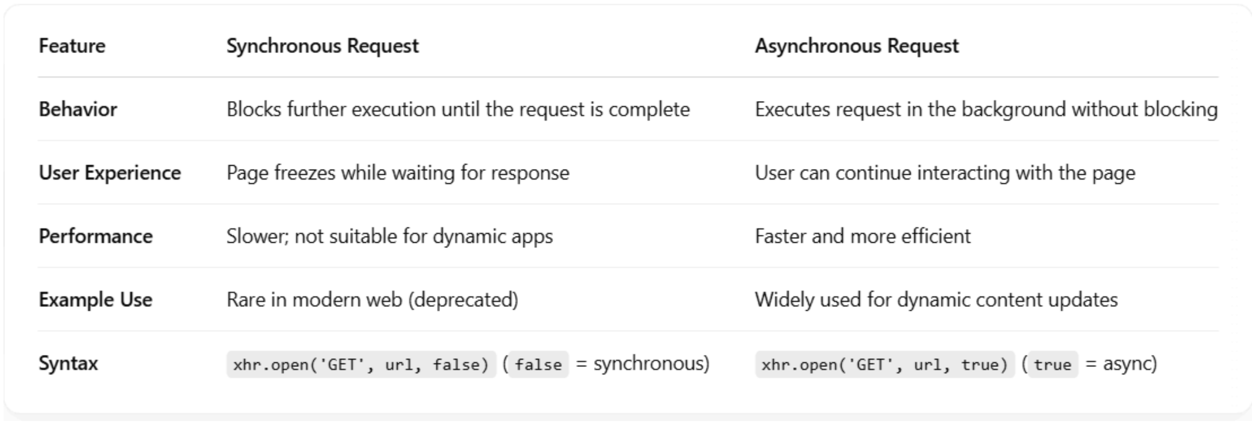
Experiment – 9: AJAX

Name of Student	**Rujuta Medhi**
Class Roll no	**D15A-27**
D.O.P	**14.04.25**
D.O.S	
Sign and Grade	

1. Aim: To study AJAX

2. Theory

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



Feature	Synchronous Request	Asynchronous Request
Behavior	Blocks further execution until the request is complete	Executes request in the background without blocking
User Experience	Page freezes while waiting for response	User can continue interacting with the page
Performance	Slower; not suitable for dynamic apps	Faster and more efficient
Example Use	Rare in modern web (deprecated)	Widely used for dynamic content updates
Syntax	`xhr.open('GET', url, false)` (`false` = synchronous)	`xhr.open('GET', url, true)` (`true` = async)

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

Properties of XMLHttpRequest Object:

readyState – This indicates the current state of the request. It changes through the following stages:

a) 0: UNSENT – Request is created but not opened.

b) 1: OPENED – open() method has been called.

c) 2: HEADERS_RECEIVED – send() has been called, and headers are available.

d) 3: LOADING – The response is being downloaded.

e) 4: DONE – The entire response has been received.

status – This gives the HTTP status code returned by the server, such as 200 for success or 404 for not found.
statusText – This provides the status message as a string, like "OK" or "Not Found".
responseText – This property holds the response body as a string (useful when expecting plain text or JSON).
responseXML – If the response is XML and the server sets the correct MIME type, this property gives a parsed XML document.
onreadystatechange – This is an event handler triggered whenever the readyState changes. You use this to handle the server's response when the request completes.

Methods of XMLHttpRequest Object:
open(method, url, async) – Initializes a request. The method is typically "GET" or "POST", url is the server URL, and async is a boolean indicating whether the request should be asynchronous (true) or synchronous (false).

send(data) – Sends the request. For GET, no data is passed. For POST, data like form content can be passed in the argument.

setRequestHeader(header, value) – Sets custom headers for the request. This is typically used to set content type, like application/json or application/x-www-form-urlencoded.

abort() – Cancels the request if it's still in progress.

