Manav 19 D15A

DATE:

Experiment 6

Aim - To Study AJAX

Problem Statement -
Create a registration page having fields like Name, College, Username and password.

Theory -

A. How do Synchronous and Asynchronous Requests differ?

→ Synchronous Requests -
1. Block the user interface until the server responds.
2. Browser waits for the response before executing other code.
3. User cannot interact with the page while waiting.
4. Not recommended due to slow and unresponsive user experience

Asynchronous Requests -
1. Don't block the user interface.
2. Browser sends the request and continues executing other code while waiting for response.
3. User can interact with the page while waiting for the response
4. A callback function is executed to handle the response when server responds.

DATE:

B. Describe various properties and method used in XMLHttp Request object.

-). Properties -

1. ready State - Indicates the current state of XML HttpRequest object.
2. status - Returns the HTTP status code of the servers response to request
3. response - Returns the response content as a string

Methods -

1. open (): Initializes the request by specifying the method and URL
2. send () : sends the request to server
3. abort(): cancels the current request
4. add Event listner (): Registers an event listener for the object.

Code -

index.html

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Registration Form</title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <h1>Registration Form</h1>
    <form>
      <div>
        <label for="name">Name:</label>
        <input type="text" id="name" name="name">
        <span id="name-error"></span>
      </div>
      <div>
        <label for="college">College:</label>
        <input
          type="text"
          id="college"
          name="college"
          placeholder="Enter your college"
          list="colleges"
          required
        />
        <datalist id="colleges">
          <option value="VESIT"></option>
          <option value="TSEC"></option>
          <option value="RAIT"></option>
          <option value="KJ Somaiya"></option>
        </datalist>

        <span id="college-error"></span>
      </div>
      <div>
        <label for="username">Username:</label>
        <input type="text" id="username" name="username">
        <span id="username-error"></span>
      </div>
      <div>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password">
        <span id="password-error"></span>
      </div>
      <div>
        <label for="confirm-password">Retype Password:</label>
        <input type="password" id="confirm-password" name="confirm-password">
        <span id="confirm-password-error"></span>
      </div>
      <div>
        <button type="submit">Register</button>
      </div>
      <div>
        <span id="registration-message"></span>
      </div>
    </form>
    <script src="script.js"></script>
  </body>
</html>
```

script.js

```
// Get the form element from the HTML document
const form = document.querySelector('form');

// Get the input fields from the form
const nameField = form.querySelector('#name');
const collegeField = form.querySelector('#college');
const usernameField = form.querySelector('#username');
const passwordField = form.querySelector('#password');
const confirmPasswordField = form.querySelector('#confirm-password');

// Get the error messages elements
const nameError = form.querySelector('#name-error');
const collegeError = form.querySelector('#college-error');
const usernameError = form.querySelector('#username-error');
const passwordError = form.querySelector('#password-error');
const confirmPasswordError = form.querySelector('#confirm-password-error');
const registrationMessage = form.querySelector('#registration-message');

// Initialize the users array to store registered users in local storage
let users = JSON.parse(localStorage.getItem('users')) || [];

// Function to check if a username is already registered
function isUsernameTaken(username) {
  return users.some(user => user.username === username);
}

// Function to add a new user to the users array
function addUser(user) {
  users.push(user);
  localStorage.setItem('users', JSON.stringify(users));
}

// Function to handle form submission using XMLHttpRequest
function handleFormSubmitXHR(event) {
  event.preventDefault();

