

ASSIGNMENT 04

Build an Authentication System with React

Prepared By: ARM. Afrih

Student Id: TED/240063R

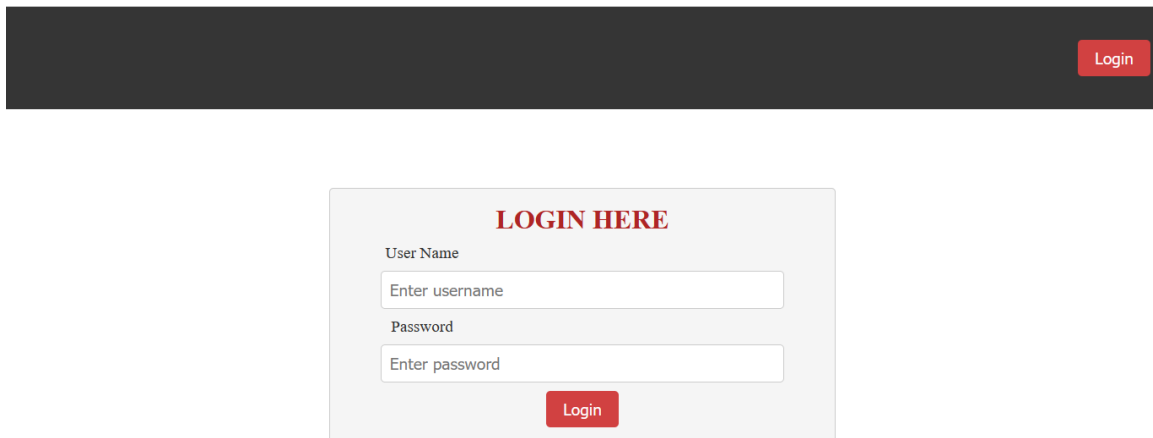
Contents

1. Login Page.....	3
2. Login Invalid Message.....	3
3. Code - Home	4
4. Authentication code.....	4
5. Protection Route code.....	5
6. App.jsx code	6

Table of Figure

FIGURE 1 LOGIN PAGE.....	3
FIGURE 2 LOGIN INVALID MESSAGE	3
FIGURE 3 HOME PAGE	4
FIGURE 4 AUTHENTICATION CODE 1	4
FIGURE 5 AUTHENTICATION CODE 2	5
FIGURE 6 PROTECTION ROUTE.....	5
FIGURE 7 APP.JSX.....	6

