CIE-2

Login Page:  
import React, { useState } from 'react';

import { Link, useNavigate } from 'react-router-dom';

import { validateLoginForm } from '../utils/validation';

import { getUserByEmail } from '../utils/userStorage';

import { useAuth } from '../context/AuthContext';

const **Login** = () => {

  const **navigate** = useNavigate();

  const { login } = useAuth();

  const [formData, **setFormData**] = useState({

    email: '',

    password: ''

  });

  const [errors, **setErrors**] = useState({});

  const [isSubmitting, **setIsSubmitting**] = useState(false);

  const **handleChange** = (*e*) => {

    const { name, value } = *e*.target;

    setFormData(*prev* => ({

      ...*prev*,

      [name]: value

    }));

*// Clear error when user starts typing*

    if (errors[name]) {

      setErrors(*prev* => ({

        ...*prev*,

        [name]: ''

      }));

    }

  };

  const **handleSubmit** = async (*e*) => {

*e*.preventDefault();

    setIsSubmitting(true);

    const validation = validateLoginForm(formData);

    if (!validation.isValid) {

      setErrors(validation.errors);

      setIsSubmitting(false);

      return;

    }

    try {

*// Check if user exists and password matches*

      const user = getUserByEmail(formData.email);

      if (!user) {

        setErrors({ email: 'No account found with this email address' });

        setIsSubmitting(false);

        return;

      }

      if (user.password !== formData.password) {

        setErrors({ password: 'Invalid password' });

        setIsSubmitting(false);

        return;

      }

*// Login successful*

      login(user);

*// Redirect to dashboard*

      navigate('/dashboard');

    } catch (error) {

      setErrors({ general: 'An error occurred during login' });

    } finally {

      setIsSubmitting(false);

    }

  };

  return (

    <div *className*="min-h-screen bg-gradient-to-br from-blue-50 to-indigo-100 flex items-center justify-center px-4">

      <div *className*="max-w-md w-full space-y-8">

        <div *className*="bg-white rounded-lg shadow-xl p-8">

          <div *className*="text-center">

            <h2 *className*="text-3xl font-bold text-gray-900 mb-2">Welcome Back</h2>

            <p *className*="text-gray-600">Sign in to your account</p>

          </div>

          <form *className*="mt-8 space-y-6" *onSubmit*={handleSubmit}>

            {errors.general && (

              <div *className*="bg-red-50 border border-red-200 text-red-700 px-4 py-3 rounded-md">

                {errors.general}

              </div>

            )}

            <div>

              <label *htmlFor*="email" *className*="block text-sm font-medium text-gray-700 mb-1">

                Email Address

              </label>

              <input

*id*="email"

*name*="email"

*type*="email"

*required*

*value*={formData.email}

*onChange*={handleChange}

*className*={`w-full px-3 py-2 border rounded-md shadow-sm focus:outline-none focus:ring-2 focus:ring-blue-500 focus:border-blue-500 ${

                  errors.email ? 'border-red-300' : 'border-gray-300'

                }`}

*placeholder*="Enter your email"

              />

              {errors.email && (

                <p *className*="mt-1 text-sm text-red-600">{errors.email}</p>

              )}

            </div>

            <div>

              <label *htmlFor*="password" *className*="block text-sm font-medium text-gray-700 mb-1">

                Password

              </label>

              <input

*id*="password"

*name*="password"

*type*="password"

*required*

*value*={formData.password}

*onChange*={handleChange}

*className*={`w-full px-3 py-2 border rounded-md shadow-sm focus:outline-none focus:ring-2 focus:ring-blue-500 focus:border-blue-500 ${

                  errors.password ? 'border-red-300' : 'border-gray-300'

                }`}

*placeholder*="Enter your password"

              />

              {errors.password && (

                <p *className*="mt-1 text-sm text-red-600">{errors.password}</p>

              )}

            </div>

            <button

*type*="submit"

