Experiment – 9: AJAX

Name of Student	Snehal Anilkumar Patil
Class Roll No	D15A / 38
D.O.P.	
D.O.S.	
Sign and Grade	

Aim: To study AJAX

Theory:
How do Synchronous and Asynchronous Requests differ?

Synchronous and Asynchronous requests refer to how tasks are executed in relation to the main program flow. In a **synchronous request**, the execution of code is paused until the request completes and a response is received from the server. This can cause the browser or application to become unresponsive, especially if the server takes a long time to respond. It is a blocking operation and is generally not recommended for modern web applications due to poor user experience.

In contrast, an **asynchronous request** allows the program to continue executing other operations while the request is still being processed in the background. Once the response is received, a callback function is triggered to handle the result. This non-blocking behavior makes asynchronous requests ideal for enhancing interactivity and responsiveness in web applications, such as updating parts of a webpage without reloading it entirely.

Describe various properties and methods used in XMLHttpRequest Object

The **XMLHttpRequest object** is a built-in JavaScript object used to interact with servers. It allows web pages to make HTTP requests to retrieve or send data without refreshing the page, enabling AJAX-based dynamic content loading.

Common Properties:

- **readyState**: Holds the status of the request. Values range from 0 (uninitialized) to 4 (request finished and response is ready).

- **status**: Returns the HTTP status code (e.g., 200 for success, 404 for not found).
- **statusText**: Provides a short message corresponding to the status code.
- **responseText**: Returns the response data as a string.
- **responseXML**: Returns the response data as XML (if available and parsed).

Common Methods:

- **open(method, url, async)**: Initializes a request with the HTTP method (GET, POST, etc.), target URL, and a boolean indicating whether the request is asynchronous.
- **send(data)**: Sends the request to the server. Data can be included in POST requests.
- **setRequestHeader(header, value)**: Sets HTTP headers before sending the request.
- **abort()**: Cancels an ongoing request.
- **onreadystatechange**: An event handler that is called every time the readyState changes. Commonly used to check when the response is ready and handle it accordingly.

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.

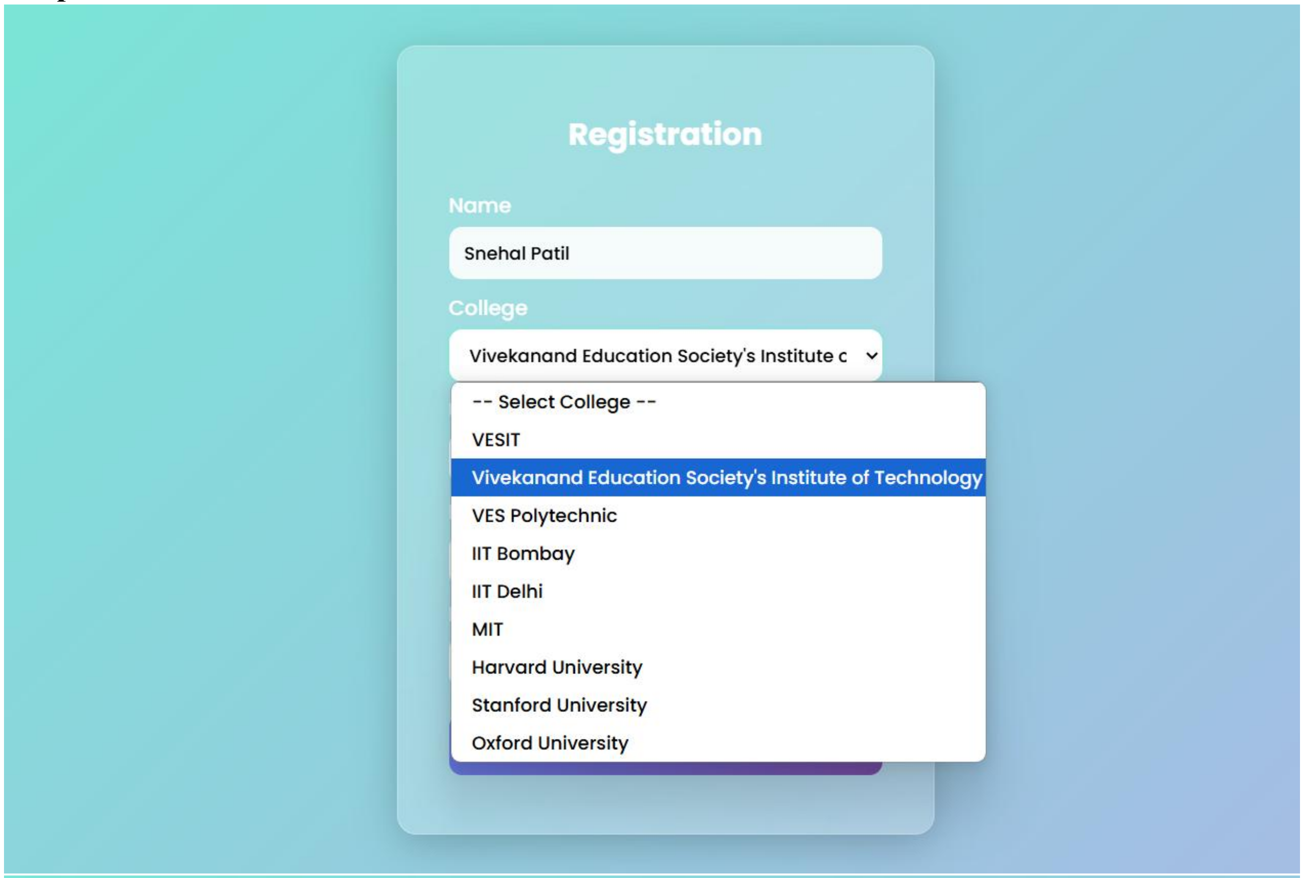
Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Registration Form - AJAX</title>

