# **[PropTypes](https://drive.google.com/uc?id=1lu9pqa3FEl-pMBg_MEeWN_E-B2fmVhL4&export=download" \t "_blank)**

## Bài 1:



import React from "react";

import PropTypes from "prop-types";

// Component UserProfile

const UserProfile = ({ name, age }) => {

  // Kiểm tra nếu `name` không được truyền hoặc rỗng

  if (!name || typeof name !== "string") {

    return (

      <div>

        <h3>Thông Tin Người Dùng</h3>

        <p style={{ color: "red" }}>

          Tên không hợp lệ hoặc không được cung cấp!

        </p>

      </div>

    );

  }

  // Kiểm tra nếu `age` không được nhập hoặc không phải là một số hợp lệ

  if (!age) {

    return (

      <div>

        <h3>Thông Tin Người Dùng</h3>

        <p>Tên: {name}</p>

        <p style={{ color: "red" }}>Tuổi không được để trống!</p>

      </div>

    );

  } else if (isNaN(age)) {

    return (

      <div>

        <h3>Thông Tin Người Dùng</h3>

        <p>Tên: {name}</p>

        <p style={{ color: "red" }}>Tuổi phải là một số hợp lệ!</p>

      </div>

    );

  }

  return (

    <div>

      <h3>Thông Tin Người Dùng</h3>

      <p>Tên: {name}</p>

      <p>Tuổi: {age}</p>

    </div>

  );

};

// Xác định PropTypes cho UserProfile

UserProfile.propTypes = {

  name: PropTypes.string.isRequired, // 'name' phải là một chuỗi và là bắt buộc

  age: PropTypes.oneOfType([PropTypes.string, PropTypes.number]).isRequired, // 'age' có thể là chuỗi hoặc số và là bắt buộc

};

export default UserProfile;

import React from "react";

import UserProfile from "./components/UserProfile";