*disabled*={isSubmitting}

*className*="w-full flex justify-center py-2 px-4 border border-transparent rounded-md shadow-sm text-sm font-medium text-white bg-blue-600 hover:bg-blue-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-blue-500 disabled:opacity-50 disabled:cursor-not-allowed"

            >

              {isSubmitting ? 'Signing In...' : 'Sign In'}

            </button>

            <div *className*="text-center">

              <p *className*="text-sm text-gray-600">

                Don't have an account?{' '}

                <Link *to*="/signup" *className*="font-medium text-blue-600 hover:text-blue-500">

                  Sign up

                </Link>

              </p>

            </div>

          </form>

        </div>

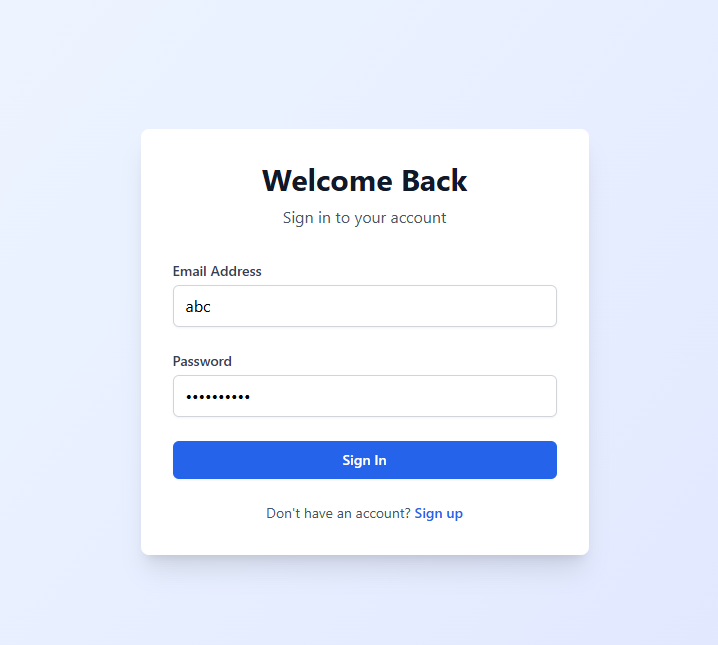
      </div>

    </div>

  );

};

export default Login;

Ouput:  


SignUp:  
import React, { useState } from 'react';

import { Link, useNavigate } from 'react-router-dom';

import { validateSignupForm } from '../utils/validation';

import { saveUser } from '../utils/userStorage';

import { useAuth } from '../context/AuthContext';

const **Signup** = () => {

  const **navigate** = useNavigate();

  const { login } = useAuth();

  const [formData, **setFormData**] = useState({

    name: '',

    email: '',

    password: '',

    confirmPassword: '',

    userType: 'user'

  });

  const [errors, **setErrors**] = useState({});

  const [isSubmitting, **setIsSubmitting**] = useState(false);

  const **handleChange** = (*e*) => {

    const { name, value } = *e*.target;

    setFormData(*prev* => ({

      ...*prev*,

      [name]: value

    }));

*// Clear error when user starts typing*

    if (errors[name]) {

      setErrors(*prev* => ({

        ...*prev*,

        [name]: ''

      }));

    }

  };

  const **handleSubmit** = async (*e*) => {

*e*.preventDefault();

    setIsSubmitting(true);

    const validation = validateSignupForm(formData);

    if (!validation.isValid) {

      setErrors(validation.errors);

      setIsSubmitting(false);

      return;

    }

    try {

*// Check if user already exists*

      const existingUsers = JSON.parse(localStorage.getItem('users') || '[]');

      const userExists = existingUsers.find(*user* => *user*.email === formData.email);

      if (userExists) {

        setErrors({ email: 'User with this email already exists' });

        setIsSubmitting(false);

        return;

      }

*// Save user to localStorage*

      const userData = {

        name: formData.name,

        email: formData.email,

        password: formData.password, *// In real app, this should be hashed*

        userType: formData.userType

      };

      saveUser(userData);