  // Reset the error messages
  nameError.textContent = '';
  collegeError.textContent = '';
  usernameError.textContent = '';
  passwordError.textContent = '';
  confirmPasswordError.textContent = '';
  registrationMessage.textContent = '';
```

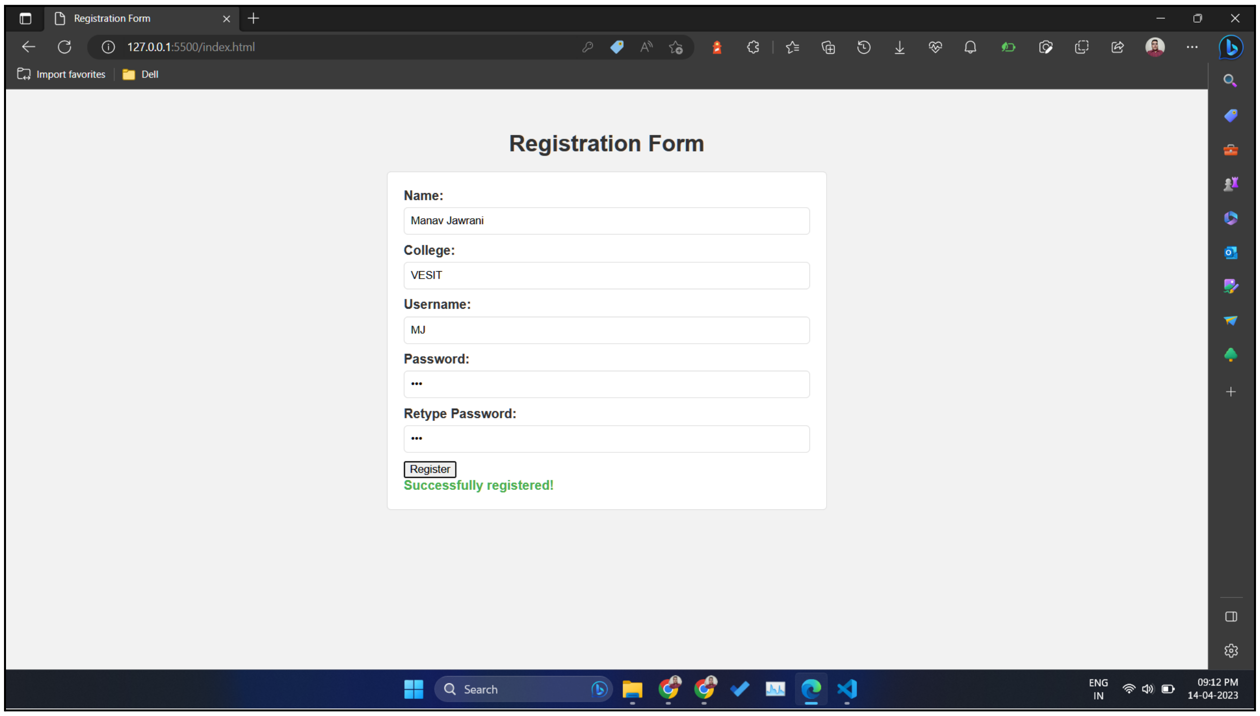
```
// Get the form values
const name = nameField.value.trim();
const college = collegeField.value.trim();
const username = usernameField.value.trim();
const password = passwordField.value;
const confirmPassword = confirmPasswordField.value;

let isValid = true;

// Validate the name field
if (name === '') {
  nameError.textContent = 'Please enter your name';
  isValid = false;
}

// Validate the college field
if (college === '') {
  collegeError.textContent = 'Please enter your college';
  isValid = false;
}

// Validate the username field
if (username === '') {
  usernameError.textContent = 'Please enter a username';
  isValid = false;
} else if (isUsernameTaken(username)) {
  usernameError.textContent = 'This username is already taken';
  isValid = false;
}

// Validate the password fields
if (password === '') {
  passwordError.textContent = 'Please enter a password';
  isValid = false;
```

```
}
if (confirmPassword === '') {
  confirmPasswordError.textContent = 'Please retype your password';
  isValid = false;
}
if (password !== confirmPassword) {
  confirmPasswordError.textContent = 'The passwords do not match';
  isValid = false;
}

if (isValid) {
  // Create a new user object
  const user = {
    name,
    college,
    username,
    password
  };

  // Send a POST request to the server to add the new user
  const xhr = new XMLHttpRequest();
  xhr.open('POST', '/register');
  xhr.setRequestHeader('Content-Type', 'application/json');
  xhr.onload = function() {
    if (xhr.status === 200) {
      // Add the new user to the users array and save it to local storage
      addUser(user);

