

## EXPERIMENT NO. 9: AJAX

Name of Student	Swaraj Patil
Class Roll No	D15A_39
D.O.P.	06/03/2025
D.O.S.	13/03/2025
Sign and Grade	

**AIM :** To study AJAX

### **PROBLEM STATEMENT :**

Create a registration page having fields like **Name**, **College**, **Username**, and **Password** (password is to be entered twice).

Validate the form by checking:

- Name field is not empty
- Username is not same as existing entries
- Password and Confirm Password fields match
- Auto-suggest college names
- On successful registration, show the message "**Successfully Registered**" below the Submit button

Let all page updates be **asynchronously loaded**. Implement using **XMLHttpRequest Object**

### **THEORY :**

#### **1. How do Synchronous and Asynchronous Requests differ?**

Synchronous Requests	Asynchronous Requests
Blocks the execution of code	Doesn't block the execution
Waits for the server response	Continues executing while waiting for response

Slower user experience	Faster, smoother user experience
Used less in modern web applications	Preferred in modern web applications

## 2. Describe various properties and methods used in XMLHttpRequest Object

Properties:

- `readyState`: Describes the state of the request (0 to 4)
- `status`: HTTP status code (e.g., 200 = OK)
- `responseText`: Gets the response data as a string
- `responseXML`: Gets the response data as XML

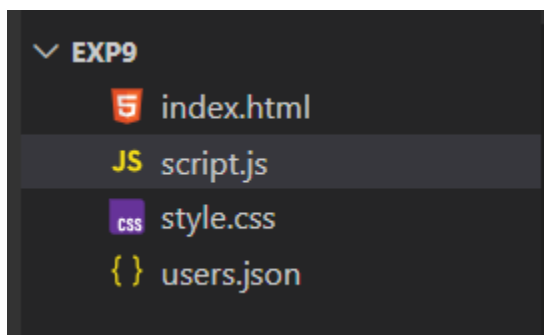
Methods:

- `open(method, url, async)`: Initializes the request
- `send(data)`: Sends the request
- `setRequestHeader(header, value)`: Sets HTTP headers
- `onreadystatechange`: Event triggered when `readyState` changes

**GITHUB LINK** - [https://github.com/Anuprita2022-26/WebX\\_Exp9](https://github.com/Anuprita2022-26/WebX_Exp9)

**CODE:**

### a) Folder Structure



### b) index.html

```

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

```

</head>
<body>
  <div class="container">
    <h1>Register</h1>
    <form id="registerForm">
      <input type="text" id="name" placeholder="Name"
required />
      <input type="text" id="college" placeholder="College"
list="collegeList" required />
      <datalist id="collegeList"></datalist>
      <input type="text" id="username"
placeholder="Username" required />
      <input type="password" id="password"
placeholder="Password" required />
      <input type="password" id="confirmPassword"
placeholder="Confirm Password" required />
      <button type="submit">Register</button>
    </form>
    <div id="message"></div>
    <button id="addNewBtn" style="display:none;">Add
New</button>
  </div>
  <script src="script.js"></script>
</body>
</html>

```

### c) script.js

```

document.getElementById('registerForm').addEventListener('s
ubmit', function (e) {
  e.preventDefault();

  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 confirmPassword =
document.getElementById('confirmPassword').value;
const messageBox = document.getElementById('message');
const addNewBtn = document.getElementById('addNewBtn');

messageBox.innerText = '';
messageBox.style.color = 'red';
addNewBtn.style.display = 'none';

if (!name) {
  messageBox.innerText = 'Name cannot be empty!';
  return;
}

if (password !== confirmPassword) {
  messageBox.innerText = 'Passwords do not match!';
  return;
}

const xhr = new XMLHttpRequest();
xhr.open('GET', 'http://localhost:3000/users', true);
xhr.onload = function () {
  if (xhr.status === 200) {
    const users = JSON.parse(xhr.responseText);
    const userExists = users.some(user => user.username
=== username);

    if (userExists) {
      messageBox.innerText = 'Username already exists!';
    } else {
      const newUser = {
```

```

        name,
        college,
        username,
        password
    };

    const xhrPost = new XMLHttpRequest();
    xhrPost.open('POST', 'http://localhost:3000/users',
true);

    xhrPost.setRequestHeader('Content-Type',
'application/json');
    xhrPost.onload = function () {
        if (xhrPost.status === 201) {
            messageBox.innerText = '✔ Successfully
Registered!';
            messageBox.style.color = 'green';
            addNewBtn.style.display = 'inline-block';