*// Auto-login after successful signup*

      login(userData);

*// Redirect to dashboard*

      navigate('/dashboard');

    } catch (error) {

      setErrors({ general: 'An error occurred during signup' });

    } finally {

      setIsSubmitting(false);

    }

  };

  return (

    <div *className*="min-h-screen bg-gradient-to-br from-blue-50 to-indigo-100 flex items-center justify-center px-4">

      <div *className*="max-w-md w-full space-y-8">

        <div *className*="bg-white rounded-lg shadow-xl p-8">

          <div *className*="text-center">

            <h2 *className*="text-3xl font-bold text-gray-900 mb-2">Create Account</h2>

            <p *className*="text-gray-600">Join us today!</p>

          </div>

          <form *className*="mt-8 space-y-6" *onSubmit*={handleSubmit}>

            {errors.general && (

              <div *className*="bg-red-50 border border-red-200 text-red-700 px-4 py-3 rounded-md">

                {errors.general}

              </div>

            )}

            <div>

              <label *htmlFor*="name" *className*="block text-sm font-medium text-gray-700 mb-1">

                Full Name

              </label>

              <input

*id*="name"

*name*="name"

*type*="text"

*required*

*value*={formData.name}

*onChange*={handleChange}

*className*={`w-full px-3 py-2 border rounded-md shadow-sm focus:outline-none focus:ring-2 focus:ring-blue-500 focus:border-blue-500 ${

                  errors.name ? 'border-red-300' : 'border-gray-300'

                }`}

*placeholder*="Enter your full name"

              />

              {errors.name && (

                <p *className*="mt-1 text-sm text-red-600">{errors.name}</p>

              )}

            </div>

            <div>

              <label *htmlFor*="email" *className*="block text-sm font-medium text-gray-700 mb-1">

                Email Address

              </label>

              <input

*id*="email"

*name*="email"

*type*="email"

*required*

*value*={formData.email}

*onChange*={handleChange}

*className*={`w-full px-3 py-2 border rounded-md shadow-sm focus:outline-none focus:ring-2 focus:ring-blue-500 focus:border-blue-500 ${

                  errors.email ? 'border-red-300' : 'border-gray-300'

                }`}

*placeholder*="Enter your email"

              />

              {errors.email && (

                <p *className*="mt-1 text-sm text-red-600">{errors.email}</p>

              )}

            </div>

            <div>

              <label *htmlFor*="password" *className*="block text-sm font-medium text-gray-700 mb-1">

                Password

              </label>

              <input

*id*="password"

*name*="password"

*type*="password"

*required*

*value*={formData.password}

*onChange*={handleChange}

*className*={`w-full px-3 py-2 border rounded-md shadow-sm focus:outline-none focus:ring-2 focus:ring-blue-500 focus:border-blue-500 ${

                  errors.password ? 'border-red-300' : 'border-gray-300'

                }`}

*placeholder*="Enter your password"

              />

              {errors.password && (

                <p *className*="mt-1 text-sm text-red-600">{errors.password}</p>

              )}

            </div>

            <div>

              <label *htmlFor*="confirmPassword" *className*="block text-sm font-medium text-gray-700 mb-1">

                Confirm Password

              </label>

              <input

*id*="confirmPassword"

*name*="confirmPassword"

*type*="password"

*required*

*value*={formData.confirmPassword}

*onChange*={handleChange}

*className*={`w-full px-3 py-2 border rounded-md shadow-sm focus:outline-none focus:ring-2 focus:ring-blue-500 focus:border-blue-500 ${

                  errors.confirmPassword ? 'border-red-300' : 'border-gray-300'

                }`}

*placeholder*="Confirm your password"

              />

              {errors.confirmPassword && (

                <p *className*="mt-1 text-sm text-red-600">{errors.confirmPassword}</p>

              )}

            </div>

            <div>

              <label *htmlFor*="userType" *className*="block text-sm font-medium text-gray-700 mb-1">

