**Task: Order Billing Form with React.js and Tailwind CSS**

**Task Description**

You are required to create a fully functional and responsive **Order Billing Form** using **React.js** and **Tailwind CSS**. The design must match the attached image closely, but you are encouraged to add your unique styling or interactive elements.

**Key Requirements**

1. **Responsive Design:**
   * The form should look good on all screen sizes (mobile, tablet, desktop).
2. **Input Validation:**
   * All required fields must be validated before the form is submitted.
   * Display error messages for invalid or missing inputs.
3. **Custom Features:**
   * Encouraged to add their own touches, such as hover effects, animations, or additional interactive elements.

**Design Details**

Refer to the attached image for the design layout. The form includes:

**Basic Information**

* **Email** (required, validate for proper email format)
* **First Name** (required)
* **Last Name** (required)
* **Full Name** (auto-generated by combining First Name and Last Name; read-only)
* **Status** (default value: "Pending")
* **Tracking** (auto-generated unique value; read-only)

**Address Information**

* **Address** (required)
* **Address 2** (optional)
* **City** (required)
* **State** (required)
* **Zip Code** (required, validate for proper length)
* **County** (required)

**Steps to Complete the Task**

**1. Set Up the Project**

* Create a new React project using Vite:

npm create vite@latest order-billing --template react

* Install **Tailwind CSS**:

npm install -D tailwindcss postcss autoprefixer

npx tailwindcss init

* Configure Tailwind CSS in your tailwind.config.js and add the required styles in index.css.

@tailwind base;

@tailwind components;

@tailwind utilities;

**2. Create the Form Component**

* Create a React component named OrderBillingForm.jsx.
* Use Tailwind CSS classes to style the form to match the design.

Example:

import React, { useState } from 'react';

const OrderBillingForm = () => {

const [formData, setFormData] = useState({

email: '',

firstName: '',

lastName: '',

fullName: '',

status: 'Pending',

tracking: `TRACK-${Date.now()}`,

address: '',

address2: '',

city: '',

state: '',

zip: '',

county: ''

});

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

const handleChange = (e) => {

const { name, value } = e.target;

setFormData({ ...formData, [name]: value });

if (name === 'firstName' || name === 'lastName') {

setFormData((prev) => ({

...prev,

fullName: `${prev.firstName} ${prev.lastName}`

}));

}

};

const validate = () => {

const newErrors = {};

if (!formData.email || !/^[\w-.]+@([\w-]+\.)+[\w-]{2,4}$/.test(formData.email)) {

newErrors.email = 'Please enter a valid email address';

}

if (!formData.firstName) newErrors.firstName = 'First Name is required';

if (!formData.lastName) newErrors.lastName = 'Last Name is required';

if (!formData.address) newErrors.address = 'Address is required';

if (!formData.city) newErrors.city = 'City is required';

if (!formData.state) newErrors.state = 'State is required';

if (!formData.zip || formData.zip.length !== 5) {

newErrors.zip = 'Zip Code must be 5 digits';

}

if (!formData.county) newErrors.county = 'County is required';

setErrors(newErrors);

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

};

const handleSubmit = (e) => {

e.preventDefault();

if (validate()) {

alert('Form submitted successfully!');

}

};

return (

<form className="max-w-4xl mx-auto p-6 bg-white shadow-md rounded-lg" onSubmit={handleSubmit}>

<h2 className="text-xl font-semibold mb-4">Order Billing</h2>

{/\* Basic Information \*/}

<div className="grid grid-cols-1 md:grid-cols-2 gap-4">

<div>

<label className="block text-sm font-medium">Email \*</label>

<input

type="email"

name="email"

className="w-full mt-1 p-2 border border-gray-300 rounded-md"

value={formData.email}

onChange={handleChange}

/>

{errors.email && <p className="text-red-500 text-sm">{errors.email}</p>}

</div>

{/\* Add more fields similarly \*/}

</div>

<button

type="submit"

className="mt-4 bg-blue-500 text-white py-2 px-4 rounded hover:bg-blue-600">

Order Now

</button>

</form>

);

};

export default OrderBillingForm;

**3. Make the Design Responsive**

* Use Tailwind's responsive utilities (e.g., grid-cols-1 md:grid-cols-2) to ensure the layout adapts to different screen sizes.
* Test the form on mobile, tablet, and desktop resolutions.

**4. Add Validation**

* Implement client-side validation for all required fields as shown in the example above.
* Display user-friendly error messages below invalid fields.

**5. Add Your Unique Touch**

* Customize the design with animations, hover effects, or icons.
* Add your unique color scheme or style to make the form more engaging.

**Bonus Challenge**

* Add a confirmation page after form submission.
* Integrate with a mock API to save the form data.
* Implement dark mode support.

*Attached Image:*

