**Lesson 01 Demo 02**

**Creating Form Validation Using a Custom Hook**

**Objective:** To create the form validation using a custom Hook for enhanced user experience

**Tools required:** Node.js and React.js

**Prerequisites:** None

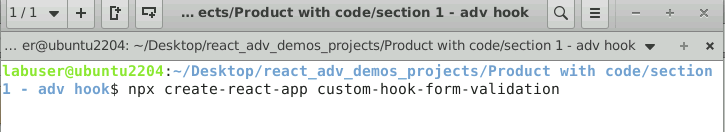
Steps to be followed:

1. Create and set up the React project
2. Create a custom Hook file
3. Use a custom Hook in user-defined components
4. Import user-defined functional components in the App.js file
5. Add bootstrap features in the index.html file
6. Run the application

**Step 1:** **Create and set up the React project**

1. Open a terminal window to run the following command to create a React application:

**npx create-react-app custom-hook-form-validation**

****

1. Open the created React application folder (**custom-hook-form-validation**) in VS Code by clicking on **File** in the top left corner and selecting **Open Folder**

**A screenshot of a computer

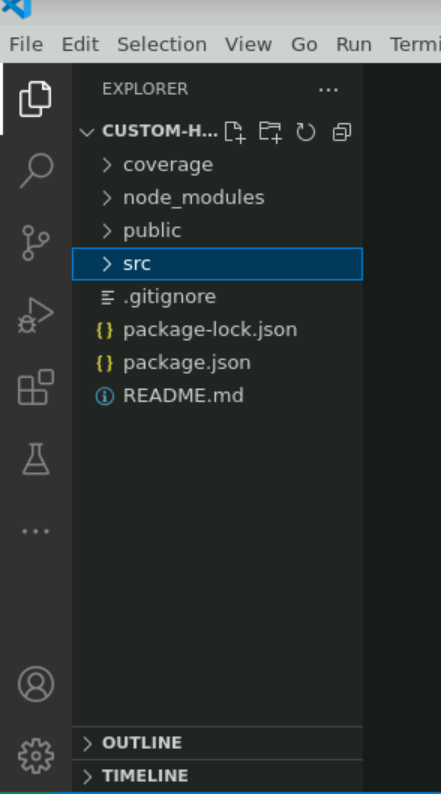
Description automatically generated**

1. Click on **Open** button

**A screenshot of a computer

Description automatically generated**

The folder structure appears as follows:



1. Inside the folder, open the **TERMINAL** and run the following command to install the required dependencies:

**npm install**

**A screen shot of a computer

Description automatically generated**

**Note**: This command helps to install all the required dependencies mentioned in the **package.json** file in the local machine in the form of a **node\_module** folder.

**Step 2:** **Create a custom Hook file**

1. Right-click on the **src** folder and select **New File**

A screenshot of a computer

Description automatically generated

1. Create a folder named **useFormValidation.js**

**A computer screen with a black box

Description automatically generated**

1. Inside the **useFormValidation.js** file, enter the following code that consists of the logic to create the custom Hook:

**import { useState, useEffect } from 'react';**

**const useFormValidation = (initialState, validationRules) => {**

**const [values, setValues] = useState(initialState);**

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

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

**useEffect(() => {**

**const validate = () => {**

**let valid = true;**

**const newErrors = {};**

**for (const fieldName in validationRules) {**

**if (Object.prototype.hasOwnProperty.call(validationRules, fieldName)) {**

**const rules = validationRules[fieldName];**

**const value = values[fieldName];**

**for (const rule of rules) {**

**if (rule.required && !value.trim()) {**

**newErrors[fieldName] = `${fieldName} is required`;**

**valid = false;**

**break;**

**}**

**if (rule.minLength && value.length < rule.minLength) {**

**newErrors[fieldName] = `${fieldName} must be at least ${rule.minLength} characters`;**

**valid = false;**

**break;**

**}**

**// Add more validation rules as needed**

**}**

**}**

**}**

**setErrors(newErrors);**

**return valid;**

**};**

**if (isSubmitting) {**

**const isValid = validate();**

**if (isValid) {**

**// Perform form submission, e.g., send data to server**

**console.log('Form data:', values);**

**setValues(initialState);**

**setIsSubmitting(false);**

**} else {**

**setIsSubmitting(false);**

**}**

**}**

**}, [isSubmitting, values, validationRules, initialState]);**

**const handleSubmit = (event) => {**

**event.preventDefault();**

**setIsSubmitting(true);**

**};**

**const handleChange = (event) => {**

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

**setValues({ ...values, [name]: value });**

**};**

**return {**

**values,**

**handleChange,**

**handleSubmit,**

**errors,**

**isSubmitting,**

**};**

**};**

**export default useFormValidation;**

**A screen shot of a computer

Description automatically generated**

**Step 3:** **Use a custom Hook in user-defined components**

1. Inside the **src** folder, create a **LoginPage.js** file and enter the following code to import the custom Hook and use its functionality:

**import React from 'react';**

**import useFormValidation from './useFormValidation'; // Import the custom hook**

**const LoginPage = () => {**

**const INITIAL\_STATE = {**

**username: '',**

**email: '',**

**password: '',**

**};**

**const VALIDATION\_RULES = {**

**username: [{ required: true, minLength: 3 }],**

**email: [{ required: true }],**

**password: [{ required: true, minLength: 6 }],**

**};**

**const {**

**values,**

**handleChange,**

**handleSubmit,**

**errors,**

**isSubmitting,**

**} = useFormValidation(INITIAL\_STATE, VALIDATION\_RULES);**

**return (**

**<div className='container'>**

**<form onSubmit={handleSubmit}>**

**<div className='row'>**

**<div className='col-4 offset-2'>**

**<input**

**type="text"**

**name="username"**

**value={values.username}**

**onChange={handleChange}**

**placeholder="Username"**

**/>**

**</div>**

**<div className='col-6'>**

**{errors.username && <span style={{"color":"red"}}>{errors.username}</span>}**

**</div>**

**</div>**

**<div className='row'>**

**<div className='col-4 offset-2'>**

**<input**

**type="email"**

**name="email"**

**value={values.email}**

**onChange={handleChange}**

**placeholder="Email"**

**/>**

**</div>**

**<div className='col-6'>**

**{errors.email && <span style={{"color":"red"}}>{errors.email}</span>}**

**</div>**

**</div>**

**<div className='row'>**

**<div className='col-4 offset-2'>**

**<input**

**type="password"**

**name="password"**

**value={values.password}**

**onChange={handleChange}**

**placeholder="Password"**

**/>**

**</div>**

**<div className='col-6'>**

**{errors.password && <span style={{"color":"red"}}>{errors.password}</span>}**

**</div>**

**</div>**

**<div className='row'>**

**<div className='col-4 offset-2'>**

**<button type="submit" disabled={isSubmitting} className='btn btn-success'>**

**{isSubmitting ? 'Submitting...' : 'Submit'}**

**</button>**

**</div>**

**</div>**

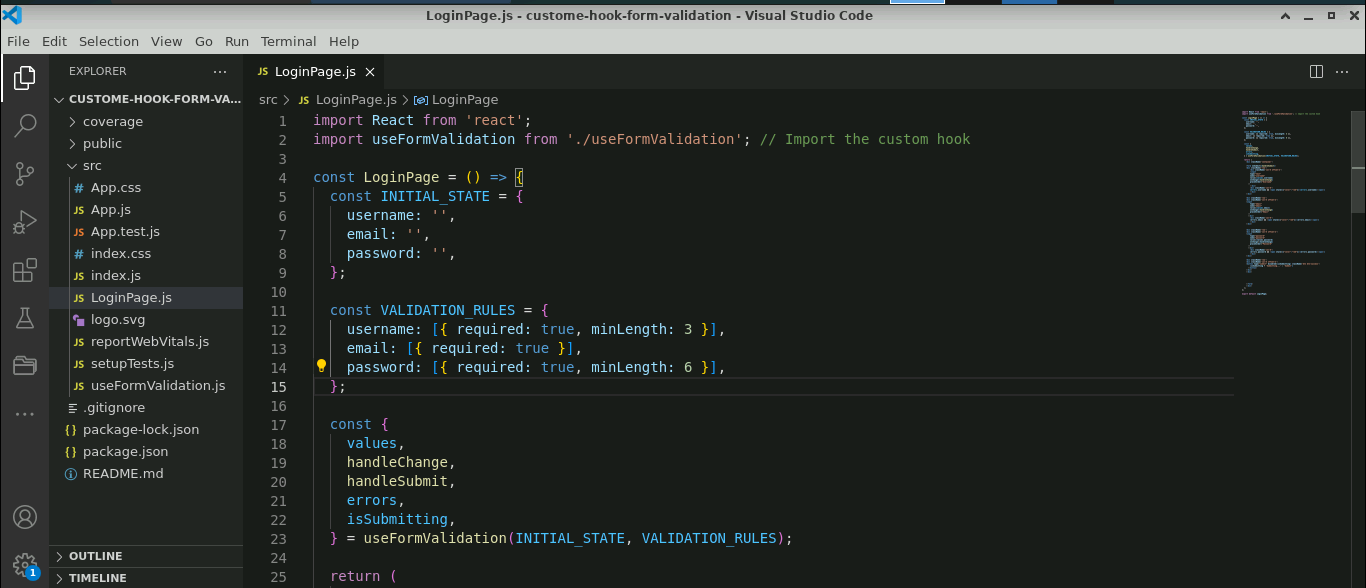
**</form>**

**</div>**

**);**

**};**

**export default LoginPage;**



**Step 4:** **Import user-defined functional component in the App.js file**

1. Inside the **App.js** file, import the user-defined component **LoginPage**