                User Type

              </label>

              <select

*id*="userType"

*name*="userType"

*value*={formData.userType}

*onChange*={handleChange}

*className*={`w-full px-3 py-2 border rounded-md shadow-sm focus:outline-none focus:ring-2 focus:ring-blue-500 focus:border-blue-500 ${

                  errors.userType ? 'border-red-300' : 'border-gray-300'

                }`}

              >

                <option *value*="user">Normal User</option>

                <option *value*="admin">Admin</option>

              </select>

              {errors.userType && (

                <p *className*="mt-1 text-sm text-red-600">{errors.userType}</p>

              )}

            </div>

            <button

*type*="submit"

*disabled*={isSubmitting}

*className*="w-full flex justify-center py-2 px-4 border border-transparent rounded-md shadow-sm text-sm font-medium text-white bg-blue-600 hover:bg-blue-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-blue-500 disabled:opacity-50 disabled:cursor-not-allowed"

            >

              {isSubmitting ? 'Creating Account...' : 'Sign Up'}

            </button>

            <div *className*="text-center">

              <p *className*="text-sm text-gray-600">

                Already have an account?{' '}

                <Link *to*="/login" *className*="font-medium text-blue-600 hover:text-blue-500">

                  Sign in

                </Link>

              </p>

            </div>

          </form>

        </div>

      </div>

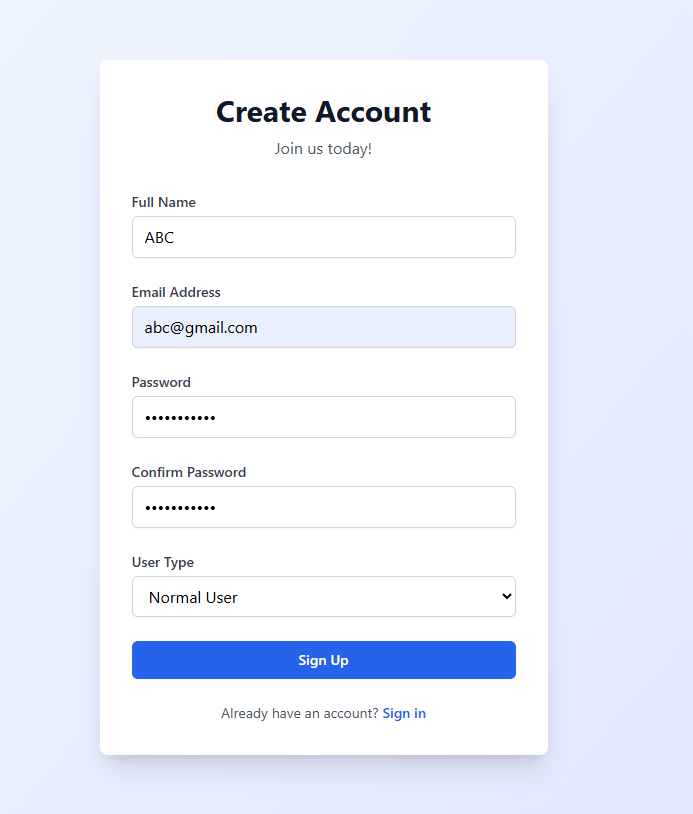
    </div>

  );

};

export default Signup;

Output:



Dashboard:  
import React from 'react';

const **Dashboard** = () => {

  return (

    <div *className*="min-h-screen bg-gradient-to-br from-blue-50 to-indigo-100 flex items-center justify-center px-4">

      <div *className*="max-w-md w-full space-y-8">

        <div *className*="bg-white rounded-lg shadow-xl p-8">

          <div *className*="text-center">

            <h2 *className*="text-3xl font-bold text-gray-900 mb-2">Dashboard</h2>

            <p *className*="text-gray-600">Welcome to your dashboard!</p>

          </div>

        </div>

      </div>

    </div>

  );

};

export default Dashboard;

Output:  
