

EXPERIMENT NO. 9: AJAX

Name of Student	Eesha Chavan
Class Roll No	D15A 08
D.O.P.	
D.O.S.	
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

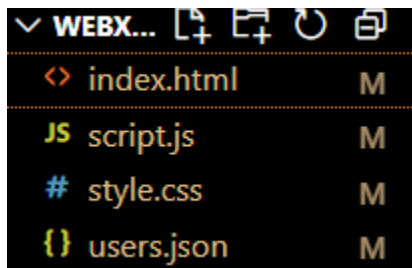
Methods:

- `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/eeshachavan/WebX-Exp9-Eesha>

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">

    <h2>Registration Form</h2>

    <form id="registerForm" class="form-box">

      <label>Name <input type="text" id="name" required /></label>

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

      <datalist id="collegeList">

        <option value="VESIT" />

        <option value="VJTI" />

        <option value="SPIT" />

        <option value="DJ Sanghvi" />

      </datalist>

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

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

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

      <button type="submit">Register</button>

    </form>

    <div id="message" class="message-box"></div>

    <button id="addNewBtn" class="add-new-btn" style="display: none;">Add
New</button>

  </div>

  <script src="script.js"></script>

</body>

</html>
```

c) script.js

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

    // Clear previous error messages
    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';
        }
        else {
            messageBox.innerText = 'Something went wrong!';
        }
    };
    xhrPost.send(JSON.stringify(newUser));
}
}
};
xhr.send();
});

// Reset form on clicking "Add New"
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 = true;
    });
});
});

```

d) users.json

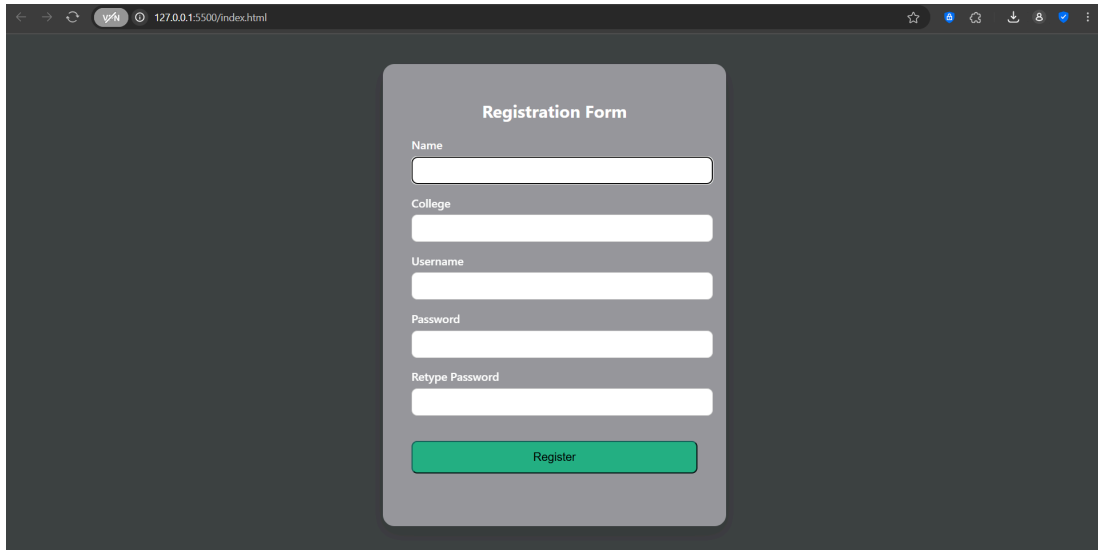
```

{
  "users": []
}

```

OUTPUT:

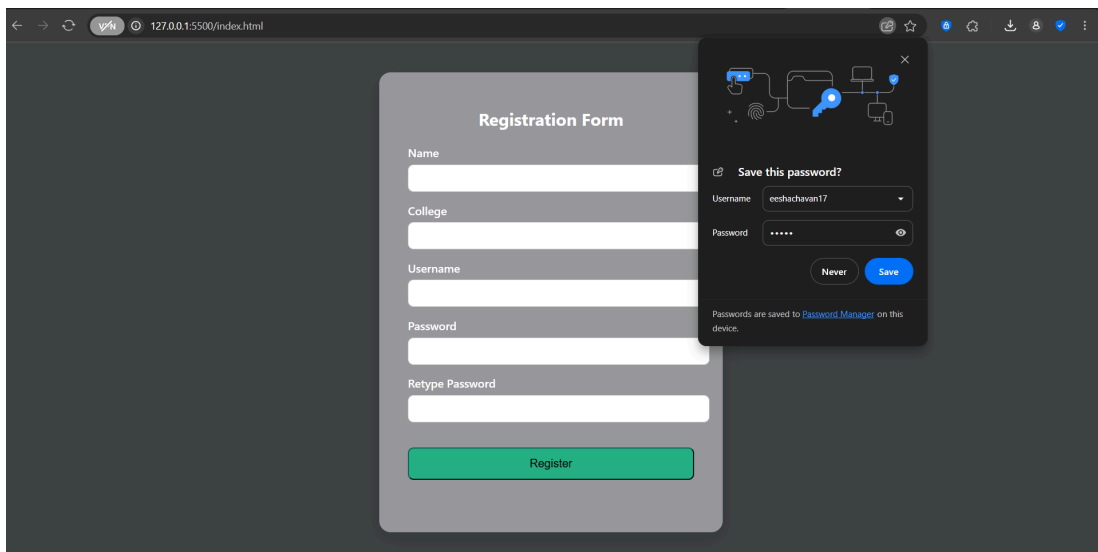
Registration Form Display



A screenshot of a web browser displaying a registration form. The browser's address bar shows the URL "127.0.0.1:5500/index.html". The form is titled "Registration Form" and contains five input fields: "Name", "College", "Username", "Password", and "Retype Password". A green "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.

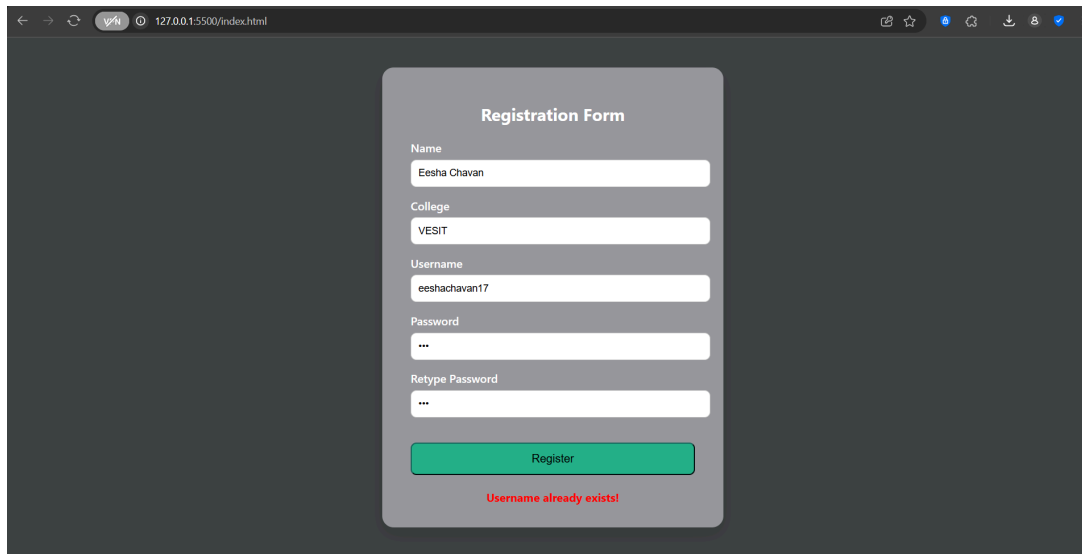
a) Successful Registration Message



A screenshot of a web browser displaying a successful registration message. The browser's address bar shows the URL "127.0.0.1:5500/index.html". The registration form is visible in the background. A dark overlay window is displayed in the foreground, showing a "Save this password?" dialog. The dialog contains the username "eeshachavan17" and a masked password "*****". There are "Never" and "Save" buttons. Below the buttons, it says "Passwords are saved to Password Manager on this device."