**import logo from './logo.svg';**

**import './App.css';**

**import LoginPage from './LoginPage';**

**function App() {**

**return (**

**<div className='container'>**

**<h3>Custom Hook for Form Validation</h3>**

**<LoginPage></LoginPage>**

**</div>**

**);**

**}**

**export default App;**

A screenshot of a computer

Description automatically generated

**Step 5: Add Bootstrap features in the index.html file**

1. Open the terminal and run the below command to execute the application:

**npm start**



1. The following output will appear; then, right-click on the screen and select **Inspect**

A screenshot of a computer

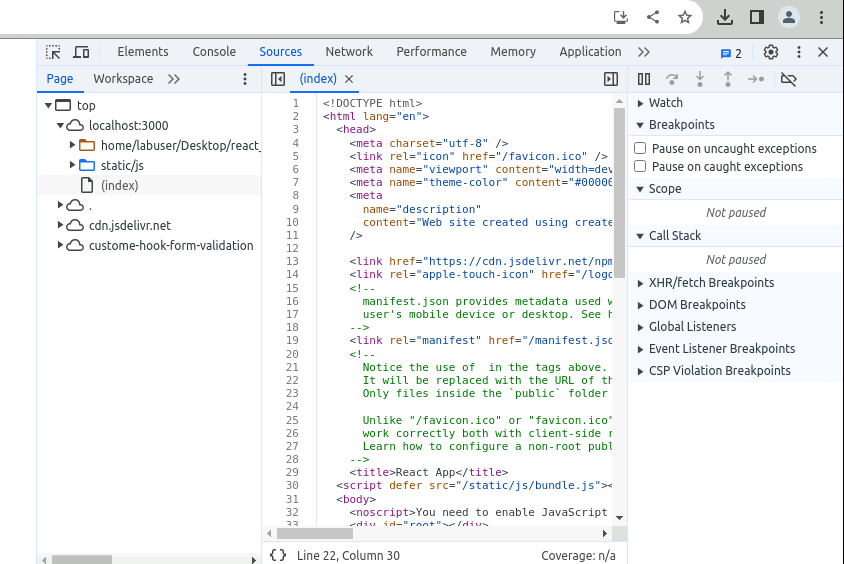
Description automatically generated

1. Navigate to **Sources** and choose (**index**) from the **Page** tab

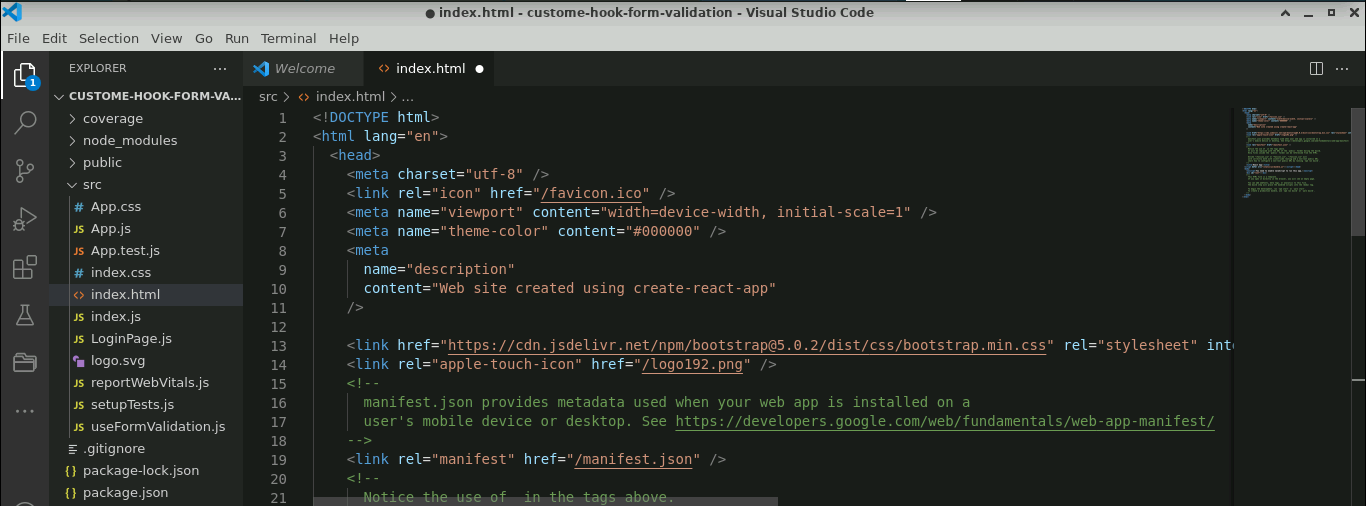
A screenshot of a computer

Description automatically generated

The following screenshot shows the **index** file:



1. Inside the **src** folder, create an **index.html** file and copy and paste the **index** file under the **index.html** file



**Step 6: Run the application**

* 1. Run the **npm start** command to get the final output

**A screen shot of a computer

Description automatically generated**

The output appears as shown below:

A screenshot of a computer

Description automatically generated

* 1. Enter the **Username**, **Email,** and **Password** under the **Custom Hook for Form Validation** and click on the **Submit** button

A screenshot of a computer

Description automatically generated

**Note**: The **Username** must be at least 3 characters, the **Password** must be at least 6 characters, andthe **Email** is mandatory. You can see the minimum length validation.

**A screenshot of a computer

Description automatically generated**

* 1. Now, right-click on the screen and select **Inspect**

A screenshot of a computer

Description automatically generated

* 1. Navigate to **Console** and click on **Form data: Object;** the **email**, **password**,and **username** of the user will appear

A screenshot of a computer

Description automatically generated

With this, you have successfully created a form validation using a custom Hook for emphasizing modularity, reusability, and enhanced user experience through Bootstrap styling.