**1. Install Required Dependencies**

If you haven't already, install axios for making API requests:

bash

npm install axios

**2. Create Login.js Component**

This component will handle user authentication and send login data to the FastAPI backend.

jsx

import React, { useState } from "react";

import axios from "axios";

import "./Login.css"; // Import CSS file

const Login = () => {

const [username, setUsername] = useState("");

const [password, setPassword] = useState("");

const handleLogin = async (e) => {

e.preventDefault();

// Create FormData since the API expects multipart/form-data

const formData = new FormData();

formData.append("username", username);

formData.append("password", password);

try {

const response = await axios.post("http://127.0.0.1:8000/login", formData, {

headers: { "Content-Type": "multipart/form-data" },

});

localStorage.setItem("authToken", response.data.access\_token);

alert("Login Successful!");

} catch (error) {

alert("Login Failed: " + error.response?.data?.detail || "Unknown error");

}

};

return (

<div className="login-container">

<h2>Login</h2>

<form onSubmit={handleLogin}>

<input

type="email"

placeholder="Email"

value={username}

onChange={(e) => setUsername(e.target.value)}

required

/>

<input

type="password"

placeholder="Password"

value={password}

onChange={(e) => setPassword(e.target.value)}

required

/>

<button type="submit">Login</button>

</form>

</div>

);

};

export default Login;

**3. Basic Login.css Styling**

Add simple styles to enhance the login form appearance.

css

.login-container {

width: 300px;

margin: 50px auto;

padding: 20px;

border: 1px solid #ccc;

border-radius: 5px;

box-shadow: 2px 2px 10px rgba(0, 0, 0, 0.1);

text-align: center;

}

input {

width: 100%;

padding: 10px;

margin: 10px 0;

border: 1px solid #ccc;

border-radius: 5px;

}

button {

width: 100%;

padding: 10px;

background-color: #007bff;

color: white;

border: none;

border-radius: 5px;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

**4. Secure Routes After Login**

Once the user logs in successfully, store the token and restrict access to protected pages.

**Modify App.js**

jsx

import { BrowserRouter as Router, Routes, Route, Navigate } from "react-router-dom";

import Login from "./Login";

import Dashboard from "./Dashboard"; // Protected Page

const PrivateRoute = ({ element }) => {

const token = localStorage.getItem("authToken");

return token ? element : <Navigate to="/login" />;

};

function App() {

return (

<Router>

<Routes>

<Route path="/login" element={<Login />} />

<Route path="/dashboard" element={<PrivateRoute element={<Dashboard />} />} />

</Routes>

</Router>

);

}

export default App;

**1. Install or Reinstall react-router-dom**

Run the following command in your project directory:

npm install react-router-dom