  <link
href="https://fonts.googleapis.com/css2?family
=Poppins:wght@300;500;700&display=swap"
rel="stylesheet">

  <style>

    * {

      box-sizing: border-box;

      font-family: 'Poppins', sans-serif;

    }


    body {

      background: linear-gradient(135deg,
#74ebd5 0%, #ACB6E5 100%);

      height: 100vh;

      margin: 0;

      display: flex;

      align-items: center;

      justify-content: center;

    }


    .container {

      background: rgba(255, 255, 255, 0.2);

  padding: 30px 40px;

  border-radius: 15px;

  box-shadow: 0 8px 32px rgba(0, 0, 0, 0.2);

  backdrop-filter: blur(10px);

  border: 1px solid rgba(255, 255, 255, 0.3);

  width: 100%;

  max-width: 420px;

}


h2 {

  text-align: center;

  color: white;

  margin-bottom: 25px;

}


label {

  color: white;

  font-weight: 500;

}


input, select {

  width: 100%;

  padding: 10px 12px;

  margin-top: 5px;

  margin-bottom: 10px;
  border: none;

  border-radius: 10px;

  background: rgba(255, 255, 255, 0.9);

  font-size: 14px;

}


input:focus, select:focus {

  outline: none;

  background-color: #fff;

  box-shadow: 0 0 5px #74ebd5;

}


.error {

  color: #ff4d4f;

  font-size: 13px;

}


.success {

  color: #38a169;

  font-weight: 600;

}


button {

  width: 100%;

  padding: 12px;

  border: none;

      border-radius: 10px;

      background: linear-gradient(135deg,
#667eea, #764ba2);

      color: white;

      font-weight: 600;

      font-size: 16px;

      cursor: pointer;

      margin-top: 10px;

    }


    button:hover {

      background: linear-gradient(135deg,
#5a67d8, #6b46c1);

    }


    .message {

      text-align: center;

      margin-top: 15px;

    }

  </style>

</head>

<body>

  <div class="container">

    <h2>Registration</h2>

    <form id="registrationForm">

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

      <input type="text" id="name" required>
    <div id="nameError" class="error"></div>


