Alright ✅ this is the **final exercise** of your Chapter 15 —
We'll build a **Signup Form** step-by-step with **error handling** and all features mentioned in your list.

I'll give you **two versions**:

1. **Beginner-friendly manual validation** with useState
2. **Bonus version** with **Formik + Yup** for cleaner code

📝 Version 1 — Manual Validation with useState

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


function SignupForm() {
  // 🧠 State for form fields
  const [name, setName] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  // 🧠 State for error messages
  const [errors, setErrors] = useState({});

  // 🖱️ Handle form submission
  const handleSubmit = (e) => {
    e.preventDefault(); // 🚫 Stop page reload
    let newErrors = {};

    // 🔍 Validation rules
    if (!name.trim()) {
      newErrors.name = "Name is required";
    }
    if (!email.includes("@")) {
```

```
    newErrors.email = "Invalid email";
  }
  if (password.length < 6) {
    newErrors.password = "Password must be at least 6 characters";
  }

  // ⚠ If errors exist, show them
  if (Object.keys(newErrors).length > 0) {
    setErrors(newErrors);
  } else {
    // ✅ No errors → log values
    console.log({ name, email, password });

    // Clear errors and form fields
    setErrors({});
    setName("");
    setEmail("");
    setPassword("");
  }
};

return (
  <div className="max-w-md mx-auto mt-10 bg-white p-6 rounded shadow">
    <h2 className="text-xl font-bold mb-4">Signup Form</h2>
    <form onSubmit={handleSubmit} className="space-y-4">

      {/* Name Field */}
      <input
```

```
  type="text"
  placeholder="Enter your name"
  value={name}
  onChange={(e) => setName(e.target.value)}
  className="w-full p-2 border rounded"
/>
{errors.name && <p className="text-red-500 text-sm">{errors.name}</p>}

{/* Email Field */}
<input
  type="email"
  placeholder="Enter your email"
  value={email}
  onChange={(e) => setEmail(e.target.value)}
  className="w-full p-2 border rounded"
/>
{errors.email && <p className="text-red-500 text-sm">{errors.email}</p>}

{/* Password Field */}
<input
  type="password"
  placeholder="Enter your password"
  value={password}
  onChange={(e) => setPassword(e.target.value)}
  className="w-full p-2 border rounded"
/>
{errors.password && (
  <p className="text-red-500 text-sm">{errors.password}</p>
```

```
      )}

      {/* Submit Button */}
      <button
        type="submit"
        className="w-full bg-blue-500 text-white py-2 rounded hover:bg-blue-600"
      >
        Submit
      </button>
    </form>
  </div>
 );
}

export default SignupForm;
```

🔍 **How it works:**

1. **3 useState hooks** → store name, email, password.
2. **1 useState for errors** → stores messages for each field.
3. On **Submit**:
    - Validate each field.
    - If invalid → store error in errors state.
    - If valid → print values & clear form.
4. Error messages display **below each field**.

🎁 **Version 2 — Bonus: Using Formik + Yup**

```
import React from "react";
import { Formik } from "formik";
```

```
import * as Yup from "yup";


function SignupFormFormik() {
  return (
    <Formik
      initialValues={{ name: "", email: "", password: "" }}
      validationSchema={Yup.object({
        name: Yup.string().required("Name is required"),
        email: Yup.string()
          .email("Invalid email format")
          .required("Email is required"),
        password: Yup.string()
          .min(6, "Password must be at least 6 characters")
          .required("Password is required"),
      })}
      onSubmit={(values, { resetForm }) => {
        console.log(values);
        resetForm(); // ✅ Clears form
      }}
    >
      {({ handleChange, handleSubmit, values, errors }) => (
        <form
          onSubmit={handleSubmit}
          className="max-w-md mx-auto mt-10 bg-white p-6 rounded shadow space-y-4"
        >
          <h2 className="text-xl font-bold mb-4">Signup Form</h2>


          {/* Name */}
```

```
<input
  name="name"
  placeholder="Enter your name"
  onChange={handleChange}
  value={values.name}
  className="w-full p-2 border rounded"
/>
{errors.name && <p className="text-red-500 text-sm">{errors.name}</p>}

{/* Email */}
<input
  name="email"
  placeholder="Enter your email"
  onChange={handleChange}
  value={values.email}
  className="w-full p-2 border rounded"
/>
{errors.email && (
  <p className="text-red-500 text-sm">{errors.email}</p>
)}

{/* Password */}
<input
  name="password"
  type="password"
  placeholder="Enter your password"
  onChange={handleChange}
  value={values.password}
```

```
          className="w-full p-2 border rounded"
        />
        {errors.password && (
          <p className="text-red-500 text-sm">{errors.password}</p>
        )}

        <button
          type="submit"
          className="w-full bg-green-500 text-white py-2 rounded hover:bg-green-600"
        >
          Submit
        </button>
      </form>
    )}
  </Formik>
 );
}

export default SignupFormFormik;
```

💡 **Formik + Yup version**:

- Cleaner and easier to extend
- No need for multiple useState or manual validation
- Just define rules once inside Yup.object()

If you want, I can now **merge this signup form into your Todo App project structure** so both work together and share the same form validation style.
That way your learning is consistent.