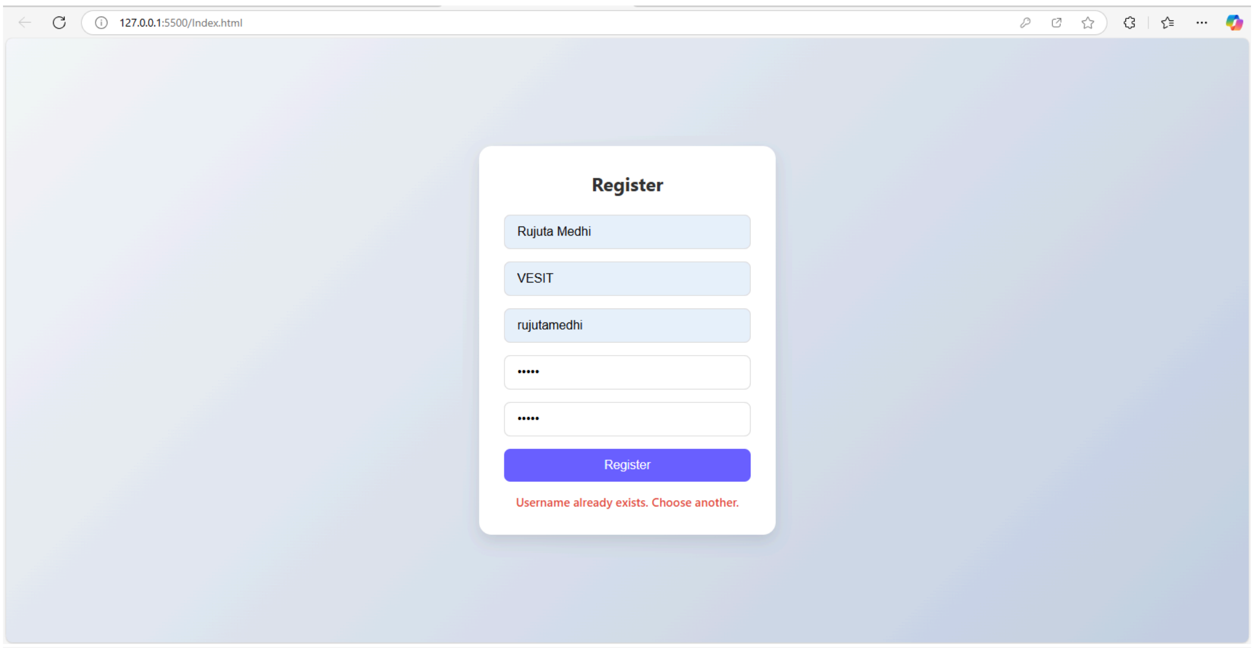
3. Problem Statement:

Create a registration page having fields like Name, College, Username and Password (read password twice).
Validate the form by checking for
1. Usernameis 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.