            // Disable form fields
            document.querySelectorAll('#registerForm input,
#registerForm button[type="submit"]').forEach(el => {
                el.disabled = true;
            });
        } else {
            messageBox.innerText = 'Something went wrong!';
        }
    };
    xhrPost.send(JSON.stringify(newUser));
}
};
xhr.send();
});

// Add new button reset

```

```
document.getElementById('addNewBtn').addEventListener('click', function () {
    document.getElementById('registerForm').reset();
    document.getElementById('message').innerText = '';
    this.style.display = 'none';

    // Enable form again
    document.querySelectorAll('#registerForm input,
    #registerForm button[type="submit"]').forEach(el => {
        el.disabled = false;
    });
});

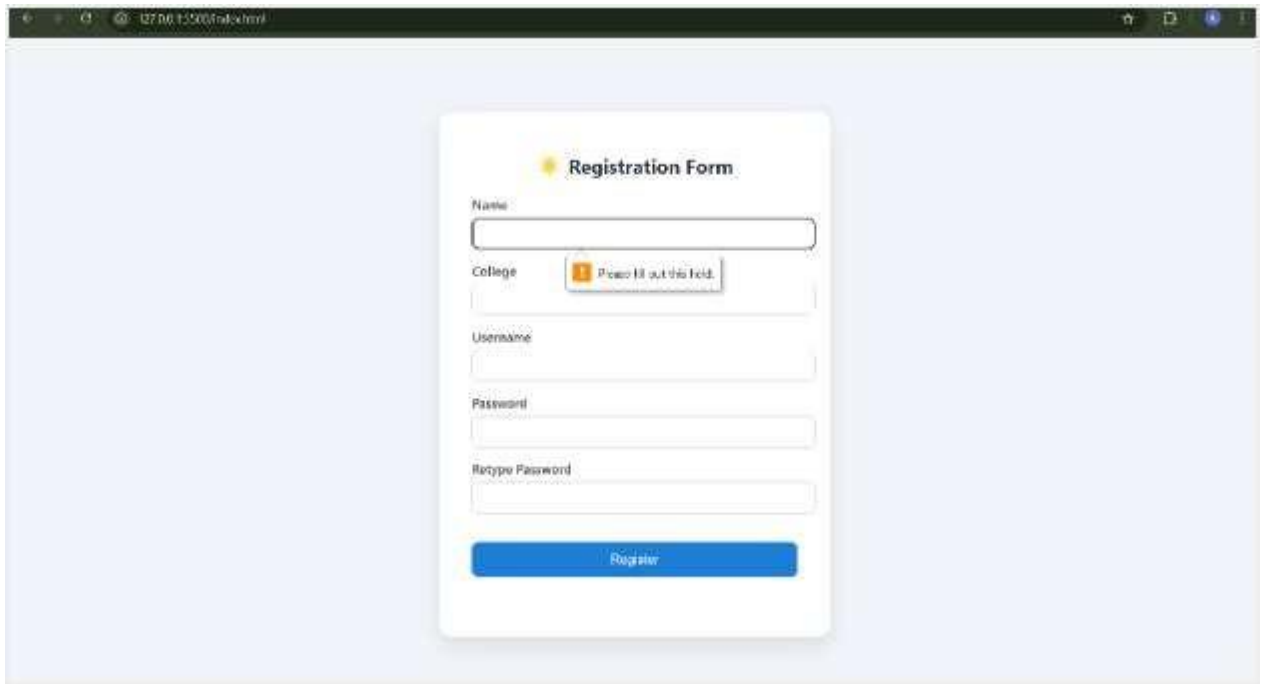
// Auto-suggest colleges
const collegeNames = ["VESIT", "DJ Sanghvi", "Sardar
Patel", "KJ Somaiya", "VJTI"];
const datalist = document.getElementById("collegeList");
collegeNames.forEach(name => {
    const option = document.createElement("option");
    option.value = name;
    datalist.appendChild(option);
});
```

**d) users.json**

