

**EXPERIMENT NO. 9 : AJAX**

<b>Name of Student</b>	<b><u>Komal Milind Deolekar</u></b>
<b>Class Roll No</b>	<b><u>14</u></b>
<b>D.O.P.</b>	<b><u>08-04-2025</u></b>
<b>D.O.S.</b>	<b><u>15-04-2025</u></b>
<b>Sign and Grade</b>	

**Aim:** To study AJAX

**Problem Statement:**

Create a registration page having fields like Name, College, Username and Password (read password twice).

Validate the form by checking for

1. Username is not same as existing entries
2. Name field is not empty
3. Retyped password is matching with the earlier one. Prompt a message is

And also auto suggest college names.

Show the message "Successfully Registered" on the same page below the submit button, on Successfully registration. Let all the updations on the page be Asynchronously loaded. Implement the same using XMLHttpRequest Object.

**Github Link :**

[https://github.com/KomalDeolekar0607/Webx\\_Lab/tree/main/Webx\\_Lab\\_Exp\\_9](https://github.com/KomalDeolekar0607/Webx_Lab/tree/main/Webx_Lab_Exp_9)

**Theory:****A. How do Synchronous and Asynchronous Requests differ?**

AJAX (Asynchronous JavaScript and XML) is a technique used in web development to create fast and dynamic web pages. With AJAX, web applications can send and retrieve data from a server asynchronously without interfering with the display and behavior of the existing page.

AJAX uses the **XMLHttpRequest** object to interact with the server behind the scenes.

**Synchronous vs. Asynchronous Requests:**

<b>Synchronous Requests</b>	<b>Asynchronous Requests</b>
Blocks the execution of code until response is received	Executes other code while waiting for response.
Slower due to waiting for response.	Faster and more responsive.
Can freeze the browser UI.	Keeps the UI responsive.
Rarely used in modern web apps.	Commonly used in modern AJAX applications.
<code>xhr.open("GET", url, false);</code>	<code>xhr.open("GET", url, true);</code>

**B. Describe various properties and methods used in XMLHttpRequest Object****XMLHttpRequest Object:**

The XMLHttpRequest object is the core of AJAX and is used to exchange data with a web server.

**Properties:**

1. **readyState** – Returns the state of the request:
  - 0: UNSENT
  - 1: OPENED
  - 2: HEADERS\_RECEIVED
  - 3: LOADING
  - 4: DONE
2. **status** – Returns the HTTP status code of the response (e.g., 200 for OK, 404 for Not Found).
3. **responseText** – Returns the response data as a string.
4. **responseXML** – Returns the response data as XML (if applicable).
5. **onreadystatechange** – An event handler that is triggered whenever the readyState property changes.

**Methods:**

- 1) **open(method, url, async)** – Initializes a request (GET or POST), specifies the URL and if it's async.
- 2) **send(data)** – Sends the request to the server. For GET requests, data is null; for POST, it contains data.
- 3) **setRequestHeader(header, value)** – Sets HTTP request headers (used with POST).
- 4) **abort()** – Cancels the current request.