const App = () => {

  return (

    <div className="App">

      <h1>Ứng Dụng React</h1>

      <UserProfile name="Nguyễn Văn A" age={25} />

      <UserProfile name="" age={25} />

      <UserProfile name="Nguyễn Văn B" age="twenty five" />

      <UserProfile name="Nguyễn Văn C" age={null} />

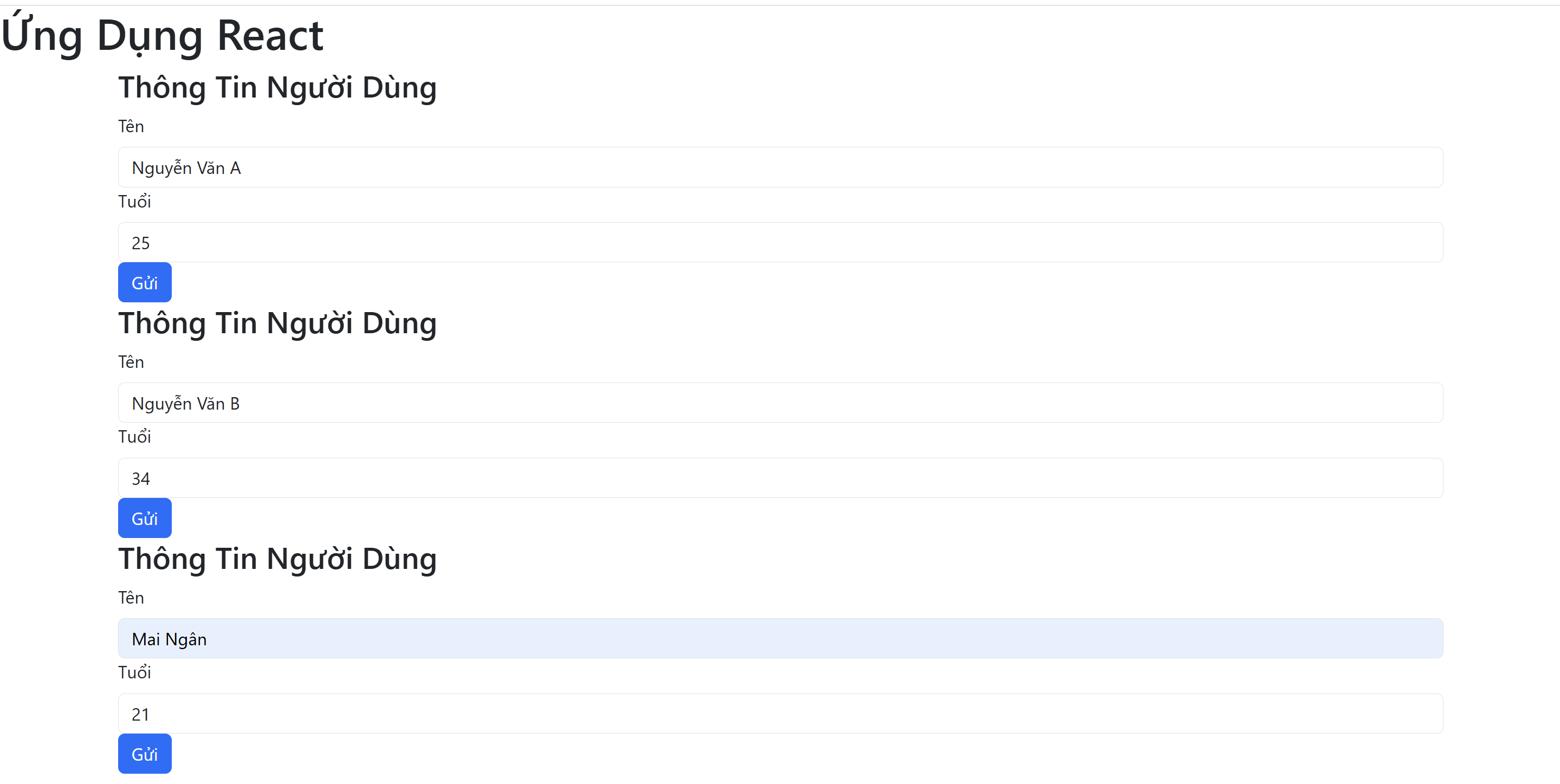
    </div>

  );

};

export default App;

## Bài 2:



import React, { useState } from "react";

import PropTypes from "prop-types";

import { Form, Button, Container } from "react-bootstrap";

// Component UserProfile

const UserProfile2 = ({ name, age, onSubmit }) => {

  const [formData, setFormData] = useState({

    name: name,

    age: age,

  });

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

  // Hàm xử lý thay đổi giá trị input

  const handleChange = (e) => {

    const { name, value } = e.target;

    setFormData({

      ...formData,

      [name]: value,

    });

  };

  // Kiểm tra lỗi dữ liệu trước khi submit

  const validateForm = () => {

    const newErrors = {};

    // Kiểm tra tuổi phải từ 18 đến 100

    const age = parseInt(formData.age, 10);

    if (!formData.name) {

      newErrors.name = "Tên là bắt buộc";

    }

    if (!formData.age) {

      newErrors.age = "Tuổi không được để trống!";

    } else if (isNaN(age)) {

      newErrors.age = "Tuổi phải là một số hợp lệ!";

    } else if (age < 18 || age > 100) {

      newErrors.age = "Tuổi phải nằm trong khoảng từ 18 đến 100!";

    }

    setErrors(newErrors);

    return Object.keys(newErrors).length === 0;

  };

  // Hàm submit form

  const handleSubmit = (e) => {

    e.preventDefault();

    if (validateForm()) {

      onSubmit(formData);

