//index.js

// This file handles the reducer, INCREMENT and RESET.

import React from "react";

import ReactDOM from "react-dom";

import "./index.css";

import App from "./App";

import reportWebVitals from "./reportWebVitals";

import { createStore } from "redux";

import { Provider } from "react-redux";

const initialState = {

score: 0,

users: [],

authenticated: false,

currentUser: null,

};

function reducer(state = initialState, action) {

switch (action.type) {

case "LOGIN":

return {

// Creating a copy of state by deconstructing it, so we can access the values and update them without actually changing state - immutable data

...state,

// setting the current user to the user in the list of users that matches the credentials entered - if this doesn't work, it becomes undefined

currentUser: state.users.find(

(user) =>

user.userName === action.user.userName &&

user.password === action.user.password

),

// This checks to see if the user exists, and determines whether they're authenticated or not. array.find will return 'undefined' if the user doesn't exist in the list of users, so if it is undefined, authenticated should stay false

authenticated:

state.users.find(

(user) =>

user.userName === action.user.userName &&

user.password === action.user.password

) === undefined

? false

: true,

};

case "CREATE\_USER":

return {

...state,

users: [

...state.users,

// Creates new user in the users array with the values of the create user form, and generates new score fields - this will change with the backend

{

...action.user,

totalPoints: 0,

totalQuizCompleted: 0,

currentScore: 0,

},

],

authenticated: true,

currentUser: {

...action.user,

totalPoints: 0,

totalQuizCompleted: 0,

currentScore: 0,

},

};

case "LOGOUT":

return {

...state,

authenticated: false,

currentUser: null,

};

case "INCREMENT":

return {

...state,

score: state.score + 10, // this number is only applicable towards quizes with 10 questions. Future redesign would need to be done to accomodate different number of questions to the score.

};

case "RESET":

return {

...state,

score: 0,

};

default:

return state;

}

}

const store = createStore(reducer); // Creates the store in Redux using Redux's createStore, updated by function "reducer" in this file.

ReactDOM.render(

<React.StrictMode>

{/\* this creates the Redux within JSX to allow for storing state outside of containers via the Provider tag. \*/}

<Provider store={store}>

<App />

</Provider>

</React.StrictMode>,

document.getElementById("root")

);

// This is for future improvement to do with User Management.

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

// reportWebVitals();