Professor Name : Prof. Prachi Sonawane	Class : FY.BSC-CS Semester : Sem II (2024-2025)
Course code : VSC	Subject: Web Technologies

Sr. No.	Date	Index	Page No.	Sign
1		**Design a Web Page with Basic HTML Elements:** Create a web page that includes: a) A title, heading, and paragraph text. b) An ordered list and an unordered list. c) A table with at least two rows and two columns, styled with a border	....	....
2		**Add Images and Links:** a) Use an image map to make specific areas of the image clickable. b) Add hyperlinks to external websites and within the same document (anchors).	....	....
3		**Style a Web Page Using CSS:** Use inline, internal, and external CSS to: a) Change the background color or add a background image to the web page. b) Customize text styles, such as font size, color, and spacing.	....	....
4		**Interactive Features with JavaScript:** Write a script that: a) Displays a welcome message using a popup box when the page loads. b) Implements a simple form validation to ensure all required fields are filled and a valid email address is entered.	....	....
5		**Work with Multimedia and Browser Objects:** a) Embed an audio or video file in your web page with playback controls. b) Create a script to display the current date and time on the page. c) Create a script to use the browser objects to display the browser and operating system details.	....	....
6		**Create and Validate an XML Document:** a) Design an XML document to store data about books (e.g., title, author, genre, and price). b) b) Write a DTD (either internal or external) to define the structure of the XML document and validate it against the DTD.	....	....
7		**Transform XML with XSLT:** a) Create an XSLT file to transform the XML document of books into an HTML table.	....	....

8		**Create an Asynchronous Web Application:** a) Build a web page where: A button fetches the current weather information from a server or mock API using an AJAX request & Display the fetched data dynamically without reloading the page.	……………….	……………………
9		**Interactive Elements Using AJAX:** a) Develop a feature where: A dropdown list dynamically updates its options based on another dropdown selection using AJAX.	……………….	……………………
10		**Build a Simple Web Application with PHP:** Create a web application that: a) Allows users to fill out a form to submit their name, email, and message. b) Stores the form data in a database (e.g., MySQL) using PHP. c) Implements session management to display a personalized greeting for logged-in users.	……………….	……………………

Experiment 1: Design a Web Page with Basic HTML Elements

Aim:

Create a web page that includes:

a) A title, heading, and paragraph text.

b) An ordered list and an unordered list.

c) A table with at least two rows and two columns, styled with a border.

Tools Used:

- **Text Editor**: Any text editor (e.g., Visual Studio Code, Sublime Text, Notepad++) to write the HTML code.
- **Web Browser**: Google Chrome, Mozilla Firefox, or any modern browser to view the output.

Learning Objectives:

1. Learn how to structure a basic HTML web page.
2. Understand the use of common HTML elements like headings, paragraphs, lists, and tables.
3. Learn how to style tables using basic CSS properties like border.
4. Understand how HTML elements are used for structuring content on the web.

Theory:

- **HTML (HyperText Markup Language)** is the standard language for creating webpages. It defines the structure of web pages.
- **Elements in HTML** include tags like <h1>, <p>, <ul>, <ol>, <li>, and <table> which are used to display headings, paragraphs, lists, and tables.

- **Ordered List <ol>**: Used to display items in a numbered list.
- **Unordered List <ul>**: Used to display items in a bullet-point list.
- **Tables <table>**: Used to display data in rows and columns.

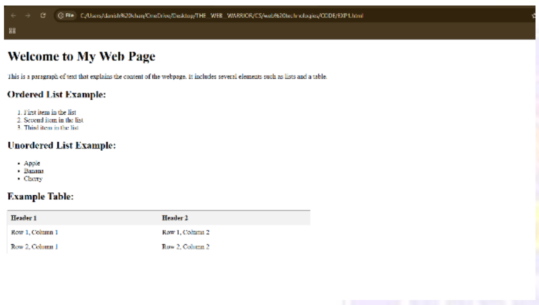
Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>My Web Page</title>
  <style>
    table {
      border: 1px solid black;
      border-collapse: collapse;
      width: 50%;
    }
    table, th, td {
      padding: 8px;
      text-align: left;
    }
    th {
      background-color: #f2f2f2;
    }
  </style>
</head>
```

```
<body>

  <h1>Welcome to My Web Page</h1>

  <p>This is a paragraph of text that explains the content of the webpage. It includes several
elements such as lists and a table.</p>

