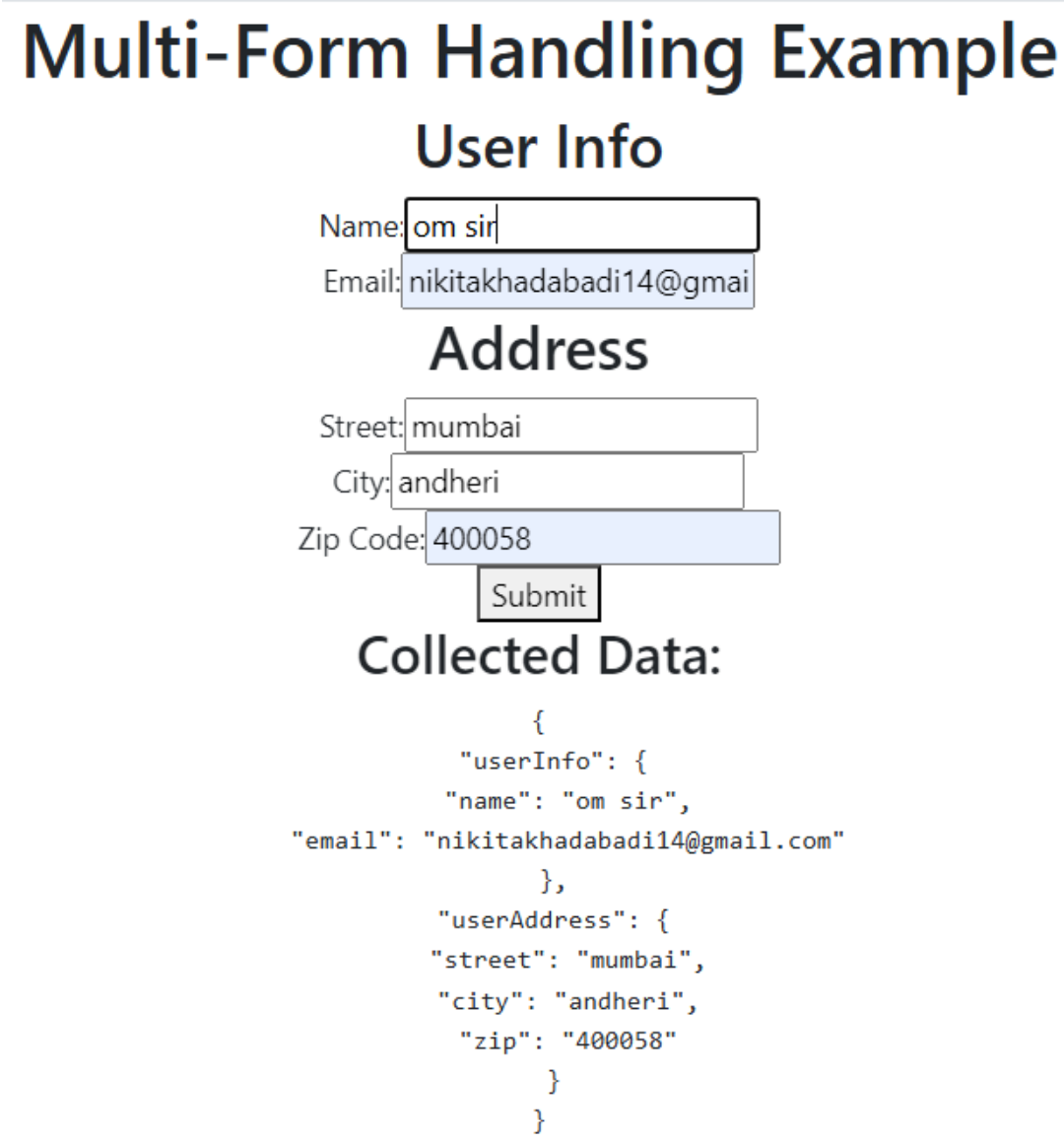

To handle multiple forms in React and collect their data, you can create a structure where each form has its own state and its own handler functions. Below is an example of how you can handle multiple forms in a React component and collect form data for each.

Example: Handling Multiple Forms in React

Let's say you have two forms: one to collect the user's name and email, and another one to collect the user's address.

Step 1: Create the React Component

In this example, we'll use the `useState` hook to manage the state of each form.

