

EXPERIMENT NO. 9

Name of Student	<u>Tejas Dhondibhau Gunjal</u>
Class Roll No	<u>18</u>
D.O.P.	<u>08-04-25</u>
D.O.S.	<u>15-04-25</u>
Sign and Grade	

AIM:

To study AJAX

PROBLEM STATEMENT:

Create a registration page having fields like Name, College, Username and Password (read password twice).

Validate the form by checking for

1. Username is 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.

THEORY:

1. How do Synchronous and Asynchronous Requests differ?

Synchronous Requests and **Asynchronous Requests** are two ways a browser communicates with the server.

- **Synchronous Request:**

The browser sends a request to the server and **waits** for the response. During this time, the user **cannot interact** with the page. It blocks the execution of further code.

- **Asynchronous Request:**

The browser sends a request to the server but **does not wait** for the response. The page remains interactive and the code continues running. When the response arrives, it is handled separately (commonly using AJAX).

Feature	Synchronous Request	Asynchronous Request
Execution	Waits for server response before continuing.	Continues executing code without waiting.
User Experience	Freezes the page until response received.	Page remains interactive.
Blocking	Blocking – stops everything else.	Non-blocking–background processing.
Use in Modern Web	Rarely used.	Commonly used (AJAX, Fetch API, etc).
Example in JS	xhr.open('GET', url, false)	xhr.open('GET', url, true)

2. Describe various properties and methods used in XMLHttpRequest Object

The **XMLHttpRequest** object is a built-in JavaScript object that allows the **browser (client-side)** to send and receive data from a **server** without reloading the entire web page. It is the main tool used in **AJAX** to create fast, dynamic, and responsive web applications.

With XMLHttpRequest, you can:

- Fetch data from a server (like text, JSON, or XML)
- Submit form data
- Update parts of a webpage **asynchronously** (in the background)
- Improve user experience by avoiding full page reloads

Important Properties of XMLHttpRequest

Property	Description
readyState	Represents the current state of the request. It can have values between 0 (request not initialized) and 4 (request complete).
status	Contains the HTTP status code of the response (e.g., 200 for successful requests, 404 for not found).
statusText	Contains the textual description of the HTTP status (e.g., "OK" for 200).
responseText	Returns the response as a plain text string, which is commonly used for handling server responses.
responseXML	Returns the response as XML (if the server sends XML data).
response	A general property that holds the response data (can be text, XML, JSON, etc.).
timeout	Specifies the number of milliseconds a request is allowed to take before being aborted.
withCredentials	Allows cross-origin requests to send credentials (cookies, authorization headers, etc.).

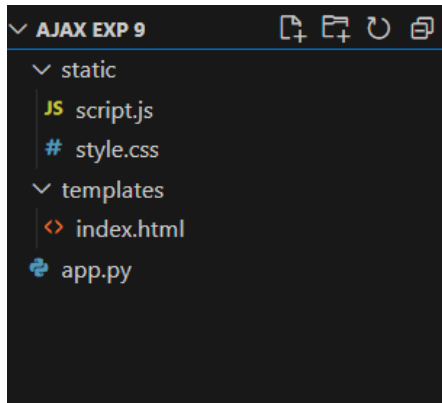
Common Methods of XMLHttpRequest

Method	Description
open(method, url, async)	Initializes a request. method is usually GET or POST, url is the request URL, and async is a boolean that specifies whether the request should be asynchronous (true) or synchronous (false).
send(data)	Sends the request to the server. If the request method is POST, data is the body of the request (e.g., form data or JSON).
setRequestHeader(header, value)	Sets an HTTP request header. Useful when sending data with POST requests.
abort()	Cancels the request if it's still ongoing.
getAllResponseHeaders()	Returns all response headers as a string.
getResponseHeader(header)	Returns the value of a specific response header.
onreadystatechange	An event handler that is called when the readyState changes. It's often used to check if the request has completed (readyState == 4).
onload	Event handler for when the request has successfully completed (status 200).
onerror	Event handler for when the request fails (e.g., status 404 or network error).

