INTERNSHIP REPORT

WEEK 6 DAY 5

Submitted to:	**Ali Hyder**	Submission Date:	**1st August, 2025**
Internship Domain:	**Front Development**	Internship Name:	**ProSensia**
Student Name:	**Yasal Qamar**	Roll No.	**S25031**

React.js Basics

Topics: Forms in React, Lifting State Up

Objective

To understand how form handling works in React, how to manage form data using component state (useState), and how to lift shared state up to a parent component when needed.

Topics Covered:

- Forms in React
- Controlled vs Uncontrolled Components
- Handling Form Inputs with `useState`
- Lifting State Up

What is React?

React is a **JavaScript library** used to build **user interfaces (UI)**, especially for **web applications**. It helps you create **dynamic, reusable components** that can change based on data without needing to reload the page.

Why Use React?

- **Component-Based**: Break your UI into small, reusable pieces (like buttons, forms, headers).
- **Fast and Efficient**: Uses a virtual DOM to update only the changed parts of the UI.
- **One-Way Data Flow**: Data flows in one direction, making the app easier to debug and manage.
- **Reusable JSX Code**: Write HTML-like code inside JavaScript using JSX.

1. Forms in React

- In React, form inputs like <input>, <textarea>, and <select> are usually handled as controlled components, where form data is synced with React's useState.
- This ensures two-way binding: React controls the input values, and user actions update the state.

2. Handling Form Inputs

Examples:

```
const [name, setName] = useState("");

<input

 type="text"

 value={name}

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

/>
```

- The input is tied to the state `name`.
- Any change in the input updates the state using `setName`.

3. Lifting State Up

- When two or more components need to share or access the same piece of state, we "lift" the state up to their common parent.
- This promotes single source of truth and clear data flow.

Example

- If two sibling components need to access the same input value:
- Move the state to their parent.
- Pass value and update function as props.

```
function Parent() {

 const [sharedValue, setSharedValue] = useState("");
```

```
 return (

  <>

   <ChildA value={sharedValue} />

   <ChildB onChange={setSharedValue} />

  </>

 );

}
```

CODING:

.app.js

```
import './App.css';
import React, { useState } from 'react';
import InputForm from './InputForm';
import DisplayData from './DisplayData';

export default function App() {
  const [formData, setFormData] = useState({ name: '', email: '' });
  const [submitted, setSubmitted] = useState(false);
  const [errors, setErrors] = useState({});

  const validateForm = () => {
    const newErrors = {};
    if (!formData.name.trim()) newErrors.name = 'Name is required';
    if (!formData.email.trim()) {
      newErrors.email = 'Email is required';
    } else if (!/\S+@\S+\.\S+/.test(formData.email)) {
      newErrors.email = 'Email is invalid';
    }
    return newErrors;
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    const validationErrors = validateForm();
    if (Object.keys(validationErrors).length === 0) {
      setSubmitted(true);
      setErrors({});
```

```
      setTimeout(() => setSubmitted(false), 3000);
    } else {
      setSubmitted(false);
      setErrors(validationErrors);
    }
  };

  const handleClear = () => {
    setFormData({ name: '', email: '' });
    setErrors({});
    setSubmitted(false);
  };

  return (
    <div className="container">
      <h1>React Form – Validation & Popup</h1>
      <form onSubmit={handleSubmit}>
        <InputForm formData={formData} setFormData={setFormData}
errors={errors} />
        <div className="button-group">
          <button type="submit" className="submit-btn">Submit</button>
          <button type="button" className="clear-btn"
onClick={handleClear}>Clear</button>
        </div>
      </form>

      {submitted && <div className="popup">Form submitted successfully!
</div>}

      <DisplayData formData={formData} />
    </div>
  );
}
```

InputForm.js

```
import React from 'react';

export default function InputForm({ formData, setFormData, errors }) {
  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData(prev => ({ ...prev, [name]: value }));
  };
```

```
  return (
    <>
      <input
        type="text"
        name="name"
        placeholder="Enter your name"
        value={formData.name}
        onChange={handleChange}
      />
      {errors.name && <div className="error">{errors.name}</div>}

      <input
        type="email"
        name="email"
        placeholder="Enter your email"
        value={formData.email}
        onChange={handleChange}
      />
      {errors.email && <div className="error">{errors.email}</div>}
    </>
  );
}
```

