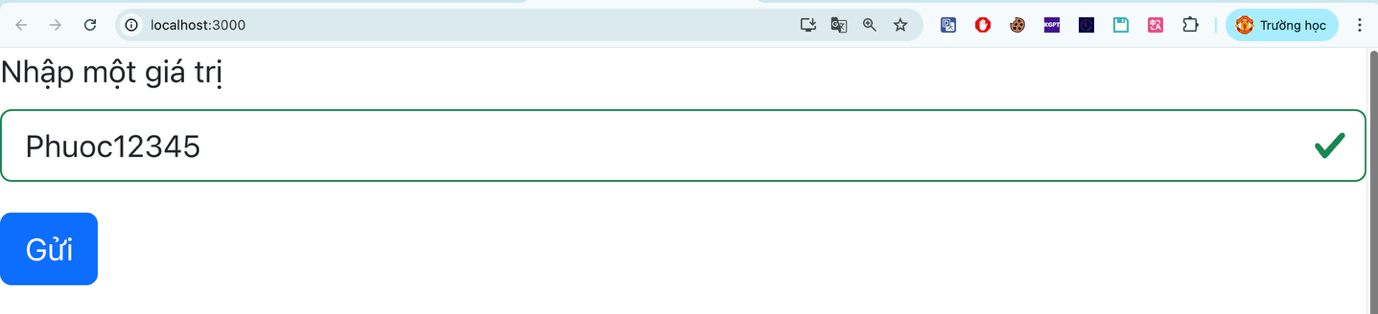
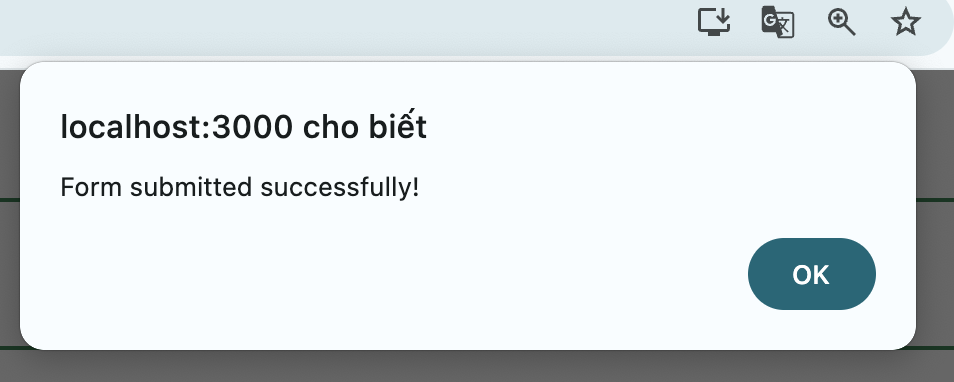
Bài 5:





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

import 'bootstrap/dist/css/bootstrap.min.css';

import { Form, Button } from "react-bootstrap"; // ĐẢM BẢO CÓ DÒNG NÀY

const validatedInput = (value) => {

return value.length >= 5;

};

function ValidatedInput() {

const [value, setValue] = useState("");

const [isValid , setIsValid] = useState(true);

const [errorMessage, setErrorMessage] = useState("");

const [touched , setTouched] = useState(false);

// userEffect để thực hiện xác thực mỗi khi gía trị đầu vào thay đổi

useEffect(() => {

if (touched) {

const isValidInput = validatedInput(value);

setIsValid(isValidInput);

if (!isValidInput) {

setErrorMessage("Giá trị phải có ít nhất 5 ký tự");

} else {

setErrorMessage("");

}

}

}, [value , touched]);

const handleBlur = () => {

setTouched(true);

};

const handleSubmit = (e) => {

e.preventDefault();

if (isValid && value.length >= 5) {

alert("Form submitted successfully!");

} else {

alert("Please correct the errors before submitting.");

}

}

return (

<Form onSubmit={handleSubmit}>

<Form.Group controlId="validatedInput">

<Form.Label>Nhập một giá trị</Form.Label>

<Form.Control

type="text"

value={value}

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

onBlur={handleBlur}

isValid={touched && isValid && value.length >= 5}

isInvalid={touched && !isValid}

/>

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

{errorMessage}

</Form.Control.Feedback>

</Form.Group>

<Button

variant="primary"

type="submit"

disabled={!isValid || value.length < 5}

className="mt-3"