GitHub Link : - https://github.com/tejasgunjal021/WEBX_EXP9

Code : -

DIRECTORY STRUCTURE



Templates/Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Registration Form</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
  <div class="container">
    <h2>Register Now</h2>
    <form id="regForm" onsubmit="return false;">
      <input type="text" id="name" placeholder="Full Name" required oninput="validateName()">
      <span id="nameStatus" style="font-size: 13px;">

      <input type="text" id="college" placeholder="College Name" onkeyup="suggestCollege()"
autocomplete="off" required>
      <ul id="suggestions"></ul>

      <input type="text" id="username" placeholder="Username" required
oninput="checkUsernameAvailability()">
      <span id="usernameStatus" style="font-size: 13px;"></span>

      <input type="password" id="password" placeholder="Password" required
oninput="validatePasswordMatch()">
      <input type="password" id="retypePassword" placeholder="Retype Password" required
oninput="validatePasswordMatch()">
      <span id="passwordStatus" style="font-size: 13px;"></span>

      <button onclick="registerUser()">Register</button>
    </form>
    <div id="result"></div>
  </div>

  <script src="{{ url_for('static', filename='script.js') }}"></script>
</body>
</html>
```

Static/style.css

```
body {
  font-family: 'Segoe UI', sans-serif;
  background: linear-gradient(to right,
#8e9eab, #eef2f3);
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}
```

```
.container {
  background: #ffffff;
  padding: 30px 40px;
  border-radius: 15px;
  box-shadow: 0px 8px 24px rgba(0, 0, 0,
0.2);
  width: 350px;
}
```

```
h2 {
  text-align: center;
  margin-bottom: 20px;
  color: #333;
}
```

```
input {
  width: 100%;
  padding: 10px 14px;
  margin: 10px 0;
  border: 1px solid #ccc;
  border-radius: 10px;
  font-size: 15px;
}
```

```
button {
  width: 100%;
  padding: 10px;
  background: #4CAF50;
  color: white;
  font-weight: bold;
  border: none;
  border-radius: 10px;
  cursor: pointer;
}
```

```
margin-top: 10px;
}
```

```
button:hover {
  background: #45a049;
}
```

```
#result {
  margin-top: 15px;
  text-align: center;
  color: #333;
}
```

```
#suggestions {
  background: #f1f1f1;
  border-radius: 6px;
  padding: 5px 10px;
  font-size: 13px;
  color: #666;
}
```

```
#suggestions {
  list-style: none;
  padding: 0;
  margin-top: -5px;
  max-height: 120px;
  overflow-y: auto;
  border: 1px solid #ccc;
  border-radius: 0 0 10px 10px;
  background-color: white;
  position: relative;
  z-index: 10;
}
```

```
#suggestions li {
  padding: 8px 12px;
  cursor: pointer;
}
```

```
#suggestions li:hover {
  background-color: #f0f0f0;
}
```

Static/script.js

```

function registerUser() {
    let name = document.getElementById("name").value;
    let college = document.getElementById("college").value;
    let username = document.getElementById("username").value;
    let password = document.getElementById("password").value;
    let retypePassword = document.getElementById("retypePassword").value;

    let xhr = new XMLHttpRequest();
    xhr.open("POST", "/register", true);
    xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");

    xhr.onload = function () {
        let response = JSON.parse(xhr.responseText);
        document.getElementById("result").innerText = response.message;
    };

    xhr.send(`name=${encodeURIComponent(name)}&college=${encodeURIComponent(college)}&username=${encodeURIComponent(username)}&password=${encodeURIComponent(password)}&retype_password=${encodeURIComponent(retypePassword)}`);
}

function suggestCollege() {
    const input = document.getElementById("college").value;
    const suggestionsList = document.getElementById("suggestions");

    if (input.length < 2) {
        suggestionsList.innerHTML = "";
        return;
    }

    let xhr = new XMLHttpRequest();
    xhr.open("GET", "/suggest_college?query=" + encodeURIComponent(input), true);
    xhr.onload = function () {
        const response = JSON.parse(xhr.responseText);
        suggestionsList.innerHTML = ""; // Clear previous

        if (response.suggestions.length > 0) {
            response.suggestions.forEach(function (college) {
                const li = document.createElement("li");
                li.textContent = college;
                li.onclick = function () {
                    document.getElementById("college").value = college;
                    suggestionsList.innerHTML = "";
                };
                suggestionsList.appendChild(li);
            });
        }
    };
    xhr.send();
}

```