  <h2>Ordered List Example:</h2>

  <ol>

    <li>First item in the list</li>

    <li>Second item in the list</li>

    <li>Third item in the list</li>

  </ol>

  <h2>Unordered List Example:</h2>

  <ul>

    <li>Apple</li>

    <li>Banana</li>

    <li>Cherry</li>

  </ul>

  <h2>Example Table:</h2>

  <table>

    <tr>

      <th>Header 1</th>

      <th>Header 2</th>

    </tr>

    <tr>

      <td>Row 1, Column 1</td>

      <td>Row 1, Column 2</td>

    </tr>
```

```
        <tr>
            <td>Row 2, Column 1</td>
            <td>Row 2, Column 2</td>
        </tr>
    </table>
</body>
</html>
```

Output:



C:/Users/danish%20khan/OneDrive/Desktop/THE_WEB_WARRIOR/CS/web%20technologies/CODE/EXP1.html

Welcome to My Web Page

This is a paragraph of text that explains the content of the webpage. It includes several elements such as lists and a table.

Ordered List Example:

1. First item in the list
2. Second item in the list
3. Third item in the list

Unordered List Example:

- Apple
- Banana
- Cherry

Example Table:

Header 1	Header 2
Row 1, Column 1	Row 1, Column 2
Row 2, Column 1	Row 2, Column 2

Learning Outcomes:

- Understand the role of HTML in web page creation.
- Be able to use HTML tags to organize text, lists, and tables.
- Gain familiarity with basic CSS for styling web elements like tables.

Course Outcomes:

- Understanding the fundamentals of web development.
- Practical experience in writing HTML code.
- Building a foundation for advanced web development concepts such as CSS, JavaScript, and web frameworks.

Conclusion:

This project demonstrates how to create a basic webpage using HTML elements. By completing this exercise, students gain a practical understanding of the fundamental elements that make up a website. These elements form the building blocks for more advanced web design.

Viva Questions:

1. What is the difference between an ordered list and an unordered list in HTML?
2. How can you style a table using CSS?
3. What is the role of the <head> and <body> sections in an HTML document?
4. How do you insert an image into a webpage in HTML?
5. What is the use of the <th> tag in HTML tables?
6. Can you explain the purpose of the border-collapse property in CSS?
7. How can you create hyperlinks in HTML?
8. What are the basic HTML tags you would use to create a webpage layout?

For Faculty use:

Correction Parameters	Formative Assessment[40%]	Timely Completion of Practical[40%]	Attendance Learning Attitude[20%]

Experiment 2: Add Images and Links

Aim:

a) Use an image map to make specific areas of the image clickable.

b) Add hyperlinks to external websites and within the same document (anchors).

Tools Used:

- **Text Editor**: Any text editor (e.g., Visual Studio Code, Sublime Text, Notepad++) to write the HTML code.
- **Web Browser**: Google Chrome, Mozilla Firefox, or any modern browser to view the output.

Learning Objectives:

1. Learn how to create **image maps** to add clickable areas to an image.
2. Understand the usage of **anchor tags** to link to external websites and internal sections of a webpage.
3. Gain an understanding of how to structure and link content effectively within a webpage.

Theory:

1. Image Maps:

An **Image Map** is a way to make certain parts of an image clickable. The clickable areas are defined by coordinates on the image, and each area can link to a different URL. To create an image map:

- Use the <map> tag to define the image map.
- The <area> tag is used to specify the clickable areas within the image.

2. Hyperlinks:

Hyperlinks are used to navigate to another location or page. The **anchor tag** (<a>) is used to create links:

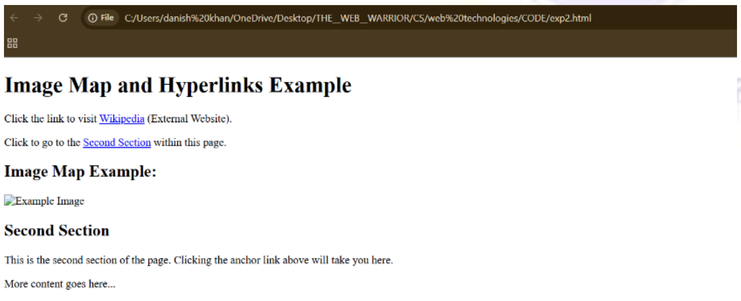
- **External Links**: Links to other websites are created by specifying a URL in the href attribute.
- **Internal Links (Anchors)**: Internal links are used to navigate to different sections within the same document. This is achieved by linking to specific id attributes on elements within the same page.

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Image Map and Links Example</title>
</head>
<body>
   <h1>Image Map and Hyperlinks Example</h1>
   <!-- External Link -->
   <p>Click the link to visit <a href="https://www.wikipedia.org" target="_blank">Wikipedia</a>
(External Website).</p>
   <!-- Internal Link (Anchor within the document) -->
   <p>Click to go to the <a href="#section2">Second Section</a> within this page.</p>
   <h2>Image Map Example:</h2>
   <img src="example-image.jpg" usemap="#image-map" alt="Example Image" width="500">
   <!-- Image Map Definition -->
   <map name="image-map">
```

```
        <!-- Coordinates for the clickable area -->
        <area shape="rect" coords="34,44,270,350" alt="Wikipedia" href="https://www.wikipedia.org" target="_blank">
        <area shape="circle" coords="400,150,40" alt="Google" href="https://www.google.com" target="_blank">
        <area shape="poly" coords="180,200,250,250,200,300" alt="Anchor Link" href="#section2">
    </map>

    <h2 id="section2">Second Section</h2>
    <p>This is the second section of the page. Clicking the anchor link above will take you here.</p>
    <p>More content goes here...</p>
</body>
</html>
```