>

Gửi

</Button>

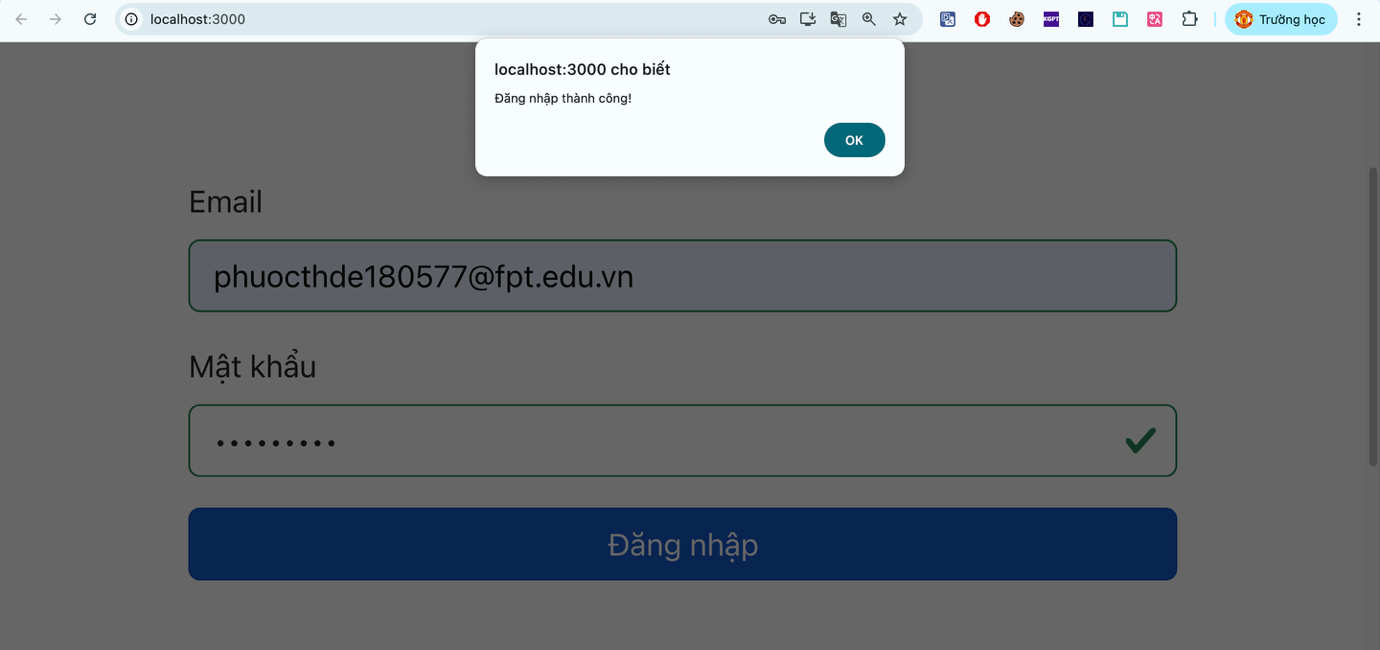
</Form>

)

}

export default ValidatedInput;

Bài 5:



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

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

import 'bootstrap/dist/css/bootstrap.min.css';

// Hàm xác thực email

const validateEmail = (email) => {

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

return emailRegex.test(email);

};

// Hàm xác thực password

const validatePassword = (password) => {

return password.length >= 8;

};

function EmailPasswordForm() {

const [email, setEmail] = useState("");

const [password, setPassword] = useState("");

const [emailValid, setEmailValid] = useState(true);

const [passwordValid, setPasswordValid] = useState(true);

const [emailTouched, setEmailTouched] = useState(false);

const [passwordTouched, setPasswordTouched] = useState(false);

const [emailError, setEmailError] = useState("");

const [passwordError, setPasswordError] = useState("");

// useEffect cho email validation

useEffect(() => {

if (emailTouched) {

const isValid = validateEmail(email);

setEmailValid(isValid);

if (!isValid && email.length > 0) {

setEmailError("Vui lòng nhập một địa chỉ email hợp lệ!");

} else if (email.length === 0) {

setEmailError("Email không được để trống!");

} else {

setEmailError("");