```
function checkUsernameAvailability() {
  const usernameInput = document.getElementById("username");
  const username = usernameInput.value.trim();
  const usernameStatus = document.getElementById("usernameStatus");

  if (username.length < 3) {
    usernameStatus.innerText = "";
    return;
  }

  let xhr = new XMLHttpRequest();
  xhr.open("GET", "/check_username?username=" + encodeURIComponent(username), true);
  xhr.onload = function () {
    const response = JSON.parse(xhr.responseText);
    if (response.exists) {
      usernameStatus.innerText = "Username already exists.";
      usernameStatus.style.color = "red";
    } else {
      usernameStatus.innerText = "Username is available.";
      usernameStatus.style.color = "green";
    }
  };
  xhr.send();
}

function validateName() {
  const name = document.getElementById("name").value.trim();
  const nameStatus = document.getElementById("nameStatus");

  if (name === "") {
    nameStatus.innerText = "Name cannot be empty.";
    nameStatus.style.color = "red";
  } else {
    nameStatus.innerText = "";
  }
}

function validatePasswordMatch() {
  const password = document.getElementById("password").value;
  const retypePassword = document.getElementById("retypePassword").value;
  const passStatus = document.getElementById("passwordStatus");

  if (retypePassword !== "" && password !== retypePassword) {
    passStatus.innerText = "Passwords do not match.";
    passStatus.style.color = "red";
  } else {
    passStatus.innerText = "";
  }
}
```

App.py

```

from flask import Flask, render_template, request, jsonify

app = Flask(__name__)

existing_users = [
    'tejas02',
    'rahul123',
    'neha_k',
    'amit09',
    'priya_singh',
]

college_names = [
    'Veermata Jijabai Technological Institute (VJTI)',
    'Sardar Patel Institute of Technology (SPIT)',
    'Thadomal Shahani Engineering College (TSEC)',
    'Dwarkadas J. Sanghvi College of Engineering (DJSCE)',
    'K. J. Somaiya College of Engineering',
    'Fr. Conceicao Rodrigues College of Engineering (FRCRCE)',
    'Rajiv Gandhi Institute of Technology (RGIT)',
    'Don Bosco Institute of Technology',
    'Shah & Anchor Kutchhi Engineering College',
    'Vivekanand Education Society\'s Institute of Technology (VESIT)',
    'Atharva College of Engineering',
    'Xavier Institute of Engineering (XIE)',
    'Vidyalankar Institute of Technology (VIT)',
    'Lokmanya Tilak College of Engineering (LTCE)',
    'Watumull Institute of Electronics Engineering and Computer Technology'
]

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/register', methods=['POST'])
def register():
    name = request.form['name'].strip()
    college = request.form['college'].strip()
    username = request.form['username'].strip()
    password = request.form['password']
    retype_password = request.form['retype_password']

    if not name:
        return jsonify({'message': 'Name cannot be empty.'})

    if username in existing_users:
        return jsonify({'message': 'Username already exists.'})

```



```

if password != retype_password:
    return jsonify({'message': 'Passwords do not match.'})

return jsonify({'message': 'Successfully Registered!'})

@app.route('/suggest_college')
def suggest_college():
    query = request.args.get('query', "").lower()
    suggestions = [name for name in college_names if query in name.lower()]
    return jsonify({'suggestions': suggestions})

