Step 1 : For validation, install these packages



Step 2 : create the form component

return (

        <form className="formcontainer" onSubmit={handleSubmit}>//event will be triggered when form is submitted i.e when there is no validation errors

            <div>

                <label htmlFor="rollno">Enter Roll No</label> //htmlFor should match the id attribute of the textbox

                <input

                    type="text"

                    name="rollno"

                    id="rollno"

                    onChange={handleChange} //event will be triggered for each char we type into the textbox

                    onBlur={handleBlur} //event will be triggered when the textbox will loose focus

                />

            </div>

            <div>

                <label htmlFor="name">Enter Name</label>

                <input

                    type="text"

                    name="name"

                    id="name"

                    onChange={handleChange} //NOTE: handleSubmit, handleChange, handleBlur are Formik methods

                    onBlur={handleBlur}

                />

            </div>

            <div>

                <label htmlFor="age">Enter Age</label>

                <input

                    type="text"

                    name="age"

                    id="age"

                    onChange={handleChange}

                    onBlur={handleBlur}

                />

            </div>

            <div>

                <label htmlFor="email">Enter Email</label>

                <input

                    type="text"

                    name="email"

                    id="email"

                    onChange={handleChange}

                    onBlur={handleBlur}

                />

            </div>

            <div>

                <label htmlFor="dob">Enter Date of Birth</label>

                <input

                    type="date"

                    name="dob"

                    id="dob"

                    onChange={handleChange}

                    onBlur={handleBlur}

                />

            </div>

            <div>

                <label htmlFor="photo">Upload Photo</label>

                <input

                    type="file"

                    name="photo"

                    id="photo"

                    onChange={(event) => setFieldValue("photo", event.currentTarget.files[0])}

                    onBlur={handleBlur}

                />

            </div>

            <div className="buttongroup">

                <button type="submit">Register</button>

                <button type="button">Cancel</button>

            </div>

        </form>

    );

};

export default RegisterationForm;

Step 3 : initialise all the form data

const initialValues = { //NOTE : the properties must match the name attribute of the form elements

    rollno: "",

    name: "",

    email: "",

    age: "",

    dob: "",

    photo: null

}

Step 4 : make a Validation.js file (NOTE its a .js file not a .jsx as it is not a component)

Validation.js

import \* as yup from 'yup';

export const registartionValidation = yup.object({

    rollno: yup

        .number()

        .typeError("Roll Number must be a number")

        .min(1, "Roll Number must be greater than 0")

        .max(100, "Roll Number must be less than or equal to 100")

        .required("Enter Roll Number"),

    name: yup

        .string()

        .matches(/^[a-zA-Z ]{3,20}$/, "Enter a valid name with 3-20 characters")

        .required("Enter Name"),

    age: yup

        .number()

        .typeError("Age must be a number")

        .min(18, "You must be at least 18 years old")

        .max(100, "Age must be less than or equal to 100")

        .required("Enter Age"),

    email: yup

        .string()

        .email("Enter a valid email address")

        .required("Enter Email"),

    dob: yup

        .date()

        .min(new Date(1947,0,1),  "Date of birth cannot before 1947")

        .max(new Date(), "Date of birth cannot be in the future")

        .required("Choose Date of Birth"),

    photo: yup

        .mixed()

        .test("fileSize", "The file is too large", (value) => {

            return value && value.size <= 2000000;

        })

        .required("Choose a file"),

});

Step 5: write the css

RegisterationForm.css

/\* Container for the entire form \*/

.formcontainer {

    max-width: 400px;

    margin: 0 auto;

    padding: 20px;

    background-color: #f9f9f9;

    border-radius: 10px;

    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

  }

  /\* Each form field container \*/

  .formcontainer > div {

    margin-bottom: 15px;

  }

  /\* Label styling \*/

  .formcontainer label {

    display: block;

    font-weight: bold;

    margin-bottom: 5px;

    color: #333;

  }

  /\* Input field styling \*/

  .formcontainer input[type="text"] {

    width: 100%;

    padding: 10px;

    border: 1px solid #ccc;

    border-radius: 5px;

    font-size: 14px;

    color: #333;

    box-sizing: border-box;

    transition: border-color 0.3s ease;

  }

  /\* Input field focus effect \*/

  .formcontainer input[type="text"]:focus {

    border-color: #007bff;

    outline: none;

  }

  /\* Optional: Add some spacing around the form \*/

  body {

    padding: 20px;

    font-family: Arial, sans-serif;

  }

  /\* Responsive design \*/

  @media (max-width: 480px) {

    .formcontainer {

      padding: 15px;

    }

    .formcontainer input[type="text"] {

      padding: 8px;

      font-size: 12px;

    }

    .formcontainer label {

      font-size: 14px;

    }

  }

  .buttongroup

  {

  display: flex;

  justify-content: space-around;

  }

  .buttongroup button{

    padding: 5px 8px;

    background-color: black;

    color: white;

    border-radius: 5px;

    display: flex;

    align-items: center;

    gap: 8px;

    font-size: 16px;

    justify-content: center;

    cursor: pointer;

  }

  p{

    color: red;

  }

Step 6 : write below the initialization (done in step 3)

import React from "react";

import "./RegisterationForm.css";

import { useFormik } from 'formik';

import { registartionValidation } from "../../validationcode/Validation";

const initialValues = {

    rollno: "",

    name: "",

    email: "",

    age: "",

    dob: "",

    photo: null

}

const RegisterationForm = () => { //useFormik hook

    const { errors, values, touched, handleBlur, handleSubmit, handleChange, setFieldValue } = useFormik({

        initialValues,

        validationSchema: registartionValidation,

        onSubmit: function(){

            alert("Form submitted")

         }

    })