}

}

}, [email, emailTouched]);

// useEffect cho password validation

useEffect(() => {

if (passwordTouched) {

const isValid = validatePassword(password);

setPasswordValid(isValid);

if (!isValid && password.length > 0) {

setPasswordError("Mật khẩu phải có ít nhất 8 ký tự!");

} else if (password.length === 0) {

setPasswordError("Mật khẩu không được để trống!");

} else {

setPasswordError("");

}

}

}, [password, passwordTouched]);

const handleEmailBlur = () => {

setEmailTouched(true);

};

const handlePasswordBlur = () => {

setPasswordTouched(true);

};

const handleSubmit = (e) => {

e.preventDefault();

if (emailValid && passwordValid && email && password) {

alert("Đăng nhập thành công!");

console.log("Email:", email);

console.log("Password:", password);

}

};

const isFormValid = emailValid && passwordValid && email && password;

return (

<Container className="mt-5">

<Row className="justify-content-center">

<Col md={6}>

<h2 className="text-center mb-4">Đăng nhập</h2>

<Form onSubmit={handleSubmit}>

<Form.Group controlId="email" className="mb-3">

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

<Form.Control

type="email"

value={email}

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

onBlur={handleEmailBlur}

isValid={emailTouched && emailValid && email}

isInvalid={emailTouched && (!emailValid || !email)}

placeholder="Nhập email của bạn"

/>

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

{emailError}

</Form.Control.Feedback>

</Form.Group>

<Form.Group controlId="password" className="mb-3">

<Form.Label>Mật khẩu</Form.Label>

<Form.Control

type="password"

value={password}

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

onBlur={handlePasswordBlur}

isValid={passwordTouched && passwordValid && password}

isInvalid={passwordTouched && (!passwordValid || !password)}

placeholder="Nhập mật khẩu của bạn"

/>

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

{passwordError}

</Form.Control.Feedback>

</Form.Group>

<Button

variant="primary"

type="submit"

disabled={!isFormValid}

className="w-100"