    }

  };

  return (

    <Container>

      <h3>Thông Tin Người Dùng</h3>

      <Form onSubmit={handleSubmit}>

        {/\* Trường tên \*/}

        <Form.Group controlId="formName">

          <Form.Label>Tên</Form.Label>

          <Form.Control

            type="text"

            name="name"

            value={formData.name}

            onChange={handleChange}

            isInvalid={!!errors.name}

          />

          <Form.Control.Feedback type="invalid">

            {errors.name}

          </Form.Control.Feedback>

        </Form.Group>

        {/\* Trường tuổi \*/}

        <Form.Group controlId="formAge">

          <Form.Label>Tuổi</Form.Label>

          <Form.Control

            type="text"

            name="age"

            value={formData.age}

            onChange={handleChange}

            isInvalid={!!errors.age}

          />

          <Form.Control.Feedback type="invalid">

            {errors.age}

          </Form.Control.Feedback>

        </Form.Group>

        {/\* Nút submit \*/}

        <Button variant="primary" type="submit">

          Gửi

        </Button>

      </Form>

    </Container>

  );

};

// Xác định PropTypes cho UserProfile

UserProfile2.propTypes = {

  name: PropTypes.string.isRequired, // 'name' là chuỗi bắt buộc

  age: PropTypes.oneOfType([PropTypes.string, PropTypes.number]), // 'age' có thể là chuỗi hoặc số

  onSubmit: PropTypes.func.isRequired, // Hàm onSubmit để xử lý khi người dùng submit form

};

export default UserProfile2;

import React from "react";

import UserProfile2 from "./components/UserProfile2";

const App = () => {

  const handleFormSubmit = (formData) => {

    console.log("Dữ liệu đã gửi:", formData);

  };

  return (

    <div className="App">

      <h1>Ứng Dụng React</h1>

      <UserProfile2 name="Nguyễn Văn A" age={25} onSubmit={handleFormSubmit} />

      <UserProfile2

        name="Nguyễn Văn B"

        age="twenty five"

        onSubmit={handleFormSubmit}

      />

      <UserProfile2 name="" age={30} onSubmit={handleFormSubmit} />

    </div>

  );

};

export default App;

## Bài 3:



import React, { useState, useReducer } from "react";

import { Button, Form, Container, Alert } from "react-bootstrap";

import PropTypes from "prop-types";

// Reducer để quản lý trạng thái form

const initialState = {

    name: "",

    email: "",

    password: "",

    isSubmitted: false,

};

const formReducer = (state, action) => {

    switch (action.type) {

        case "SET\_FIELD":

            return { ...state, [action.field]: action.value };

        case "SUBMIT":

            return { ...state, isSubmitted: true };

        default:

            return state;

    }

};

// Component Form

const MyForm = ({ title, onSubmit }) => {

    const [state, dispatch] = useReducer(formReducer, initialState);

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

    const [showAlert, setShowAlert] = useState(false); // Biến để kiểm soát việc hiển thị alert

    // Hàm xử lý thay đổi giá trị input

    const handleChange = (e) => {

        const { name, value } = e.target;

        dispatch({ type: "SET\_FIELD", field: name, value });

    };

    // Hàm kiểm tra lỗi trước khi submit

    const handleValidation = () => {

        const newErrors = {};

        if (!state.name) newErrors.name = "Tên không được để trống!";

        if (!state.email) newErrors.email = "Email không được để trống!";

        if (!state.password) newErrors.password = "Mật khẩu không được để trống!";

        // Nếu có lỗi, hiển thị alert

        if (Object.keys(newErrors).length > 0) {

            setShowAlert(true);

        } else {

            setShowAlert(false);

        }

        setErrors(newErrors);

        return Object.keys(newErrors).length === 0;

    };

    const handleSubmit = (e) => {

        e.preventDefault();

        if (handleValidation()) {

            dispatch({ type: "SUBMIT" });

            onSubmit(state);

        }

    };

    return (

        <Container>

            <h3>{title}</h3>

            {/\* Hiển thị Alert nếu có lỗi \*/}

            {showAlert && (

                <Alert variant="danger">

                    <strong>Lỗi:</strong> Vui lòng điền đầy đủ thông tin.