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

    <select id="college" required>

      <option value="">-- Select College --
</option>

      <option value="VESIT">VESIT</option>

      <option value="Vivekanand Education
Society's Institute of Technology">Vivekanand
Education Society's Institute of
Technology</option>

      <option value="VES Polytechnic">VES
Polytechnic</option>

      <option value="IIT Bombay">IIT
Bombay</option>

      <option value="IIT Delhi">IIT
Delhi</option>

      <option value="MIT">MIT</option>

      <option value="Harvard
University">Harvard University</option>

      <option value="Stanford
University">Stanford University</option>

      <option value="Oxford
University">Oxford University</option>

    </select>

    <div id="collegeError"
class="error"></div>


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

    <input type="text" id="username" required>

    <div id="usernameError"
class="error"></div>


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

    <input type="password" id="password"
required>


    <label for="confirmPassword">Retype
Password</label>

    <input type="password"
id="confirmPassword" required>

    <div id="confirmPasswordError"
class="error"></div>


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

    <div id="registrationMessage"
class="message"></div>

   </form>

  </div>


  <script>

   const existingUsernames = ['admin', 'user1',
'test123'];


   document.getElementById('registrationForm'
).addEventListener('submit', function(e) {

    e.preventDefault();


    // Clear previous errors

    document.getElementById('nameError').text
Content = '';
```

```
    document.getElementById('collegeError').textContent = '';

    document.getElementById('usernameError').textContent = '';

    document.getElementById('confirmPasswordError').textContent = '';

    const messageDiv = document.getElementById('registrationMessage');

    messageDiv.textContent = '';

    messageDiv.className = 'message';



    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;



    let valid = true;



    if (!name) {

      document.getElementById('nameError').textContent = 'Name is required';

      valid = false;

    }



    if (!college) {

      document.getElementById('collegeError').textContent = 'College is required';

      valid = false;

    }



    if (existingUsernames.includes(username)) {

      document.getElementById('usernameError').textContent = 'Username already exists';

      valid = false;

    }



    if (password !== confirmPassword) {

      document.getElementById('confirmPasswordError').textContent = 'Passwords do not match';

      valid = false;

    }



    if (!valid) return;



    const xhr = new XMLHttpRequest();

    xhr.open('POST', '/register', true); // Change URL if needed

    xhr.setRequestHeader('Content-Type', 'application/json');
```

```
      xhr.onload = function () {

        if (xhr.status === 200) {

          messageDiv.textContent = 'Successfully
Registered';

          messageDiv.classList.add('success');

          document.getElementById('registrationF
orm').reset();

        } else {

          messageDiv.textContent = 'Registration
failed. Try again.';

          messageDiv.classList.add('error');

        }

      };

      xhr.onerror = function () {

        messageDiv.textContent = 'An error
occurred during request.';

        messageDiv.classList.add('error');

      };


      xhr.send(JSON.stringify({ name, college,
username, password }));

    });

