Understanding Formik and Yup in React Forms

Formik and Yup are widely used in React for handling forms efficiently with **state management, validation, and form submission handling**.

What is Formik?

Formik is a library that simplifies **form handling in React** by managing:

- Form state (values)
- Validation
- Error messages
- Submission handling

Installation

To use Formik, install it via npm:

npm install formik

What is Yup?

Yup is a schema validation library that works well with Formik.
It helps define **rules for form validation** easily.

Installation
npm install yup

Building a Form with Formik and Yup

1. **Basic Form Without Validation**

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

const BasicForm = () => {
  const [formData, setFormData] = useState({
    name: "",
    email: "",
    password: "",
  });

  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
```

```
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    console.log("Submitted Data:", formData);
  };

  return (
    <form onSubmit={handleSubmit}>
      <input type="text" name="name" placeholder="Name"
onChange={handleChange} />
      <input type="email" name="email" placeholder="Email"
onChange={handleChange} />
      <input type="password" name="password"
placeholder="Password" onChange={handleChange} />
      <button type="submit">Submit</button>
    </form>
  );
};

export default BasicForm;
```

Explanation

- We use useState to store form values.
- handleChange() updates the state as users type.
- handleSubmit() logs data to the console.

Problem:
This form has **no validation**, meaning users can submit an empty form.

2. **Form with Formik (No Validation)**

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

const FormikForm = () => {
  const formik = useFormik({
    initialValues: { name: "", email: "", password: "" },
    onSubmit: (values) => {
      console.log("Submitted Data:", values);
```

```
    },
  });

  return (
    <form onSubmit={formik.handleSubmit}>
      <input type="text" name="name" placeholder="Name"
onChange={formik.handleChange} value={formik.values.name} />
      <input type="email" name="email" placeholder="Email"
onChange={formik.handleChange} value={formik.values.email} />
      <input type="password" name="password"
placeholder="Password" onChange={formik.handleChange}
value={formik.values.password} />
      <button type="submit">Submit</button>
    </form>
  );
};

export default FormikForm;
```

Explanation

- useFormik() manages form state (formik.values).
- formik.handleChange() updates state automatically.
- formik.handleSubmit() processes form submission.

Improvement: No need for useState, **Formik handles everything**.

3. **Formik with Validation (Using Yup)**

```
import React from "react";
import { useFormik } from "formik";
import * as Yup from "yup";

const FormikYupForm = () => {
  const formik = useFormik({
    initialValues: { name: "", email: "", password: "" },
    validationSchema: Yup.object({
      name: Yup.string().required("Name is required"),
      email: Yup.string().email("Invalid email").required("Email
is required"),
      password: Yup.string().min(8, "Password must be at least 8
characters").required("Password is required"),
```

```
    }),
    onSubmit: (values) => {
      console.log("Submitted Data:", values);
    },
  });

  return (
    <form onSubmit={formik.handleSubmit}>
      <input type="text" name="name" placeholder="Name"
onChange={formik.handleChange} value={formik.values.name} />
      {formik.touched.name && formik.errors.name &&
<p>{formik.errors.name}</p>}

      <input type="email" name="email" placeholder="Email"
onChange={formik.handleChange} value={formik.values.email} />
      {formik.touched.email && formik.errors.email &&
<p>{formik.errors.email}</p>}

      <input type="password" name="password"
placeholder="Password" onChange={formik.handleChange}
value={formik.values.password} />
      {formik.touched.password && formik.errors.password &&
<p>{formik.errors.password}</p>}

      <button type="submit">Submit</button>
    </form>
  );
};

export default FormikYupForm;
```

Explanation

- **validationSchema** defines rules:
    - Name is required.
    - Email must be valid.
    - Password must be at least **8 characters**.
- formik.errors stores validation errors.
- formik.touched ensures errors are shown **only after touching the field**.

Result: The form **shows real-time validation** errors.

4. Enhancing with Bootstrap Styling

```
npm install react-bootstrap bootstrap


import React from "react";
import { useFormik } from "formik";
import * as Yup from "yup";
import { Form, Button, Container } from "react-bootstrap";
import "bootstrap/dist/css/bootstrap.min.css";

const BootstrapFormikForm = () => {
  const formik = useFormik({
    initialValues: { name: "", email: "", password: "" },
    validationSchema: Yup.object({
      name: Yup.string().required("Name is required"),
      email: Yup.string().email("Invalid email").required("Email
is required"),
      password: Yup.string().min(8, "Password must be at least 8
characters").required("Password is required"),
    }),
    onSubmit: (values) => {
      console.log("Submitted Data:", values);
    },
  });

  return (
    <Container className="mt-4">
      <Form onSubmit={formik.handleSubmit}>
        <Form.Group className="mb-3">
          <Form.Label>Name</Form.Label>
          <Form.Control
            type="text"
            name="name"
            placeholder="Enter name"
            onChange={formik.handleChange}
            onBlur={formik.handleBlur}
            value={formik.values.name}
            isInvalid={formik.touched.name && formik.errors.name}
          />
          <Form.Control.Feedback
```

```
type="invalid">{formik.errors.name}</Form.Control.Feedback>
        </Form.Group>

        <Form.Group className="mb-3">
          <Form.Label>Email</Form.Label>
          <Form.Control
            type="email"
            name="email"
            placeholder="Enter email"
            onChange={formik.handleChange}
            onBlur={formik.handleBlur}
            value={formik.values.email}
            isInvalid={formik.touched.email &&
formik.errors.email}
          />
          <Form.Control.Feedback
type="invalid">{formik.errors.email}</Form.Control.Feedback>
        </Form.Group>

        <Form.Group className="mb-3">
          <Form.Label>Password</Form.Label>
          <Form.Control
            type="password"
            name="password"
            placeholder="Enter password"
            onChange={formik.handleChange}
            onBlur={formik.handleBlur}
            value={formik.values.password}
            isInvalid={formik.touched.password &&
formik.errors.password}
          />
          <Form.Control.Feedback
type="invalid">{formik.errors.password}</Form.Control.Feedback>
        </Form.Group>

        <Button type="submit">Register</Button>
      </Form>
    </Container>
  );
};

export default BootstrapFormikForm;
```