>

Đăng nhập

</Button>

</Form>

</Col>

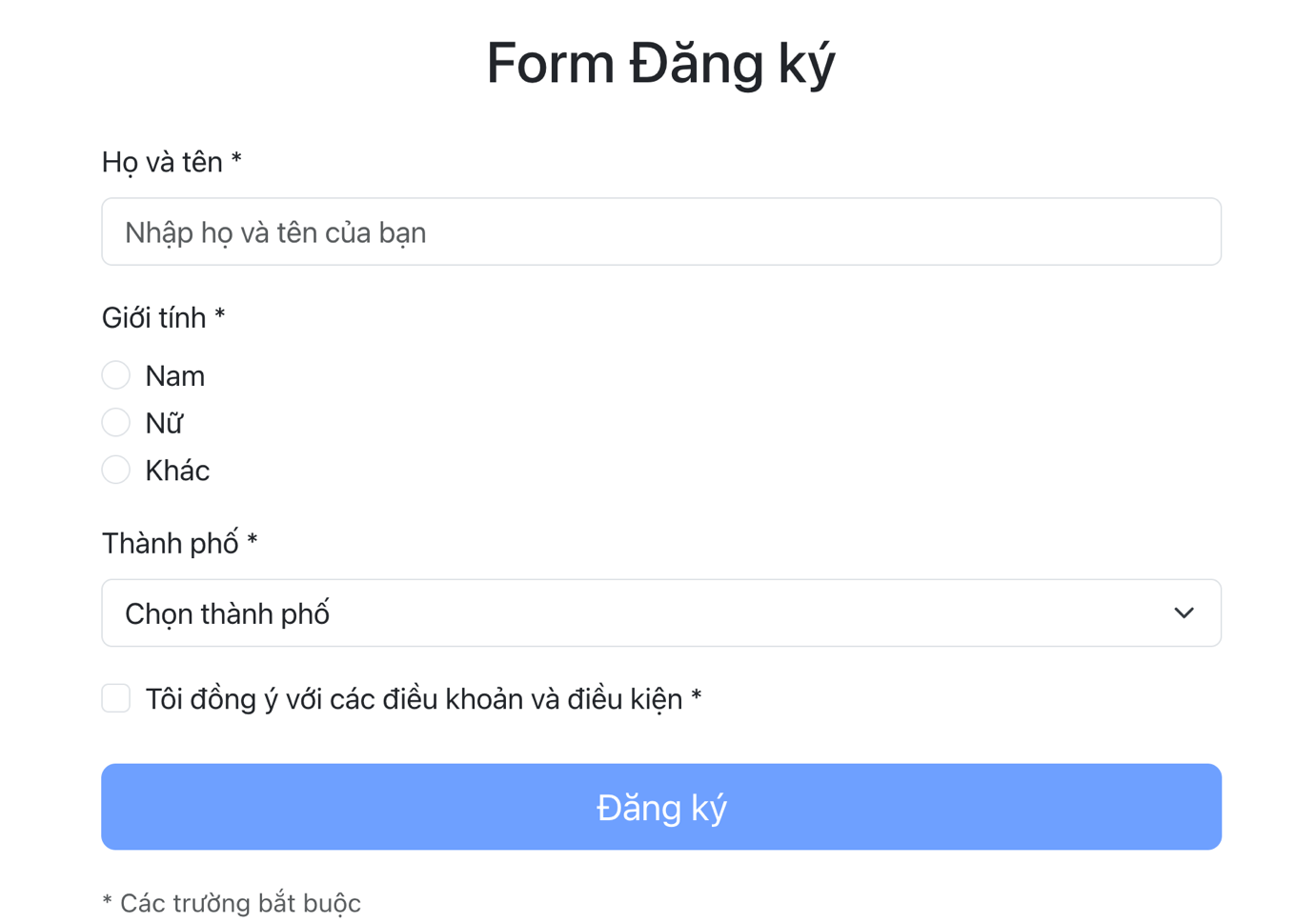
</Row>

</Container>

);

}

export default EmailPasswordForm;

Bài 6



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

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

import 'bootstrap/dist/css/bootstrap.min.css';

function CompleteForm() {

// States cho các trường input

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

const [gender, setGender] = useState("");

const [city, setCity] = useState("");

const [agreeTerms, setAgreeTerms] = useState(false);

// States cho validation

const [nameValid, setNameValid] = useState(true);

const [genderValid, setGenderValid] = useState(true);

const [cityValid, setCityValid] = useState(true);

const [termsValid, setTermsValid] = useState(true);

// States cho touched

const [nameTouched, setNameTouched] = useState(false);

const [genderTouched, setGenderTouched] = useState(false);

const [cityTouched, setCityTouched] = useState(false);

const [termsTouched, setTermsTouched] = useState(false);

// States cho error messages

const [nameError, setNameError] = useState("");

const [genderError, setGenderError] = useState("");

const [cityError, setCityError] = useState("");

const [termsError, setTermsError] = useState("");

// State cho form submission

const [showSuccess, setShowSuccess] = useState(false);

// Validation functions

const validateName = (value) => {

return value.trim().length >= 2;

};

const validateGender = (value) => {

return value !== "";

};

const validateCity = (value) => {

return value !== "";

};

const validateTerms = (value) => {

return value === true;

};

// useEffect cho name validation

useEffect(() => {

if (nameTouched) {

const isValid = validateName(name);

setNameValid(isValid);

if (!isValid) {

setNameError("Tên phải có ít nhất 2 ký tự!");

} else {

setNameError("");

}

}

}, [name, nameTouched]);

// useEffect cho gender validation

useEffect(() => {

if (genderTouched) {

const isValid = validateGender(gender);

setGenderValid(isValid);

if (!isValid) {

setGenderError("Vui lòng chọn giới tính!");

} else {

setGenderError("");

}

}

}, [gender, genderTouched]);

// useEffect cho city validation

useEffect(() => {

if (cityTouched) {

const isValid = validateCity(city);

setCityValid(isValid);

if (!isValid) {

setCityError("Vui lòng chọn thành phố!");

} else {

setCityError("");

}

}

}, [city, cityTouched]);

// useEffect cho terms validation

useEffect(() => {

if (termsTouched) {

const isValid = validateTerms(agreeTerms);

setTermsValid(isValid);

if (!isValid) {

setTermsError("Bạn phải đồng ý với điều khoản!");

} else {

setTermsError("");

}

}

}, [agreeTerms, termsTouched]);