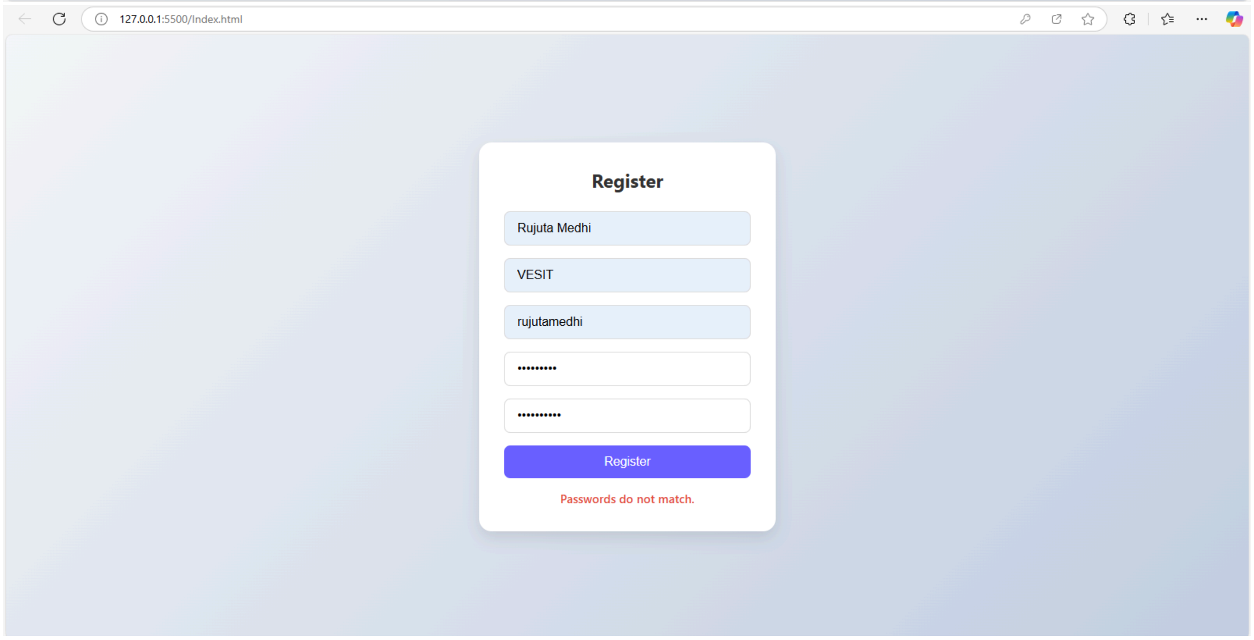
4. Output:

<Include Code and Screenshot of Output>

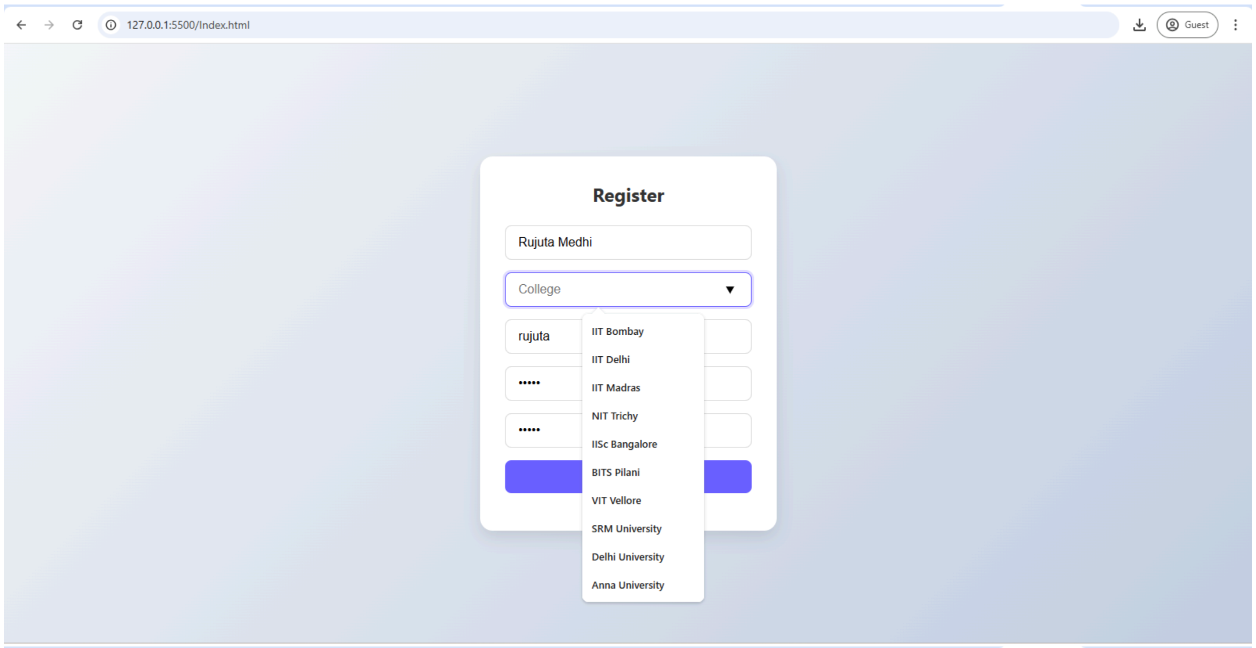
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>AJAX Registration Experiment</title>
  <style>
    /* Reset & Base */
    * { box-sizing: border-box; margin: 0; padding: 0; }
    body {
      display: flex;
      align-items: center;
      justify-content: center;
      min-height: 100vh;
      background: linear-gradient(135deg, #f5f7fa, #c3cfe2);
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    }

