Experiment 9 : To study AJAX

Name of Student	**Sanket More**
Class Roll No	**D15A_29**
D.O.P.	**03/04/2025**
D.O.S.	**10/04/2025**
Sign and Grade	

Aim: To study AJAX

Theory:-

1. How do Synchronous and Asynchronous Requests differ?

Synchronous and Asynchronous requests are two different methods used in web development to communicate with a server, especially when using technologies like **AJAX (Asynchronous JavaScript and XML)**.

Synchronous Requests:

- In a synchronous request, the **browser waits** for the server to respond before continuing to execute the rest of the JavaScript code.
- The entire user interface (UI) **freezes or becomes unresponsive** until the request is complete.
- This approach can lead to a **poor user experience**, especially if the server takes a long time to respond.

Example:
If a synchronous request is made to load data from a server, the user cannot interact with the webpage (like clicking buttons or typing) until the response is received.

Asynchronous Requests:

- In an asynchronous request, the browser **does not wait** for the response.
- The remaining JavaScript code continues to execute, and the browser stays responsive.
- Once the server responds, a **callback function** (or event listener) is triggered to handle the response.

Example:
Fetching live search results without reloading the page. The user continues typing while results update in real time.

Key Differences:

Feature	Synchronous	Asynchronous
Browser Behavior	Waits for response	Does not wait
User Experience	Freezes UI	UI remains responsive
Implementation	Simpler, but less efficient	Requires callbacks or promises
Usage	Rare in modern web apps	Common and preferred

2. Describe various properties and methods used in XMLHttpRequest Object

The `XMLHttpRequest` object is a built-in JavaScript object used to interact with servers and fetch data without reloading the webpage. It is a core part of AJAX-based applications.

Commonly Used Properties:

Property	Description
`readyState`	Returns the current state of the request (0 to 4).
`status`	Returns the HTTP status code (e.g., 200 for OK, 404 for Not Found).
`statusText`	Returns the textual status (e.g., "OK", "Not Found").
`responseText`	Returns the response data as a string.
`responseXML`	Returns the response data as an XML document (if the response is XML).

readyState Values:

Value	State	Description
0	UNSENT	Request not initialized
1	OPENED	`open()` has been called
2	HEADERS_RECEIVED	`send()` has been called
3	LOADING	Receiving the response
4	DONE	Request finished and response is ready

Commonly Used Methods:

Method	Description
`open(method, url, async)`	Initializes a request. `method` is "GET" or "POST", `async` is a boolean.
`send()`	Sends the request to the server.
`setRequestHeader(header, value)`	Sets a request header (e.g., Content-Type). Used after `open()`, before `send()`.
`abort()`	Cancels an ongoing request.

Example:

Javascript

```
let xhr = new XMLHttpRequest();
xhr.open("GET", "data.json", true);  // Asynchronous GET request
xhr.onreadystatechange = function() {
  if (xhr.readyState === 4 && xhr.status === 200) {
    console.log(xhr.responseText);  // Handle response
  }
};
xhr.send();
```

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:-

abc.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Registration Page</title>
  <style>
    body {
      font-family: 'Segoe UI', sans-serif;
      background: #e3f2fd;
      margin: 0;
      padding: 20px;
    }

    .container {
      background: white;
      max-width: 500px;
      margin: 40px auto;
      padding: 30px;
      border-radius: 10px;
      box-shadow: 0 8px 16px rgba(0,0,0,0.2);
    }

    h2 {
      text-align: center;
      color: #0d47a1;
```

```
}

label {
  display: block;
  margin-top: 15px;
  font-weight: bold;
}

input[type="text"],
input[type="password"],
input[list] {
  width: 100%;
  padding: 10px;
  margin-top: 6px;
  border: 1px solid #ccc;
  border-radius: 5px;
  box-sizing: border-box;
}

button {
  margin-top: 20px;
  width: 100%;
  padding: 12px;
  background-color: #1976d2;
  color: white;
  border: none;
  border-radius: 5px;
  font-size: 16px;
  cursor: pointer;
  transition: background 0.3s;
}