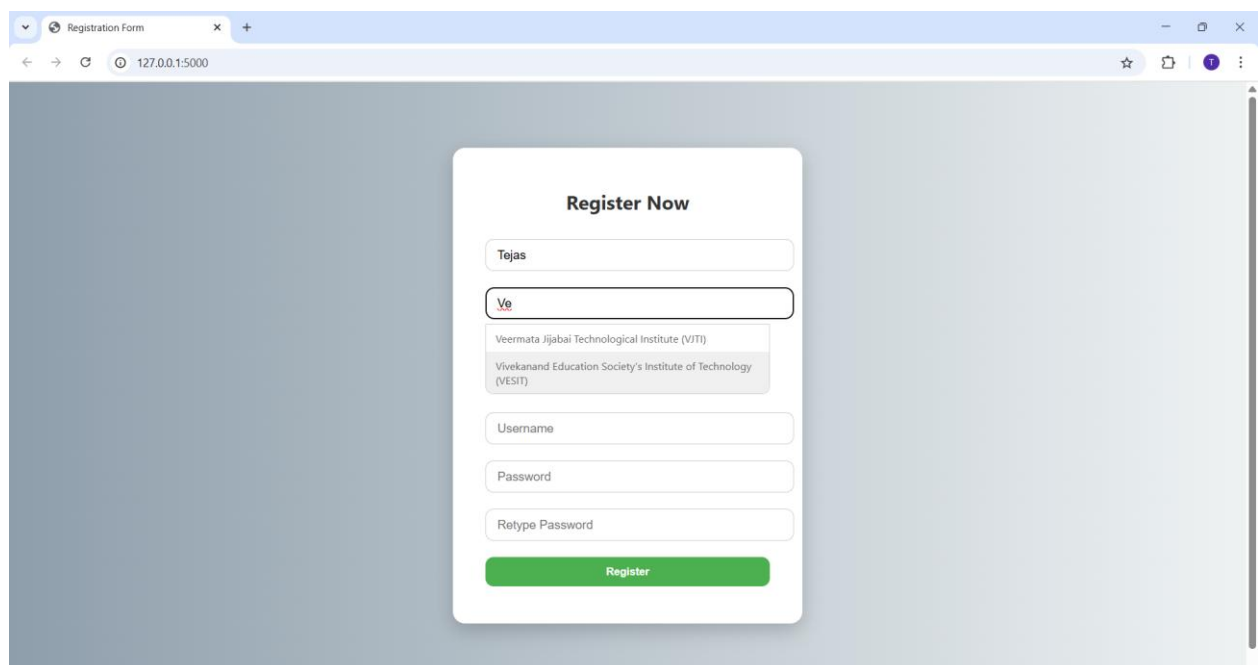
@app.route('/check_username')
def check_username():
    username = request.args.get('username', "").strip()
    exists = username in existing_users
    return jsonify({'exists': exists})

if __name__ == '__main__':
    app.run(debug=True)