    /* Card Container */
    .card {
      background: #ffffff;
      padding: 2rem;
      border-radius: 1rem;
      box-shadow: 0 8px 20px rgba(0, 0, 0, 0.1);
      width: 100%;
      max-width: 380px;
      text-align: center;
    }

    .card h2 {
      margin-bottom: 1.5rem;
      color: #333;
      font-size: 1.5rem;
    }

    /* Form Fields */
    form {
      display: flex;
      flex-direction: column;
      gap: 1rem;
    }

    input {
      padding: 0.75rem 1rem;
      border: 1px solid #ddd;
      border-radius: 0.5rem;
      font-size: 1rem;
      transition: border-color 0.2s, box-shadow 0.2s;
    }

    input:focus {
      border-color: #6c63ff;
      box-shadow: 0 0 0 3px rgba(108, 99, 255, 0.2);
      outline: none;
    }

    button {
      padding: 0.75rem;
      background: #6c63ff;
      color: #fff;
      border: none;
      border-radius: 0.5rem;
      font-size: 1rem;
      cursor: pointer;
      transition: background 0.2s, transform 0.1s;
    }

    button:hover {
      background: #5a54d1;
    }

    button:active {
      transform: scale(0.98);
    }

    /* Message Styles */
    .message {
      margin-top: 1rem;
      font-weight: 600;
      font-size: 0.95rem;
    }

    .error {
      color: #e74c3c;
    }

    .success {
      color: #27ae60;
    }
  </style>
</head>
<body>

  <div class="card">
    <h2>Register</h2>
    <form id="regForm">
      <input type="text" id="name" placeholder="Name" />
      <input type="text" id="college" placeholder="College" list="colleges" />
      <datalist id="colleges"></datalist>
      <input type="text" id="username" placeholder="Username" />
      <input type="password" id="password" placeholder="Password" />
      <input type="password" id="retype" placeholder="Retype Password" />

      <button type="submit">Register</button>
    </form>
    <div id="message" class="message"></div>
  </div>

  <script>
    // --- Data for simulation ---
    const existingUsernames = ["Shravani Patil", "priya01", "arjun_99", "neha123"];
    const collegeList = [
      "IIT Bombay", "IIT Delhi", "IIT Madras",
      "NIT Trichy", "IISc Bangalore", "BITS Pilani",
      "VIT Vellore", "SRM University", "Delhi University",
      "Anna University"
    ];

    // Populate datalist
    const dl = document.getElementById('colleges');
    collegeList.forEach(c => {
      const opt = document.createElement('option');
      opt.value = c;
      dl.appendChild(opt);
    });

    // Form submission handler
    document.getElementById('regForm').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 retype    = document.getElementById('retype').value;
      const msgDiv    = document.getElementById('message');

