# Web Stack Development

## Lab Exercise 6

**Q.) Implementing a User Registration Form with Validation for Gym Management System**

### 1. Key Features:

- Real-Time Validation:
  - The form includes validation for name, email, password, confirm password, and date of birth (DOB).
  - Users receive instant feedback with visual cues (like color changes and icons) as they fill out the form.
- Responsive Design:
  - Tailwind CSS ensures that the form is fully responsive, adapting to various screen sizes while maintaining a clean, user-friendly layout.
- Age Verification:
  - The form calculates the user's age based on the DOB and disables the submit button if the user is under 18.

### 2. Tools and Technologies Used

- HTML: Structure of the registration form.
- JavaScript: Handles the validation logic, ensuring that each field meets the specified criteria.
- Tailwind CSS: Provides utility-first styling, making the form responsive and visually appealing without the need for extensive custom CSS.

# 3. Implementation

## 3.1 HTML Structure (index.html)

The HTML structure of the form includes the following key elements:

- **Form Elements**: Input fields for the user's name, email, password, confirm password, and date of birth.
- **Validation Messages**: Error and success messages displayed dynamically based on the validation state.

## CODE SNIPPET: -

```
<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>User Registration</title>

   <link href="https://cdn.jsdelivr.net/npm/tailwindcss@latest/dist/tailwind.min.css" rel="stylesheet">

</head>

<body class="bg-gray-100">

   <h2 class="text-2xl text-center font-bold text-gray-800 my-6">User Registration Form for Gym</h2>

   <form id="registrationForm" class="max-w-md mx-auto bg-white p-6 rounded-lg shadow-md">

      <label for="name" class="block text-gray-700">Name:</label>

      <input type="text" id="name" name="name" class="w-full p-2 mt-1 mb-3 border rounded focus:outline-none focus:ring-2 focus:ring-green-400">

      <span class="error-message text-red-500 text-sm"></span>


      <label for="email" class="block text-gray-700">Email:</label>

      <input type="email" id="email" name="email" class="w-full p-2 mt-1 mb-3 border rounded focus:outline-none focus:ring-2 focus:ring-green-400">

      <span class="error-message text-red-500 text-sm"></span>
```

```
    <label for="password" class="block text-gray-700">Password:</label>
    <input type="password" id="password" name="password" class="w-full p-2 mt-1 mb-3 border rounded focus:outline-none focus:ring-2 focus:ring-green-400">
    <span class="error-message text-red-500 text-sm"></span>

    <label for="confirmPassword" class="block text-gray-700">Confirm Password:</label>
    <input type="password" id="confirmPassword" name="confirmPassword" class="w-full p-2 mt-1 mb-3 border rounded focus:outline-none focus:ring-2 focus:ring-green-400">
    <span class="error-message text-red-500 text-sm"></span>

    <label for="dob" class="block text-gray-700">Date of Birth:</label>
    <input type="text" id="dob" name="dob" placeholder="YYYY-MM-DD" class="w-full p-2 mt-1 mb-3 border rounded focus:outline-none focus:ring-2 focus:ring-green-400">
    <span class="error-message text-red-500 text-sm"></span>

    <input type="submit" id="submitBtn" value="Register" class="w-full p-3 mt-4 bg-green-500 text-white rounded hover:bg-green-600 disabled:bg-gray-400 cursor-pointer">
  </form>

  <script src="script.js"></script>
</body>
</html>
```

### 3.2 JavaScript Validation (script.js)

The JavaScript file handles the form validation, ensuring that the data entered by the user meets the required criteria before submission. It provides real-time feedback as the user interacts with the form.

**Key Validation Functions**

1. **Name Validation**
    - Ensures the name contains only alphabetic characters and spaces, with a minimum of 3 characters.
    - **Regex Used**: /^[A-Za-z\s]{3,}$/
2. **Email Validation**
    - Checks for a valid email format.
    - **Regex Used**: /^[^\s@]+@[^\s@]+\.[^\s@]+$/
3. **Password Validation**
    - Ensures the password is at least 8 characters long and contains both letters and numbers.
    - **Regex Used**: /^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{8,}$/
4. **Confirm Password Validation**
    - Checks if the confirm password matches the original password.
5. **Date of Birth Validation**
    - Validates that the date is in YYYY-MM-DD format and calculates if the user is at least 18 years old.

