RegisterPage.jsx

```
import React, { useState } from 'react';

function RegisterPage({ onRegisterSuccess, onLoginClick }) {
  const [formData, setFormData] = useState({
    name: '',
    email: '',
    password: '',
    gender: '',
    country: '',
    terms: false,
    bio: ''
  });

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

  const validateEmail = (email) => /^\S+@\S+\.\S+$/.test(email);
  const validatePassword = (password) => password.length >= 6;

  const handleChange = (e) => {
    const { name, value, type, checked } = e.target;
    setFormData((prev) => ({
      ...prev,
      [name]: type === 'checkbox' ? checked : value
    }));
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    const newErrors = {};

    if (!formData.name) newErrors.name = 'Name is required';
    if (!formData.email || !validateEmail(formData.email)) newErrors.email
= 'Valid email is required';
    if (!formData.password || !validatePassword(formData.password))
newErrors.password = 'Password must be at least 6 characters';
    if (!formData.gender) newErrors.gender = 'Select a gender';
    if (!formData.country) newErrors.country = 'Select a country';
```

```
    if (!formData.terms) newErrors.terms = 'You must accept terms';

    setErrors(newErrors);

    if (Object.keys(newErrors).length === 0) {
      const existingUsers = JSON.parse(localStorage.getItem('users')) ||
[];

      const userExists = existingUsers.find((user) => user.email ===
formData.email);
      if (userExists) {
        alert('User with this email already exists.');
        return;
      }

      const updatedUsers = [...existingUsers, formData];
      localStorage.setItem('users', JSON.stringify(updatedUsers));

      onRegisterSuccess();
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <h2>Register</h2>

      <div>
        <label>Name:</label>
        <input name="name" value={formData.name} onChange={handleChange}
/>
        {errors.name && <p>{errors.name}</p>}
      </div>

      <div>
        <label>Email:</label>
        <input name="email" value={formData.email} onChange={handleChange}
/>
        {errors.email && <p>{errors.email}</p>}
      </div>
```

```
      <div>
        <label>Password:</label>
        <input name="password" type="password" value={formData.password}
onChange={handleChange} />
        {errors.password && <p>{errors.password}</p>}
      </div>

      <div>
        <label>Gender:</label>
        <input type="radio" name="gender" value="male"
onChange={handleChange} /> Male
        <input type="radio" name="gender" value="female"
onChange={handleChange} /> Female
        {errors.gender && <p>{errors.gender}</p>}
      </div>

      <div>
        <label>Country:</label>
        <select name="country" value={formData.country}
onChange={handleChange}>
          <option value="">--Select--</option>
          <option value="India">India</option>
          <option value="USA">USA</option>
          <option value="UK">UK</option>
        </select>
        {errors.country && <p>{errors.country}</p>}
      </div>

      <div>
        <label>Accept Terms:</label>
        <input type="checkbox" name="terms" checked={formData.terms}
onChange={handleChange} />
        {errors.terms && <p>{errors.terms}</p>}
      </div>

      <div>
        <label>Bio:</label>
        <textarea name="bio" value={formData.bio}
onChange={handleChange}></textarea>
      </div>
```

```
      <button type="submit">Register</button>

      <p>
        Already have an account?{' '}
        <button type="button" onClick={onLoginClick}>
          Login here
        </button>
      </p>
    </form>
  );
}

export default RegisterPage;
```

```
js                                                    Copy  Edit

setFormData((prev) => ({
  ...prev,
  [name]: type === 'checkbox' ? checked : value
}));
```

- `setFormData` is used to update the form state.
- `prev` is the previous state of `formData`.
- `...prev` keeps all the previous values.
- `[name]`: This uses **computed property names** to dynamically set the value of the input field that triggered the event.
- `type === 'checkbox' ? checked : value`:
  - If the input is a checkbox, use `checked` (a boolean).
  - Otherwise, use the input's `value`.

js Copy Edit

```
/^\S+@\S+\.\S+$/
```

This is a **regular expression** used to test email formatting.