Step 7 : add the values attributes in the text boxes & also show the error messages <p> for each textbox for incorrect value

return (

        <form className="formcontainer" onSubmit={handleSubmit}>//predefined dunction of Formik hook

            <div>

                <label htmlFor="rollno">Enter Roll No</label>

                <input

                    type="text"

                    name="rollno"

                    id="rollno"

                    onChange={handleChange} //predefined function of Formik hook

                    value={values.rollno}

                    onBlur={handleBlur} //predefined function of Formik hook

                />

                {errors?.rollno && touched?.rollno && (<p>{errors?.rollno}</p>)} //the red label under the textbox

            </div>

            <div>

                <label htmlFor="name">Enter Name</label>

                <input

                    type="text"

                    name="name"

                    id="name"

                    onChange={handleChange}

                    value={values.name}

                    onBlur={handleBlur}

                />

                {errors?.name && touched?.name && (<p>{errors?.name}</p>)}

            </div>

            <div>

                <label htmlFor="age">Enter Age</label>

                <input

                    type="text"

                    name="age"

                    id="age"

                    onChange={handleChange}

                    value={values.age}

                    onBlur={handleBlur}

                />

                {errors?.age && touched?.age && (<p>{errors?.age}</p>)}

            </div>

            <div>

                <label htmlFor="email">Enter Email</label>

                <input

                    type="text"

                    name="email"

                    id="email"

                    onChange={handleChange}

                    value={values.email}

                    onBlur={handleBlur}

                />

                {errors?.email && touched?.email && (<p>{errors?.email}</p>)}

            </div>

            <div>

                <label htmlFor="dob">Enter Date of Birth</label>

                <input

                    type="date"

                    name="dob"

                    id="dob"

                    onChange={handleChange}

                    value={values.dob}

                    onBlur={handleBlur}

                />

                {errors?.dob && touched?.dob && (<p>{errors?.dob}</p>)}

            </div>

            <div>

                <label htmlFor="photo">Upload Photo</label>

                <input

                    type="file"

                    name="photo"

                    id="photo"

                    onChange={(event) => setFieldValue("photo", event.currentTarget.files[0])}

                    onBlur={handleBlur}

                />

                {errors?.photo && touched?.photo && (<p>{errors?.photo}</p>)}

            </div>

            <div className="buttongroup">

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

                <button type="button">Cancel</button>

            </div>

        </form>

    );

};

export default RegisterationForm;

Formik

is a hook provided by Formik to manage form state and handle form submission.

It returns an object with several properties and methods to manage form fields, validation, and submission.

**Formik Properties**:

* values: An object containing the current values of form fields.
* errors: An object containing error messages corresponding to each field.
* touched: An object containing information about which fields have been "touched" (i.e., focused and then blurred).
* handleBlur: A function to handle the blur event for each field to mark it as touched.
* handleChange: A function to handle the change event for each field and update the corresponding value.
* handleSubmit: A function to handle the form's submission.
* setFieldValue: A function to manually set the value of a specific field.

**validationSchema**:

* The validationSchema is an object created using Yup to define validation rules for the form fields.

import React, { useState } from 'react';

const RegisterProduct = () => {

    const [userValues, setUserValues] = useState({

        title: '',

        price: 0,

        description: '',

        category: '',

        image: ''

    });

    // Handle text input changes

    const handleChange = (e) => {

        const { name, value } = e.target;

        setUserValues((prev) => ({

            ...prev,

            [name]: value

        }));

    };

    // Handle file input changes

    const handleFileChange = (e) => {

        const { name, files } = e.target;

        setUserValues((prev) => ({

            ...prev,

            [name]: URL.createObjectURL(files[0]) // Store the first file selected

        }));

    };

    // Handle form submission

    async function handleSubmit(event){

        event.preventDefault();  // Prevent default form submission

        console.log(userValues);  // Log the user input data

        try {

            const response = await fetch('http://localhost:8086/saveProduct', {

                method: 'POST',

                headers: {

                    'Content-Type': 'application/json',

                },

                body: JSON.stringify(userValues)

            });

            if (response.ok) {

                const data = await response.json();

                //alert(data.sname + " " +" user registered successfully!");

                alert("Product Added successfully!");

                console.log(data)

                //resetForm();

            } else {

                alert("Failed to Add Product");

            }

        } catch (error) {

            console.error("Error submitting the form", error);

        }

       // alert('Form Submitted');

    };

    return (

        <div>

            <form className="form-container" onSubmit={handleSubmit}>

                <div>

                    <label htmlFor="title">Product Title</label>

                    <input

                        type="text"

                        name="title"

                        id="title"

                        value={userValues.title}

                        onChange={handleChange}

                    />

                </div>

                <div>

                    <label htmlFor="price">Product Price</label>

                    <input

                        type="text"

                        name="price"

                        id="price"

                        value={userValues.price}

                        onChange={handleChange}

                    />

                </div>

                <div>

                    <label htmlFor="description">Product Description</label>

                    <input

                        type="text"

                        name="description"

                        id="description"

                        value={userValues.description}

                        onChange={handleChange}

                    />

                </div>

                <div>

                    <label htmlFor="category">Product Category</label>

                    <input

                        type="text"

                        name="category"

                        id="category"

                        value={userValues.category}

                        onChange={handleChange}

                    />

                </div>

                <div>

                    <label htmlFor="image">Upload Photo</label>

                    <input

                        type="file"

                        name="image"

                        id="image"

                        onChange={handleFileChange}  // File input should not have value set

                    />

                </div>

                <div className="button-group">

                    <button type="submit">Register</button>

                    <button type="button">Cancel</button>

                </div>

            </form>

        </div>

    );

};

export default RegisterProduct;