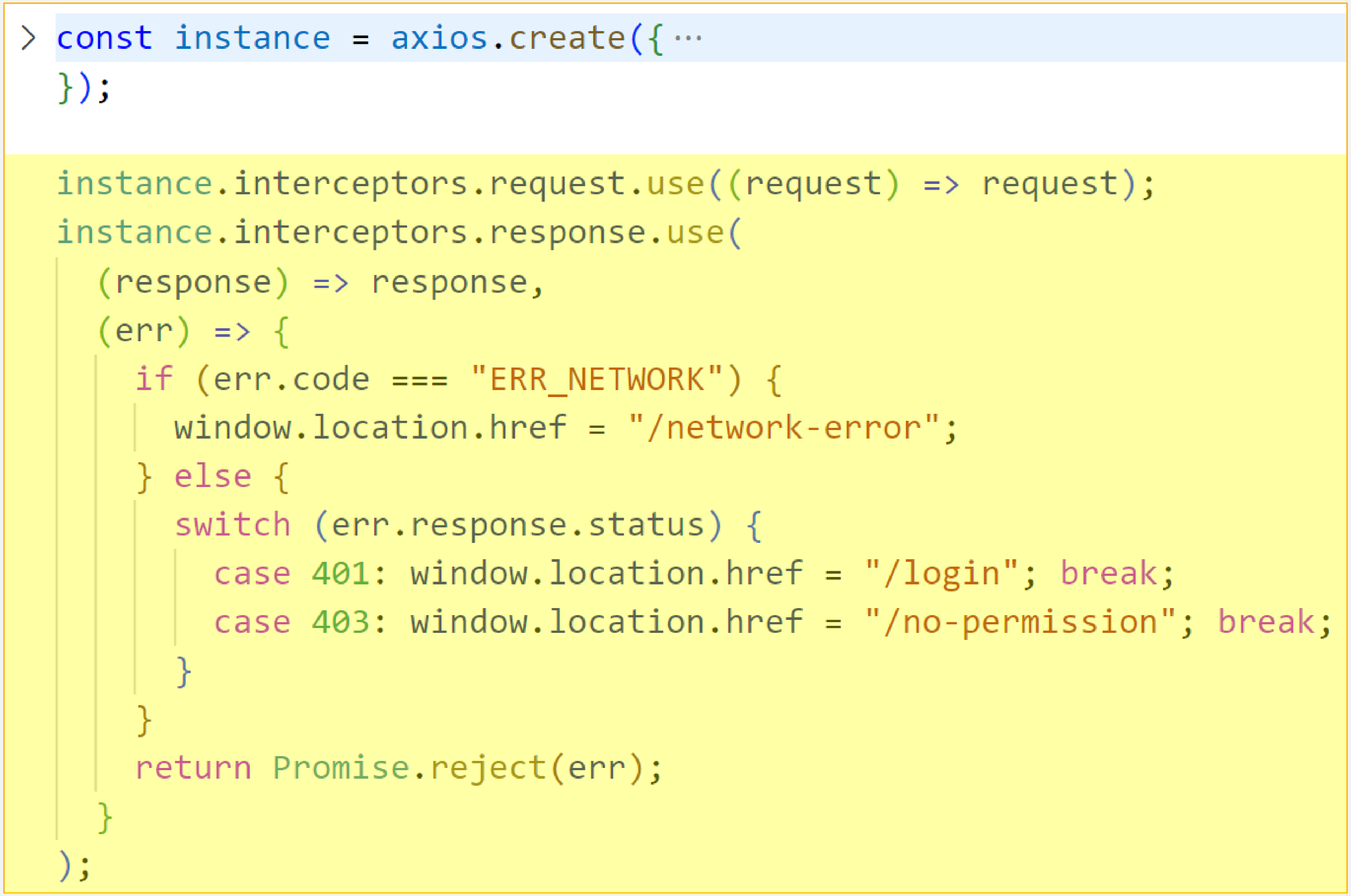


Key	Value
persist:auth	{"isLoggedIn":"false","token":"null","userInfo":"{}","_persist":"{\"version\":-1,\"rehydr...

