EXPERIMENT NO. 1

Title: Development of a Simple Web Application with Input Validation using HTML, CSS, and JavaScript

Aim: To design and develop a basic web application using HTML, CSS, and JavaScript with client-side form validation.

Theory:

Form validation is an essential part of web development. It ensures that the user has provided valid and complete data before the information is processed or sent to a server. Client-side validation is handled using JavaScript, which checks user inputs directly in the browser, without involving the server.

This improves user experience by giving instant feedback and reduces unnecessary network requests. HTML provides the structure of the form, CSS enhances its visual appearance, and JavaScript handles validation tasks such as:

- Checking if required fields are filled
- Verifying email format
- Ensuring password length or strength

Steps of Execution:

1. Create the HTML structure for the registration form.
2. Add CSS to style the form elements and layout.
3. Implement JavaScript for form validation:
    - Check for empty fields
    - Validate email format
    - Validate password length
4. Test the form with various input values to verify the validation logic.

Stepwise Procedure:

Step 1: Create a folder with the following files:

- index.html
- style.css
- script.js

Step 2: HTML Code – index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Simple Registration Form</title>
  <link rel="stylesheet" href="style.css">
```

```
</head>
<body>
  <div class="container">
    <h2>User Registration</h2>
    <form id="registrationForm">
      <label for="name">Name:</label>
      <input type="text" id="name" placeholder="Enter your name">
      <label for="email">Email:</label>
      <input type="text" id="email" placeholder="Enter your email">
      <label for="password">Password:</label>
      <input type="password" id="password" placeholder="Enter your password">
      <button type="submit">Register</button>
      <p id="message"></p>
    </form>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

Step 3: CSS Code – style.css

```
body {
  font-family: Arial, sans-serif;
  background-color: #f2f2f2;
}
.container {
  width: 350px;
  margin: 50px auto;
  padding: 25px;
  background-color: #ffffff;
  border-radius: 10px;
  box-shadow: 0 0 10px #aaa;
}
input, button {
  width: 100%;
  padding: 10px;
  margin: 8px 0;
  border-radius: 5px;
  border: 1px solid #ccc;
```

```
}
button {
  background-color: #28a745;
  color: white;
  font-weight: bold;
  cursor: pointer;
}
button:hover {
  background-color: #218838;
}
#message {
  margin-top: 15px;
  font-weight: bold;
}
```

Step 4: JavaScript Code – script.js

```
document.getElementById("registrationForm").addEventListener("submit", function(event) {
  event.preventDefault(); // prevent default form submission
  let name = document.getElementById("name").value.trim();
  let email = document.getElementById("email").value.trim();
  let password = document.getElementById("password").value.trim();
  let message = document.getElementById("message");
  // Clear previous message
  message.textContent = "";
  message.style.color = "red";
  if (name === "" || email === "" || password === "") {
    message.textContent = "All fields are required!";
    return;
  }
  let emailPattern = /^[^ ]+@[^ ]+\.[a-z]{2,3}$/;
  if (!emailPattern.test(email)) {
    message.textContent = "Invalid email format!";
    return;
  }
  if (password.length < 6) {
    message.textContent = "Password must be at least 6 characters long!";
    return;
  }
```

```
  message.style.color = "green";
  message.textContent = "Registration Successful!";
});
```

Key Points:

- HTML provides form structure for collecting user input.
- CSS improves the layout and appearance of the web form.
- JavaScript performs real-time client-side validation.
- Helps prevent invalid data from being submitted.
- Enhances user experience and reduces server processing.

Conclusion:

This experiment demonstrated how to build a simple web application using HTML, CSS, and JavaScript to validate form input on the client side. Validation ensures that users enter data in the correct format and prevents incomplete or incorrect data from being submitted. Such techniques are widely used in real-world applications like registration, login, feedback, and e-commerce forms, improving both usability and reliability.