button:hover {
  background-color: #1565c0;
}

.message {
  margin-top: 20px;
  font-size: 15px;
  color: #d32f2f;
```

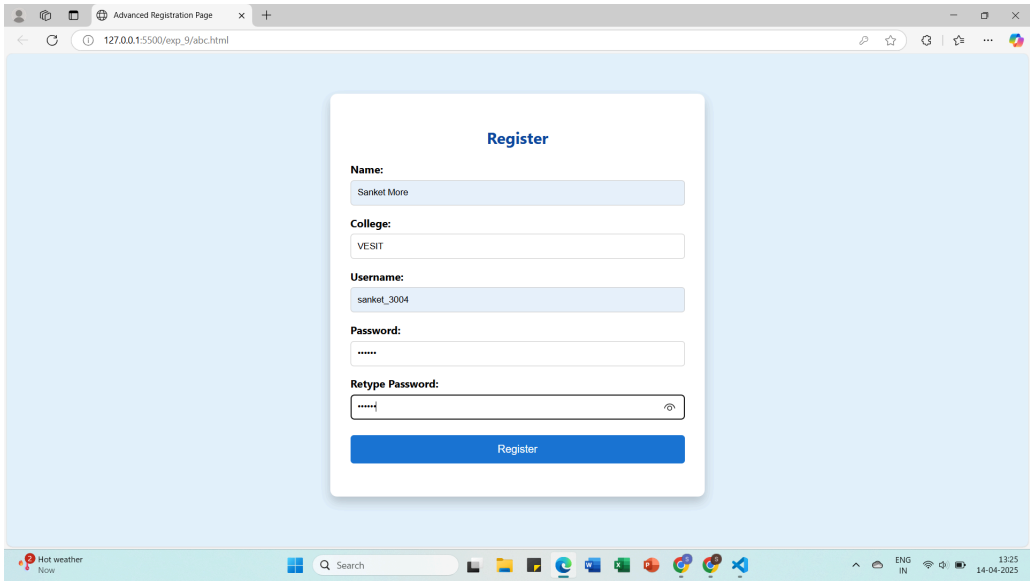
```
    }

    .success {
      color: #388e3c;
    }

    .input-feedback {
      font-size: 13px;
      color: #d32f2f;
      margin-top: 3px;
    }
  </style>
</head>
<body>

<div class="container">
  <h2>Register</h2>
  <form id="regForm" onsubmit="return false;">
    <label for="name">Name:</label>
    <input type="text" id="name">
    <div class="input-feedback" id="nameFeedback"></div>

    <label for="college">College:</label>
    <input list="colleges" id="college">
    <datalist id="colleges">
      <option value="VESIT">
      <option value="IIT Bombay">
      <option value="Stanford University">
      <option value="MIT">
      <option value="BITS Pilani">
      <option value="University of Mumbai">
      <option value="Harvard University">
    </datalist>

    <label for="username">Username:</label>
    <input type="text" id="username" onblur="checkUsernameAsync()">
    <div class="input-feedback" id="usernameFeedback"></div>

