RegistrationForm Component

Yeh component ek registration form hai jo Formik aur Yup libraries ka istemal karti hai.

Imports:

- React ko import kiya gaya hai taake React ki functionalities ka istemal kiya ja sake.
- Formik, Form, Field, aur ErrorMessage ko import kiya gaya hai jo form ke handling aur validation ke liye use hote hain.
- Yup ko import kiya gaya hai validation rules define karne ke liye.
- ToastContainer aur toast ko import kiya gaya hai taake form submission ke baad ek success message dikhaya ja sake.
- CSS file ko bhi import kiya gaya hai taake styling ki ja sake.

Validation Schema:

- validationSchema variable me Yup ka object use kiya gaya hai taake form fields ke liye validation rules set kiye ja sakein. Jaise:
  - firstName, lastName, email, mobileNumber, cnic, gender, aur address ko required banaya gaya hai.
  - mobileNumber ke liye specific format aur length check ki gayi hai.
  - cnic ke liye ek specific format check ki gayi hai.

Handle Submit Function:

- handleSubmit function ko define kiya gaya hai jo form submit hone par call hota hai.
- Is function me toast.success ka istemal karke ek success message dikhaya jata hai aur console me form values ko print kiya jata hai.
- resetForm ko call karke form ko reset kar diya jata hai.

Formik Component:

- Formik component ko form ke initial values aur validation schema ke saath configure kiya gaya hai.
- Initial values me sabhi fields ke liye empty string ya default values set ki gayi hain.

Form Structure:

- Form component ke andar fields aur labels hain:
  - **Name Fields**: firstName, middleName, aur lastName fields hain. ErrorMessage component ko errors dikhane ke liye use kiya gaya hai.
  - **Birth Date aur Gender**: Birth Date ke liye date picker aur Gender ke liye dropdown menu diya gaya hai.

   - **CNIC**: CNIC field me validation rules lagaye gaye hain.
   - **Address**: Address fields ke liye different inputs hain jaise street address, city, aur postal code.
   - **Contact Information**: Email, mobile number, aur phone number ke fields hain.
   - **Additional Info**: Company aur courses ke fields hain. Courses ke liye dropdown menu diya gaya hai.
   - **Comments**: Additional comments ke liye ek text area diya gaya hai.

Submit Button:

- Form submit hone par Submit button disable ho jata hai jab form submit ho raha hota hai (isSubmitting flag se).

Toast Container:

- ToastContainer ko use kiya gaya hai taake success message form submission ke baad dikhaya ja sake.

Yeh component ek comprehensive registration form provide karta hai jo user se detailed information leta hai aur uski validation bhi karta hai.

Yup aur Formik dono kaam aate hain React me forms ko handle karne aur validate karne ke liye. Dono mil kar ek smooth aur efficient way provide karte hain taake form validation aur state management React applications me easy ho sake.

Main ab tumhe **Formik** aur **Yup** ka kaam aur unka role detail se samjhaata hoon.

1. **Formik**:

Formik ek React library hai jo forms ko asaani se handle karne aur manage karne me madad deti hai. React me jab form banate ho, toh har input ko manually handle karna padta hai, jisme form data ki state manage karni hoti hai aur validation ko bhi manually lagana padta hai. Yeh kaafi tedious ho sakta hai jab forms complex ho jaayein. Is problem ko solve karne ke liye **Formik** kaam aata hai.

Formik ke Kaam:

1. **Form State ko Manage karna**:

   - Jab form banate ho toh har input field ka value aur state manage karna hota hai. For example, agar tum ek input field me kuch likhte ho, toh React me uska value ko track karna padta hai. Formik is process ko simplify karta hai. Tumhe manually har input ke liye state handle nahi karni padti, Formik ye kaam asaani se kar leta hai.

2. **Field Handling**:

   - Formik ka `Field` component direct input fields ko control karta hai. Tum har form field ko `Field` component ke zariye handle kar sakte ho. Yeh automatically har field ko form ke state ke sath sync karta hai. Tumhe har field ke liye `onChange`, `onBlur`, aur `value` functions manually define nahi karne padte.

3. **Form Submission Handling**:

   - Jab form submit hota hai, Formik ka `onSubmit` function trigger hota hai. Is function ke zariye tum form ke data ko handle kar sakte ho, jaise API call karna, ya user ko koi feedback dena. `isSubmitting` property ke zariye tum form submit hone tak button ko disable bhi kar sakte ho.