// Handle blur events

const handleNameBlur = () => setNameTouched(true);

const handleGenderBlur = () => setGenderTouched(true);

const handleCityBlur = () => setCityTouched(true);

const handleTermsBlur = () => setTermsTouched(true);

// Check if form is valid

const isFormValid = nameValid && genderValid && cityValid && termsValid &&

name && gender && city && agreeTerms;

const handleSubmit = (e) => {

e.preventDefault();

// Mark all fields as touched

setNameTouched(true);

setGenderTouched(true);

setCityTouched(true);

setTermsTouched(true);

if (isFormValid) {

setShowSuccess(true);

console.log("Form Data:", {

name,

gender,

city,

agreeTerms

});

// Reset form after 3 seconds

setTimeout(() => {

setShowSuccess(false);

setName("");

setGender("");

setCity("");

setAgreeTerms(false);

setNameTouched(false);

setGenderTouched(false);

setCityTouched(false);

setTermsTouched(false);

}, 3000);

}

};

return (

<Container className="mt-5">

<Row className="justify-content-center">

<Col md={8}>

<h2 className="text-center mb-4">Form Đăng ký</h2>

{showSuccess && (

<Alert variant="success" className="mb-4">

<Alert.Heading>Thành công!</Alert.Heading>

<p>Form đã được gửi thành công. Cảm ơn bạn đã đăng ký!</p>

</Alert>

)}

<Form onSubmit={handleSubmit}>

{/\* Textbox - Name \*/}

<Form.Group controlId="name" className="mb-3">

<Form.Label>Họ và tên \*</Form.Label>

<Form.Control

type="text"

value={name}

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

onBlur={handleNameBlur}

isValid={nameTouched && nameValid && name}

isInvalid={nameTouched && (!nameValid || !name)}

placeholder="Nhập họ và tên của bạn"

/>

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

{nameError}

</Form.Control.Feedback>

</Form.Group>

{/\* Radio Button - Gender \*/}

<Form.Group className="mb-3">

<Form.Label>Giới tính \*</Form.Label>

<div onBlur={handleGenderBlur}>

<Form.Check

type="radio"

id="male"

name="gender"

label="Nam"

value="male"

checked={gender === "male"}

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

isInvalid={genderTouched && !genderValid}

/>

<Form.Check

type="radio"

id="female"

name="gender"

label="Nữ"

value="female"

checked={gender === "female"}

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

isInvalid={genderTouched && !genderValid}

/>

<Form.Check

type="radio"

id="other"

name="gender"

label="Khác"

value="other"

checked={gender === "other"}

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

isInvalid={genderTouched && !genderValid}

/>

</div>

{genderTouched && !genderValid && (

<div className="invalid-feedback d-block">

{genderError}

</div>

)}

</Form.Group>

{/\* Dropdown - City \*/}

<Form.Group controlId="city" className="mb-3">

<Form.Label>Thành phố \*</Form.Label>

<Form.Select

value={city}

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

onBlur={handleCityBlur}

isValid={cityTouched && cityValid && city}

isInvalid={cityTouched && (!cityValid || !city)}

>

<option value="">Chọn thành phố</option>

<option value="hanoi">Hà Nội</option>

<option value="hcm">Hồ Chí Minh</option>

<option value="danang">Đà Nẵng</option>

<option value="haiphong">Hải Phòng</option>

<option value="cantho">Cần Thơ</option>

</Form.Select>

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

{cityError}

</Form.Control.Feedback>

</Form.Group>

{/\* Checkbox - Terms \*/}

<Form.Group className="mb-4">

<Form.Check

type="checkbox"

id="agreeTerms"

label="Tôi đồng ý với các điều khoản và điều kiện \*"

checked={agreeTerms}

onChange={(e) => setAgreeTerms(e.target.checked)}

onBlur={handleTermsBlur}

isInvalid={termsTouched && !termsValid}

/>

{termsTouched && !termsValid && (

<div className="invalid-feedback d-block">

{termsError}

</div>

)}

</Form.Group>

{/\* Submit Button \*/}

<Button

variant="primary"

type="submit"

disabled={!isFormValid}

className="w-100"

size="lg"

>

Đăng ký

</Button>

</Form>

<div className="mt-3 text-muted">

<small>\* Các trường bắt buộc</small>

</div>

</Col>

</Row>

</Container>

);

}

export default CompleteForm;