Web Stack Development

Lab Exercise 6

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

1. Introduction

This project involves creating a user registration form for the Gym Management System website. The form collects essential user information and ensures all data is valid before submission. The goal is to deliver a user-friendly experience by implementing form validation using JavaScript, providing real-time feedback to users as they interact with the form.

2. Form Requirements

The registration form collects the following information:

- **Full Name:** Must contain only alphabetic characters and spaces, with a minimum of 3 characters.
- **Email:** Must follow a proper email format (e.g., "user@example.com").
- **Password:** Must be at least 8 characters long and contain both letters and numbers.
- **Confirm Password:** Must match the password entered.
- **Date of Birth:** Must be in "YYYY-MM-DD" format, and the user must be at least 18 years old.

3. Project Structure

The project consists of three main files:

- index.html: The HTML structure of the registration form.
- styles.css: The CSS file for styling the form and its validation states.
- script.js: The JavaScript file responsible for client-side validation.

4. Implementation

4.1 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 rel="stylesheet" href="styles.css"> </head>
<body>
   <h2>User Registration Form for Gym</h2>
   <form id="registrationForm">
      <label for="name">Name:</label>
      <input type="text" id="name" name="name">
      <span class="error-message"></span>

      <label for="email">Email:</label>
      <input type="email" id="email" name="email">
      <span class="error-message"></span>

      <label for="password">Password:</label>
      <input type="password" id="password" name="password">
      <span class="error-message"></span>
```

```
        <label for="confirmPassword">Confirm Password:</label>
        <input type="password" id="confirmPassword" name="confirmPassword">
        <span class="error-message"></span>

        <label for="dob">Date of Birth:</label>
        <input type="text" id="dob" name="dob" placeholder="YYYY-MM-DD">
        <span class="error-message"></span>
        <input type="submit" id="submitBtn" value="Register">

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

4.2 CSS Styling (styles.css)

The CSS file styles the form, making it visually appealing and user-friendly. It includes styles for the general layout, input fields, submit button, and validation states.

CODE SNIPPET: -

```
/* General Form Styles */
form {
    width: 300px;
    margin: 0 auto;
 }
label {
    display: block;
    margin-top: 10px;
 }
input[type="text"],
input[type="email"],
 input[type="password"],
 input[type="date"] {
```

```
    width: 100%;
    padding: 8px;
    margin-top: 5px;
    margin-bottom: 5px;
    border: 1px solid #ccc;
    border-radius: 4px;
    box-sizing: border-box;
}
input[type="submit"] {
    width: 100%;
    padding: 10px;
    margin-top: 20px;
    background-color: #4CAF50;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
input[type="submit"]: disabled {
    background-color: #ccc; }

/* Validation States */
input.valid {
    border-color: green;
}
input.invalid {
    border-color: red;
}
```

```
/* Error and Success Messages */
.error-message {
    display: none;
    font-size: 12px;
    margin-left: 5px;
}
span.invalid {
    color: red;
    display: inline-block;
}
span.valid {
    color: green;
    display: inline-block;
}
/* Center-align the h2 tag */
h2 {
    text-align: center;
    margin-top: 20px; /* Optional: add some top margin for spacing */
    font-family: Arial, sans-serif; /* Optional: choose a specific font */
    font-size: 24px; /* Optional: set the font size */
    color: #333; /* Optional: change the color */
}
```

- **Validation States**: The valid and invalid classes are used to highlight input fields based on their validation status.
- **Error Messages**: Displayed next to the input fields when validation fails, using the “. error-message” class.

4.3 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.");
        }
    });
});
```

5. 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.