Let's break it into pieces:

Part	Meaning
`^`	Anchors the pattern to the **start** of the string
`\S+`	Matches **one or more non-whitespace characters** before the `@`
`@`	Matches the literal `@` symbol
`\S+`	Again, matches one or more non-whitespace characters after the `@`
`\.`	Escaped dot `.` – matches the literal dot (.) in the domain
`\S+`	Matches one or more non-whitespace characters after the `.`
`$`	Anchors the pattern to the **end** of the string

🔄 Full Pattern Example Match:

For the string `"john.doe@example.com"`:

- `"john.doe"` matches `\S+`
- `"@"` matches `@`
- `"example"` matches `\S+`
- `"."` matches `\.`
- `"com"` matches `\S+`

✅ Match → returns `true`

Suppose:

- `formData = { name: 'Alice', email: 'alice@example.com', password: 'abc123' }`

After submission:

```
js                                                                    Copy  Edit

localStorage.getItem('users') → null

existingUsers → []

updatedUsers → [ { name: 'Alice', email: 'alice@example.com', password: 'abc123' } ]

localStorage.setItem('users', JSON.stringify(updatedUsers))
→ localStorage now stores:
   'users': '[{"name":"Alice","email":"alice@example.com","password":"abc123"}]'
```

Login Page

```
import React, { useState } from 'react';

function LoginPage({ onRegisterClick, onLoginSuccess }) {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState('');

  const validateEmail = (email) => /^\S+@\S+\.\S+$/.test(email);

  const handleSubmit = (e) => {
    e.preventDefault();

    if (!validateEmail(email)) {
      setError('Invalid email format');
      return;
    }
    if (password.length < 6) {
      setError('Password must be at least 6 characters');
      return;
    }
```

```
    const users = JSON.parse(localStorage.getItem('users')) || [];
    const matchedUser = users.find(
      (user) => user.email === email && user.password === password
    );

    if (matchedUser) {
      setError('');
      onLoginSuccess(matchedUser);
    } else {
      setError('Invalid email or password');
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <h2>Login</h2>

      <div>
        <label>Email:</label>
        <input value={email} onChange={(e) => setEmail(e.target.value)} />
      </div>

      <div>
        <label>Password:</label>
        <input type="password" value={password} onChange={(e) =>
setPassword(e.target.value)} />
      </div>

      {error && <p>{error}</p>}

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

      <p>
        Don't have an account?{' '}
        <button type="button" onClick={onRegisterClick}>
          Please register
        </button>
      </p>
    </form>
```

```
  );
}

export default LoginPage;
```

Dashboard

```
import React from 'react';

function Dashboard({ user, onLogout }) {
  return (
    <div>
      <h2>Welcome, {user.name}</h2>
      <p><strong>Email:</strong> {user.email}</p>
      <p><strong>Gender:</strong> {user.gender}</p>
      <p><strong>Country:</strong> {user.country}</p>
      <p><strong>Bio:</strong> {user.bio}</p>

      <button onClick={onLogout}>Logout</button>
    </div>
  );
}

export default Dashboard;
```

App

```
import React, { useState } from 'react';
import LoginPage from './LoginPage';
import RegisterPage from './RegisterPage';
import Dashboard from './Dashboard';

function App() {
  const [currentView, setCurrentView] = useState('login');
  const [currentUser, setCurrentUser] = useState(null);

  // Called on successful login: set user and show dashboard
```

```
  const handleLoginSuccess = (user) => {
    setCurrentUser(user);
    setCurrentView('dashboard');
  };

  // Called on successful registration: show login and alert
  const handleRegisterSuccess = () => {
    alert('Registration successful!');
    setCurrentView('login');
  };

  // Render based on currentView
  return (
    <div>
      {currentView === 'login' && (
        <LoginPage
          onRegisterClick={() => setCurrentView('register')}
          onLoginSuccess={handleLoginSuccess}
        />
      )}

      {currentView === 'register' && (
        <RegisterPage
          onRegisterSuccess={handleRegisterSuccess}
          onLoginClick={() => setCurrentView('login')}
        />
      )}

      {currentView === 'dashboard' && currentUser && (
        <Dashboard user={currentUser} onLogout={() => {
          setCurrentUser(null);
          setCurrentView('login');
        }} />
      )}
    </div>
  );
}
export default App;
```