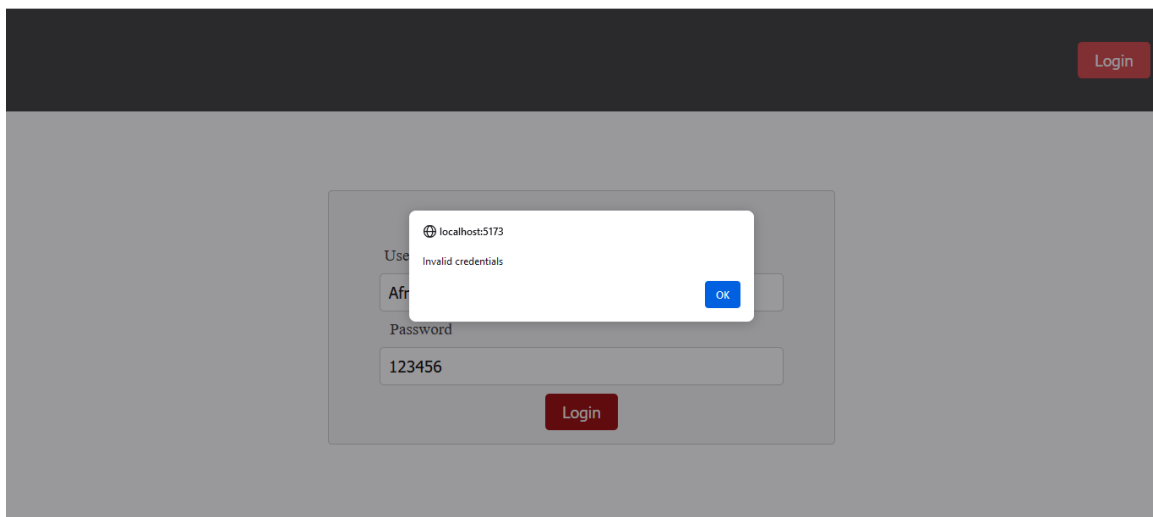
1. Login Page



The screenshot shows a login page with a dark header bar containing a red 'Login' button. Below the header is a light gray login form. The form has a title 'LOGIN HERE' in red. It contains two input fields: 'User Name' with the placeholder 'Enter username' and 'Password' with the placeholder 'Enter password'. A red 'Login' button is positioned below the password field.

Figure 1Login Page

2. Login Invalid Message



The screenshot shows the login page with a modal dialog box displayed over the login form. The modal dialog has a title bar with a close button and the text 'localhost:5173'. The main content of the dialog says 'Invalid credentials'. There is a blue 'OK' button in the bottom right corner of the dialog. The login form in the background is dimmed and shows the password field filled with '123456' and a red 'Login' button.

Figure 2 Login Invalid Message

3. Code - Home



WEL COME TO THIS PAGE.

We created using some set of use states and some react hooks

Figure 3 Home Page

4. Authentication code

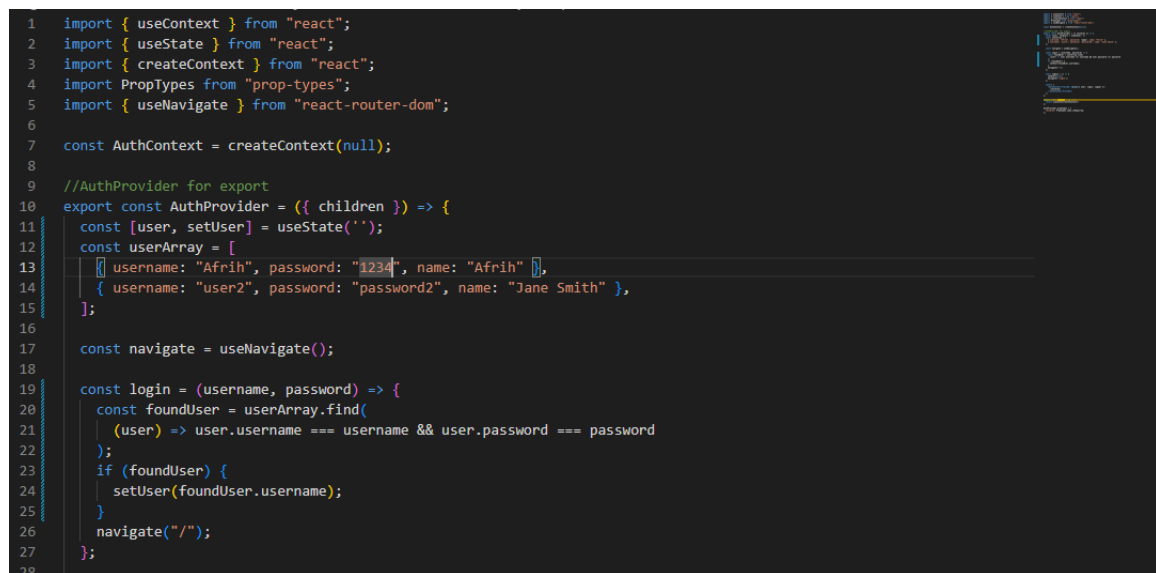


Figure 4Authentication code 1

```

export const AuthProvider = ({ children }) => {

  const login = (username, password) => {
    const foundUser = userArray.find(
      (user) => user.username === username && user.password === password
    );
    if (foundUser) {
      setUser(foundUser.username);
    }
    navigate("/");
  };

  const logout = () => {
    setUser(null);
    navigate("/login");
  };

  return (
    <AuthContext.Provider value={{ user, login, logout }}>
      {children}
    </AuthContext.Provider>
  );
};

export const useAuth = () => {
  return useContext(AuthContext);
};

AuthProvider.propTypes = {
  children: PropTypes.node.isRequired,
};