**CODE SNIPPET**: -

```
document.addEventListener("DOMContentLoaded", function () {
    const form = document.getElementById("registrationForm");
    const nameInput = document.getElementById("name");
    const emailInput = document.getElementById("email");
    const passwordInput = document.getElementById("password");
    const confirmPasswordInput = document.getElementById("confirmPassword");
    const dobInput = document.getElementById("dob");
    const submitBtn = document.getElementById("submitBtn");
```

```
    // Validation Functions
    function validateName() {
        const name = nameInput.value.trim();
        const regex = /^[A-Za-z\s]{3,}$/;
        if (regex.test(name)) {
            setValid(nameInput);
            return true;
        } else {
            setInvalid(nameInput, "Name must be at least 3 characters and contain only letters
and spaces.");
            return false;  }
    }

    function validateEmail() {
        const email = emailInput.value.trim();
        const regex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
        if (regex.test(email)) {
            setValid(emailInput);
            return true;
        } else {
            setInvalid(emailInput, "Please enter a valid email address.");
            return false;
        }
    }

    function validatePassword() {
        const password = passwordInput.value.trim();
        const regex = /^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{8,}$/;
        if (regex.test(password)) {
            setValid(passwordInput);
            return true;
        } else {
```

```
        setInvalid(passwordInput, "Password must be at least 8 characters long and contain both letters and numbers.");
        return false;
    }
}

function validateConfirmPassword() {
    const confirmPassword = confirmPasswordInput.value.trim();
    if (confirmPassword === passwordInput.value.trim()) {
        setValid(confirmPasswordInput);
        return true;
    } else {
        setInvalid(confirmPasswordInput, "Passwords do not match.");
        return false;
    }
}

function validateDOB() {
    const dob = dobInput.value.trim();
    const regex = /^\d{4}-\d{2}-\d{2}$/;

    if (!regex.test(dob)) {
        setInvalid(dobInput, "Date must be in YYYY-MM-DD format.");
        return false;
    }

    const dobDate = new Date(dob);
    const today = new Date();
    let age = today.getFullYear() - dobDate.getFullYear();
    const monthDiff = today.getMonth() - dobDate.getMonth();

    if (monthDiff < 0 || (monthDiff === 0 && today.getDate() < dobDate.getDate())) {
```

```
        age--;
    }

    if (age >= 18) {
        setValid(dobInput);
        submitBtn.disabled = false;
        return true;
    } else {
        setInvalid(dobInput, "You must be at least 18 years old.");
        submitBtn.disabled = true;
        return false;
    }
}

// Helper Functions to Set Validation States
function setValid(element) {
    element.classList.add("valid");
    element.classList.remove("invalid");
    element.nextElementSibling.textContent = '✓';
    element.nextElementSibling.style.color = "green";
    element.nextElementSibling.style.display = "inline";
}

function setInvalid(element, message) {
    element.classList.add("invalid");
    element.classList.remove("valid");
    element.nextElementSibling.textContent = message;
    element.nextElementSibling.style.color = "red";
    element.nextElementSibling.style.display = "inline";
}

// Attach Event Listeners for Real-Time Validation
```

```
    nameInput.addEventListener("input", validateName);
    emailInput.addEventListener("input", validateEmail);
    passwordInput.addEventListener("input", validatePassword);
    confirmPasswordInput.addEventListener("input", validateConfirmPassword);
    dobInput.addEventListener("input", validateDOB);

    // Event Listener for Form Submission
    form.addEventListener("submit", function (e) {
        e.preventDefault(); // Prevent form from submitting if validation fails

        const isNameValid = validateName();
        const isEmailValid = validateEmail();
        const isPasswordValid = validatePassword();
        const isConfirmPasswordValid = validateConfirmPassword();
        const isDOBValid = validateDOB();

        if (isNameValid && isEmailValid && isPasswordValid && isConfirmPasswordValid
&& isDOBValid) {
            alert("Registration successful!");
            form.submit(); // Submit the form if all fields are valid
        } else {
            alert("Please correct the errors in the form.");
        }
    });
});
```

## 4. Styling with Tailwind CSS

Tailwind CSS simplifies the styling of the form by using utility classes directly in the HTML. This eliminates the need for extensive custom CSS, speeding up the development process while ensuring a responsive and clean design.

- **Layout**: max-w-md mx-auto centers the form on the page.
- **Typography**: Tailwind utilities like text-2xl, font-bold, and text-center are used to style the form header.
- **Form Inputs**: Classes like block w-full px-3 py-2 border rounded-md are applied to input fields for consistent styling.
-

## 5. Conclusion -

## 6. Form Submission Workflow

1. **Real-Time Validation**: As the user interacts with the form, each field is validated in real-time. Invalid fields are highlighted, and error messages are displayed.
2. **Final Validation on Submission**: When the form is submitted, all fields are revalidated. If any field is invalid, the form submission is prevented, and an alert is shown.
3. **Successful Submission**: If all fields are valid, the form is submitted, and a success message is displayed.