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

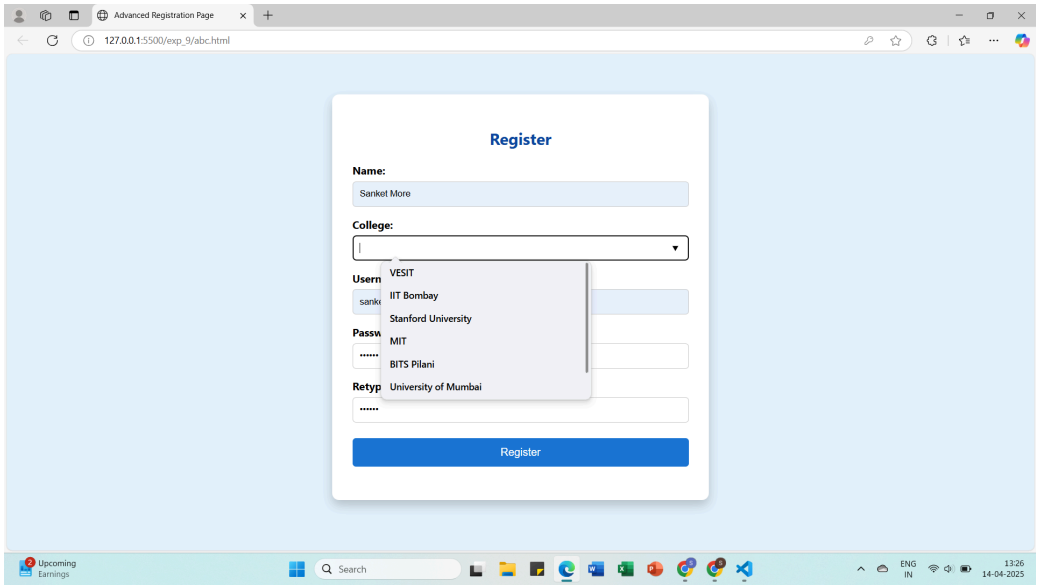
```
    <label for="confirmPassword">Retype Password:</label>
    <input type="password" id="confirmPassword">
    <div class="input-feedback" id="passwordFeedback"></div>

    <button type="button" onclick="submitForm()">Register</button>
  </form>

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

<script>
    const existingUsernames = ["sanket123", "john_doe", "admin", "guest"];

    function checkUsernameAvailable(username) {
      return new Promise((resolve) => {
        const taken = existingUsernames.includes(username.trim());
        const result = {
          available: !taken,
          message: taken ? "Username already exists." : "Username is
available."
        };

        setTimeout(() => resolve(result), 500); // simulate async delay
      });
    }

    async function submitForm() {
      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 nameFeedback = document.getElementById("nameFeedback");
      const usernameFeedback = document.getElementById("usernameFeedback");
      const passwordFeedback = document.getElementById("passwordFeedback");
      const resultMessage = document.getElementById("resultMessage");
```

```
      // Reset messages
      nameFeedback.textContent = "";
      usernameFeedback.textContent = "";
      passwordFeedback.textContent = "";
      resultMessage.textContent = "";
      resultMessage.classList.remove("success");

      let valid = true;

      if (name === "") {
        nameFeedback.textContent = "Name cannot be empty.";
        valid = false;
      }

      if (password !== confirmPassword) {
        passwordFeedback.textContent = "Passwords do not match.";
        valid = false;
      }

      // Username availability check
      const usernameCheck = await checkUsernameAvailable(username);
      usernameFeedback.textContent = usernameCheck.message;
      usernameFeedback.style.color = usernameCheck.available ? "green" :
"#d32f2f";

      if (!usernameCheck.available) {
        valid = false;
      }

      if (!valid) return;

      // Simulate form submission
      setTimeout(() => {
        resultMessage.textContent = "Successfully Registered";
        resultMessage.classList.add("success");
      }, 1000);
    }

    document.getElementById("username").addEventListener("input", () => {
      document.getElementById("usernameFeedback").textContent = "";
```

```
    });
  </script>


</body>
</html>
```

OUTPUT:-





Username:

sanket_3004

Please choose a different username.

Register

Name:

Sanket More

College:

VESIT

Username:

sanket_3003

Please choose a different username.

Password:

••••••

Retype Password:

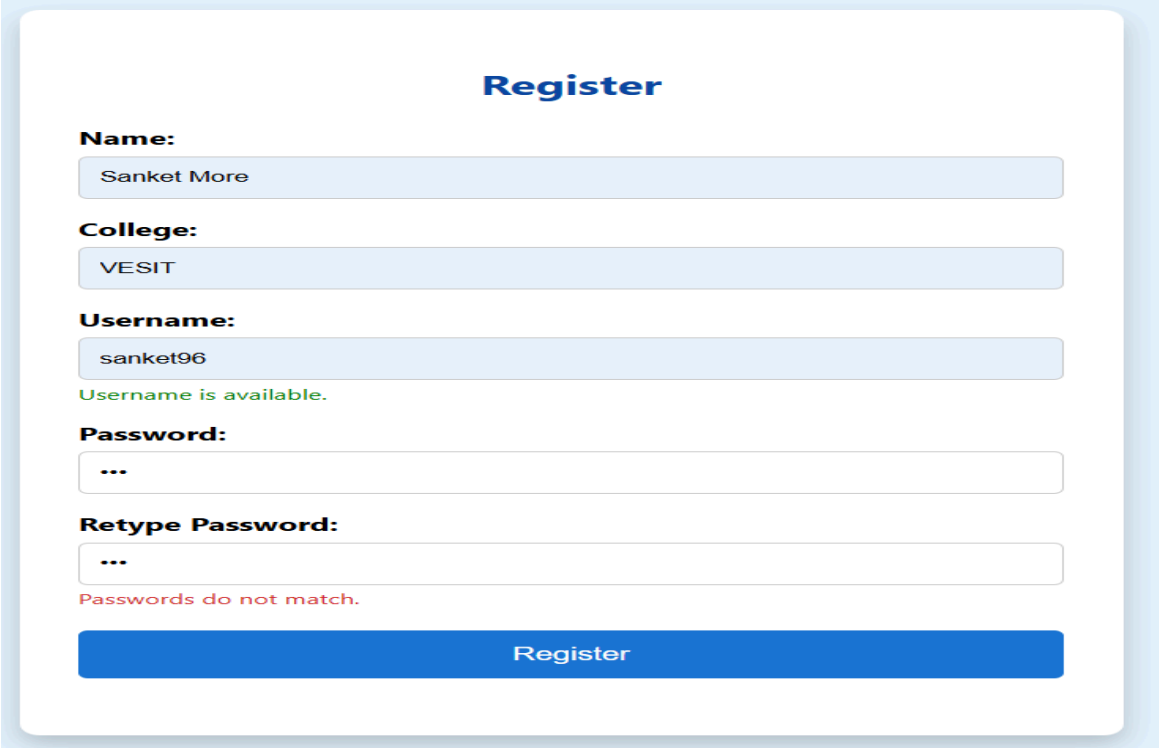
••••••

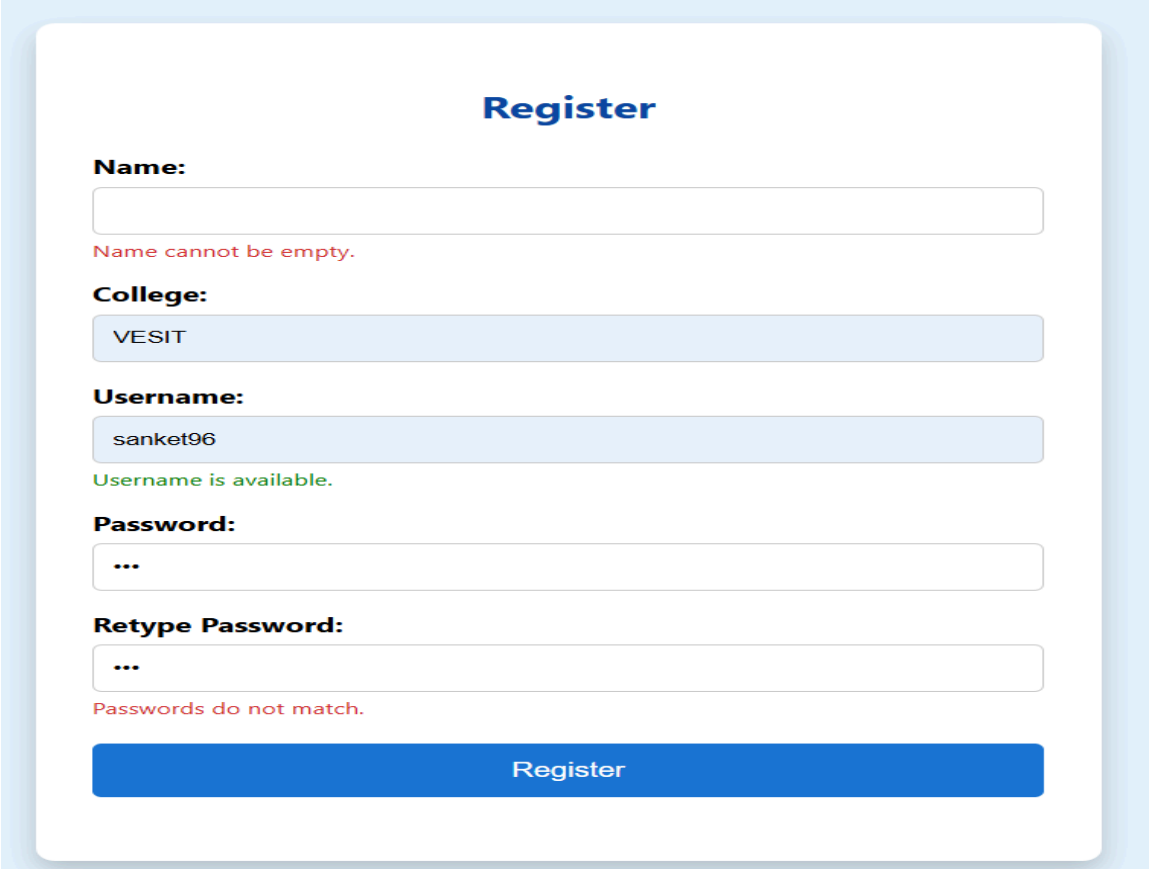
Register

Username:

sanket123

Username already exists.


Register
Name:
Sanket More
College:
VESIT
Username:
sanket96
Username is available.
Password:
Retype Password:
Passwords do not match.
Register


Register
Name:
Name cannot be empty.
College:
VESIT
Username:
sanket96
Username is available.
Password:
Retype Password:
Passwords do not match.
Register

Register

Name:

Sanket More

College:

VESIT

Username:

sanket96

Username is available.

Password:

•••

Retype Password:

•••

Register

Successfully Registered