TAILWIND CONFIG

* Go to TailwindCSS and click Get Started and select Framework guides and Choose Create React App and follow the instructions (<https://tailwindcss.com/docs/installation/framework-guides>)
* Commands to setup
  + npm install -D tailwindcss
  + npx tailwindcss init
* Update the tailwind.config.js with the following

/\*\* @type {import('tailwindcss').Config} \*/

module.exports = {

  content: [

    "./src/\*\*/\*.{js,jsx,ts,tsx}",

  ],

  theme: {

    extend: {},

  },

  plugins: [],

}

* Update index.css with the following code

@tailwind base;

@tailwind components;

@tailwind utilities;

FORM VALIDATION

* If we are having a complex form with a huge number of form controls then its better to use an external library such as formik
* If we are having a small form, we can write custom validators
* It’s a best practice to write validators in a different file
* Ex:

export const checkValidData = (email, password, name, isSignInForm) => {

    const isEmailValid = /^[a-zA-Z0-9\_.±]+@[a-zA-Z0-9-]+.[a-zA-Z0-9-.]+$/.test(email);

    const isPasswordValid = /^(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[a-zA-Z]).{8,}$/.test(password);

    if (!isEmailValid) return 'Email is not valid';

    if (!isPasswordValid) return 'Password is not valid';

    if (!isSignInForm && !name) return 'Name is required'

    return null

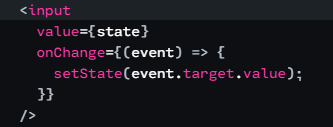
}

**Accessing form controls / Data binding**

* There are two ways to access the values form control
  + Using state variable

Ex:

const [state, setState] = useState('Hello World');



* + Using the reference of the input boxes using useRef Hook of React library
* Binding using useRef
  + useRef is a React Hook that lets you reference a value that’s not needed for rendering.
  + Ex:

const email = useRef(null);

<input ref={email} type='text' placeholder='Email Address' className='p-4 my-4 w-full bg-gray-800'></input>

* We can prevent the default form submission using event.preventDefault() method

Ex:

<form onSubmit={(event) => event.preventDefault()} >

// form controls

</form>

**Programmatic Navigation in ‘react-router-dom’**

* The ‘navigate’ method of ‘useNavigate()’ hook from ‘react-router-dom’ is used for programmatic navigation.

**React.StrictMode**

* Used to analyze the inconsistency between our calls by react. But due to which we might observe our methods are called twice, like useEffect callback. It will get automatically removed on making a prod build