

EXPERIMENT NO. 9: AJAX

Name of Student	Rohan Lalchandani
Class Roll No	D15A_25
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:

- 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:

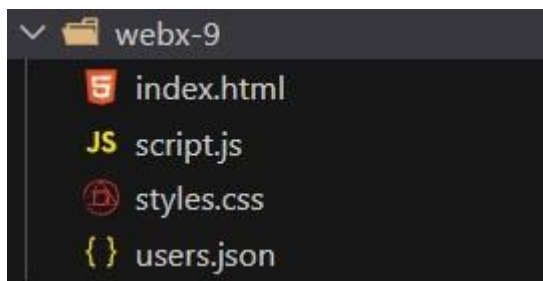
- open(method, url, async): Initializes the request
- send(data): Sends the request
- setRequestHeader(header, value): Sets HTTP headers
- onreadystatechange: Event triggered when readyState changes

GITHUB LINK –

https://github.com/Rohan-Lalchandani08/WebX_Lab/tree/main/WebX%20Exp%209

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="styles.css" />
</head>
<body>
  <div class="container">
    <h1>Register</h1>
    <form id="registerForm">
      <label for="name">Name</label>
      <input type="text" id="name" required />

      <label for="college">College</label>
      <input type="text" id="college" list="collegeList" required />
      <datalist id="collegeList"></datalist>
```

```

        <label for="username">Username</label>
        <input type="text" id="username" required />

        <label for="password">Password</label>
        <input type="password" id="password" required />

        <label for="confirmPassword">Retype Password</label>
        <input type="password" id="confirmPassword" 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

```

function showMessage(text, color = 'red') {  const
messageBox = document.getElementById('message');
messageBox.innerText = text;  messageBox.style.color =
color;  messageBox.style.opacity = 1;

    if (messageBox.timeoutId) {
        clearTimeout(messageBox.timeoutId);
    }
    messageBox.timeoutId = setTimeout(() => {
messageBox.innerText = "";
    }, 15000);
}

document.getElementById('registerForm').addEventListener('submit', 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 addNewBtn = document.getElementById('addNewBtn');
addNewBtn.style.display = 'none';

    if (!name) {  showMessage('Name
cannot be empty!');  return;
    }
}

```

```

    if (password !== confirmPassword) {
    showMessage('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) {
        showMessage('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) {
            document.querySelectorAll('#registerForm input, #registerForm
            button[type="submit"]').forEach(el => {    el.disabled = true;
            });

            setTimeout(() => {    showMessage('☑
            Successfully Registered!', 'green');
            addNewBtn.style.display = 'inline-block';    }, 100);
            } else {
            showMessage('Something went wrong!');
            }
            };

            xhrPost.send(JSON.stringify(newUser));
            }
            } else {    showMessage('Failed to
            fetch users!');
            }
            };

```

```

    xhr.onerror = function () {
    showMessage('Network error occurred!');
    };

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

      document.querySelectorAll('#registerForm input, #registerForm
  button[type="submit"]').forEach(el => { el.disabled = false;
  });
  });
  const collegeNames = ["VESIT", "DJ Sanghvi", "Sardar Patel", "KJ Somaiya", "VJTI"];
  const datalist = document.getElementById('collegeList');

  collegeNames.forEach(college => { const option
  = document.createElement('option');
  option.value = college;
  datalist.appendChild(option);
  });

```

c) users.json

```

{
  users:[]
}

```

Output:

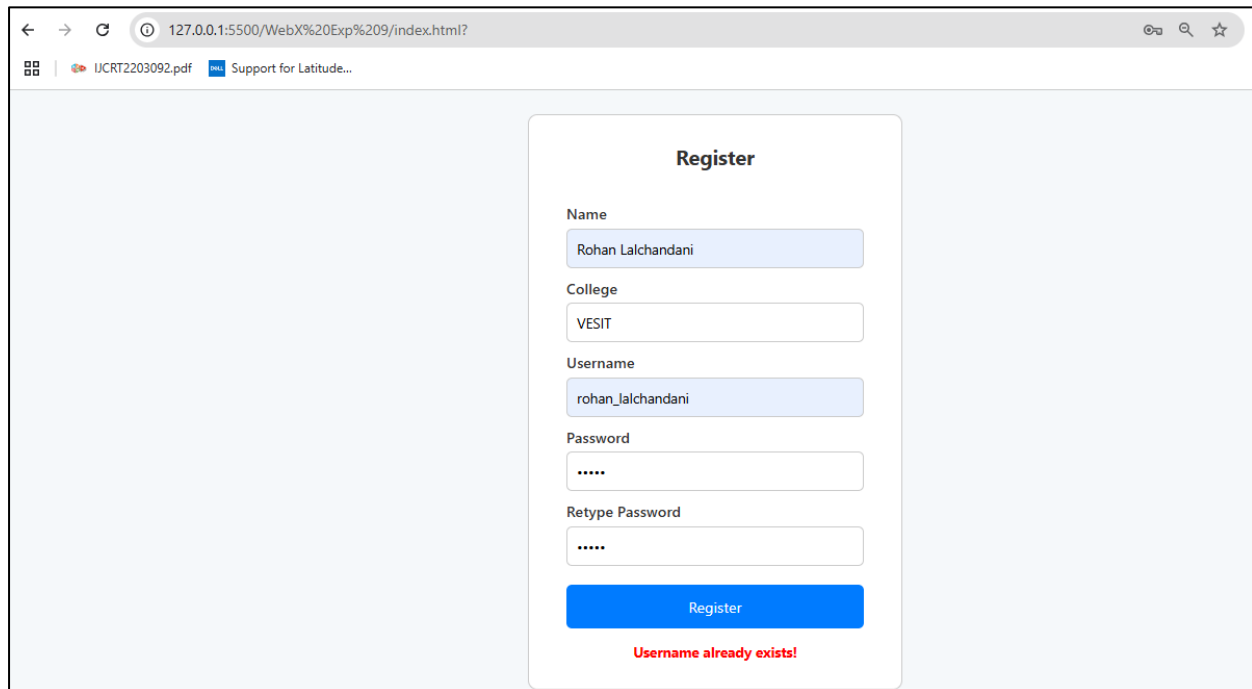
a) Successful Registration Message

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5500/WebX%20Exp%209/index.html?'. The page content is a registration form titled 'Register'. The form includes the following fields and elements:

- Name:** A text input field containing 'Rohan Lalchandani'.
- College:** A text input field containing 'VESIT'.
- Username:** A text input field containing 'rohan_lalchandani'.
- Password:** A password input field with masked characters '*****'.
- Retype Password:** A password input field with masked characters '*****'.
- Register Button:** A blue button labeled 'Register'.
- Success Message:** A green checkmark icon followed by the text 'Successfully Registered!'.
- Add New Button:** A green button labeled 'Add New' located at the bottom of the form.

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

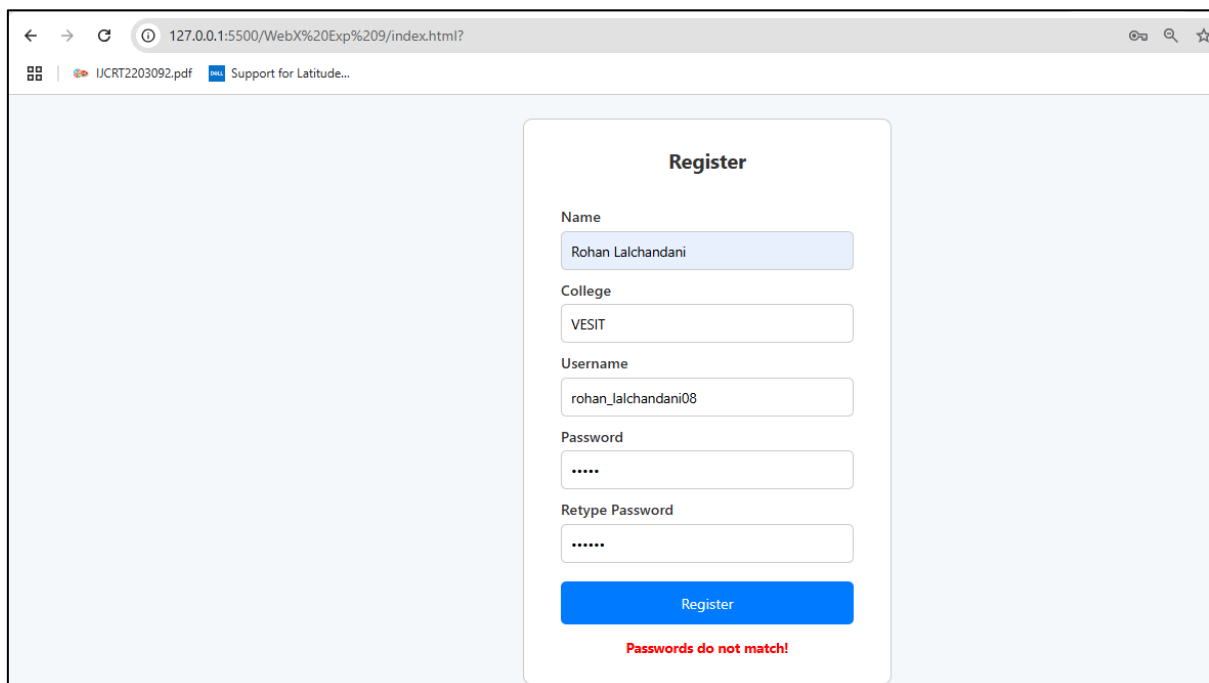
b)Duplicate Username Validation



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5500/WebX%20Exp%209/index.html?". The browser tabs include "UCRT2203092.pdf" and "Support for Latitude...". The main content area displays a "Register" form with the following fields: Name (Rohan Lalchandani), College (VESIT), Username (rohan_lalchandani), Password (masked with dots), and Retype Password (masked with dots). A blue "Register" button is at the bottom of the form. Below the button, a red error message states "Username already exists!".