MultiForm.js file code:-

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

function MultiForm() {
  // State for form 1 (User Info)
  const [userInfo, setUserInfo] = useState({
    name: '',
    email: '',
  });

  // State for form 2 (User Address)
  const [userAddress, setUserAddress] = useState({
    street: '',
    city: '',
    zip: '',
  });

  // Handler for user info form
  const handleUserInfoChange = (e) => {
    const { name, value } = e.target;
    setUserInfo((prevUserInfo) => ({
      ...prevUserInfo,
      [name]: value,
    }));
  };

  // Handler for user address form
  const handleUserAddressChange = (e) => {
    const { name, value } = e.target;
    setUserAddress((prevUserAddress) => ({
      ...prevUserAddress,
      [name]: value,
    }));
  };

  // Handle form submission
  const handleSubmit = (e) => {
    e.preventDefault();
    // Log or submit the collected data here
    console.log('User Info:', userInfo);
```

```
  console.log('User Address:', userAddress);
};

return (
  <div>
    <h1>Multi-Form Handling Example</h1>

    {/* User Info Form */}
    <form onSubmit={handleSubmit}>
      <h2>User Info</h2>
      <div>
        <label>
          Name:
          <input
            type="text"
            name="name"
            value={userInfo.name}
            onChange={handleUserInfoChange}
            placeholder="Enter your name"
          />
        </label>
      </div>
      <div>
        <label>
          Email:
          <input
            type="email"
            name="email"
            value={userInfo.email}
            onChange={handleUserInfoChange}
            placeholder="Enter your email"
          />
        </label>
      </div>

      {/* User Address Form */}
      <h2>Address</h2>
      <div>
        <label>
          Street:
          <input
            type="text"
            name="street"
            value={userAddress.street}
            onChange={handleUserAddressChange}
```

```
              placeholder="Enter your street"
            />
          </label>
        </div>
        <div>
          <label>
            City:
            <input
              type="text"
              name="city"
              value={userAddress.city}
              onChange={handleUserAddressChange}
              placeholder="Enter your city"
            />
          </label>
        </div>
        <div>
          <label>
            Zip Code:
            <input
              type="text"
              name="zip"
              value={userAddress.zip}
              onChange={handleUserAddressChange}
              placeholder="Enter your zip code"
            />
          </label>
        </div>

        <button type="submit">Submit</button>
      </form>

      <div>
        <h3>Collected Data:</h3>
        <pre>{JSON.stringify({ userInfo, userAddress }, null, 2)}</pre>
      </div>
    </div>
  );
}

export default MultiForm;
```

Step 2: Explanation

1. **State Setup:**
   - We use two state variables, `userInfo` and `userAddress`, to hold the data for each form. Each state is initialized as an object with keys corresponding to the form fields.
2. **Change Handlers:**
   - `handleUserInfoChange` and `handleUserAddressChange` are the functions responsible for updating the state when the user types in any of the form fields. The `name` attribute of each input field is used to update the correct key in the respective state object.
3. **Form Submission:**
   - The `handleSubmit` function is called when the form is submitted. It prevents the default form behavior and logs the collected data from both forms.
4. **Form Structure:**
   - The form is split into two sections: one for "User Info" (name, email) and another for "Address" (street, city, zip).
   - Each input field is bound to its respective state via the `value` attribute, and changes are handled by the corresponding `onChange` event handler.
5. **Displaying Collected Data:**
   - After the form is submitted, the collected form data is displayed in a `pre` tag to format the data nicely.

Step 3: Use the Component in `App.js`

Replace the content of `src/App.js` with the following to use the `MultiForm` component:

Jsx code:-

```
import React from 'react';
import MultiForm from './MultiForm';

function App() {
  return (
    <div className="App">
      <MultiForm />
    </div>
  );
}

export default App;
```

Step 4: Styling (Optional)

You can add some basic styling to make the form look cleaner, for example, in `App.css`:

Css code:-

```
div {
  font-family: Arial, sans-serif;
  text-align: center;
  margin-top: 20px;
}

input {
  padding: 8px;
  margin: 10px;
  width: 250px;
  display: block;
  margin-left: auto;
  margin-right: auto;
}

button {
  padding: 10px 15px;
  background-color: #4CAF50;
  color: white;
  border: none;
  cursor: pointer;
  display: block;
  margin: 20px auto;
}

button:hover {
  background-color: #45a049;
}

h1, h2, h3 {
  color: #333;
}
```

Step 5: Running the Application

You can run the app by executing the following command in the terminal:

```
Cmd (terminal):-
npm start
```

Result:

1. **Multiple Forms**: You now have two forms — one for user information and one for the address.
2. **Data Collection**: As the user fills out the forms and clicks "Submit," their information is collected and logged in the browser's console.
3. **Dynamic Form Handling**: Each form can have its own logic, and you can easily add more forms or data fields as needed.

This approach allows you to handle multiple forms and easily manage state in a modular way.