Interceptors

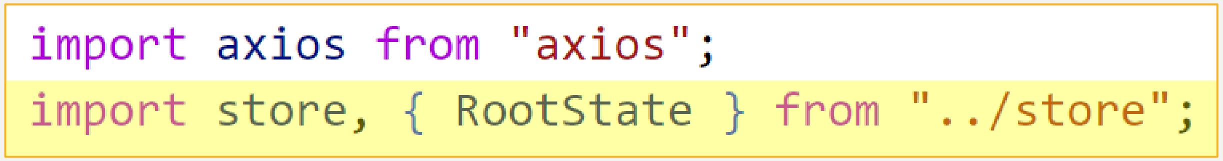


Setting Up Interceptors

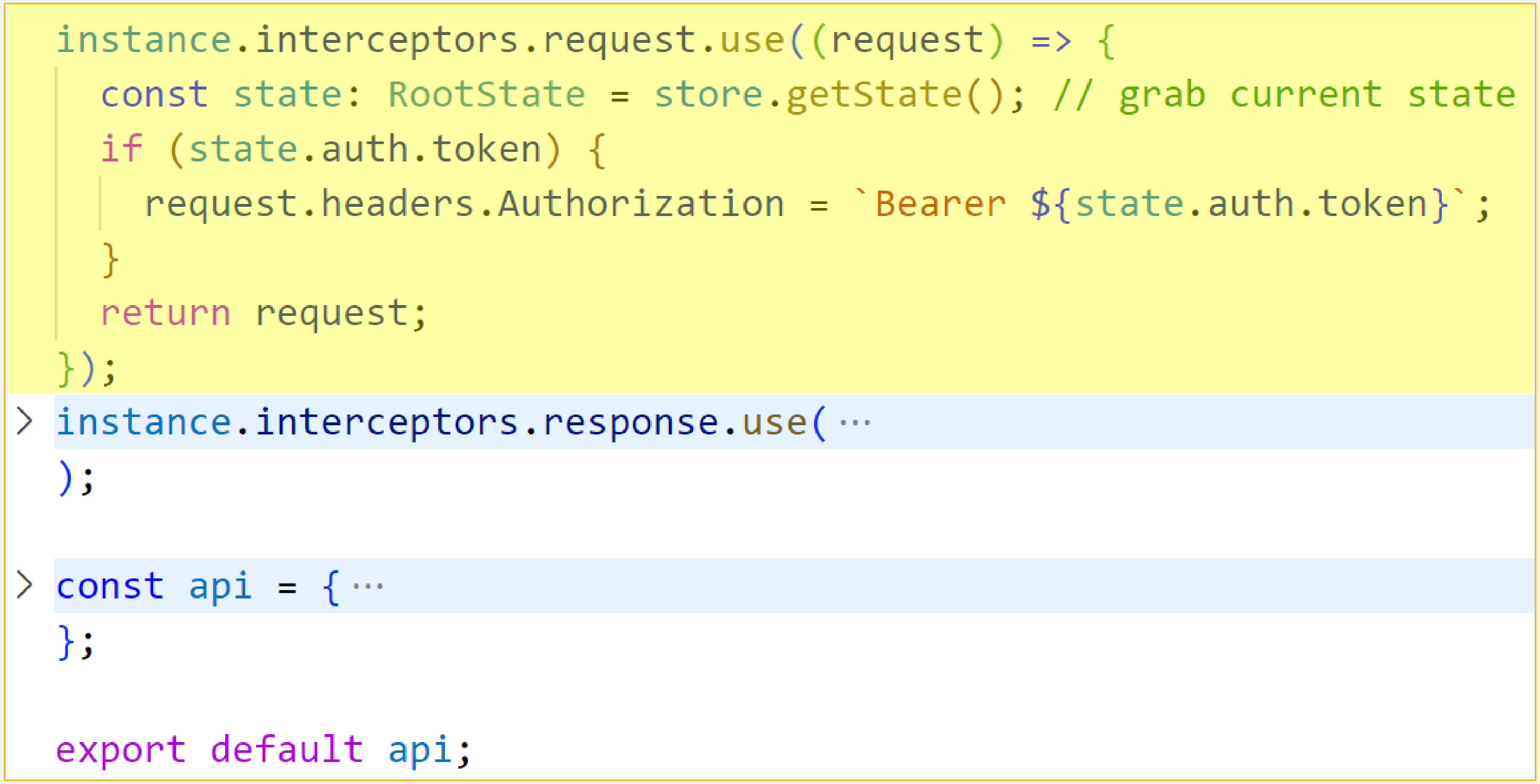


```
> const instance = axios.create({…
  });

  instance.interceptors.request.use((request) => request);
  instance.interceptors.response.use(
    (response) => response,
    (err) => {
      if (err.code === "ERR_NETWORK") {
        window.location.href = "/network-error";
      } else {
        switch (err.response.status) {
          case 401: window.location.href = "/login"; break;
          case 403: window.location.href = "/no-permission"; break;
        }
      }
      return Promise.reject(err);
    }
  );
```