  </script>

</body>

</html>
```

Output:



Registration

Name

Snehal Patil

College

Vivekanand Education Society's Institute c

-- Select College --

VESIT

Vivekanand Education Society's Institute of Technology

VES Polytechnic

IIT Bombay

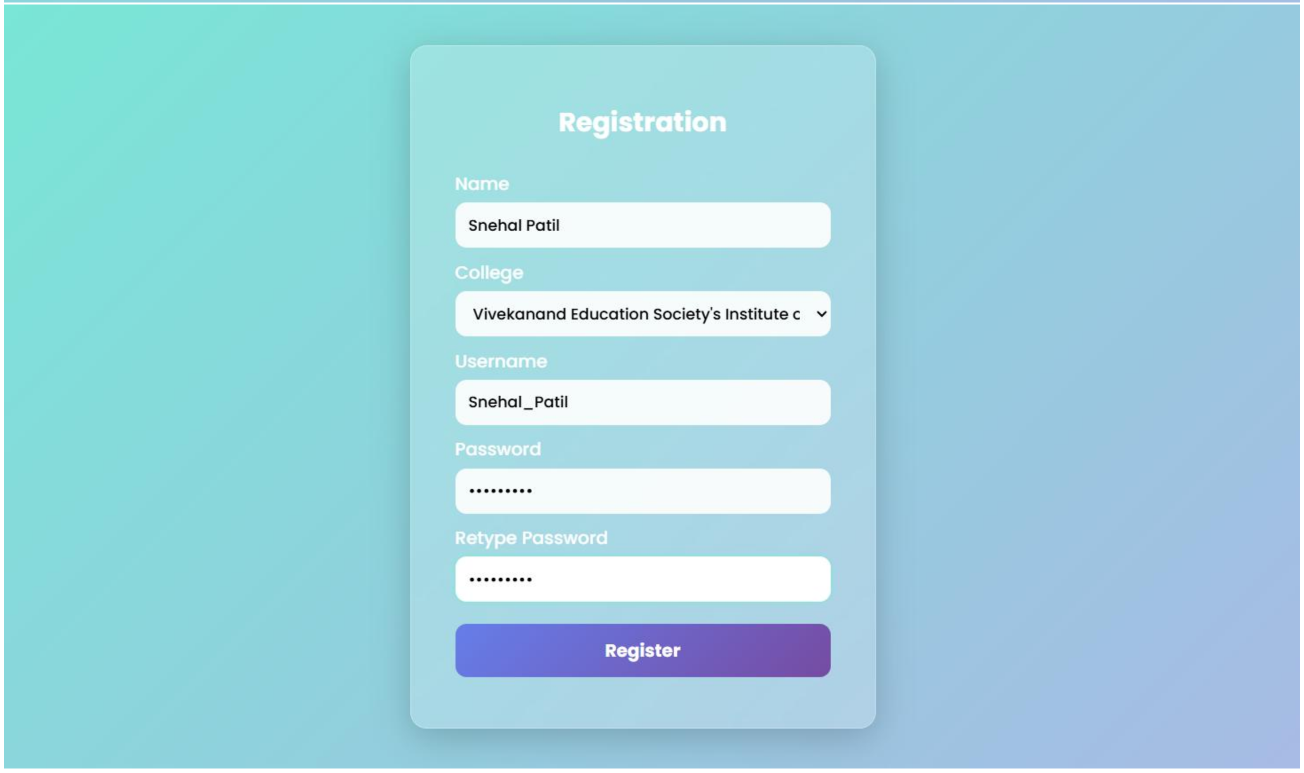
IIT Delhi

MIT

Harvard University

Stanford University

Oxford University



Registration

Name

Snehal Patil

College

Vivekanand Education Society's Institute c

Username

Snehal_Patil

