1.Create an application in ReactJSform and add client and server side validation

# App.js

```
import React from 'react';

import Form from './Form';


function App() {

  return (

    <div className="App">

      <Form />

    </div>

  );

}


export default App;
```

#Form.js

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


function Form() {

  // State to store form data

  const [formData, setFormData] = useState({

    username: '',

    email: '',

    password: ''

  });
```

```
// State to store validation errors
const [errors, setErrors] = useState({
  username: '',
  email: '',
  password: ''
});

// State to store form submission status
const [submitStatus, setSubmitStatus] = useState('');

// Handle form field change
const handleChange = (e) => {
  const { name, value } = e.target;
  setFormData({ ...formData, [name]: value });
};

// Client-side validation function
const validateForm = () => {
  let formErrors = {};
  let isValid = true;

  // Username validation
  if (!formData.username) {
    formErrors.username = 'Username is required';
    isValid = false;
  }

  // Email validation
  if (!formData.email) {
    formErrors.email = 'Email is required';
    isValid = false;
  } else if (!/\S+@\S+\.\S+/.test(formData.email)) {
    formErrors.email = 'Email is invalid';
    isValid = false;
  }

  // Password validation
  if (!formData.password) {
    formErrors.password = 'Password is required';
    isValid = false;
  } else if (formData.password.length < 6) {
    formErrors.password = 'Password must be at least 6 characters long';
    isValid = false;
  }

  setErrors(formErrors);
  return isValid;
};

// Handle form submission
const handleSubmit = async (e) => {
  e.preventDefault();

  if (validateForm()) {
    // Simulate server-side validation (for example, checking if username already exists)
    try {
      const response = await simulateServerValidation(formData);
      if (response.success) {
        setSubmitStatus('Form submitted successfully!');
      } else {
        setSubmitStatus('Server-side validation failed.');
      }
    } catch (error) {
      setSubmitStatus('Server error occurred.');
    }
  } else {
    setSubmitStatus('Please correct the errors in the form.');
  }
};

// Simulate server-side validation (for example, check if username already exists)
const simulateServerValidation = (formData) => {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      if (formData.username === 'existingUser') {
        reject('Username already exists.');
      } else {
        resolve({ success: true });
      }
    }, 1000);
  });
};

return (
  <div>
    <h2>React Form with Validation</h2>
```

```
<form onSubmit={handleSubmit}>
 <div>
  <label>Username</label>
  <input
   type="text"
   name="username"
   value={formData.username}
   onChange={handleChange}
  />
  {errors.username && <p style={{ color: 'red' }}>{errors.username}</p>}
 </div>
 <div>
  <label>Email</label>
  <input
   type="email"
   name="email"
   value={formData.email}
   onChange={handleChange}
  />
  {errors.email && <p style={{ color: 'red' }}>{errors.email}</p>}
 </div>
 <div>
  <label>Password</label>
  <input
   type="password"
   name="password"
   value={formData.password}
   onChange={handleChange}
  />
```

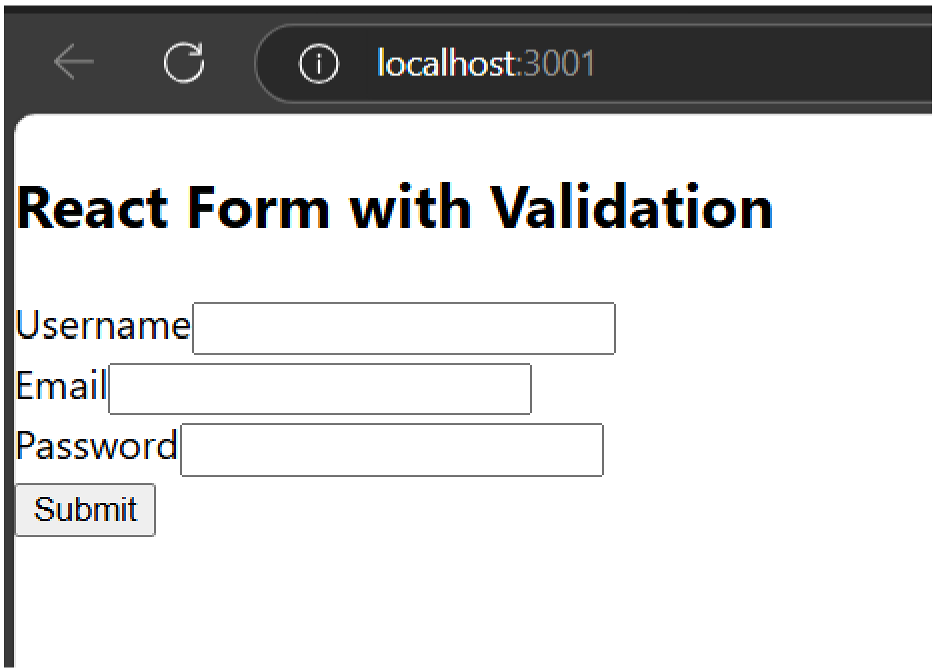
```
          {errors.password && <p style={{ color: 'red' }}>{errors.password}</p>}
        </div>
        <button type="submit">Submit</button>
      </form>
      {submitStatus && <p>{submitStatus}</p>}
    </div>
  );
}

