**Code:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Automobile Company User Registration</title>

    <link href="<https://cdnjs.cloudflare.com/ajax/libs/tailwindcss/2.2.19/tailwind.min.css>" rel="stylesheet">

    <link rel="stylesheet" href="form.css">

</head>

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

    <div class="max-w-lg mx-auto p-6 bg-white rounded-lg shadow-md mt-10">

        <h2 class="text-2xl font-bold mb-6 text-center text-blue-600">User Registration Form for booking a Bently</h2>

        <form id="registrationForm" novalidate>

            <div class="mb-4">

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

                <input type="text" id="name" name="name" class="w-full p-2 border rounded" required>

                <small id="nameError" class="text-red-600"></small>

            </div>

            <div class="mb-4">

                <label for="email" class="block text-gray-700">Enter your Email Address</label>

                <input type="email" id="email" name="email" class="w-full p-2 border rounded" required>

                <small id="emailError" class="text-red-600"></small>

            </div>

            <div class="mb-4">

                <label for="password" class="block text-gray-700">Enter your Password</label>

                <input type="password" id="password" name="password" class="w-full p-2 border rounded" required>

                <small id="passwordError" class="text-red-600"></small>

            </div>

            <div class="mb-4">

                <label for="confirmPassword" class="block text-gray-700">Please confirm your Password</label>

                <input type="password" id="confirmPassword" name="confirmPassword" class="w-full p-2 border rounded" required>

                <small id="confirmPasswordError" class="text-red-600"></small>

            </div>

            <div class="mb-4">

                <label for="dob" class="block text-gray-700">Enter your Date of Birth</label>

                <input type="date" id="dob" name="dob" class="w-full p-2 border rounded" required>

                <small id="dobError" class="text-red-600"></small>

            </div>

            <button type="submit" id="submitBtn" class="w-full bg-blue-600 text-white p-2 rounded mt-4">CLICK HERE TO SUBMIT</button>

        </form>

    </div>

    <script src="form.js"></script>

</body>

</html>

**------------CSS------------**

input:valid {

    border-color: green;

}

input:invalid {

    border-color: red;

}

small {

    display: none;

}

input:invalid + small {

    display: block;

}

.error-visible {

    display: block;

}

**--------Java Script--------**

document.getElementById('registrationForm').addEventListener('submit', function(event) {

    event.preventDefault(); // Prevent form submission

    validateForm();

});

document.getElementById('name').addEventListener('input', validateName);

document.getElementById('email').addEventListener('input', validateEmail);

document.getElementById('password').addEventListener('input', validatePassword);

document.getElementById('confirmPassword').addEventListener('input', validateConfirmPassword);

document.getElementById('dob').addEventListener('input', validateDOB);

function validateForm() {

    validateName();

    validateEmail();

    validatePassword();

    validateConfirmPassword();

    validateDOB();

    // If all fields are valid, submit the form

    if (isFormValid()) {

        alert("Form submitted successfully!");

        document.getElementById('registrationForm').submit();

    }

}

function validateName() {

    const nameInput = document.getElementById('name');

    const nameError = document.getElementById('nameError');

    const namePattern = /^[A-Za-z\s]{3,}$/;

    if (!namePattern.test(nameInput.value)) {

        nameError.textContent = "Name must be at least 3 characters long and contain only alphabets and spaces.";

        nameInput.classList.add('border-red-500');

        nameError.classList.add('error-visible');

    } else {

        nameError.classList.remove('error-visible');

        nameInput.classList.remove('border-red-500');

    }

}

function validateEmail() {

    const emailInput = document.getElementById('email');

    const emailError = document.getElementById('emailError');

    const emailPattern = /^[a-zA-Z0-9.\_-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/;

    if (!emailPattern.test(emailInput.value)) {

        emailError.textContent = "Please enter a valid email address.";

        emailInput.classList.add('border-red-500');

        emailError.classList.add('error-visible');

    } else {

        emailError.classList.remove('error-visible');

        emailInput.classList.remove('border-red-500');

    }

}

function validatePassword() {

    const passwordInput = document.getElementById('password');

    const passwordError = document.getElementById('passwordError');

    const passwordPattern = /^(?=.\*[A-Za-z])(?=.\*\d)[A-Za-z\d]{8,}$/;

    if (!passwordPattern.test(passwordInput.value)) {

        passwordError.textContent = "Password must be at least 8 characters long and contain both letters and numbers.";

        passwordInput.classList.add('border-red-500');

        passwordError.classList.add('error-visible');