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

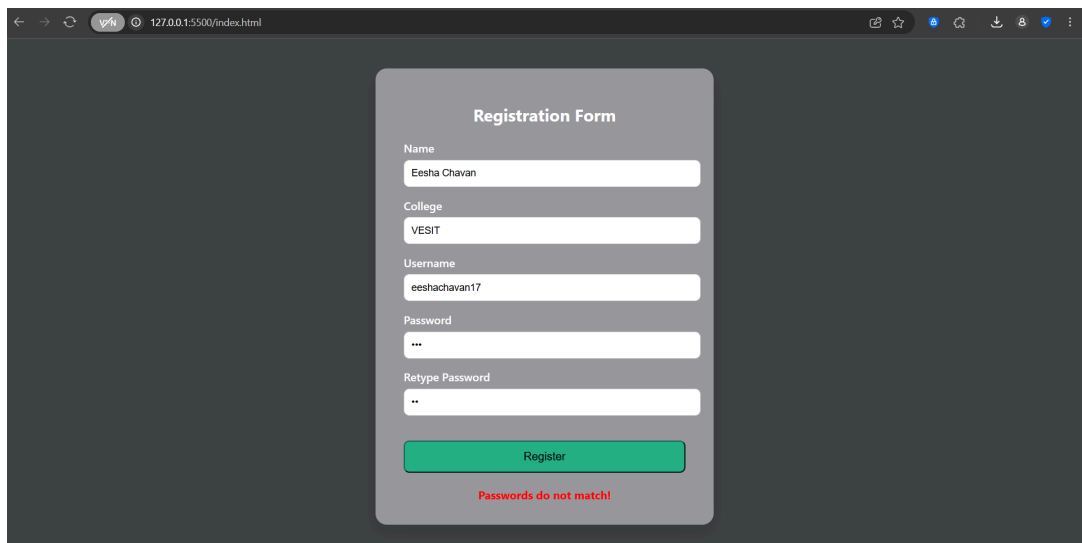
b) Duplicate Username Validation



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5500/index.html". The browser's developer tools are open, showing the "Elements" panel. The main content area displays a "Registration Form" with the following fields: Name (Esha Chavan), College (VESIT), Username (eeshachavan17), Password (masked with three dots), and Retype Password (masked with three dots). A green "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.

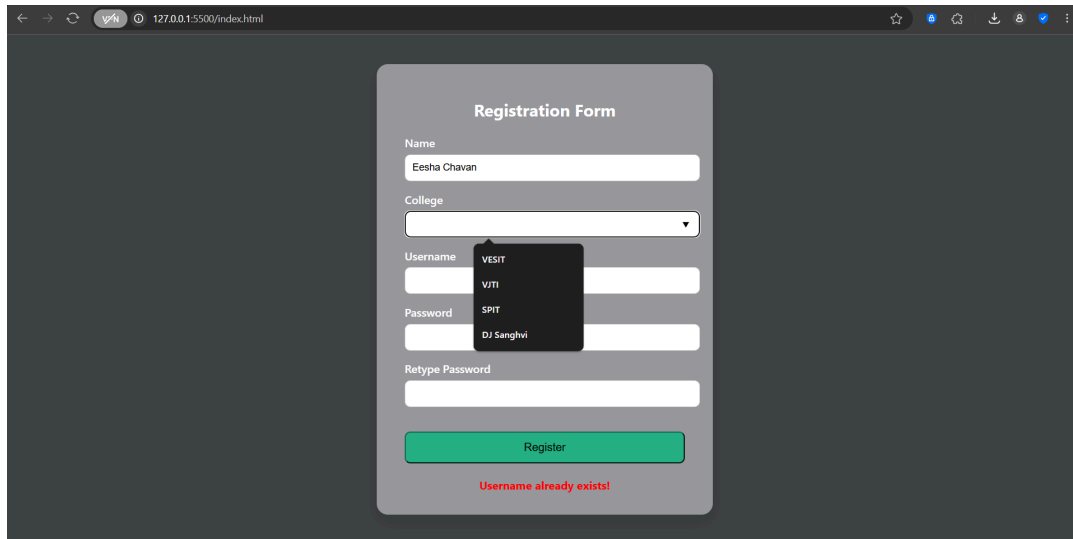
c) Password Match Confirmation



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5500/index.html". The browser's developer tools are open, showing the "Elements" panel. The main content area displays a "Registration Form" with the following fields: Name (Esha Chavan), College (VESIT), Username (eeshachavan17), Password (masked with three dots), and Retype Password (masked with two dots). A green "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.

d) College Name Auto-suggestion



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5500/index.html". The main content is a "Registration Form" with the following fields: "Name" (containing "Eesha Chavan"), "College" (a dropdown menu), "Username" (containing "VESIT"), "Password" (containing "SPIT"), and "Retype Password". A dropdown menu is open for the "College" field, showing suggestions: "VESIT", "VJI", "SPIT", and "DJ Sanghvi". Below the form is a green "Register" button. At the bottom of the form, there is a red error message: "Username already exists!".

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.