Accessing Redux out of Component



```
import axios from "axios";
import store, { RootState } from "../store";
```



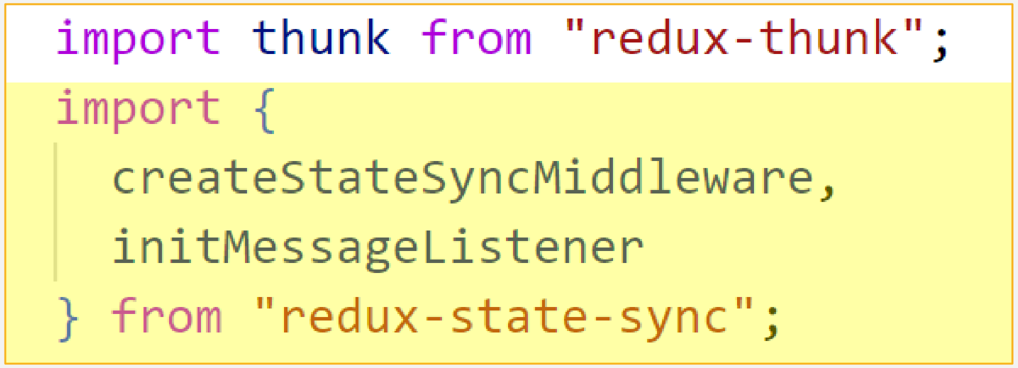
```
instance.interceptors.request.use((request) => {
  const state: RootState = store.getState(); // grab current state
  if (state.auth.token) {
    request.headers.Authorization = `Bearer ${state.auth.token}`;
  }
  return request;
});
> instance.interceptors.response.use( ···
);

> const api = { ···
};

export default api;
```