    } else {

        passwordError.classList.remove('error-visible');

        passwordInput.classList.remove('border-red-500');

    }

}

function validateConfirmPassword() {

    const confirmPasswordInput = document.getElementById('confirmPassword');

    const confirmPasswordError = document.getElementById('confirmPasswordError');

    const passwordInput = document.getElementById('password');

    if (confirmPasswordInput.value !== passwordInput.value) {

        confirmPasswordError.textContent = "Passwords do not match.";

        confirmPasswordInput.classList.add('border-red-500');

        confirmPasswordError.classList.add('error-visible');

    } else {

        confirmPasswordError.classList.remove('error-visible');

        confirmPasswordInput.classList.remove('border-red-500');

    }

}

function validateDOB() {

    const dobInput = document.getElementById('dob');

    const dobError = document.getElementById('dobError');

    const dob = new Date(dobInput.value);

    const today = new Date();

    const age = today.getFullYear() - dob.getFullYear();

    const monthDifference = today.getMonth() - dob.getMonth();

    const dayDifference = today.getDate() - dob.getDate();

    if (monthDifference < 0 || (monthDifference === 0 && dayDifference < 0)) {

        age--;

    }

    if (age < 18) {

        dobError.textContent = "You must be at least 18 years old.";

        dobInput.classList.add('border-red-500');

        dobError.classList.add('error-visible');

        document.getElementById('submitBtn').disabled = true;

    } else {

        dobError.classList.remove('error-visible');

        dobInput.classList.remove('border-red-500');

        document.getElementById('submitBtn').disabled = false;

    }

}

function isFormValid() {

    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');

    return (

        nameInput.value.trim() !== '' &&

        emailInput.value.trim() !== '' &&

        passwordInput.value.trim() !== '' &&

        confirmPasswordInput.value.trim() !== '' &&

        dobInput.value.trim() !== '' &&

        !document.querySelector('.border-red-500')

    );

}

**HTML**

1. **Structure and Semantics**:
   * **<form> Element**: Used to contain form fields and manage submission.
   * **<input> Fields**: Each field (name, email, password, confirmPassword, dob) is paired with a <label> for accessibility.
   * **Error Messages**: <small> elements are used for displaying error messages related to each input field.
2. **Accessibility**:
   * **required Attribute**: Ensures that users fill out all fields before submission.
   * **novalidate Attribute**: Prevents native browser validation, allowing custom JavaScript validation.
3. **Styling**:
   * **Tailwind CSS**: Provides utility classes for responsive design, spacing, typography, and colors.
   * **Custom CSS**: Styles error states (input:valid, input:invalid, small, and .error-visible).

**CSS**

1. **Custom Styling**:
   * **Error States**:
     + **input:invalid**: Changes the border color to red to indicate invalid input.
     + **input:valid**: Changes the border color to green for valid input.
     + **small**: Initially hidden but displayed when an input is invalid.
   * **Error Visibility**: .error-visible class ensures error messages are visible when needed.
2. **Flexibility**:
   * Tailwind CSS provides a base styling framework, while custom CSS addresses specific validation states and error visibility.

**JavaScript**

1. **Event Handling**:
   * **Form Submission**:
     + **submit Event**: Prevents default form submission to handle validation manually.
     + **validateForm Function**: Validates all fields before allowing form submission.
   * **Field Input Events**: Each field has an input event listener to provide real-time feedback.
2. **Validation Functions**:
   * **validateName**: Ensures the name is at least 3 characters long and contains only alphabets and spaces.
   * **validateEmail**: Checks if the email address conforms to a standard format.
   * **validatePassword**: Validates that the password is at least 8 characters long and includes both letters and numbers.
   * **validateConfirmPassword**: Ensures that the confirmation password matches the original password.
   * **validateDOB**: Confirms that the user is at least 18 years old based on the provided date of birth.
3. **Form Validity Check**:
   * **isFormValid Function**: Checks if all fields are filled and if there are no invalid inputs (i.e., no elements with border-red-500).

**Rationale**

* **Custom Validation**: By handling validation through JavaScript, you can provide immediate, customized feedback and ensure that users follow specific rules before form submission.
* **Visual Feedback**: Using CSS for error states and feedback improves user experience by clearly indicating where corrections are needed.
* **Accessibility and Usability**: Ensuring all form fields are required and providing clear, real-time feedback help users complete the form accurately and efficiently.
* **Tailwind CSS and Custom Styling**: Tailwind CSS simplifies layout and styling, while custom CSS fine-tunes error handling and visual feedback.