export default Form;
```

#App.css

```
.App {
  font-family: Arial, sans-serif;
  margin: 0 auto;
  max-width: 400px;
  padding: 20px;
}

form div {
  margin-bottom: 10px;
}

input {
  width: 100%;
  padding: 8px;
  margin-top: 5px;
}
```

```
button {
  padding: 10px;
  background-color: #4CAF50;
  color: white;
  border: none;
  cursor: pointer;
}

button:hover {
  background-color: #45a049;
}

p {
  font-size: 14px;
}
```

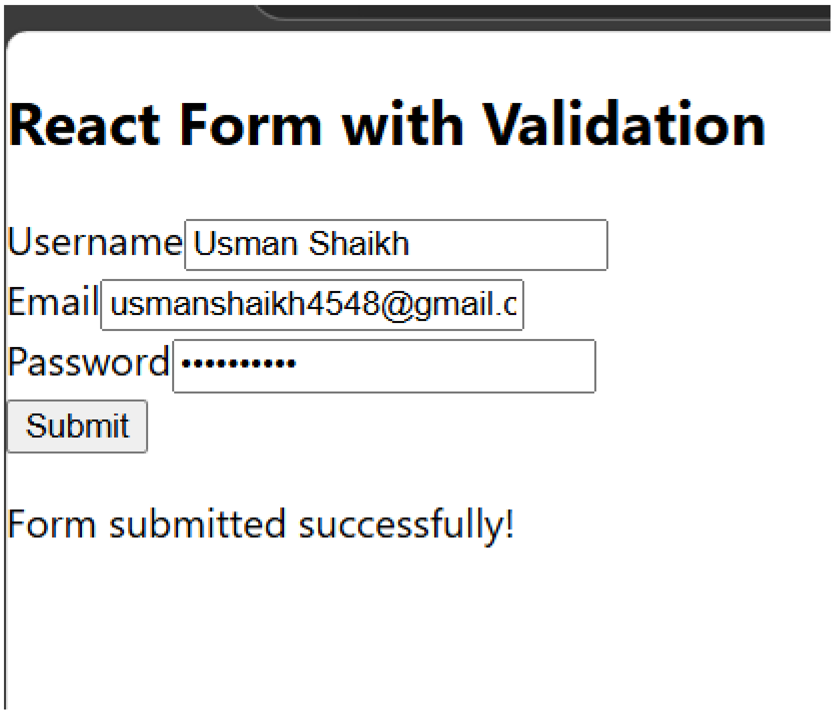
Output:-



React Form with Validation

Username
Email
Password