Syncing Redux State across Tabs

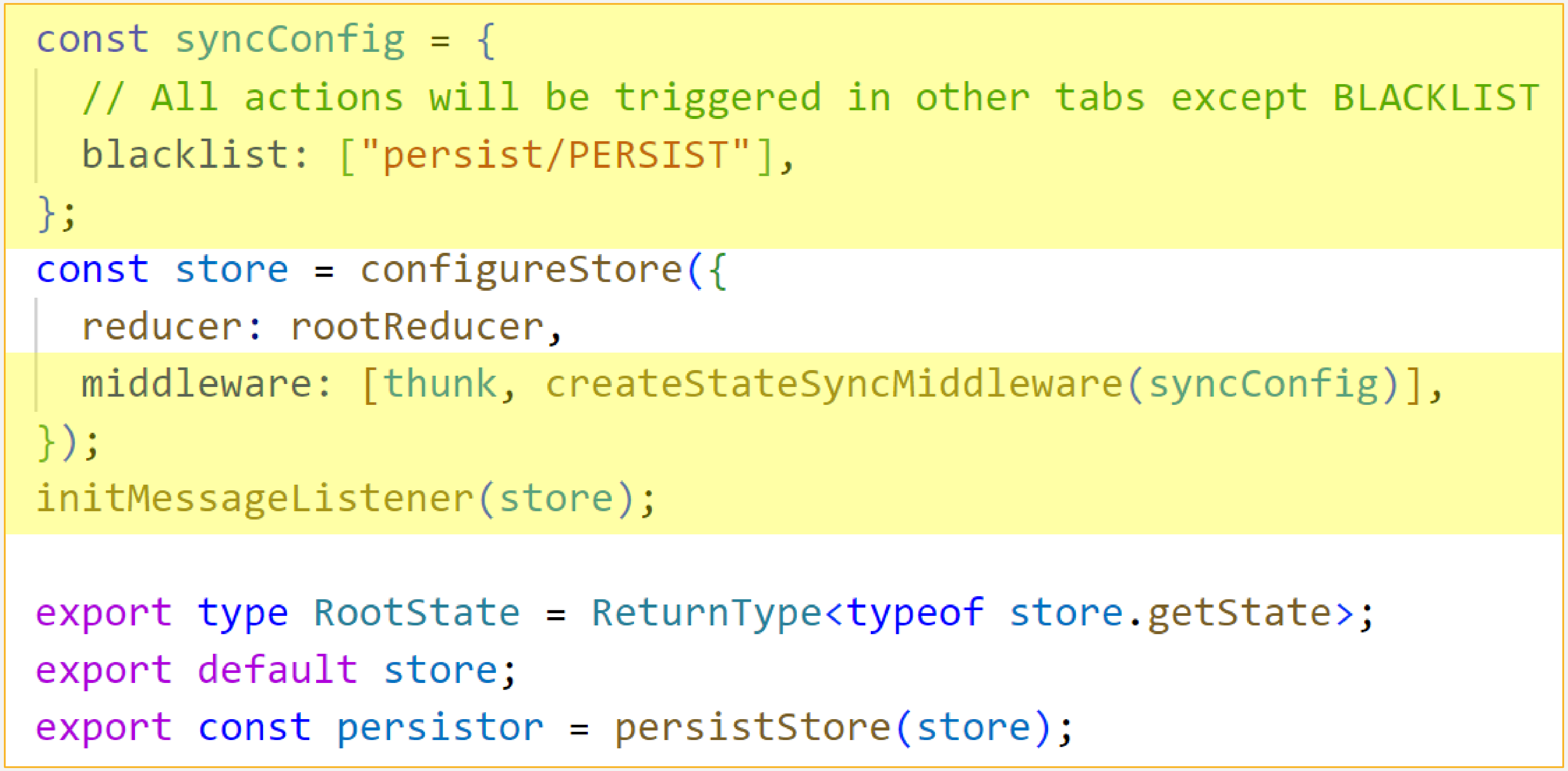
- redux-state-sync package is A lightweight middleware to sync your redux state across browser tabs. It will listen to the Broadcast Channel and dispatch exactly the same actions dispatched in other tabs to keep the redux state in sync.
- Although the author uses the Broadcast Channel API that is not supported on this date by all browsers, he was concerned to provide a fallback to make sure that the communication between tabs always works.
- In terminal, type:

```
npm i redux-state-sync @types/redux-state-sync
```

