Experiment 9

Name of Student	**Siddhant Sathe**
Class Roll No	**D15A/50**
D.O.P.	
D.O.S.	
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
readyStat e	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).
statusTex t	Returns the textual status (e.g., "OK", "Not Found").
responseT ext	Returns the response data as a string.
responseX ML	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(head er, 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.

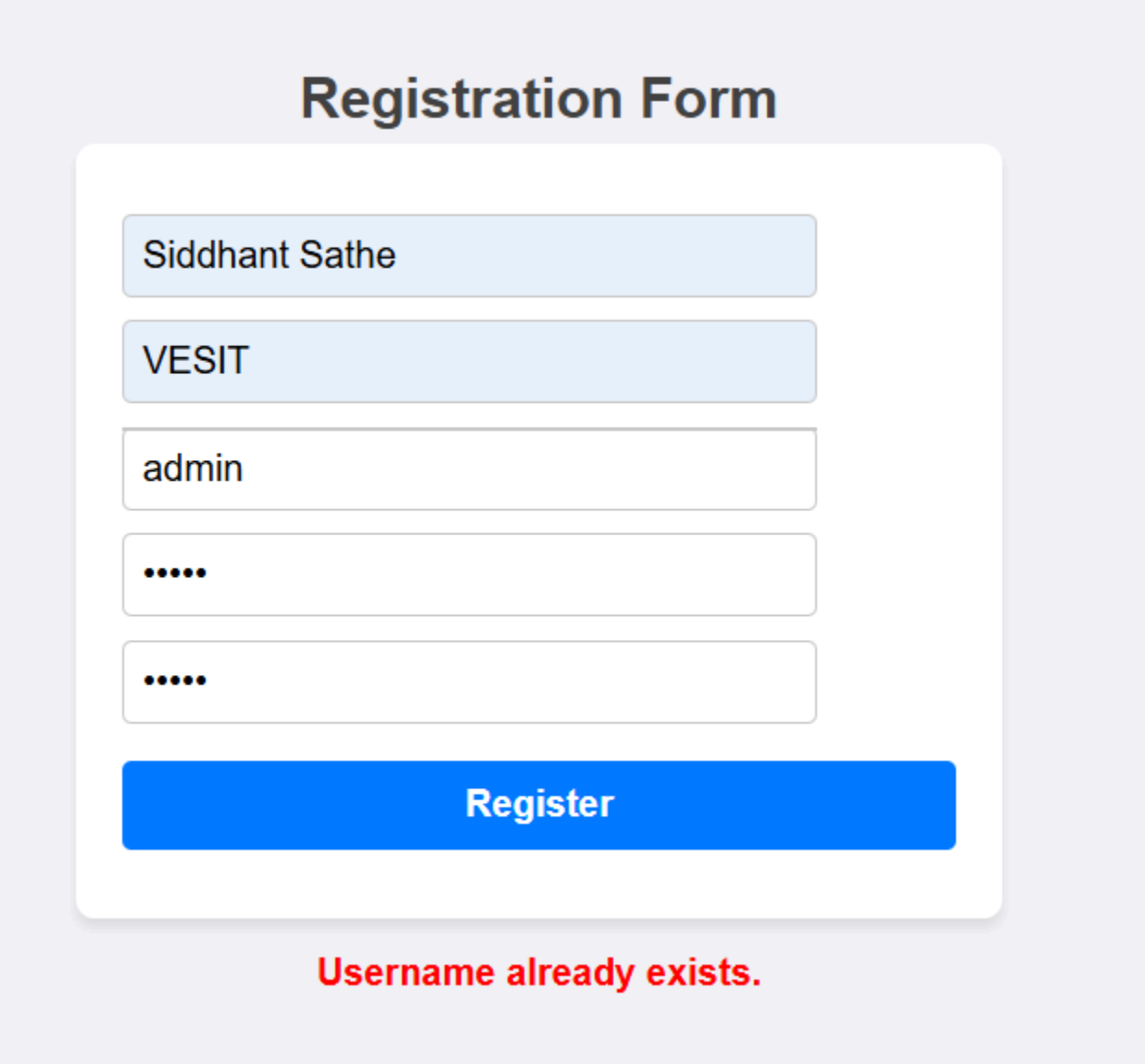
CODE:-

Index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Registration Form</title>
    <link rel="stylesheet" href="styles.css" />
    <style>
      /* Inline styles moved to styles.css */
    </style>
  </head>
  <body>
    <h2>Registration Form</h2>
    <form id="registrationForm" onsubmit="return false;">
      <input type="text" id="name" placeholder="Name" required />
      <div>
        <input
          type="text"
          id="college"
          placeholder="College"
          autocomplete="off"
          onkeyup="autoSuggestCollege()"
        />
        <div id="suggestions" class="suggestions"></div>
      </div>
      <input type="text" id="username" placeholder="Username" required />
      <input type="password" id="password" placeholder="Password" required />
      <input
        type="password"
        id="confirmPassword"
        placeholder="Confirm Password"
        required
      />
      <button onclick="submitForm()">Register</button>
    </form>
    <p id="message"></p>
    <script>
      const existingUsernames = ["Siddhant", "Pranav",”Arnav”, "admin"];
      const colleges = [
        "Indian Institute of Technology",
        "National Institute of Technology",
        "Anna University",
        "Delhi University",
        "VIT University",
```

```
  "SRM Institute",
  "MIT College",
  "BITS Pilani",
  "VESIT",
];

function autoSuggestCollege() {
  const input = document.getElementById("college").value.toLowerCase();
  const suggestionsBox = document.getElementById("suggestions");
  suggestionsBox.innerHTML = "";

  if (input === "") return;

  const matched = colleges.filter((college) =>
    college.toLowerCase().includes(input)
  );

  matched.forEach((college) => {
    const div = document.createElement("div");
    div.className = "suggestion-item";
    div.innerText = college;
    div.onclick = () => {
      document.getElementById("college").value = college;
      suggestionsBox.innerHTML = "";
    };
    suggestionsBox.appendChild(div);
  });
}