**Code :**

```
<!-- <!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Registration Form</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      background-color: #f1f1f1;

      margin: 0;

      padding: 0;

      display: flex;

      justify-content: center;

      align-items: center;

      height: 100vh;

    }
```

```
.container {  
    background-color: white;  
    padding: 20px;  
    border-radius: 10px;  
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);  
    width: 400px;  
}  
  
h2 {  
    text-align: center;  
    color: #333;  
}  
  
.form-group {  
    margin-bottom: 15px;  
}  
  
.form-group input, .form-group select {  
    width: 100%;  
    padding: 10px;  
    margin-top: 5px;  
    border: 1px solid #ccc;  
    border-radius: 5px;  
}  
  
.form-group input:focus {  
    border-color: #6c5ce7;  
}
```

```
#collegeSuggestions {  
    border: 1px solid #ccc;  
    border-radius: 5px;  
    max-height: 200px;  
    overflow-y: auto;  
    background-color: white;  
    display: none;  
}  
  
#collegeSuggestions div {  
    padding: 8px;  
    cursor: pointer;  
}  
  
#collegeSuggestions div:hover {  
    background-color: #f0f0f0;  
}  
  
#message {  
    text-align: center;  
    margin-top: 20px;  
    padding: 10px;  
    display: none;  
}  
  
#message.success {  
    background-color: #28a745;  
    color: white;
```

```
}

#message.error {

    background-color: #dc3545;

    color: white;

}

#message.info {

    background-color: #17a2b8;

    color: white;

}

button {

    width: 100%;

    padding: 10px;

    background-color: #6c5ce7;

    color: white;

    border: none;

    border-radius: 5px;

    cursor: pointer;

}

button:hover {

    background-color: #5e4eb2;

}

</style>

</head>

<body>
```

```
<div class="container">

  <h2>Registration Form</h2>

  <form id="registerForm">

    <div class="form-group">

      <label for="name">Name</label>

      <input type="text" id="name" name="name" placeholder="Enter your name" required>

    </div>

    <div class="form-group">

      <label for="college">College</label>

      <input type="text" id="college" name="college" placeholder="Enter your college"
oninput="showSuggestions()">

      <div id="collegeSuggestions"></div>

    </div>

    <div class="form-group">

      <label for="username">Username</label>

      <input type="text" id="username" name="username" placeholder="Enter your username"
required>

    </div>

    <div class="form-group">

      <label for="password">Password</label>

      <input type="password" id="password" name="password" placeholder="Enter your
password" required>

    </div>

    <div class="form-group">

      <label for="repassword">Retype Password</label>
```

```
        <input type="password" id="repassword" name="repassword" placeholder="Retype your
password" required>
```

```
    </div>
```

```
    <button type="button" onclick="registerUser()">Register</button>
```

```
</form>
```

```
<div id="message"></div>
```

```
</div>
```

```
<script>
```

```