      // Show the success message
      registrationMessage.textContent = 'Successfully registered!';

      // Replace the submit event listener with the XHR version
      form.removeEventListener('submit', handleFormSubmit);
      form.addEventListener('submit', handleFormSubmitXHR);
  }
};

};

};
```

style.css

```
body {
    font-family: Arial, sans-serif;
    background-color: #f2f2f2;
    color: #333;
  }

  h1 {
    font-size: 28px;
    text-align: center;
    margin-top: 50px;
  }

  form {
    max-width: 500px;
    margin: 0 auto;
    padding: 20px;
    background-color: #fff;
    border: 1px solid #ddd;
    border-radius: 5px;
  }

  form label {
    display: block;
    margin-bottom: 5px;
    font-weight: bold;
  }

  form input[type="text"],
form input[type="password"] {
  width: 100%;
  padding: 8px;
  border: 1px solid #ddd;
  border-radius: 5px;
  margin-bottom: 10px;
  box-sizing: border-box;
}

form input[type="submit"] {
  display: block;
  width: 100%;
  background-color: #4CAF50;
  color: #fff;
  border: none;
  border-radius: 5px;
  padding: 10px;
  cursor: pointer;
}

form input[type="submit"]:hover {
  background-color: #3e8e41;
}

#registration-message {
  text-align: center;
  margin-top: 20px;
  font-weight: bold;
  color: #4CAF50;
}
```

Output -


Registration Form

Name:
Manav Jawrani

College:
VESIT

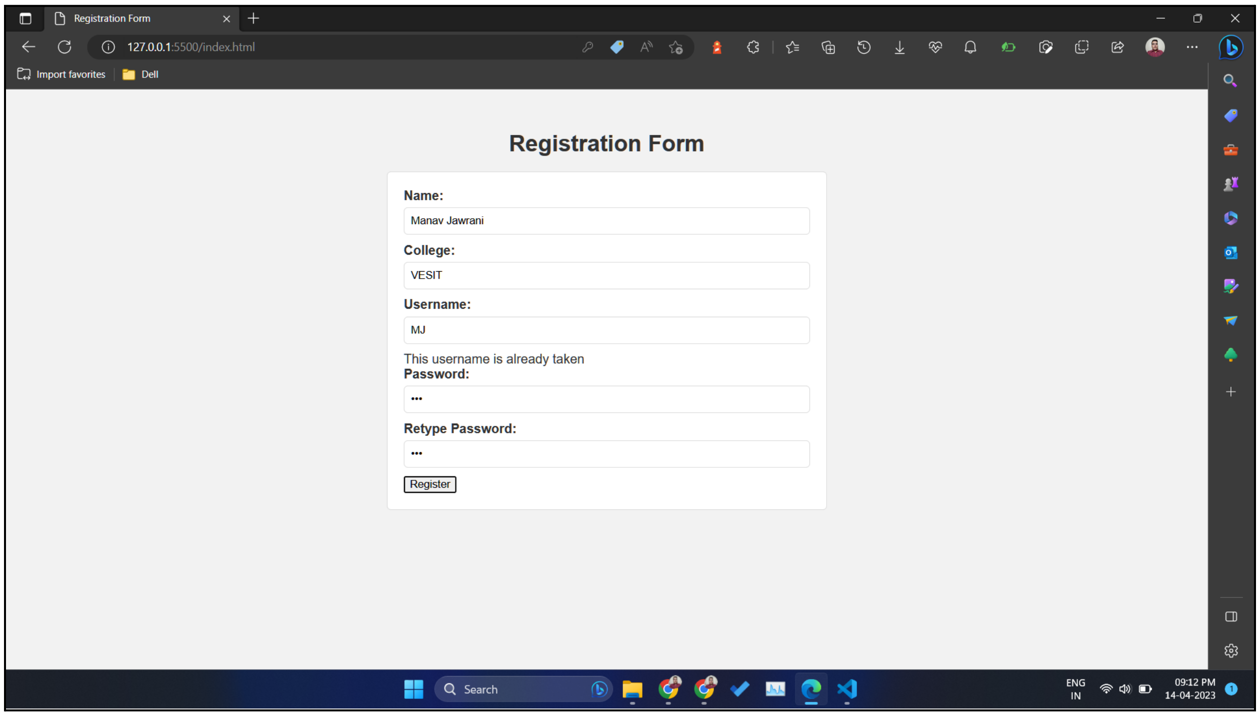
Username:
MJ

Password:
•••

Retype Password:
•••

Register

Successfully registered!


Registration Form

Name:
Manav Jawrani

College:
VESIT

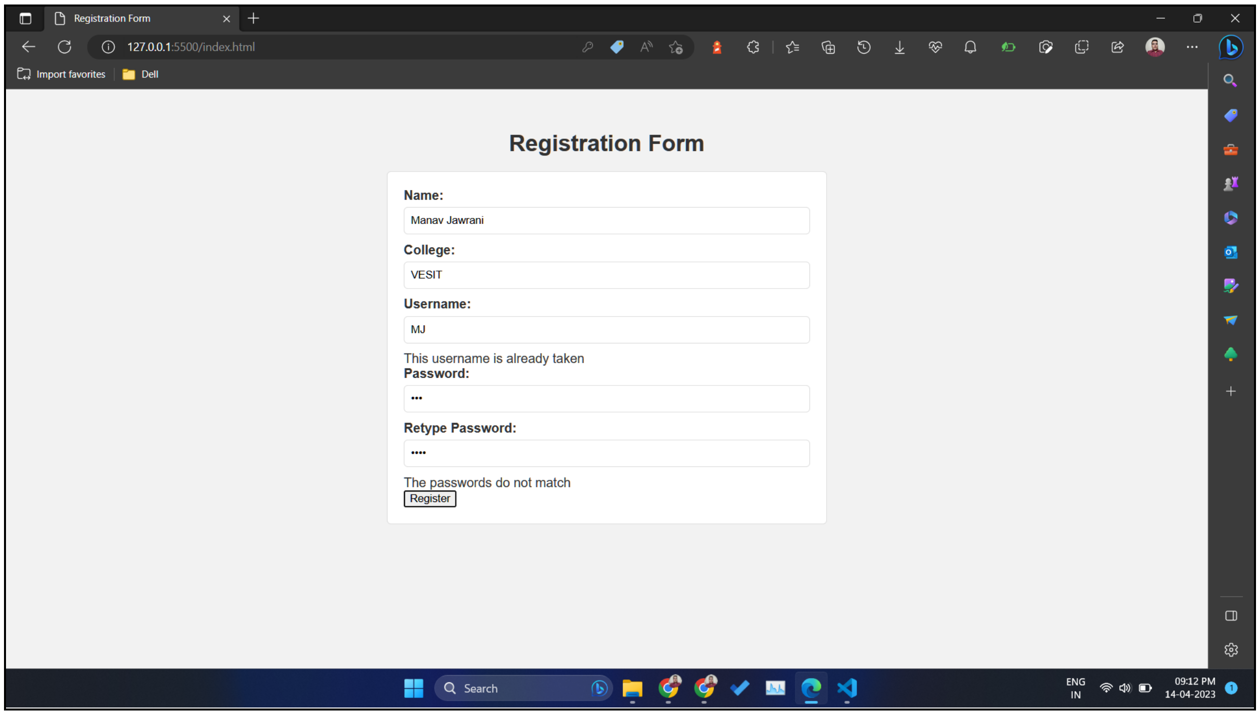
Username:
MJ

This username is already taken

Password:
•••

Retype Password:
•••

Register



Registration Form

Name:

Manav Jawrani

College:

VESIT

Username:

MJ

This username is already taken

Password:

•••

Retype Password:

••••

The passwords do not match

Register