// Live check for existing username
document.getElementById("username").addEventListener("blur", function () {
  const username = this.value.trim();
  const message = document.getElementById("message");

  if (existingUsernames.includes(username)) {
    message.innerText = "Username already exists.";
    message.style.color = "red";
  } else {
    message.innerText = "";
  }
});

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 message = document.getElementById("message");

  // Simulate async behavior
  setTimeout(() => {
    let response = "";

    if (name === "") {
      response = "Name cannot be empty.";
    } else if (existingUsernames.includes(username)) {
      response = "Username already exists.";
    } else if (password !== confirmPassword) {
      response = "Passwords do not match.";
```

```
        } else {
          response = "Successfully Registered";
        }

        message.innerText = response;
        message.style.color =
          response === "Successfully Registered" ? "green" : "red";
      }, 500); // fake delay to mimic XMLHttpRequest async loading
    }
  </script>
 </body>
</html>
```



Registration Form

Siddhant Sathe

VESIT

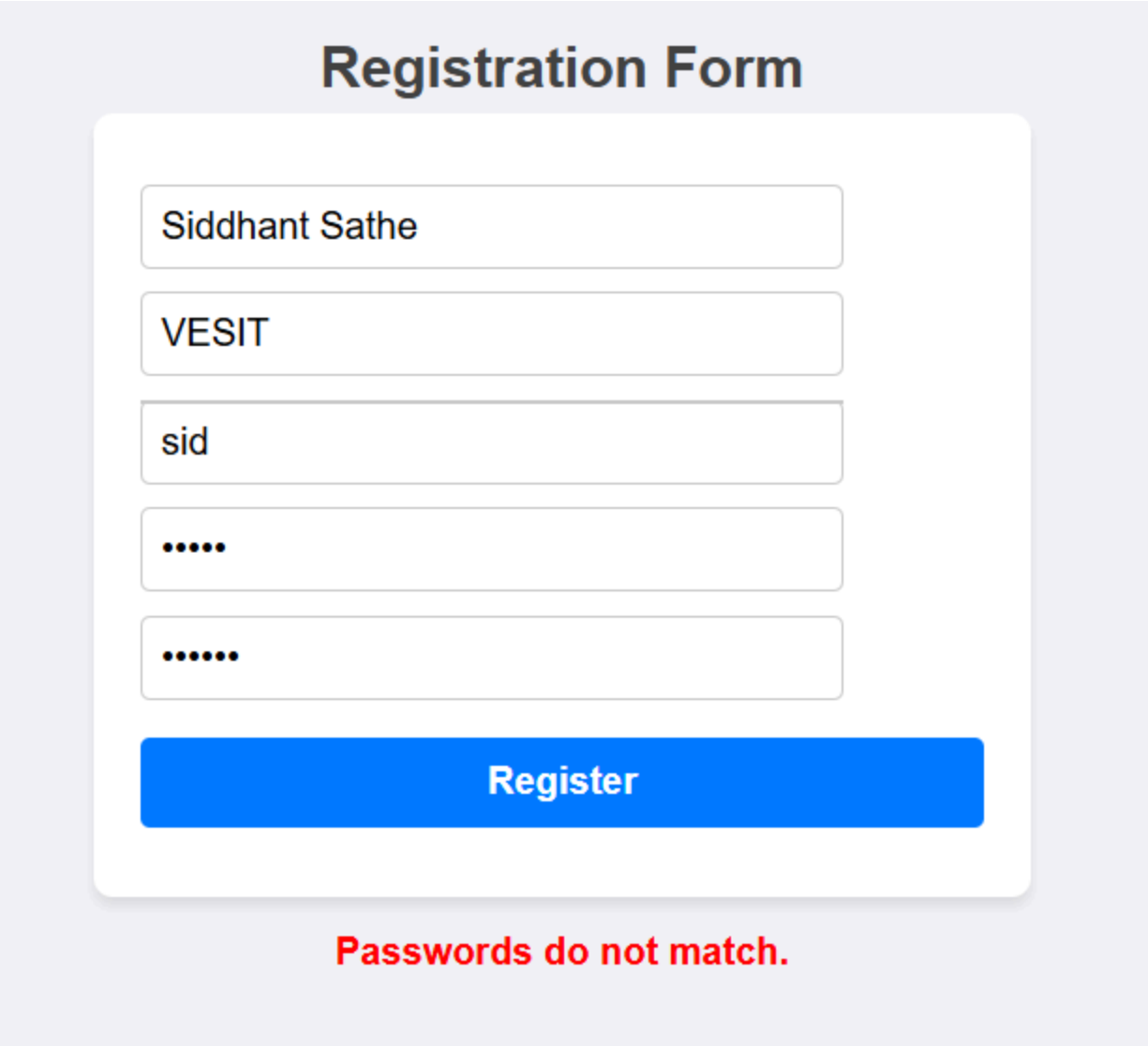
siddy

••••••

••••••

Register

Successfully Registered



Conclusion:

The provided code implements a registration form with features like live username validation and college name auto-suggestions. The form includes fields for name, college, username, password, and password confirmation. The autoSuggestCollege function dynamically displays matching college names as the user types, enhancing usability. The username field checks for duplicates against a predefined list when it loses focus, providing immediate feedback. The submitForm function validates the inputs, ensuring all fields are filled, passwords match, and the username is unique. Feedback is displayed to the user with appropriate messages and colors, simulating asynchronous behavior for a realistic user experience.