```

Figure 5 Authentication code 2

5. Protection Route code

```

import { Navigate } from "react-router-dom";
import { useAuth } from "../context/AuthContext";
import PropTypes from "prop-types";

export default function ProtectedRoutes({children}) {

  const {user} = useAuth();
  if (!user) {
    return <Navigate to={'/login'} />
  }

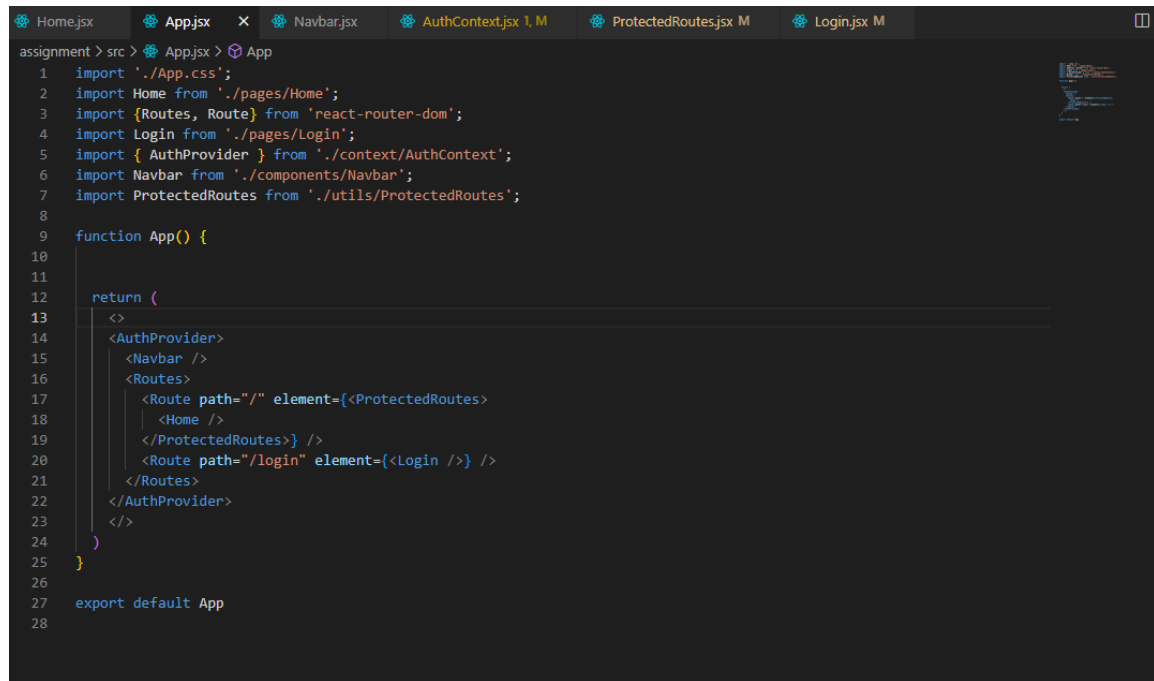
  return children;
}

ProtectedRoutes.propTypes = {
  children: PropTypes.node.isRequired
}

```

Figure 6 Protection Route

6. App.jsx code



```
assignment > src > App.jsx > App
1  import './App.css';
2  import Home from './pages/Home';
3  import {Routes, Route} from 'react-router-dom';
4  import Login from './pages/Login';
5  import {AuthProvider} from './context/AuthContext';
6  import NavBar from './components/NavBar';
7  import ProtectedRoutes from './utils/ProtectedRoutes';
8
9  function App() {
10
11
12    return (
13      <>
14        <AuthProvider>
15          <NavBar />
16          <Routes>
17            <Route path="/" element={ProtectedRoutes}>
18              <Home />
19            </ProtectedRoutes> </>
20            <Route path="/login" element={Login} />
21          </Routes>
22        </AuthProvider>
23      </>
24    )
25  }
26
27  export default App
28
```

Figure 7 App.jsx