                </Alert>

            )}

            <Form onSubmit={handleSubmit}>

                <Form.Group controlId="formName">

                    <Form.Label>Tên</Form.Label>

                    <Form.Control

                        type="text"

                        name="name"

                        value={state.name}

                        onChange={handleChange}

                        isInvalid={!!errors.name}

                    />

                    <Form.Control.Feedback type="invalid">

                        {errors.name}

                    </Form.Control.Feedback>

                </Form.Group>

                <Form.Group controlId="formEmail">

                    <Form.Label>Email</Form.Label>

                    <Form.Control

                        type="email"

                        name="email"

                        value={state.email}

                        onChange={handleChange}

                        isInvalid={!!errors.email}

                    />

                    <Form.Control.Feedback type="invalid">

                        {errors.email}

                    </Form.Control.Feedback>

                </Form.Group>

                <Form.Group controlId="formPassword">

                    <Form.Label>Mật khẩu</Form.Label>

                    <Form.Control

                        type="password"

                        name="password"

                        value={state.password}

                        onChange={handleChange}

                        isInvalid={!!errors.password}

                    />

                    <Form.Control.Feedback type="invalid">

                        {errors.password}

                    </Form.Control.Feedback>

                </Form.Group>

                <Button variant="primary" type="submit">

                    Submit

                </Button>

            </Form>

        </Container>

    );

};

// Xác định PropTypes cho MyForm

MyForm.propTypes = {

    title: PropTypes.string.isRequired, // Tiêu đề phải là một chuỗi

    onSubmit: PropTypes.func.isRequired, // Hàm onSubmit phải là một function

};

export default MyForm;

import React from "react";

import MyForm from "././components/MyForm"; // Giả sử MyForm được lưu trong thư mục cùng với App.js

const App = () => {

  const handleFormSubmit = (formData) => {

    console.log("Dữ liệu đã gửi:", formData);

  };

  return (

    <div className="App">

      <h1>Ứng Dụng React</h1>

      <MyForm title="Đăng Ký Người Dùng" onSubmit={handleFormSubmit} />

    </div>

  );

};

export default App;

## Bài 4:



import React, { useState } from "react";

import PropTypes from "prop-types";

import { Form, Button, Container, Alert } from "react-bootstrap";

const AdvancedForm = ({ onSubmit, title }) => {

  const [formData, setFormData] = useState({

    name: "",

    age: "",

    email: "",

    phone: "",

    gender: "",

    agree: false,

  });

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

  const [showAlert, setShowAlert] = useState(false);

  const [success, setSuccess] = useState(false);

  const handleChange = (e) => {

    const { name, value, type, checked } = e.target;

    setFormData({

      ...formData,

      [name]: type === "checkbox" ? checked : value,

    });

  };

  const validateForm = () => {

    const newErrors = {};

    const age = parseInt(formData.age, 10);

    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

    const phoneRegex = /^\d{10,15}$/;

    if (!formData.name) newErrors.name = "Tên không được để trống!";

    else if (formData.name.length < 3 || formData.name.length > 50)

      newErrors.name = "Tên phải từ 3-50 ký tự!";

    if (!formData.age) newErrors.age = "Tuổi không được để trống!";

    else if (isNaN(age) || age < 18 || age > 100)

      newErrors.age = "Tuổi phải từ 18-100!";

    if (!formData.email) newErrors.email = "Email không được để trống!";

    else if (!emailRegex.test(formData.email))

      newErrors.email = "Email không hợp lệ!";

    if (!formData.phone) newErrors.phone = "Số điện thoại không được để trống!";

    else if (!phoneRegex.test(formData.phone))

      newErrors.phone = "Số điện thoại phải từ 10-15 chữ số!";

    if (!formData.gender) newErrors.gender = "Vui lòng chọn giới tính!";

    if (!formData.agree) newErrors.agree = "Phải đồng ý với điều khoản!";

    setErrors(newErrors);

    setShowAlert(Object.keys(newErrors).length > 0);

    return Object.keys(newErrors).length === 0;