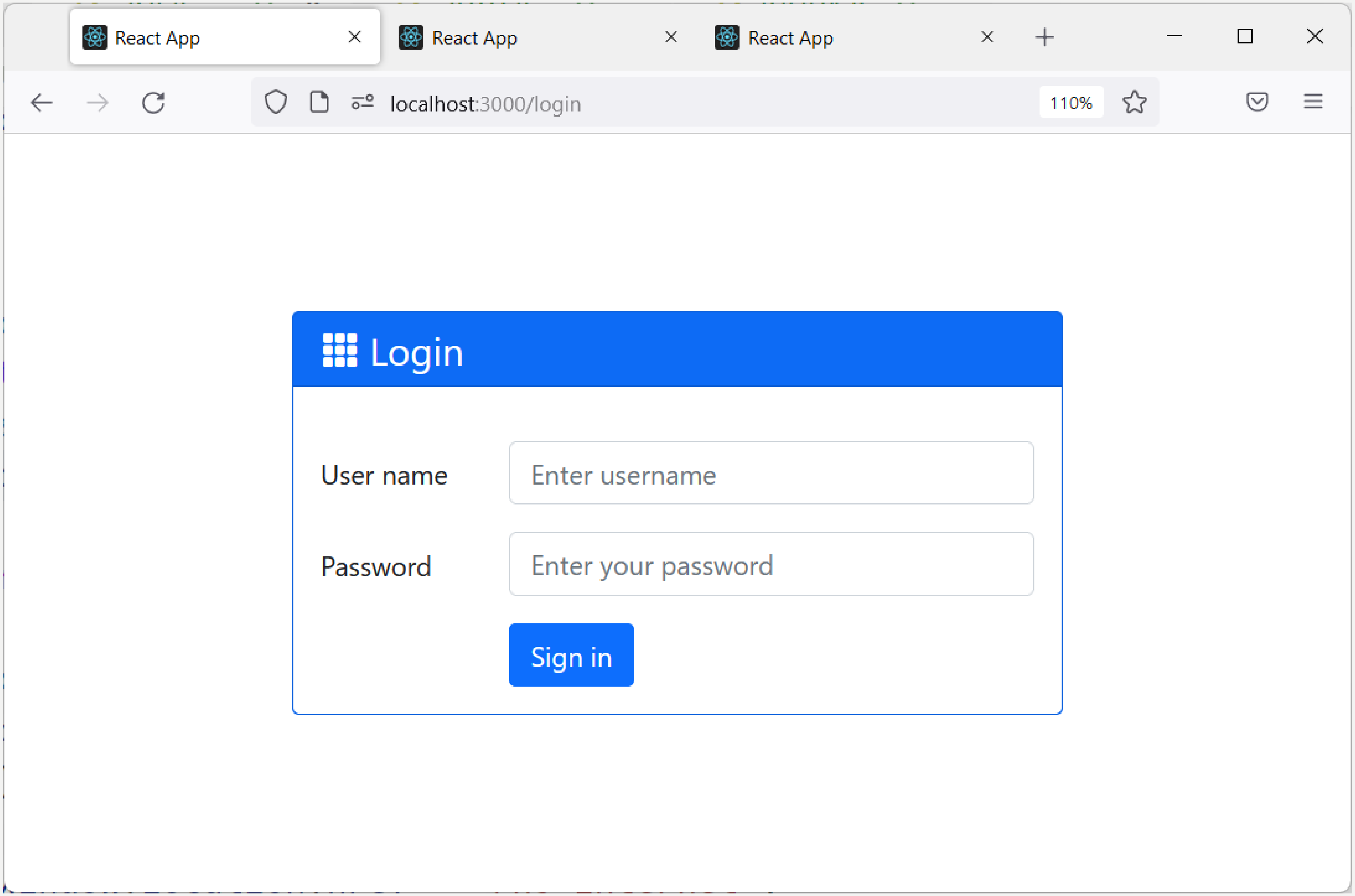
Modifying store/index.ts



```
import thunk from "redux-thunk";
import {
  createStateSyncMiddleware,
  initMessageListener
} from "redux-state-sync";
```

