ASP.NET Core React Login

Objective: Create a simple web application that includes a login form on the home page using ASP.NET Core with React.js in Visual Studio 2019.
Requirements:
The home page should contain:
A username textbox
A password textbox
A login button
When the user enters the credentials and presses the login button:
The application should call a server-side method.
The server should check if the username and password are both "admin".
If the credentials are correct, redirect to a new page.
If the credentials are incorrect, display an alert indicating failure.

To create a simple web application with a login form using ASP.NET Core and React.js, follow these steps:

Step 1: Set up the ASP.NET Core with React.js project

1. **Open Visual Studio 2019** and create a new project.
2. Select **ASP.NET Core Web Application**.
3. Choose **React.js** as the project template.
4. Name the project (e.g., `**LoginApp**`), and click **Create**.

This will create a project that combines ASP.NET Core with a React front-end.

Step 2: Modify the React component for the login form

1. Open the **ClientApp** folder in your project (this is where React.js components are located).
2. Open the `**src/App.js**` file.
3. Replace the content of `**App.js**` with the following code for the login form:

```
javascript


  import React, { useState } from "react";

  function App() {
    const [username, setUsername] = useState("");
    const [password, setPassword] = useState("");
    const [error, setError] = useState("");

    const handleLogin = async () => {
      const response = await fetch("/api/login", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ username, password }),
      });

      if (response.ok) {
```

```
        window.location.href = "/dashboard";  // Redirect to a new page on success
      } else {
        setError("Invalid credentials. Please try again.");  // Show error if login fails
      }
    };

    return (
      <div className="App">
        <h1>Login</h1>
        <div>
          <input
            type="text"
            placeholder="Username"
            value={username}
            onChange={(e) => setUsername(e.target.value)}
          />
        </div>
        <div>
          <input
            type="password"
            placeholder="Password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
          />
        </div>
        <button onClick={handleLogin}>Login</button>
        {error && <div style={{ color: "red" }}>{error}</div>}
      </div>
    );
  }

  export default App;
```

Step 3: Create the server-side login method in ASP.NET Core

1. Open the `Controllers` folder in the **Server** part of the project.
2. Create a new controller named `LoginController.cs`.

```
csharp


  using Microsoft.AspNetCore.Mvc;

  namespace LoginApp.Controllers
  {
      [Route("api/[controller]")]
      [ApiController]
      public class LoginController : ControllerBase
      {
          [HttpPost]
          public IActionResult Post([FromBody] LoginRequest request)
          {
              if (request.Username == "admin" && request.Password == "admin")
              {
                  return Ok();  // Return OK if credentials are correct
              }
              return Unauthorized();  // Return Unauthorized if credentials are incorrect
          }
      }

      public class LoginRequest
      {
          public string Username { get; set; }
          public string Password { get; set; }
      }
  }
```

This code defines a simple `LoginController` that accepts POST requests. If the credentials are `"admin"`, it returns a success response. Otherwise, it returns an unauthorized status.

Step 4: Create the "dashboard" page

1. In your **React.js** app, create a new `Dashboard.js` component inside `src/`:

```
javascript


  import React from "react";

  function Dashboard() {
    return (
      <div>
        <h1>Welcome to the Dashboard!</h1>
      </div>
    );
  }

  export default Dashboard;
```

2. Update `src/App.js` to include a basic route for navigating to the dashboard:

```
javascript


  import React, { useState } from "react";
  import { BrowserRouter as Router, Route, Switch, Link } from "react-router-dom";
  import Dashboard from "./Dashboard";

  function App() {
    const [username, setUsername] = useState("");
    const [password, setPassword] = useState("");
    const [error, setError] = useState("");

    const handleLogin = async () => {
      const response = await fetch("/api/login", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ username, password }),
      });

      if (response.ok) {
        window.location.href = "/dashboard";  // Redirect to the dashboard page
      } else {
        setError("Invalid credentials. Please try again.");
      }
    };

    return (
      <Router>
        <Switch>
          <Route path="/dashboard">
            <Dashboard />
          </Route>
          <Route path="/">
            <h1>Login</h1>
            <div>
              <input
                type="text"
                placeholder="Username"
                value={username}
                onChange={(e) => setUsername(e.target.value)}
              />
            </div>
            <div>
```

```
              <input
                type="password"
                placeholder="Password"
                value={password}
                onChange={(e) => setPassword(e.target.value)}
              />
            </div>
            <button onClick={handleLogin}>Login</button>
            {error && <div style={{ color: "red" }}>{error}</div>}
          </Route>
        </Switch>
      </Router>
    );
  }

  export default App;
```

Step 5: Add routing to the project

1. Install **React Router** by running the following command in the **ClientApp** folder:

```
bash


  npm install react-router-dom
```

2. Ensure `react-router-dom` is imported correctly in the `App.js` as shown above.

Step 6: Run the application

1. Build and run the application by pressing **F5** or clicking **Start** in Visual Studio.
2. Navigate to the home page, where you should see the login form.
3. Enter the credentials (`admin` for both username and password).
4. Upon successful login, you will be redirected to the dashboard page.
5. If the credentials are incorrect, you will see an error message.

Conclusion

This basic web application uses React.js for the front-end and ASP.NET Core for the back-end. It provides a login form that validates the credentials (both "admin"), and redirects the user to a dashboard page if the credentials are correct.