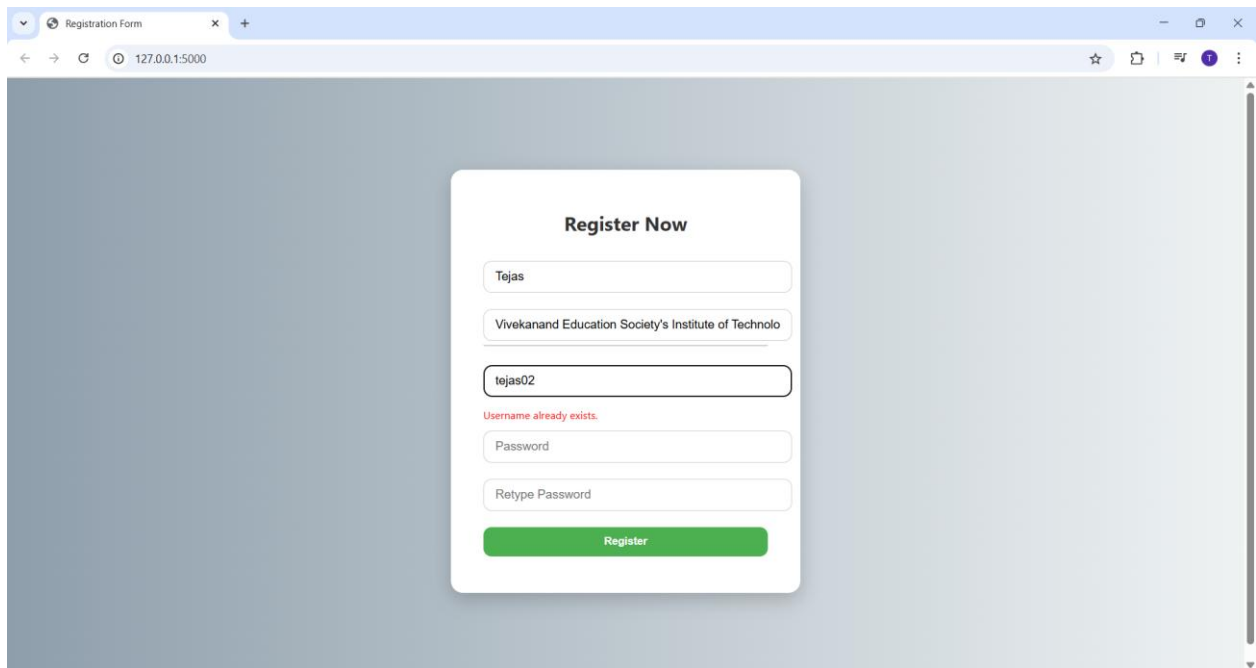
```

OUTPUT: -

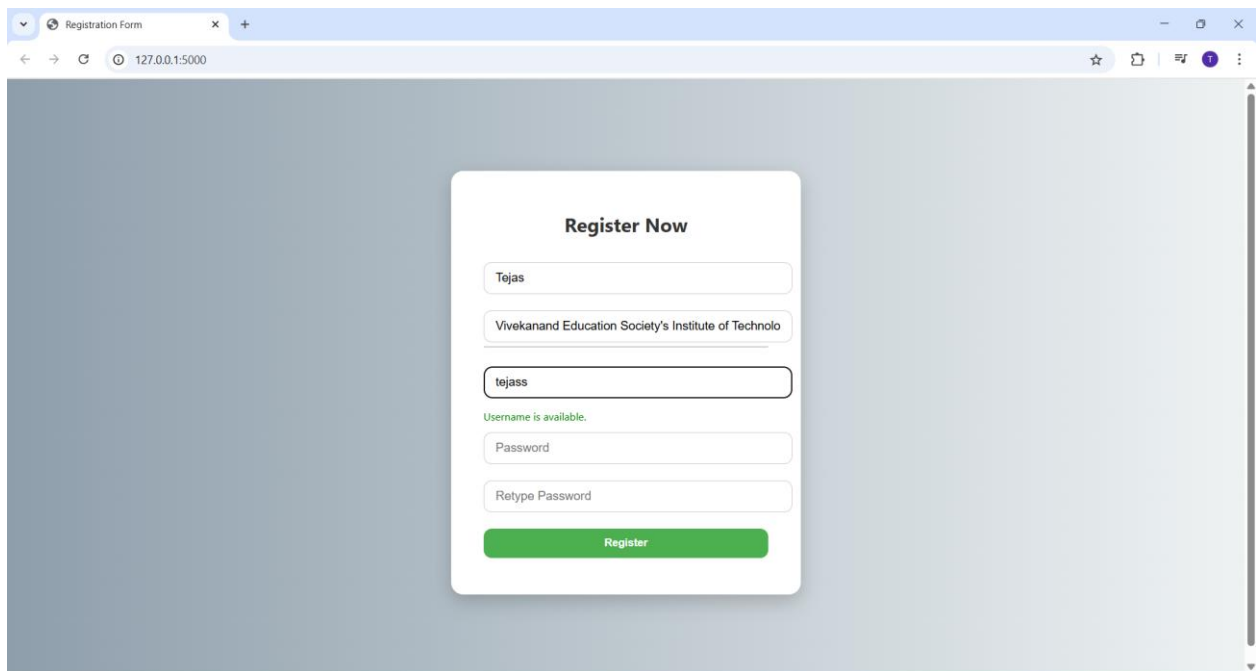
- Demonstration of dynamic college name suggestions as the user types into the College field.



- Validation alert shown when the entered username already exists in the database.

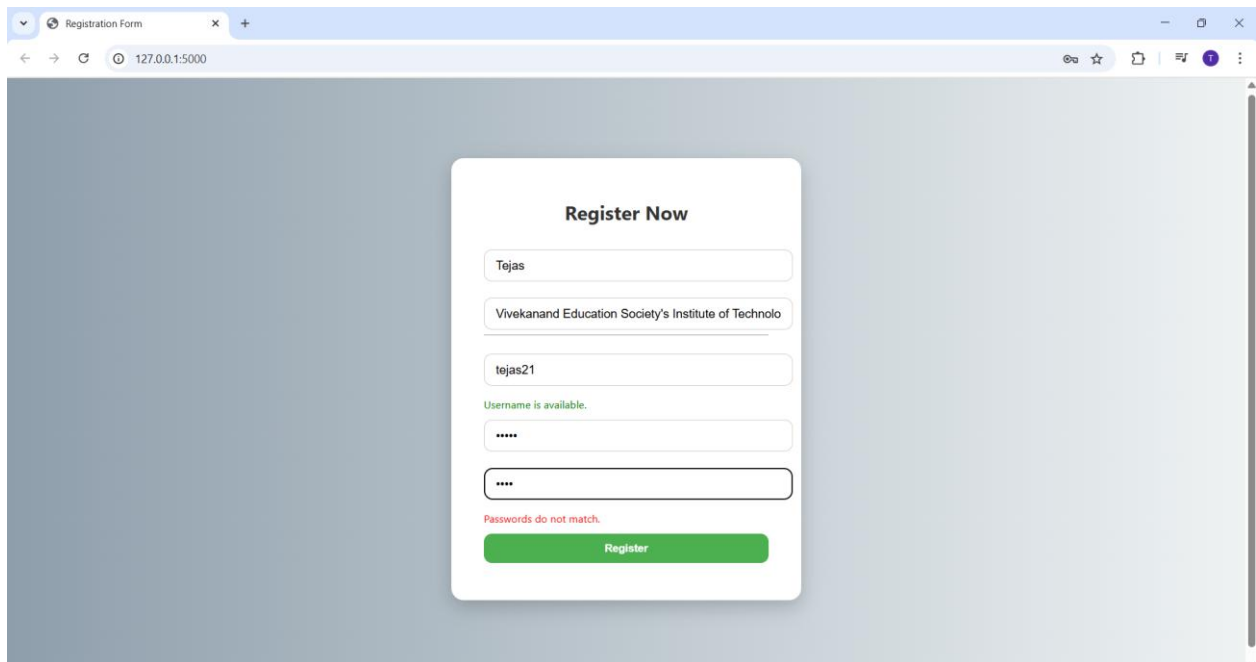


A screenshot of a web browser window titled "Registration Form" with the address bar showing "127.0.0.1:5000". The page displays a "Register Now" form with the following fields: Name (filled with "Tejas"), Institution (filled with "Vivekanand Education Society's Institute of Technolo"), Username (filled with "tejas02"), Password, and Retype Password. A red error message "Username already exists." is displayed below the Username field. A green "Register" button is at the bottom of the form.



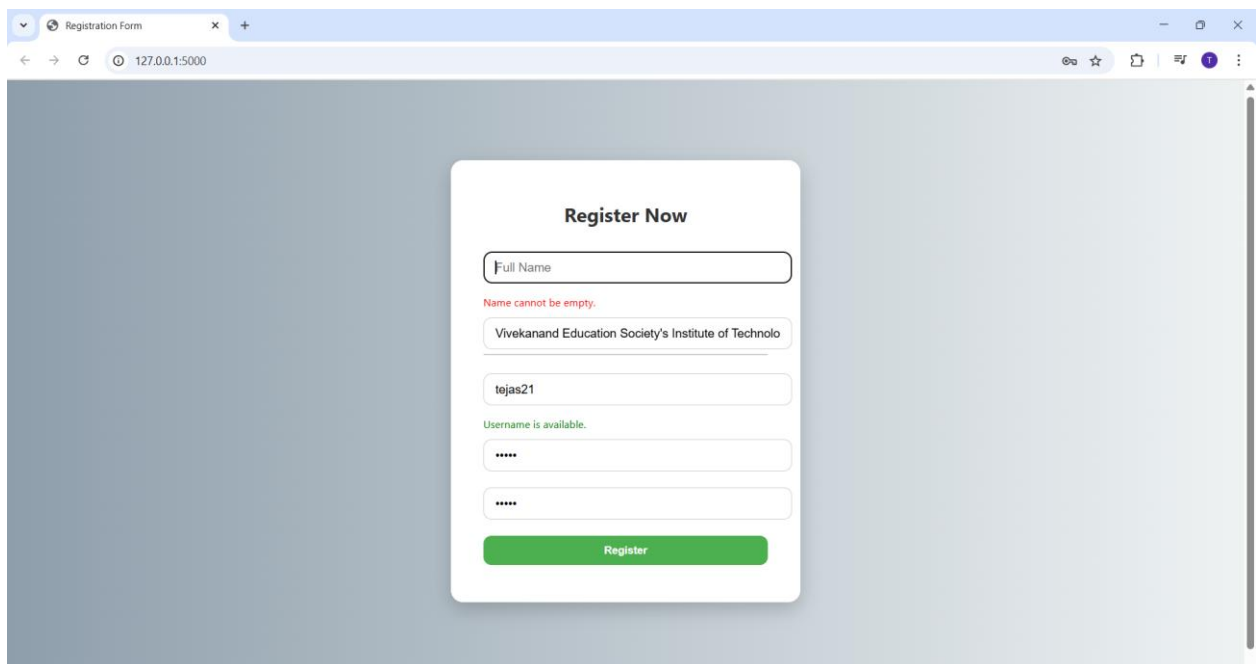
A screenshot of a web browser window titled "Registration Form" with the address bar showing "127.0.0.1:5000". The page displays a "Register Now" form with the following fields: Name (filled with "Tejas"), Institution (filled with "Vivekanand Education Society's Institute of Technolo"), Username (filled with "tejass"), Password, and Retype Password. A green success message "Username is available." is displayed below the Username field. A green "Register" button is at the bottom of the form.

- Message displayed when Password and Retype Password fields do not match.



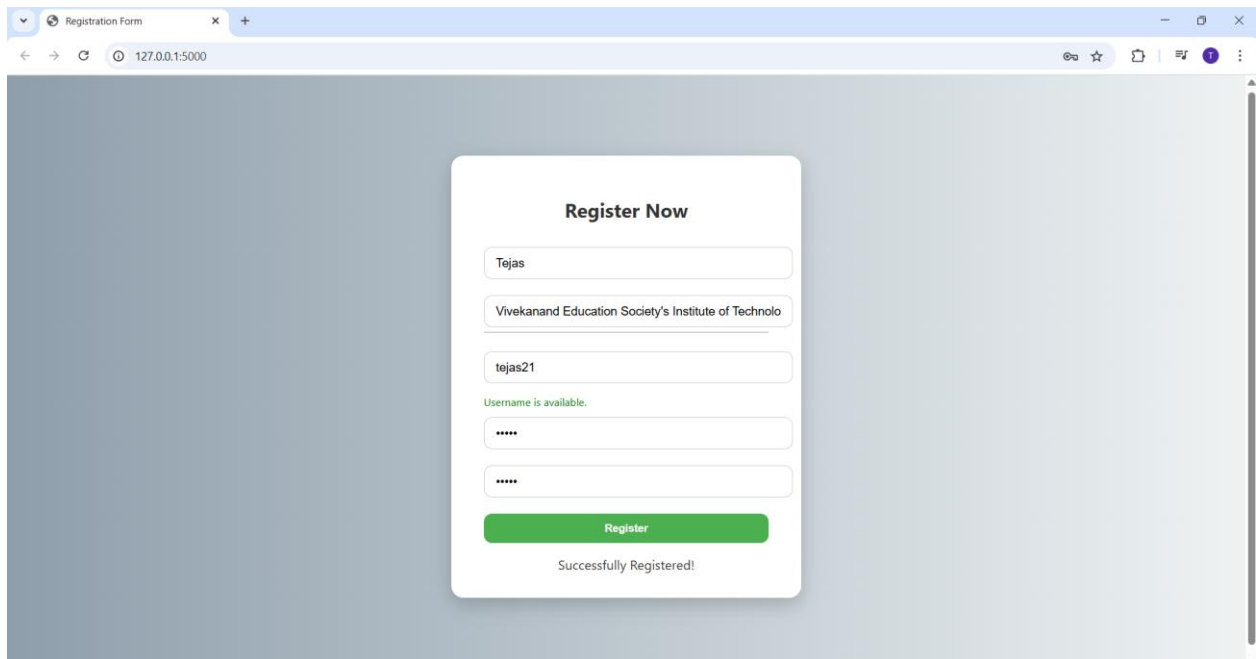
The screenshot shows a web browser window with the title "Registration Form" and the address bar displaying "127.0.0.1:5000". The main content area features a "Register Now" form. The form fields are: "Full Name" (containing "Tejas"), "Institution" (containing "Vivekanand Education Society's Institute of Technolo"), "Username" (containing "tejas21" with a green message "Username is available." below it), "Password" (containing "*****"), and "Retype Password" (containing "****"). A red error message "Passwords do not match." is displayed below the Retype Password field. A green "Register" button is at the bottom of the form.

- Error message displayed when the Name field is left empty.



The screenshot shows the same web browser window and "Register Now" form. In this instance, the "Full Name" field is empty. A red error message "Name cannot be empty." is displayed below the "Full Name" field. The other fields ("Institution", "Username", "Password", and "Retype Password") and the "Register" button remain the same as in the previous screenshot.

➤ "Successfully Registered!" message displayed after correct form submission.



Conclusion: -

In this experiment, we developed a responsive registration form using **AJAX** on the frontend and **Flask** as the backend, enabling asynchronous communication between the client and the server. Using the XMLHttpRequest object, we created a registration form that validates inputs such as username availability, empty name fields, and password matching in real-time. We also added an auto-suggestion feature for college names based on user input. All updates were handled without refreshing the page, showcasing the responsiveness of AJAX. This practical helped us understand how AJAX improves interactivity and performance in modern web applications.