St. Francis Institute of Technology

(Engineering College)

An Autonomous Institute, Affiliated to University of Mumbai NAAC

A+ Accredited | CMPN, EXTC, INFT NBA Accredited | ISO 9001:2015 Certified

Department of Artificial Intelligence and Machine Learning

Academic Year: 2025-2026 **Term**: Even (Jan. 2026 – Jun. 2026) **Class / Branch**: SE – AIML
Semester: IV
Course: Web Programming Lab. (AI4VS_LR4)
Date of Assignment: / /2026 **Date of Submission:** / /2026

Pre-Lab Exercises for Experiment-6

JavaScript Form Validation, Local Storage & UI Control

Before performing Experiment-6, understand HTML forms, basic validation logic, and local storage concepts.

Part A: Conceptual Exercises

Exercise 1: Form Validation Basics

Task: Answer the following:

1) What is Form Validation?

Form validation is the process of checking whether the data entered by the user in a form is correct, complete, and in the proper format before it is submitted to the server. It ensures that users provide valid information such as a properly formatted email address, correct phone number, or strong password. Validation can be done on the client side (using JavaScript) or on the server side. It helps maintain data accuracy and prevents incorrect or harmful data from being processed.

---

2) Why is Client-Side Validation Important?

Client-side validation is important because it checks user input immediately in the browser before sending data to the server. This improves user experience by providing instant feedback and reducing waiting time. It also reduces unnecessary server requests, which saves server resources and improves website performance. Additionally, it helps prevent submission of incomplete or invalid forms, making the application more efficient and user-friendly.

---

3) Name Any Two Form Fields That Require Validation

Some common form fields that require validation are:

- **Email Field** – It should contain a valid email format (example: user@gmail.com).
- **Phone Number Field** – It should contain exactly 10 digits (in India).
- **Password Field** – It should meet minimum length and strength requirements.
- **Username Field** – It should not be empty and may have character restrictions.

Validation ensures that these fields contain accurate and meaningful data.

Exercise 2: Error Handling

Task:

Explain How Error Messages Help Users During Form Submission

Error messages play a very important role in guiding users while filling out a form. When a user enters incorrect or incomplete information, error messages clearly indicate what went wrong. For example, if an email does not contain "@", the system displays an error message like "Invalid Email Address." This helps users understand the mistake and correct it immediately.

Error messages improve user experience by:

- Providing clear instructions on how to fix errors.
- Preventing frustration caused by failed form submissions.
- Saving time by giving instant feedback.
- Ensuring correct and valid data is submitted.

Proper error handling makes web applications more user-friendly, reliable, and efficient.

Part B: Hands-On JavaScript Practice

Exercise 3: Email Validation

Task:

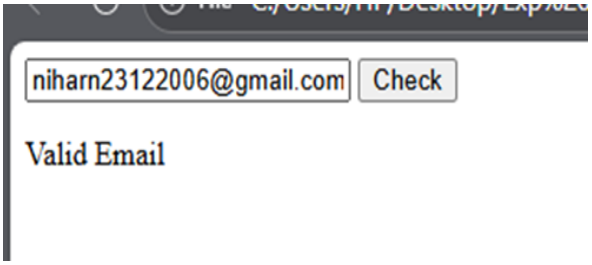
• Create an email input field.

• Check whether the input contains @.

Exp 6 exercise-3.html

C: > Users > HP > Desktop > Exp 6 exercise-3.html > ...

```
1   <input type="text" id="email" placeholder="Enter Email">
2   <button onclick="checkEmail()">Check</button>
3   <p id="msg"></p>
4
5   <script>
6   function checkEmail() {
7     let email = document.getElementById("email").value;
8     if(email.includes("@"))
9       document.getElementById("msg").innerText = "Valid Email";
10    else
11      document.getElementById("msg").innerText = "Invalid Email";
12  }
13  </script>
14
```



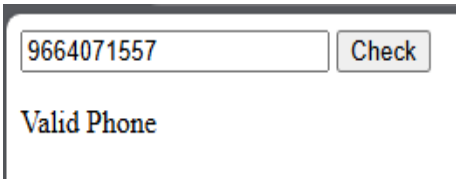
Exercise 4: Phone Number Validation

Task:

• Validate that a phone number contains exactly 10 digits.

```
C: > Users > HP > Desktop > Exp 6 exercise-4.html > ...
<input type="text" id="phone" placeholder="Enter Phone">
<button onclick="checkPhone()">Check</button>
<p id="msg"></p>

<script>
function checkPhone() {
  let phone = document.getElementById("phone").value;
  if(phone.length == 10 && !isNaN(phone))
    document.getElementById("msg").innerText = "Valid Phone";
  else
    document.getElementById("msg").innerText = "Invalid Phone";
}
</script>
```



Exercise 5: Password Strength Check

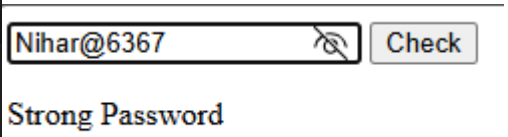
Task:

• Display a message if the password length is less than 6 characters.

Exp 6 exercise-5.html

```
C: > Users > HP > Desktop > Exp 6 exercise-5.html > ...
<input type="password" id="pass" placeholder="Enter Password">
<button onclick="checkPass()">Check</button>
<p id="msg"></p>

<script>
function checkPass() {
  let pass = document.getElementById("pass").value;
  if(pass.length < 6)
    document.getElementById("msg").innerText = "Weak Password";
  else
    document.getElementById("msg").innerText = "Strong Password";
}
</script>
```



Part C: Local Storage Practice

Exercise 6: Storing Data

Task:

• Store a username in local storage.

• Retrieve and display it on page load.

Exp 6 exercise-6.html

C: > Users > HP > Desktop > Exp 6 exercise-6.html > ...

```html
<input type="text" id="user" placeholder="Enter Username">
<button onclick="save()">Save</button>
<p id="display"></p>

<script>
function save() {
  let user = document.getElementById("user").value;
  localStorage.setItem("username", user);
  document.getElementById("display").innerText = user;
}

window.onload = function() {
  document.getElementById("display").innerText =
    localStorage.getItem("username");
}
</script>
```

Exercise 7: Theme Preference

Task:

• Create a button.

• Toggle a CSS class on click.

• Save the preference in local storage

Exp 7 exercise-6.html

C: > Users > HP > Desktop > Exp 7 exercise-6.html > ...

```html
<button onclick="toggle()">Toggle Theme</button>

<script>
function toggle() {
  document.body.classList.toggle("dark");
  localStorage.setItem("theme",
    document.body.classList.contains("dark") ? "dark" : "light");
}

window.onload = function() {
  if(localStorage.getItem("theme") == "dark")
    document.body.classList.add("dark");
}
</script>

<style>
.dark {
  background: black;
  color: white;
}
</style>
```