  };

  const handleSubmit = (e) => {

    e.preventDefault();

    setSuccess(false);

    if (validateForm()) {

      onSubmit(formData);

      setSuccess(true);

      setFormData({

        name: "",

        age: "",

        email: "",

        phone: "",

        gender: "",

        agree: false,

      });

      setShowAlert(false);

    }

  };

  return (

    <Container>

      <h3>{title}</h3>

      {showAlert && (

        <Alert variant="danger">

          <strong>Lỗi:</strong> Vui lòng kiểm tra lại thông tin.

        </Alert>

      )}

      {success && (

        <Alert variant="success">

          <strong>Thành công:</strong> Đã gửi thông tin!

        </Alert>

      )}

      <Form onSubmit={handleSubmit}>

        <Form.Group controlId="formName">

          <Form.Label>Tên</Form.Label>

          <Form.Control

            type="text"

            name="name"

            value={formData.name}

            onChange={handleChange}

            isInvalid={!!errors.name}

          />

          <Form.Control.Feedback type="invalid">

            {errors.name}

          </Form.Control.Feedback>

        </Form.Group>

        <Form.Group controlId="formAge">

          <Form.Label>Tuổi</Form.Label>

          <Form.Control

            type="text"

            name="age"

            value={formData.age}

            onChange={handleChange}

            isInvalid={!!errors.age}

          />

          <Form.Control.Feedback type="invalid">

            {errors.age}

          </Form.Control.Feedback>

        </Form.Group>

        <Form.Group controlId="formEmail">

          <Form.Label>Email</Form.Label>

          <Form.Control

            type="email"

            name="email"

            value={formData.email}

            onChange={handleChange}

            isInvalid={!!errors.email}

          />

          <Form.Control.Feedback type="invalid">

            {errors.email}

          </Form.Control.Feedback>

        </Form.Group>

        <Form.Group controlId="formPhone">

          <Form.Label>Số điện thoại</Form.Label>

          <Form.Control

            type="text"

            name="phone"

            value={formData.phone}

            onChange={handleChange}

            isInvalid={!!errors.phone}

          />

          <Form.Control.Feedback type="invalid">

            {errors.phone}

          </Form.Control.Feedback>

        </Form.Group>

        <Form.Group controlId="formGender">

          <Form.Label>Giới tính</Form.Label>

          <Form.Control

            as="select"

            name="gender"

            value={formData.gender}

            onChange={handleChange}

            isInvalid={!!errors.gender}

          >

            <option value="">Chọn giới tính</option>

            <option value="male">Nam</option>

            <option value="female">Nữ</option>

            <option value="other">Khác</option>

          </Form.Control>

          <Form.Control.Feedback type="invalid">

            {errors.gender}

          </Form.Control.Feedback>

        </Form.Group>

        <Form.Group controlId="formAgree">

          <Form.Check

            type="checkbox"

            name="agree"

            label="Đồng ý với điều khoản"

            checked={formData.agree}

            onChange={handleChange}

            isInvalid={!!errors.agree}

          />

          <Form.Control.Feedback type="invalid">

            {errors.agree}

          </Form.Control.Feedback>

        </Form.Group>

        <Button variant="primary" type="submit">

          Gửi

        </Button>

      </Form>

    </Container>

  );

};

AdvancedForm.propTypes = {

  onSubmit: PropTypes.func.isRequired,

  title: PropTypes.string.isRequired,

};

export default AdvancedForm;

import React from "react";

import AdvancedForm from "./components/AdvancedForm";

const App = () => {

  const handleFormSubmit = (formData) => {

    console.log("Dữ liệu đã gửi:", formData);

  };

  return (

    <div className="App">

      <h1>Ứng Dụng React</h1>

      <AdvancedForm onSubmit={handleFormSubmit} />

    </div>

  );

};

export default App;