Experiment-6

Design and implement a React Form that collects user input for name, email, and password. Form Fields are Name, Email, Password. Ensure all fields are filled before allowing form submission. Validate the email field to ensure it follows the correct email format (e.g., example@domain.com). Optionally enforce a minimum password length or complexity. Display error messages for invalid or missing inputs. Provide visual cues (e.g., red borders) to highlight invalid fields. Prevent form submission until all fields pass validation. Log or display the entered data upon successful submission (optional). Add a "Show Password" toggle for the password field. Implement client side sanitization to ensure clean input.

App.js

```
import React from 'react';
import './App.css';
import Form from './Form';
function App() {
return (
<div className="App">
<h1>React Form Validation</h1>
<Form />
</div>
);
}
export default App;
```

Form.js

```
import React, { useState } from 'react';
const Form = () => {
const [formData, setFormData] = useState({ name: '', email: '', password: '' });
```

```
const [errors, setErrors] = useState({});
const [submittedData, setSubmittedData] = useState(null);
const [showPassword, setShowPassword] = useState(false);
const handleInputChange = (e) => {
const { name, value } = e.target;
setFormData(prev => ({ ...prev, [name]: value }));
};
const togglePassword = () => setShowPassword(prev => !prev);
const validateForm = () => {
const newErrors = {};
const emailPattern = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/;
if (!formData.name) newErrors.name = 'Name is required';
if (!formData.email) newErrors.email = 'Email is required';
else if (!emailPattern.test(formData.email)) newErrors.email = 'Invalid email format';
if (!formData.password) newErrors.password = 'Password is required';
else if (formData.password.length < 6) newErrors.password = 'Password must be at least 6 characters';
setErrors(newErrors);
return Object.keys(newErrors).length === 0;
};
const handleSubmit = (e) => {
e.preventDefault();
if (validateForm()) {
setSubmittedData(formData);
console.log('Form data submitted:', formData);
}
```

```
};
const renderInput = (label, name, type = 'text') => (
<div>
<label htmlFor={name}>{label}:</label>
<input
type={name === 'password' ? (showPassword ? 'text' : 'password') : type}
id={name}
name={name}
value={formData[name]}
onChange={handleInputChange}
style={{ borderColor: errors[name] ? 'red' : 'black' }}
/>
{name === 'password' && (
<button type="button" onClick={togglePassword}>
{showPassword ? 'Hide Password' : 'Show Password'}
</button>
)}
{errors[name] && <p style={{ color: 'red' }}>{errors[name]}</p>}
</div>
);
return (
<div>
<form onSubmit={handleSubmit}>
{renderInput('Name', 'name')}
{renderInput('Email', 'email', 'email')}
{renderInput('Password', 'password')}
```

```
<button type="submit">Submit</button>
</form>
{submittedData && (
<div>
<h3>Submitted Data:</h3>
<p>Name: {submittedData.name}</p>
<p>Email: {submittedData.email}</p>
<p>Password: {submittedData.password}</p>
</div>
)}
</div>
);
};
export default Form;
```

App.css

```
.App {
font-family: Arial, sans-serif;
max-width: 400px;
margin: 50px auto;
padding: 20px;
border: 1px solid #ddd;
border-radius: 10px;
box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
}
form > div {
margin-bottom: 15px;
```

```
}
label {
display: block;
margin-bottom: 5px;
font-weight: bold;
}
input {
width: 100%;
padding: 8px;
box-sizing: border-box;
}
button {
padding: 8px 12px;
margin-top: 10px;
cursor: pointer;
}
p {
margin: 5px 0 0;
}
```