```
{
  "users": []
}
```

## OUTPUT:

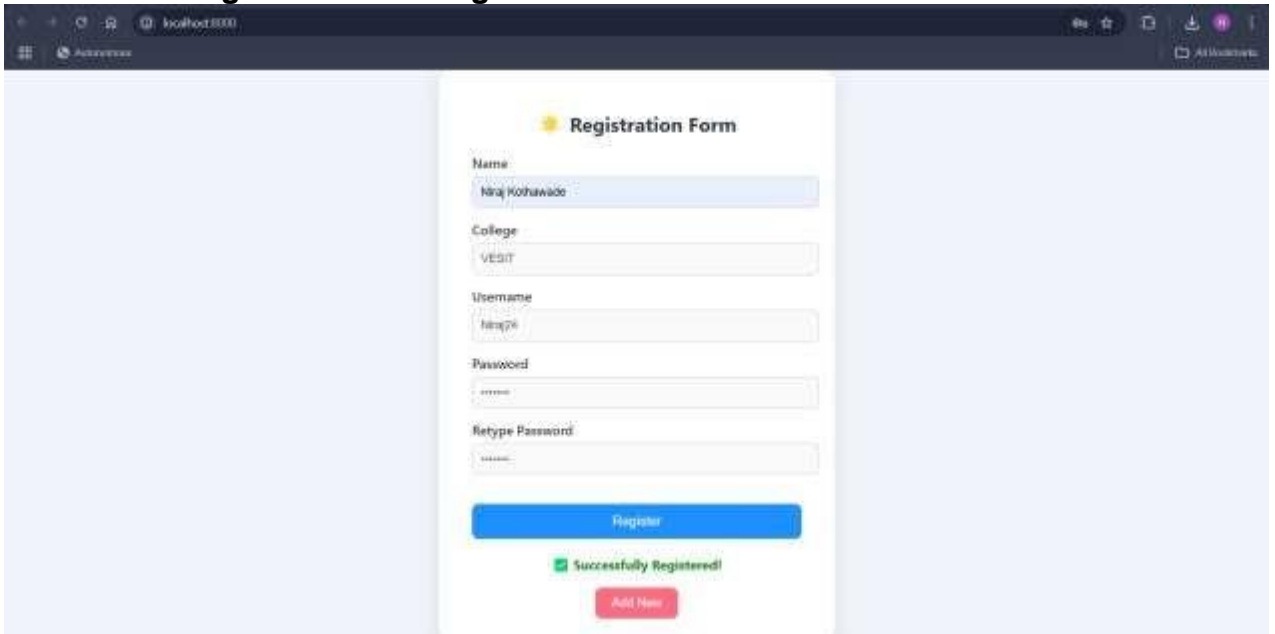
### a) Registration Form Display



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000/index.html'. The main content area features a white registration form titled 'Registration Form' with a yellow star icon. The form contains the following fields: 'Name' (empty), 'College' (empty with a red error message 'Please fill out this field.'), 'Username' (empty), 'Password' (empty), and 'Retype Password' (empty). A blue 'Register' button is located at the bottom of the form.

This screenshot displays the registration form with input fields for Name, College, Username, Password, and Confirm Password, ensuring that the Name field is not left empty.

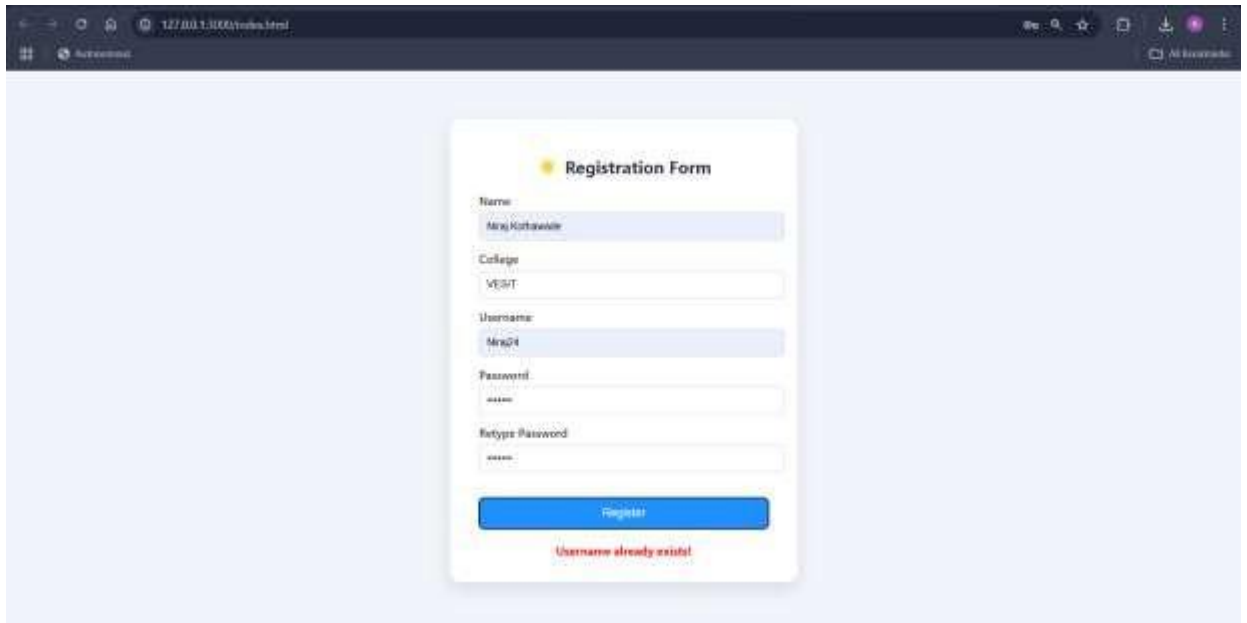
### b) Successful Registration Message



The screenshot shows the same web browser window, but the registration form is now filled out. The 'Name' field contains 'Niraj Kothawade', 'College' contains 'VESIT', 'Username' contains 'niraj76', 'Password' contains '\*\*\*\*\*', and 'Retype Password' contains '\*\*\*\*\*'. The blue 'Register' button is still present. Below the form, a green checkmark icon is followed by the text 'Successfully Registered!'. A red 'Add Note' button is located at the bottom of the form.