      // reset
      msgDiv.textContent = '';
      msgDiv.className = 'message';


      const xhr = new XMLHttpRequest();
      xhr.open('POST', '#', true);  // '#' placeholder, no real endpoint
      xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');

      xhr.onreadystatechange = function() {
        if (xhr.readyState === 4) {
          // Simulate server processing delay
          setTimeout(() => {
            // 1. Name not empty
            if (!name) {
              msgDiv.textContent = 'Name field cannot be empty.';
              msgDiv.classList.add('error');
              return;
            }
            // 2. Unique username
            if (existingUsernames.includes(username)) {
              msgDiv.textContent = 'Username already exists. Choose another.';
              msgDiv.classList.add('error');
              return;
            }
            // 3. Passwords match
            if (password !== retype) {
              msgDiv.textContent = 'Passwords do not match.';
              msgDiv.classList.add('error');
              return;
            }
            // All validations passed
            existingUsernames.push(username);
            msgDiv.textContent = 'Successfully Registered';
            msgDiv.classList.add('success');
          }, 300);
        }
      };

      // Send simulated data
                                          const          params          =
  `name=${encodeURIComponent(name)}&college=${encodeURIComponent(college)}&user
  name=${encodeURIComponent(username)}&password=${encodeURIComponent(password
  )}`;

      xhr.send(params);
    });
  </script>

  </body>
  </html>
```


127.0.0.1:5500/Index.html

Username already exists. Choose another.


127.0.0.1:5500/Index.html

Passwords do not match.



127.0.0.1:5500/Index.html

Register

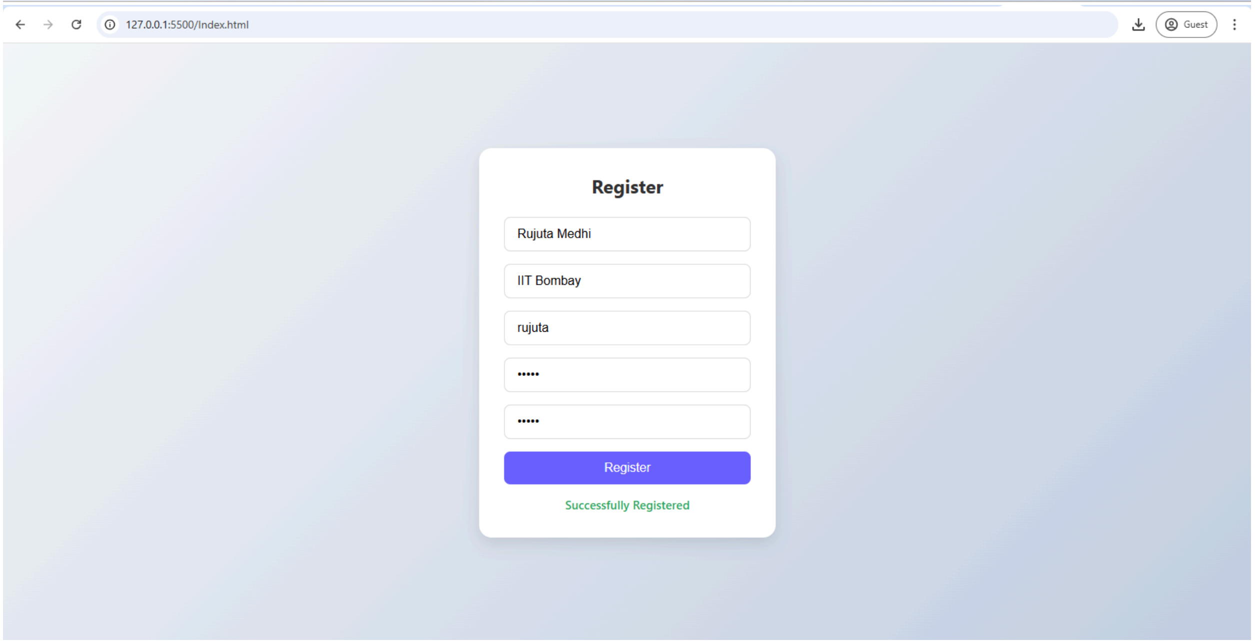
Rujuta Medhi

College

rujuta

•••••

•••••



127.0.0.1:5500/Index.html

Register

Rujuta Medhi

IIT Bombay

rujuta

•••••

•••••

Successfully Registered