DisplayData.js

```
import React from 'react';

export default function DisplayData({ formData }) {
  return (
    <>
      <h3>Entered Data:</h3>
      <p><strong>Name:</strong> {formData.name}</p>
      <p><strong>Email:</strong> {formData.email}</p>
    </>
  );
}
```

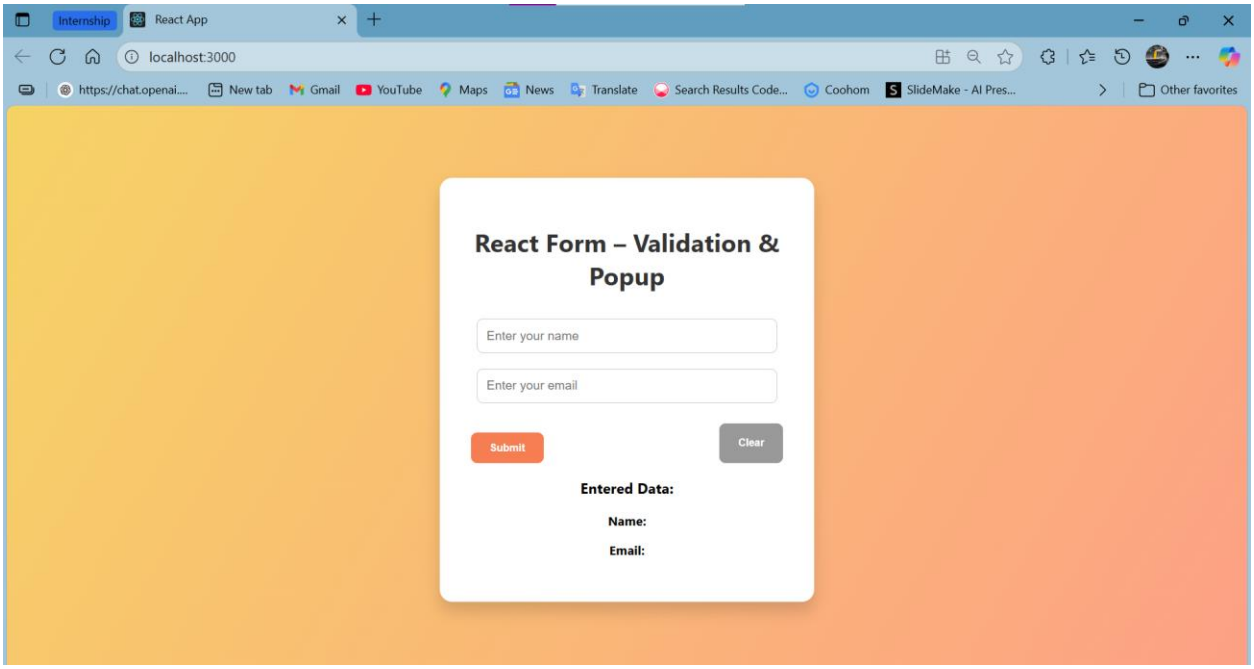
.index.css

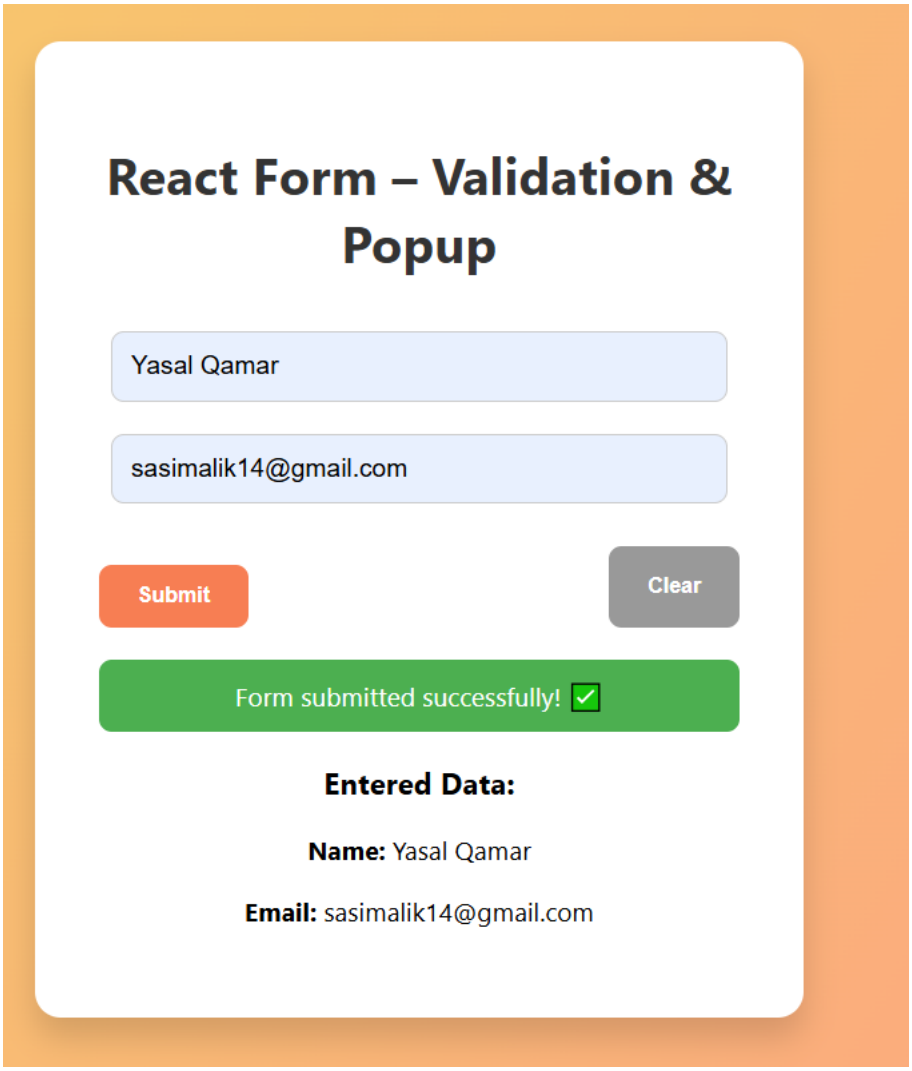
```
body {
  margin: 0;
  padding: 0;
```

```
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  background: linear-gradient(120deg, #f6d365 0%, #fda085 100%);
  min-height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
}

.container {
  background: #ffffff;
  padding: 40px;
  border-radius: 16px;
  box-shadow: 0 12px 20px rgba(0, 0, 0, 0.15);
  width: 100%;
  max-width: 400px;
  text-align: center;
}

h1 {
  margin-bottom: 24px;
  color: #333;
}

input {
  width: 90%;
  padding: 12px;
  margin: 10px 0;
  border: 1px solid #ccc;
  border-radius: 8px;
  font-size: 16px;
  outline: none;
}

input:focus {
  border-color: #f77e53;
}

.submit-btn {
  margin-top: 12px;
  padding: 12px 24px;
  background-color: #f77e53;
  color: #fff;
  font-weight: bold;
  border: none;
  border-radius: 8px;
```

```
  cursor: pointer;
  transition: background 0.3s ease;
}

.submit-btn:hover {
  background-color: #d65b3d;
}

.error {
  color: #d9534f;
  font-size: 14px;
  margin-bottom: 8px;
}

.popup {
  background-color: #4caf50;
  color: white;
  padding: 12px;
  margin-top: 20px;
  border-radius: 8px;
  animation: fade 3s forwards;
}

@keyframes fade {
  0% { opacity: 1; }
  90% { opacity: 1; }
  100% { opacity: 0; display: none; }
}
.button-group {
  display: flex;
  justify-content: space-between;
  gap: 10px;
  margin-top: 16px;
}

.clear-btn {
  padding: 12px 24px;
  background-color: #999;
  color: white;
  font-weight: bold;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  transition: background 0.3s ease;
}
```

```
.clear-btn:hover {
  background-color: #666;
}
```





CONCUSLION:

This session strengthened my understanding of form handling in React using controlled components and demonstrated how to lift state to a parent for shared access between components. I also practiced form validation, dynamic UI updates, and user feedback via popups. These are essential skills for modern frontend apps and will help in building robust user interfaces.

I now feel confident working with React forms and managing complex state relationships across components.