**Assignment**

**Learner Details**

* **Name: Nikhil k**
* **Enrollment Number: SU625MR004**
* **Batch / Class: June 2025 MERN**
* **Assignment: Registration form using typescript**
* **Date of Submission: 12/08/2025**

**Problem Solving Activity 1.1**

**1. Program Statement**

A simple React + TypeScript registration form that collects username, email, and age, updating state dynamically using useState. On submission, it logs the data to the console with basic inline styling for a clean look.

**2. Algorithm**

1. Initialize State:

2. Use useState to store an object containing username, email, and age fields.

3. Handle Input Change:

4. When the user types into an input, update the corresponding property in state.

5. Convert the value to a number when the field name is "age".

6. Handle Form Submit:

7. Prevent default browser form submission.

8. Log the state object to the console.

9. Render the Form:

10. Display labeled input fields for username, email, and age.

11. Bind the value and onChange events to maintain controlled components.

12. Add a styled "Register" button.

**3. Pseudocode**

START

DEFINE interface FormData with fields: username (string), email (string), age (number)

INITIALIZE form state with empty username, empty email, and age = 0

FUNCTION handleChange(event)

EXTRACT name, value from event.target

IF name == "age"

UPDATE state with Number(value)

ELSE

UPDATE state with value

FUNCTION handleSubmit(event)

PREVENT default form submission

PRINT "Form submitted:" and form data to console

RENDER form with:

- Username input

- Email input

- Age input

- Register button

Bind all inputs to state and change handler

END

**4. Program Code**

import React, { useState } from "react";

interface FormData {

username: string;

email: string;

age: number;

}

const RegistrationForm: React.FC = () => {

const [form, setForm] = useState<FormData>({

username: "",

email: "",

age: 0,

});

const handleChange = (e: React.ChangeEvent<HTMLInputElement>) => {

const { name, value } = e.target;

setForm(prevForm => ({

...prevForm,

[name]: name === "age" ? Number(value) : value,

}));

};

const handleSubmit = (e: React.FormEvent<HTMLFormElement>) => {

e.preventDefault();

console.log("Form submitted:", form);

};

return (

<div

style={{

color: "black",

border: "2px solid black",

padding: "50px",

borderRadius: "20px",

background: "linear-gradient(35deg, violet, skyblue, violet, skyblue, violet, skyblue)"

}}

>

<form onSubmit={handleSubmit}>

<h2 style={{ margin: "-30px 0px 50px 0px" }}>Registration Form</h2>

<label htmlFor="username">Username:</label>

<input

id="username"

type="text"

name="username"

placeholder="Username"

value={form.username}

onChange={handleChange}

style={{ width: "200px", height: "30px", margin: "0 30px 0 0" }}

/><br /><br />

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

<input

id="email"

type="email"

name="email"

placeholder="Email"

value={form.email}

onChange={handleChange}

style={{ width: "200px", height: "30px" }}

/><br /><br />

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

<input

id="age"

type="number"

name="age"

placeholder="Age"

value={form.age}

onChange={handleChange}

style={{ width: "200px", height: "30px" }}

/><br />

<button

type="submit"

style={{

border: "2px solid gray",

margin: "50px 0 0 0",

background: "blue",

color: "white",

padding: "10px 20px",

borderRadius: "5px",

cursor: "pointer"

}}

>

Register

</button>

</form>

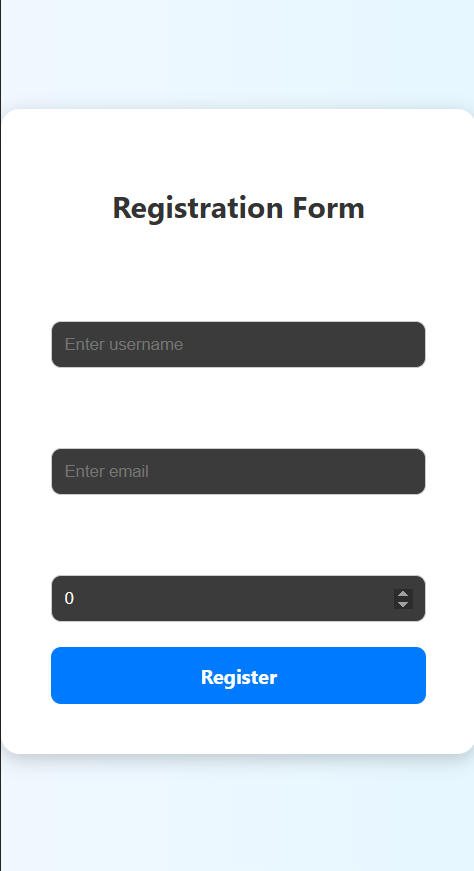
</div>

);

};

export default RegistrationForm;

**6. Screenshots of Output**



**7. Observation / Reflection**

This assignment helped me understand how to use JavaScript functions and interact with the DOM effectively. I learned how to connect inputs, dropdowns, and buttons to perform real-time calculations. Handling edge cases like empty inputs or division by zero improved my validation skills. I also enjoyed styling the interface. Next time, I’d like to add features like keyboard input, calculation history, and instant validation.