useEffect(() => {

    if (editData) {

      formik.setFieldValue('country\_name', editData.country\_id);

      formik.setFieldValue('state\_name', editData.state\_name);

    } else {

      formik.setValues(initialValues);

    }

  }, [editData]);

// useEffect(() => {

  //   if (editData) {

  //     formik.setValues(editData);

  //   } else {

  //     formik.setValues(initialValues)

  //   }

  // }, [editData]);

import { useFormik } from 'formik';

import \* as Yup from 'yup';

import { useDispatch, useSelector } from 'react-redux';

import { poststateData, updatestateData } from '../redux/slice/slice2';

import { useEffect } from 'react';

import { useState } from 'react';

import { useNavigate, useParams } from 'react-router-dom';

import { fetchCountryById2 } from '../redux/slice/slice2';

export default function StateUserForm() {

  const navigate = useNavigate()

  const dispatch = useDispatch()

  const [countrydata, setCountrydata] = useState([])

  const [loading, setLoading] = useState(false)

  const { id } = useParams()

  const editData = useSelector((state) => state.data2.editData);

  useEffect(() => {

    const fetchData = async () => {

      try {

        const response = await fetch("https://api.metaestate.ai/api/v1/country");

        if (response.ok) {

          const data = await response.json();

          setCountrydata(data.data)

          console.log(data);

        }

      } catch (error) {

        console.error("An error occurred:", error);

      }

    };

    fetchData();

  }, [editData]);

  useEffect(() => {

    if (!editData || editData.country\_id !== Number(id)) {

      dispatch(fetchCountryById2(Number(id)));

      console.log(Number(id))

    }

  }, [id, dispatch]);

  const initialValues = {

    country\_name: '',

    state\_name: '',

  };

  useEffect(() => {

    if (editData) {

      formik.setFieldValue('country\_name', editData.country\_id);

      formik.setFieldValue('state\_name', editData.state\_name);

    } else {

      formik.setValues(initialValues);

    }

  }, [editData]);

  const validationSchema = Yup.object().shape({

    country\_name: Yup.string().required(),

    state\_name: Yup.string().required(),

  });

  const formik = useFormik({

    initialValues: initialValues,

    validationSchema: validationSchema,

    onSubmit: async (values) => {

      setLoading(true);

      try {

        const selectedCountry = countrydata.find(

          (country) => country.country\_name === values.country\_name

        );

        if (selectedCountry) {

          const dataToPost = {

            country\_id: selectedCountry.country\_id,

            state\_name: values.state\_name,

          };

          if (editData) {

            await dispatch(updatestateData({ values, id: Number(id) }));

          } else {

            await dispatch(poststateData(dataToPost));

          }

        }

      } catch (error) {

        console.error('Error posting data:', error);

      }

      navigate('/statepage');

    }

  });

  return (

    <>

      <form

        style={{

          width: '500px',

          border: '2px solid black',

          position: 'absolute',

          top: '30%',

          left: '35%',

          padding: '30px',

        }}

        onSubmit={formik.handleSubmit}

      >

        <div className="mb-3">

          <label htmlFor="country\_name" className="form-label">

            country\_name

          </label>

          <select

            className="form-select" aria-label="Default select example"

            id="country\_name"

            name="country\_name"

            value={formik.values.country\_name}

            onChange={formik.handleChange}

            onBlur={formik.handleBlur}

          >

            <option value="">Open this select menu</option>

            {countrydata.map((item) => (

              <option key={item.id} value={item.country\_name}>

                {item.country\_name}

              </option>

            ))}

          </select>

          {formik.touched.country\_name && formik.errors.country\_name ? (

            <p style={{ color: 'red' }}>{formik.errors.country\_name}</p>

          ) : null}

        </div>

        <div className="mb-3">

          <label htmlFor="currency\_name" className="form-label">

            state\_name

          </label>

          <input

            type="text"

            className="form-control"

            id="state\_name"

            name="state\_name"

            value={formik.values.state\_name}

            onChange={formik.handleChange}

            onBlur={formik.handleBlur}

          />

          {formik.touched.state\_name && formik.errors.state\_name ? (

            <p style={{ color: 'red' }}>{formik.errors.state\_name}</p>

          ) : null}

        </div>

        {loading ?

          <>

            <div className="spinner-grow text-primary" role="status">

              <span className="visually-hidden">Loading...</span>

            </div>

            <div className="spinner-grow text-secondary" role="status">

              <span className="visually-hidden">Loading...</span>

            </div>

            <div className="spinner-grow text-success" role="status">

              <span className="visually-hidden">Loading...</span>

            </div>

            <div className="spinner-grow text-danger" role="status">

              <span className="visually-hidden">Loading...</span>

            </div>

            <div className="spinner-grow text-warning" role="status">

              <span className="visually-hidden">Loading...</span>

            </div>

            <div className="spinner-grow text-info" role="status">

              <span className="visually-hidden">Loading...</span>

            </div>

          </> :

          <button type="submit" className="btn btn-primary">

            {editData ? "Save Changes" : "Submit"}

          </button>

        }

      </form>

    </>

  );

}