    const existingUsernames = ["john_doe", "jane_smith", "komal28"]; // Example existing usernames
```

```
    const collegeNames = ["Harvard", "MIT", "Stanford", "Oxford", "Cambridge", "Columbia",
"VESIT", "Vidyalankar", "VJTI"]; // Example college names
```

```
function showSuggestions() {
```

```
    const collegeInput = document.getElementById("college");
```

```
    const suggestionsBox = document.getElementById("collegeSuggestions");
```

```
    const query = collegeInput.value.toLowerCase();
```

```
    const filteredColleges = collegeNames.filter(college => college.toLowerCase().includes(query));
```

```
    suggestionsBox.innerHTML = ""; // Clear previous suggestions
```

```
    if (query !== "") {
```

```
        filteredColleges.forEach(college => {
```

```
            const div = document.createElement("div");
```



```
        div.textContent = college;

        div.onclick = () => selectCollege(college);

        suggestionsBox.appendChild(div);

    });

    suggestionsBox.style.display = "block";

} else {

    suggestionsBox.style.display = "none";

}

}

function selectCollege(college) {

    document.getElementById("college").value = college;

    document.getElementById("collegeSuggestions").style.display = "none";

}

function registerUser() {

    const name = document.getElementById("name").value.trim();

    const college = document.getElementById("college").value.trim();

    const username = document.getElementById("username").value.trim();

    const password = document.getElementById("password").value;

    const repassword = document.getElementById("repassword").value;

    // Basic validation

    if (!name || !college || !username || !password || !repassword) {
```

```
    showMessage("✗ Please fill in all fields.", "error");

    return;
}

if (existingUsernames.includes(username)) {

    showMessage("✗ Username already taken. Please choose another one.", "error");

    return;
}

if (password !== repassword) {

    showMessage("✗ Passwords do not match.", "error");

    return;
}

// Simulate sending data asynchronously with XMLHttpRequest

const xhr = new XMLHttpRequest();

xhr.open("POST", "dummy-url", true); // Simulating server interaction

xhr.onload = function () {

    if (xhr.status === 200) {

        showMessage("✓ Successfully Registered!", "success");

        existingUsernames.push(username); // Simulate storing the new username

        document.getElementById("registerForm").reset();

        document.getElementById("collegeSuggestions").style.display = "none";

    } else {
```

```
        showMessage("✗ Registration failed due to server error.", "error");
    }
};

xhr.onerror = function () {
    showMessage("✗ Registration failed due to network error.", "error");
};

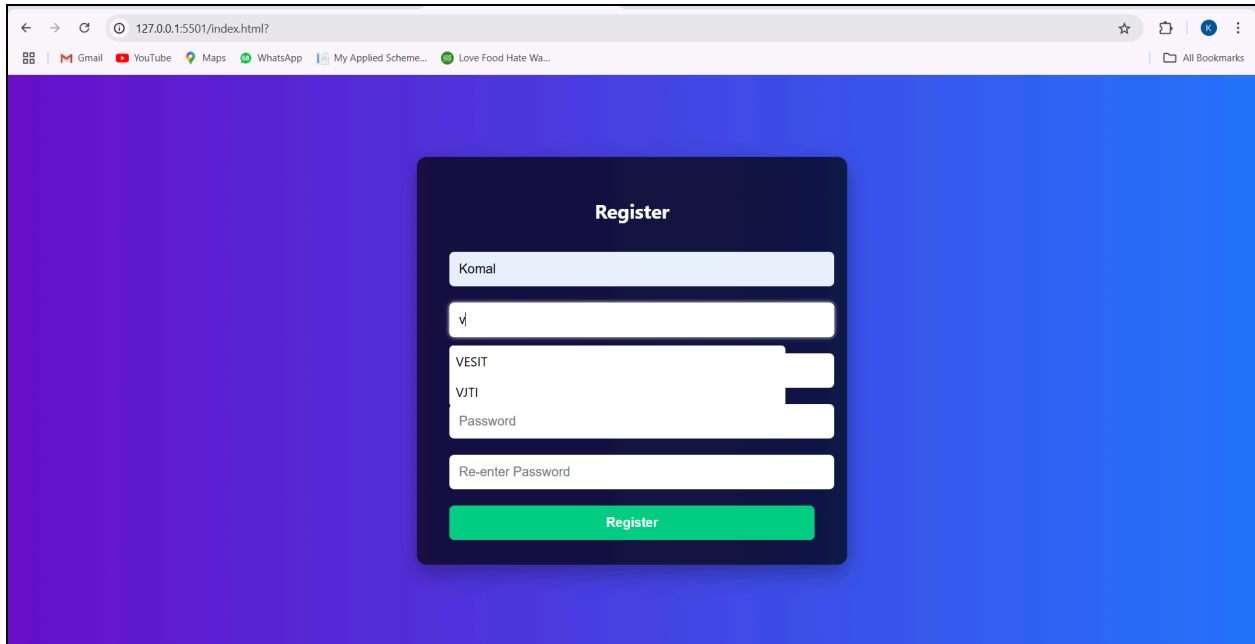
xhr.send();
}

function showMessage(message, type) {
    const messageBox = document.getElementById("message");
    messageBox.textContent = message;
    messageBox.className = type; // 'success' or 'error'
    messageBox.style.display = "block";
}
</script>

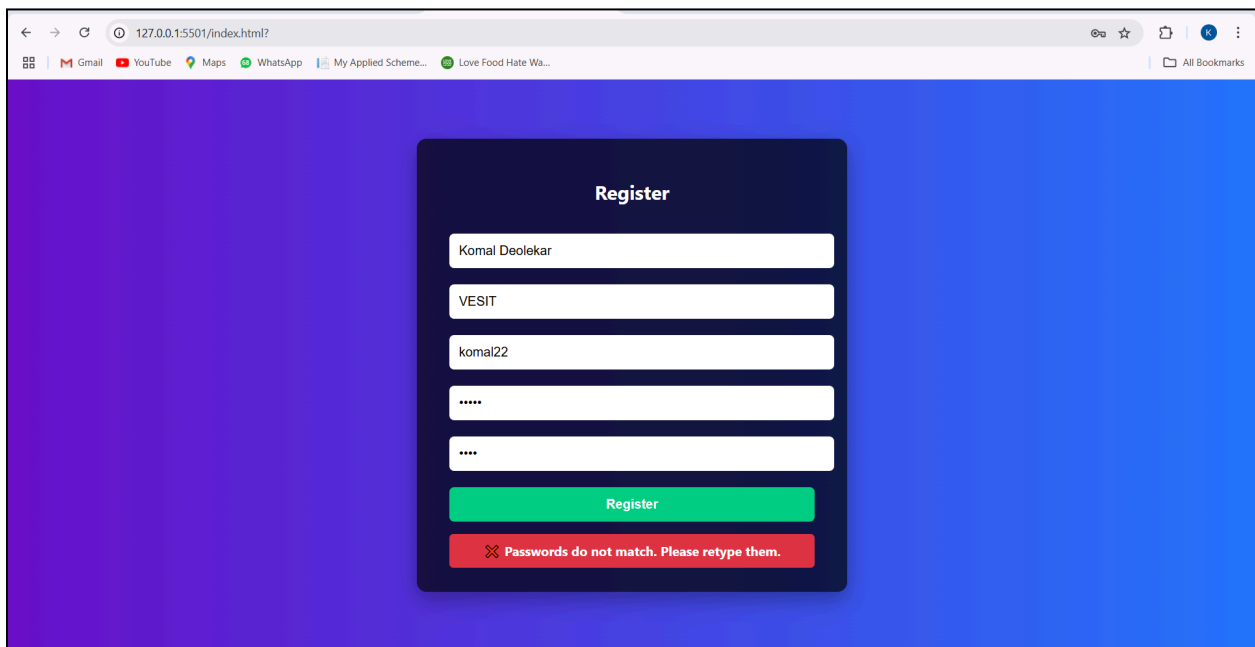
</body>

</html>
```