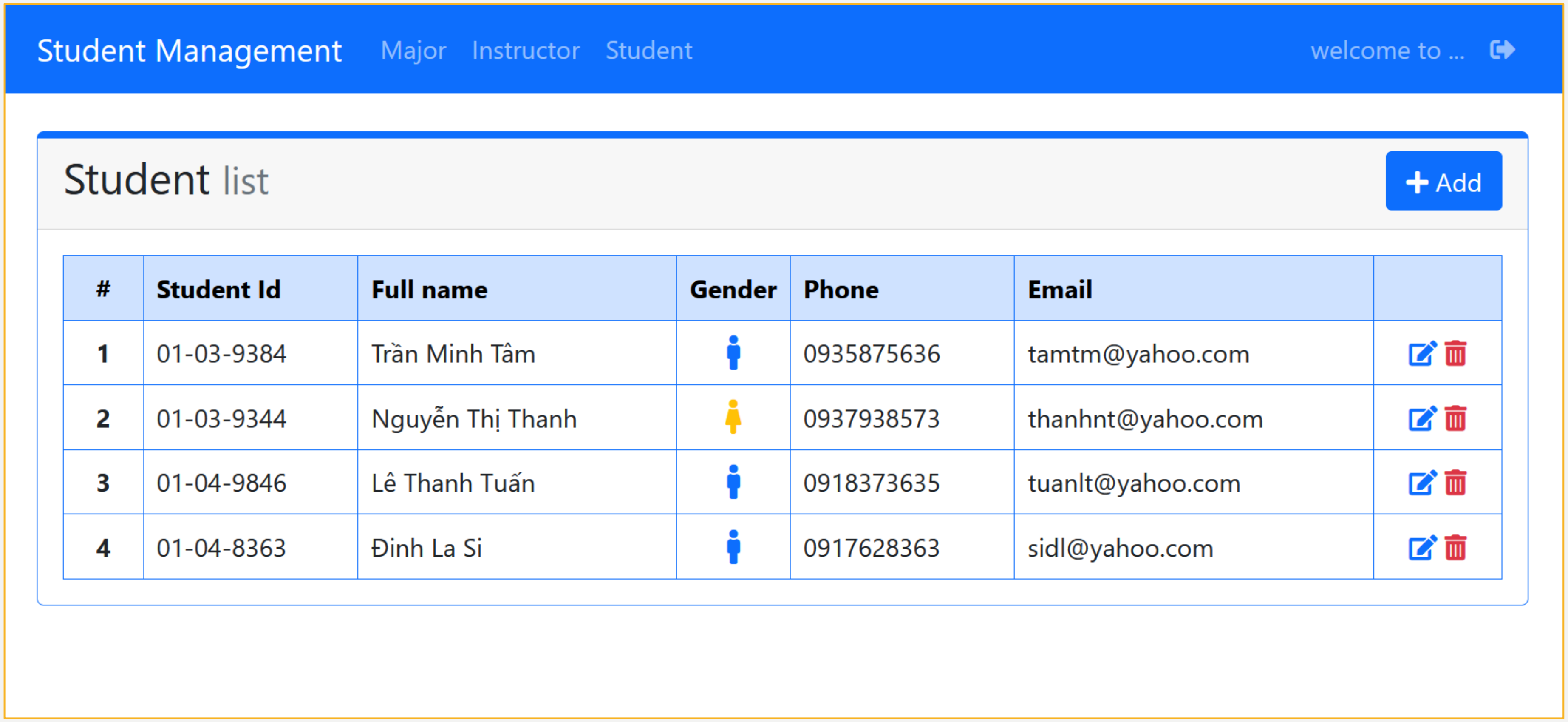


```
const syncConfig = {
  // All actions will be triggered in other tabs except BLACKLIST
  blacklist: ["persist/PERSIST"],
};
const store = configureStore({
  reducer: rootReducer,
  middleware: [thunk, createStateSyncMiddleware(syncConfig)],
});
initMessageListener(store);

export type RootState = ReturnType<typeof store.getState>;
export default store;
export const persistor = persistStore(store);
```


React App
React App
React App
localhost:3000/login
110%
Login
User name
Enter username
Password
Enter your password
Sign in