This screenshot shows the **"Successfully Registered!"** message, which appears after a successful registration.

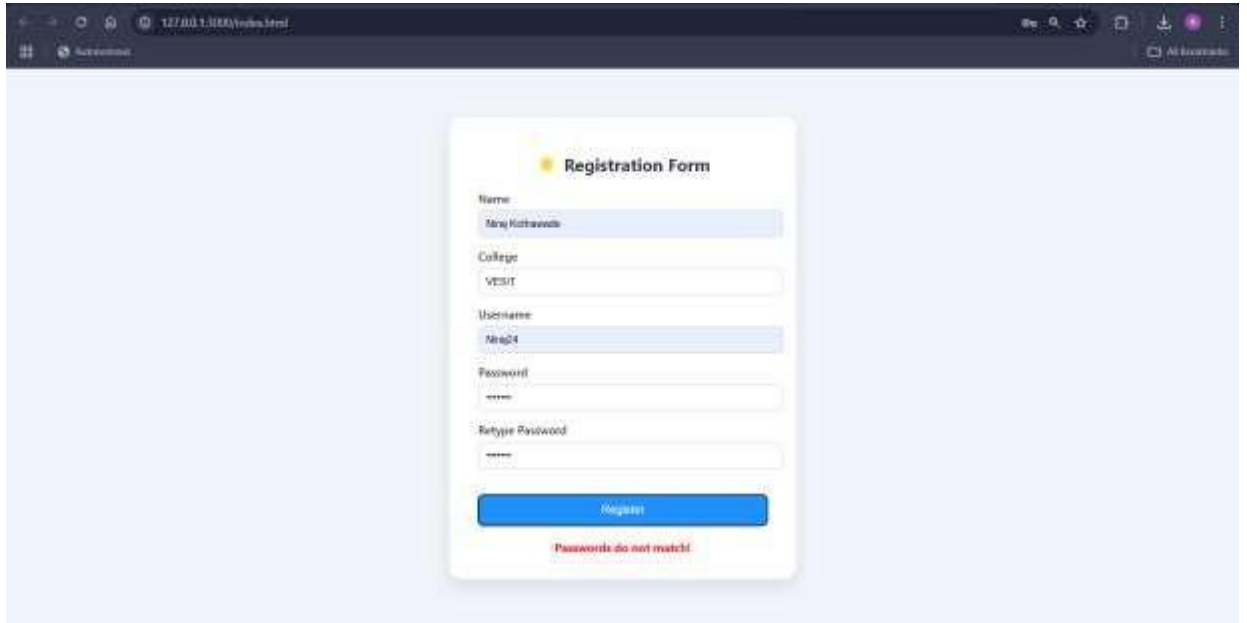
### c) Duplicate Username Validation



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:3000/index.html'. The page contains a 'Registration Form' with the following fields: Name (filled with 'Anuj Kothawale'), College (filled with 'VESIT'), Username (filled with 'Anuj04'), Password (filled with '123456'), and Retype Password (filled with '123456'). A blue 'Register' button is at the bottom. Below the button, a red error message reads 'Username already exists!'.

This screenshot validates that the **Username is not already in use**, preventing duplicate entries.

### d) Password Match Confirmation

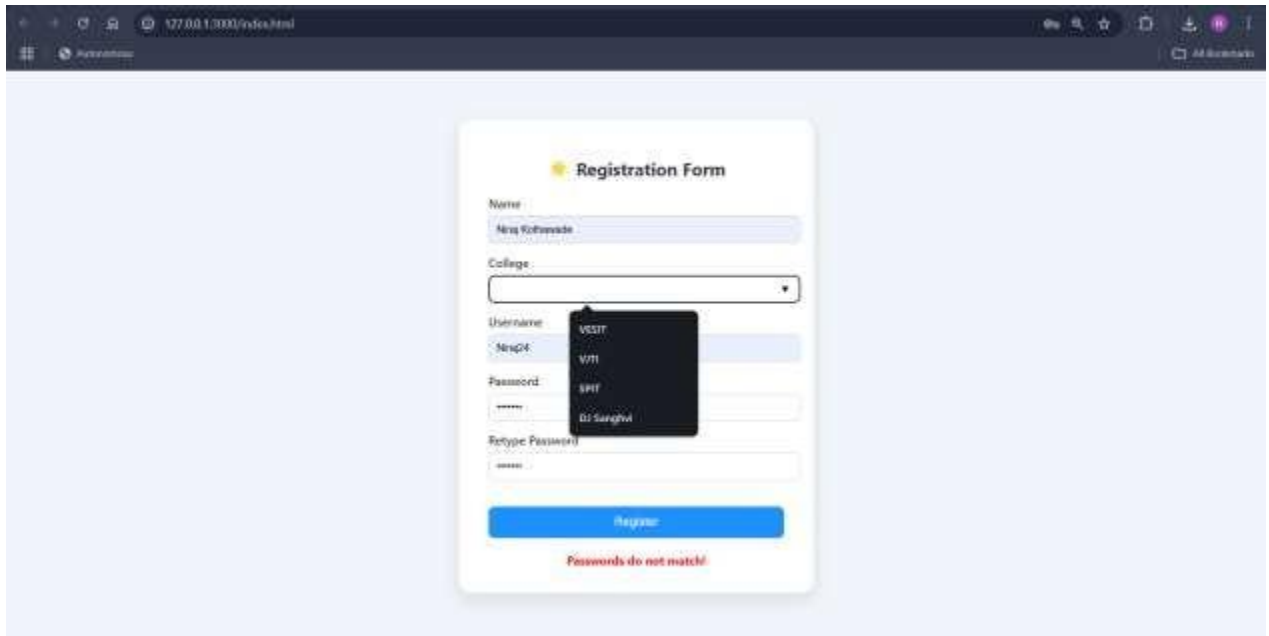


The screenshot shows a web browser window with the address bar displaying '127.0.0.1:3000/index.html'. The page contains a 'Registration Form' with the following fields: Name (filled with 'Anuj Kothawale'), College (filled with 'VESIT'), Username (filled with 'Anuj04'), Password (filled with '123456'), and Retype Password (filled with '123456'). A blue 'Register' button is at the bottom. Below the button, a red error message reads 'Passwords do not match!'.

This screenshot confirms that the **Password and Re-typed Password match**, ensuring data integrity.



### e) College Name Auto-suggestion



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:3000/index.html'. The page contains a 'Registration Form' with the following fields: 'Name' (text input with 'Nrup Kothawade'), 'College' (dropdown menu), 'Username' (text input with 'NrupG4'), 'Password' (password input with '1234'), and 'Re-type Password' (password input with '1234'). A blue 'Register' button is at the bottom. A red error message 'Passwords do not match!' is displayed below the button. A dark dropdown menu is open for the 'College' field, showing suggestions: 'VJIT', 'VIT', 'BIT', and 'BIT Sangli'.

This screenshot demonstrates the **auto-suggestion feature for the College field**, where users can choose from suggested college names.

### CONCLUSION:

The experiment successfully demonstrated the use of the **XMLHttpRequest object** to implement **AJAX-based asynchronous form submission and validation**. Key features such as **form field validation**, **duplicate username detection**, **password match checking**, and **college name auto-suggestions** were efficiently implemented without reloading the page. This experiment highlighted the effectiveness of **AJAX in enhancing user experience** by allowing **dynamic content updates** and **real-time feedback** during user interaction.