**Output:**



A screenshot of a web browser displaying a registration form. The browser's address bar shows the URL "127.0.0.1:5501/index.html?". The form is titled "Register" and is set against a dark blue background. It contains the following fields: a text input with "Komal", a text input with "vj", a text input with "VESIT", a text input with "VJTI", a password input with "Password", and a "Re-enter Password" input. A green "Register" button is at the bottom of the form.



A screenshot of the same web browser showing the registration form after an attempt. The form fields now contain: "Komal Deolekar", "VESIT", "komal22", and two password fields, both filled with dots. A green "Register" button is present. Below the button, a red error message box displays: "✖ Passwords do not match. Please retype them.".

127.0.0.1:5501/index.html? | Gmail | YouTube | Maps | WhatsApp | My Applied Scheme... | Love Food Hate Wa... | All Bookmarks

### Register

Komal Deolekar

VESIT

komal98

\*\*\*\*

\*\*\*\*

Register

✖ Username already exists. Try another one.

127.0.0.1:5501/index.html? | Gmail | YouTube | Maps | WhatsApp | My Applied Scheme... | Love Food Hate Wa... | All Bookmarks

### Register

Komal Deolekar

VESIT

komal22

\*\*\*\*\*

\*\*\*\*\*

Register

✖ Registration failed due to server error.

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5501/index.html?'. The browser's bookmark bar includes links to Gmail, YouTube, Maps, WhatsApp, My Applied Scheme..., and Love Food Hate Wa... The main content area features a dark blue gradient background with a central white 'Register' form. The form contains the following elements:

- A 'Full Name' input field.
- A 'VESIT' input field with a red error message icon and the text 'Please fill out this field.'
- An 'Email' input field containing the text 'komal25'.
- A 'Password' input field masked with four asterisks (\*\*\*\*).
- A 'Confirm Password' input field masked with four asterisks (\*\*\*\*).
- A green 'Register' button at the bottom.

## Conclusion :

This practical implementation showcases the power of AJAX in creating dynamic and responsive web applications. By using the XMLHttpRequest object, we were able to asynchronously check existing usernames, validate inputs, and provide real-time feedback to the user. The college field auto-suggestion feature added a modern touch, enhancing usability. Through this exercise, we learned the importance of asynchronous communication in web development, how to handle user input validations efficiently, and how to provide seamless interactions between client and server—all without disrupting the user's experience. This approach is fundamental in building smooth, user-friendly applications and forms the basis for more advanced features in modern web development.