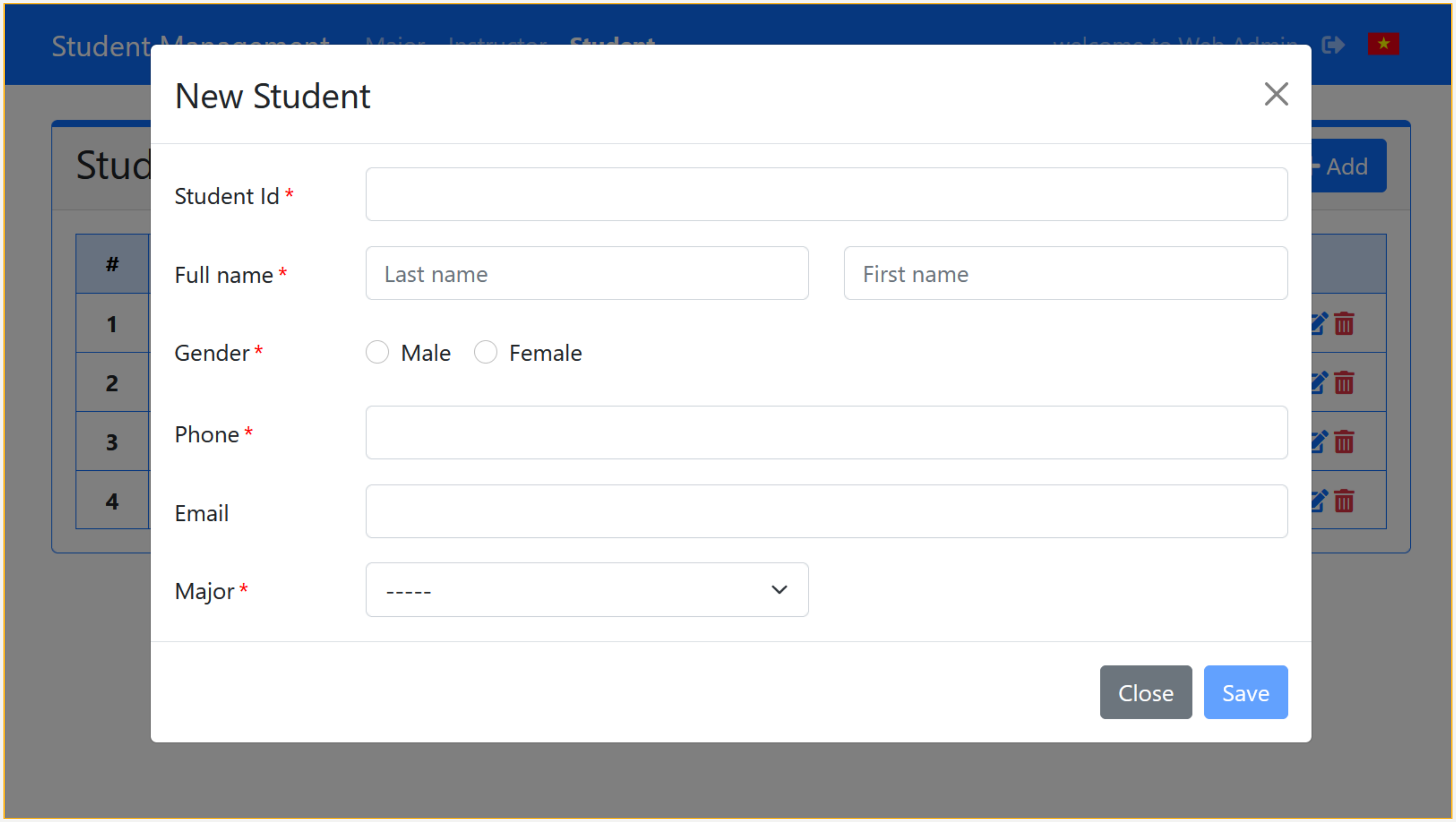
Exercise

- Implement Student Component using "/students" api.



Student list

#	Student Id	Full name	Gender	Phone	Email	
1	01-03-9384	Trần Minh Tâm	🚹	0935875636	tamtm@yahoo.com	✎ 🗑
2	01-03-9344	Nguyễn Thị Thanh	🚺	0937938573	thanhnt@yahoo.com	✎ 🗑
3	01-04-9846	Lê Thanh Tuấn	🚹	0918373635	tuanlt@yahoo.com	✎ 🗑
4	01-04-8363	Đinh La Si	🚹	0917628363	sidl@yahoo.com	✎ 🗑



New Student

Student Id *		
Full name *	Last name	First name
Gender *	○ Male ○ Female	
Phone *		
Email		
Major *	-----	

THE END