```
};
```



React Form with Validation

Username Usman Shaikh
Email usmanshaikh4548@gmail.c
Password ••••••••••
Submit

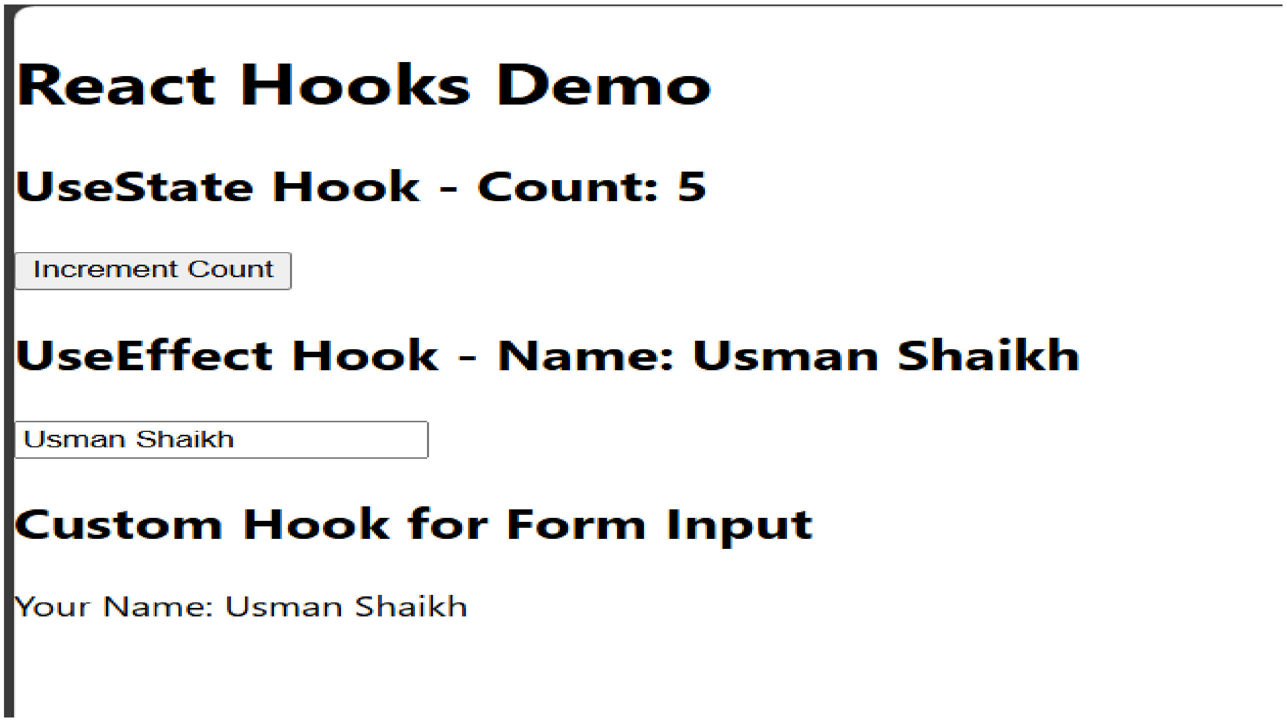
Form submitted successfully!

Create an application to implement react hook

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


// Custom hook to handle form input
function useInput(initialValue) {
  const [value, setValue] = useState(initialValue);


  const handleChange = (event) => {
    setValue(event.target.value);
  };


  return {
    value,
    onChange: handleChange,
  };
}
```

```
function App() {
  // State hooks
  const [count, setCount] = useState(0);
  const { value: name, onChange: handleNameChange } = useInput('');

  // Effect hook to run a side effect
  useEffect(() => {
    console.log('Component rendered or count/state changed:', count);
  }, [count]); // The effect will run when `count` changes

  // Fetch data when the component mounts (example)
  useEffect(() => {
    const fetchData = async () => {
      const response = await fetch('https://jsonplaceholder.typicode.com/posts/1');
      const data = await response.json();
      console.log('Fetched Data:', data);
    };
    fetchData();
  }, []); // The effect will run only once after the initial render (like componentDidMount)

  return (
    <div className="App">
      <h1>React Hooks Demo</h1>

      <div>
        <h2>UseState Hook - Count: {count}</h2>
        <button onClick={() => setCount(count + 1)}>Increment Count</button>
      </div>
      <div>
        <h2>UseEffect Hook - Name: {name}</h2>
        <input type="text" value={name} onChange={handleNameChange} placeholder="Enter your name"
/>
      </div>

      <div>
        <h2>Custom Hook for Form Input</h2>
        <p>Your Name: {name}</p>
      </div>
    </div>
  );
}

export default App;
```

Output:-



React Hooks Demo

UseState Hook - Count: 5

UseEffect Hook - Name: Usman Shaikh

Custom Hook for Form Input

Your Name: Usman Shaikh