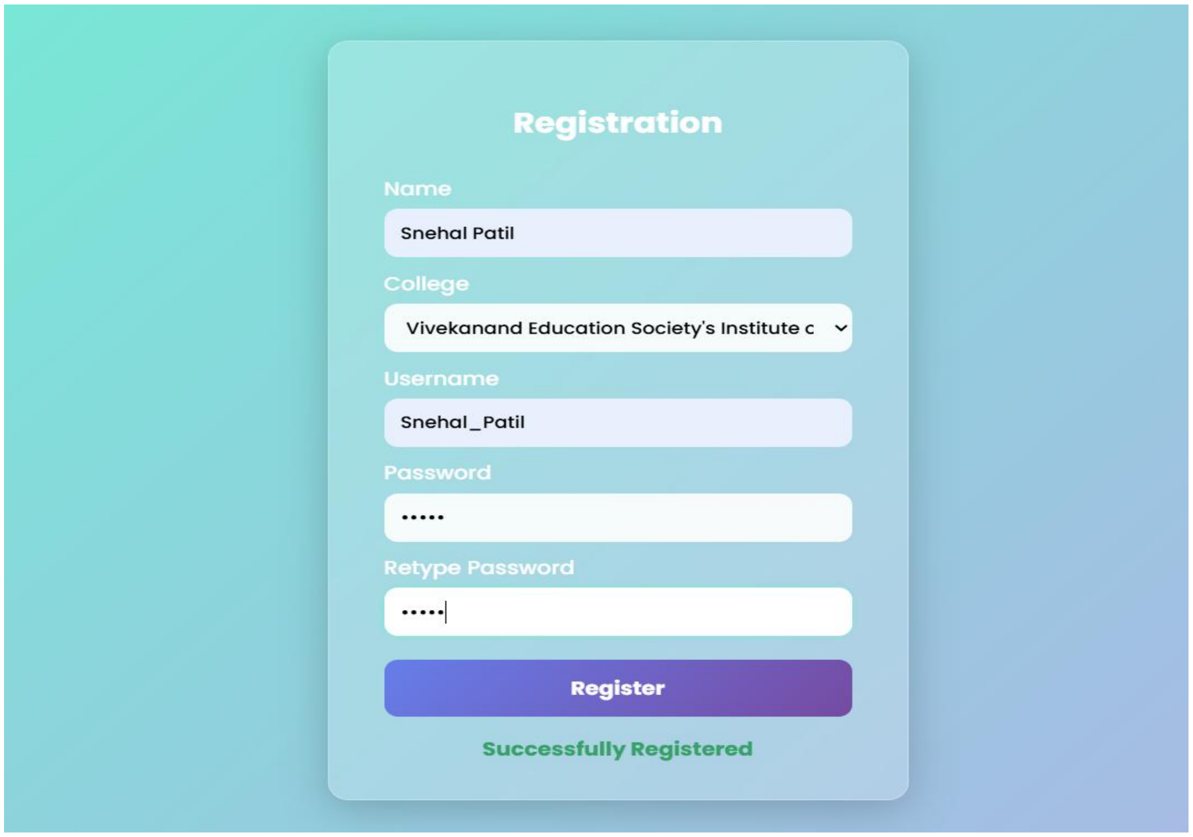
Password

•••••••••

Retype Password

•••••••••

Register



Registration

Name

Snehal Patil

College

Vivekanand Education Society's Institute c

Username

Snehal_Patil

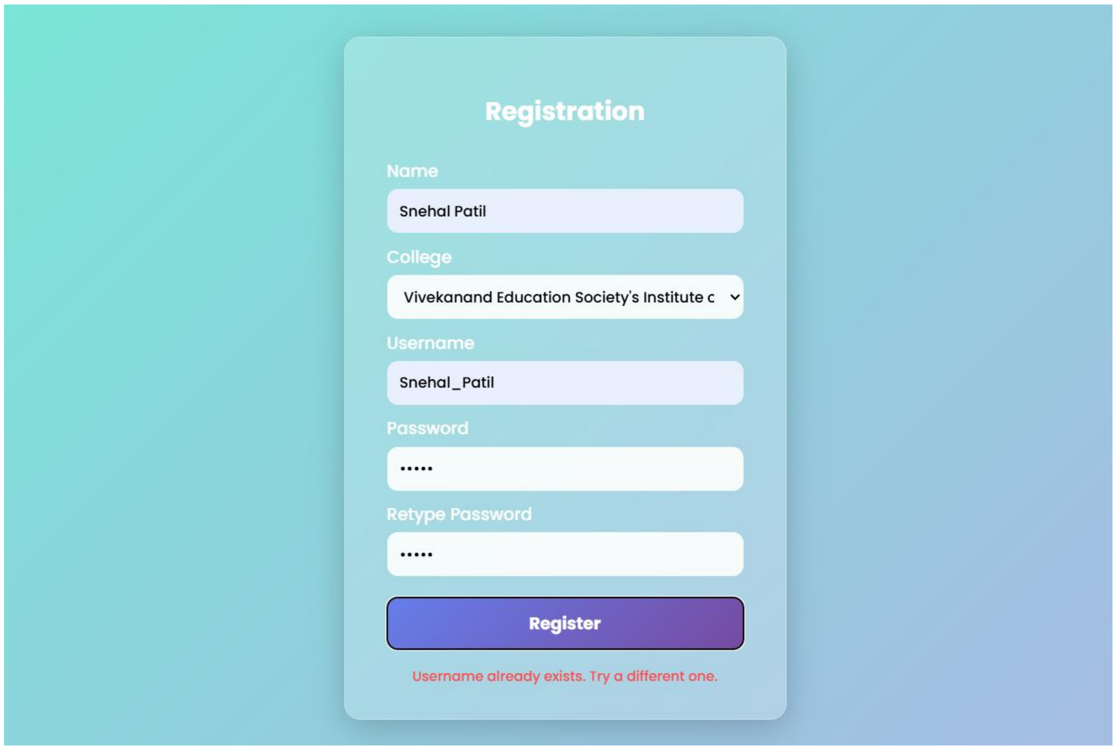
Password

•••••

Retype Password

•••••

Successfully Registered



Registration

Name

Snehal Patil

College

Vivekanand Education Society's Institute c

Username

Snehal_Patil

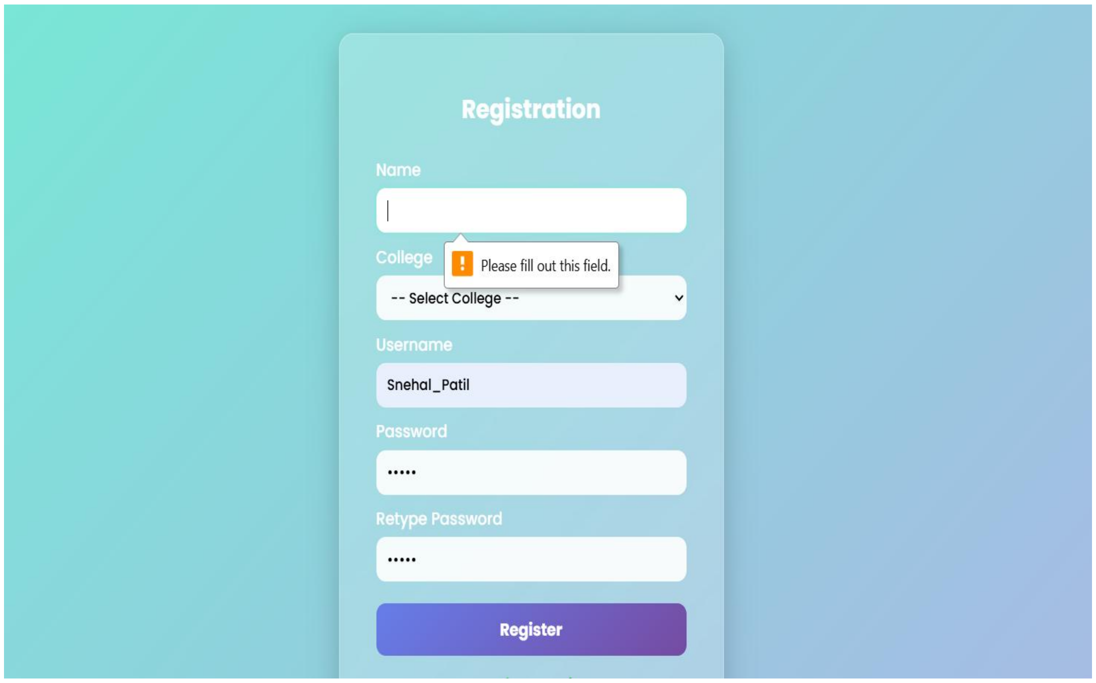
Password

•••••

Retype Password

•••••

Username already exists. Try a different one.



Conclusion : In conclusion, understanding the difference between synchronous and asynchronous requests is crucial for creating responsive web applications. The `XMLHttpRequest` object plays a vital role in enabling asynchronous communication between the client and server without reloading the page. Its properties and methods allow developers to send, receive, and handle data efficiently, improving user experience and performance.