4. **Validation**:

   - Formik ka ek important feature hai validation ko handle karna. Tum multiple validation techniques use kar sakte ho, lekin **Yup** ke sath Formik kaam karna kaafi common hai. Formik Yup ke sath mil kar form fields ko validate karta hai aur errors dikhata hai.

Formik ki Important Properties:

1. **`initialValues`**:

   - Form ke shuru me initial values define ki jaati hain. For example:

     ```javascript
     initialValues={{
       firstName: '',
       email: '',
       password: ''
     }}
     ```

   - Har input field ki default value yeh hogi jab tak user koi data input nahi karta.

2. **`onSubmit`**:

   - Jab user form submit karega, yeh function trigger hoga. Tum isme apna logic likh sakte ho, jaise form data ko API pe bhejna.

```javascript
onSubmit={(values) => {
  console.log(values);
}}
```

3. **`Field`**:

   - Formik ka `Field` component automatically har field ko manage karta hai, jaise uska value aur state. Tumhe manually koi event listeners nahi likhne padte.

```javascript
<Field name="firstName" placeholder="First Name" />
```

4. **`ErrorMessage`**:

   - Agar koi validation error hoti hai, toh `ErrorMessage` component user ko error message dikhata hai.

```javascript
<ErrorMessage name="firstName" component="div" className="error" />
```

2. **Yup**:

Yup ek JavaScript validation library hai jo forms me input fields ko validate karne ke liye use hoti hai. Iska kaam har field ke liye specific rules banana hota hai, jisse ensure kiya ja sake ke input sahi format me hai ya nahi.

Yup ke Kaam:

1. **Validation Rules Define Karna**:

   - Yup ke zariye tum apne form ke har input field ke liye validation rules define kar sakte ho. Har field ke liye alag alag requirements set kar sakte ho, jaise:

     - `required`: Yeh rule check karta hai ke field khali nahi honi chahiye.

     - `min`/`max`: Yeh field ke minimum aur maximum length ko set karta hai.

   - `matches`: Yeh field ke data ko specific pattern ya regular expression ke sath match karta hai.

2. **Complex Validations**:

   - Yup simple validations ke sath complex validations ko bhi handle karta hai, jaise email format check karna, passwords me specific characters hona, ya phir date validation.

3. **Real-Time Feedback**:

   - Yup ke zariye, tum har field ke validation errors ko real-time me show kar sakte ho. Jab user ghalat data input karta hai, toh turant error dikhayi jaati hai.

Example of Yup Validation:

```javascript
const validationSchema = Yup.object({
  firstName: Yup.string()
    .required("First Name is required")
    .min(2, "First Name must be at least 2 characters"),

  email: Yup.string()
    .required("Email is required")
    .email("Invalid email format"),

  password: Yup.string()
    .required("Password is required")
    .min(8, "Password must be at least 8 characters")
    .matches(/[a-z]/, "Password must contain at least one lowercase letter")
    .matches(/[A-Z]/, "Password must contain at least one uppercase letter")
    .matches(/[0-9]/, "Password must contain at least one number"),
});
```

Is example me:

- `firstName` ko kam se kam 2 characters hona chahiye aur yeh required hai.

- `email` ko sahi email format me hona chahiye.

- `password` ke liye kaafi strict rules rakhe gaye hain taake secure ho, jaise uppercase, lowercase, number aur special character ka hona zaroori hai.

Formik aur Yup ka Combination:

- Formik Yup ko validation ke liye use karta hai. Jab user form submit karta hai, Formik Yup se validate karwata hai ke har field ka input sahi hai ya nahi. Agar koi error hoti hai, toh `ErrorMessage` component user ko error feedback deta hai.

Conclusion:

Formik forms ko manage aur handle karne ka kaam asaan banata hai, jabke **Yup** validation ke rules ko simplify karta hai. Yeh dono libraries mil kar React me forms ke liye ek powerful solution deti hain.

Yeh solution user ko ek better experience deta hai jisme:

- Form data ki validation automatic hoti hai.

- Har input ke neeche real-time errors dikhayi jaati hain.

- Form ke submit hone par asaani se success ya error messages handle kiye ja sakte hain.

Is project me main ne Formik aur Yup ka istemal is liye kiya taake form validation aur management efficient ho, aur code ko maintainable aur reusable banaya ja sake.