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

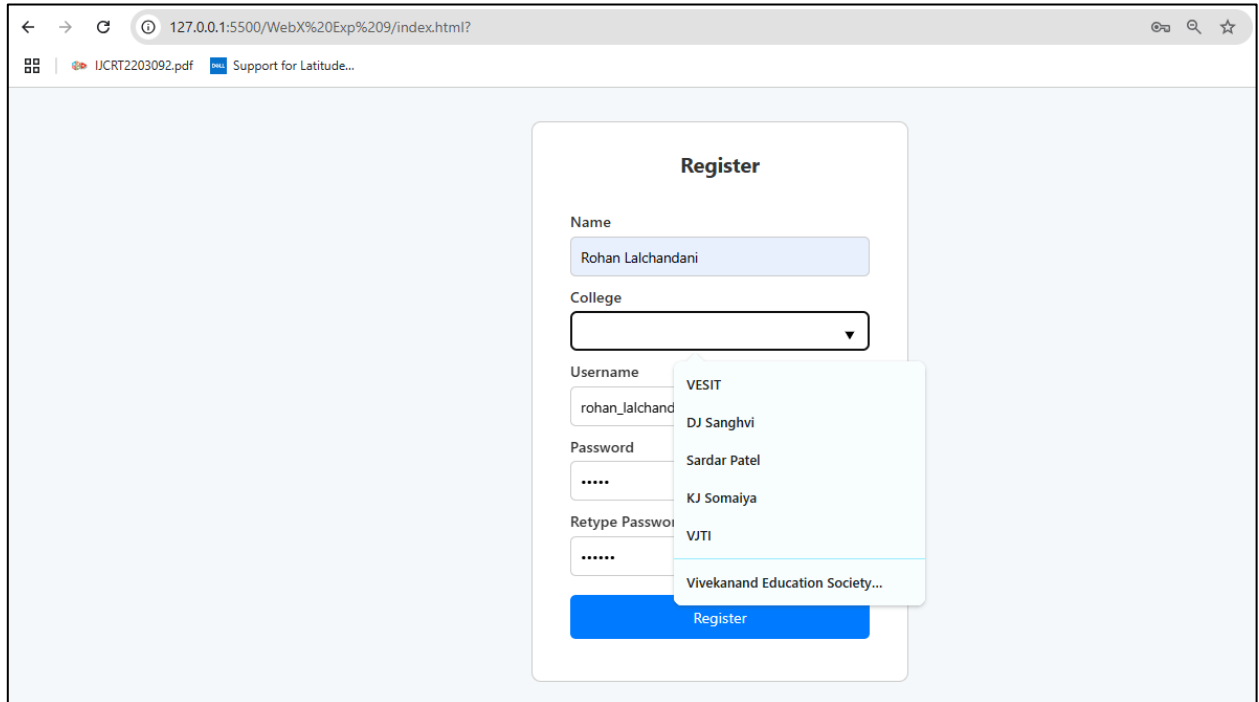
c)Password Match Confirmation



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5500/WebX%20Exp%209/index.html?". The browser tabs include "UCRT2203092.pdf" and "Support for Latitude...". The main content area displays a "Register" form with the following fields: Name (Rohan Lalchandani), College (VESIT), Username (rohan_lalchandani08), Password (masked with dots), and Retype Password (masked with dots). A blue "Register" button is at the bottom of the form. Below the button, a red error message states "Passwords do not match!".

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

d) College Name Auto-suggestion



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5500/WebX%20Exp%209/index.html?". The browser's tab bar shows two tabs: "UCRT2203092.pdf" and "Support for Latitude...". The main content area displays a "Register" form. The form has the following fields and values:

- Name:** Rohan Lalchandani
- College:** A dropdown menu is open, showing a list of suggested college names: VESIT, DJ Sanghvi, Sardar Patel, KJ Somaiya, VJTI, and Vivekanand Education Society... (truncated).
- Username:** rohan_lalchandani
- Password:** *****
- Retype Password:** *****

A blue "Register" button is located at the bottom of the form.

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.