Output:



C:/Users/danish%20khan/OneDrive/Desktop/THE__WEB__WARRIOR/CS/web%20technologies/CODE/exp2.html

Image Map and Hyperlinks Example

Click the link to visit Wikipedia (External Website).

Click to go to the Second Section within this page.

Image Map Example:

Example Image

Second Section

This is the second section of the page. Clicking the anchor link above will take you here.

More content goes here...

Learning Outcomes:

1. Understand how to create interactive images using image maps.
2. Learn how to create internal navigation using anchors and external links using the anchor (<a>) tag.
3. Gain experience in integrating images and links to make a webpage more dynamic and interactive.

Course Outcomes:

1. Understand basic HTML elements and their usage in web design.
2. Ability to create interactive and engaging web pages with images and links.
3. Learn how to implement practical web design elements like image maps and anchor links

Course Outcomes:

1. Understand basic HTML elements and their usage in web design.
2. Ability to create interactive and engaging web pages with images and links.
3. Learn how to implement practical web design elements like image maps and anchor links

Viva Questions:

1. What is an image map, and how do you create one in HTML?
2. What are the different types of shapes you can define for clickable areas in an image map?
3. How do you link to an external website in HTML?
4. How do you create internal navigation within the same webpage using anchor tags?
5. How does the target="_blank" attribute work when creating links?
6. What is the difference between the href and src attributes in HTML?
7. Can you describe the use of the coords attribute in an image map?
8. How can you make an image map accessible and user-friendly for all users?

For Faculty use:

Correction Parameters	Formative Assessment[40%]	Timely Completion of Practical[40%]	Attendance Learning Attitude[20%]

Experiment 3: Style a Web Page Using CSS

Aim:

Use inline, internal, and external CSS to:

a) Change the background color or add a background image to the web page.

b) Customize text styles, such as font size, color, and spacing.

Tools Used:

- **Text Editor**: Any text editor (e.g., Visual Studio Code, Sublime Text, Notepad++) to write the HTML and CSS code.
- **Web Browser**: Google Chrome, Mozilla Firefox, or any modern browser to view the output.

Learning Objectives:

1. Learn the three methods of applying CSS: inline, internal, and external.
2. Understand how to style backgrounds, text colors, and font properties in CSS.
3. Gain the ability to structure CSS for a webpage with different styling techniques.
4. Understand the role of CSS in enhancing the design and appearance of web pages.

Theory:

1. CSS (Cascading Style Sheets):

CSS is used to describe the presentation of an HTML document, including its layout, colors, fonts, and spacing. It allows you to separate content (HTML) from presentation (CSS).

Types of CSS:

1. **Inline CSS**: Styles applied directly to an HTML element using the style attribute.

2. **Internal CSS**: Styles defined inside the <style> tag in the <head> section of the HTML document.
3. **External CSS**: Styles are written in a separate .css file and linked to the HTML document using the <link> tag.

Key Properties:

- **Background**: background-color, background-image, background-repeat, etc.
- **Text Styling**: font-size, font-family, color, line-height, letter-spacing, etc.

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CSS Styling Example</title>
  <!-- Internal CSS -->
  <style>
    /* Background color and image */
    body {
      background-color: lightblue; /* Change background color */
      background-image: url('background-image.jpg'); /* Add a background image */
      background-size: cover; /* Ensure the background image covers the entire page */
      background-position: center center; /* Center the background image */
    }
    /* Customize text styles */
    h1 {
      font-size: 36px;
```

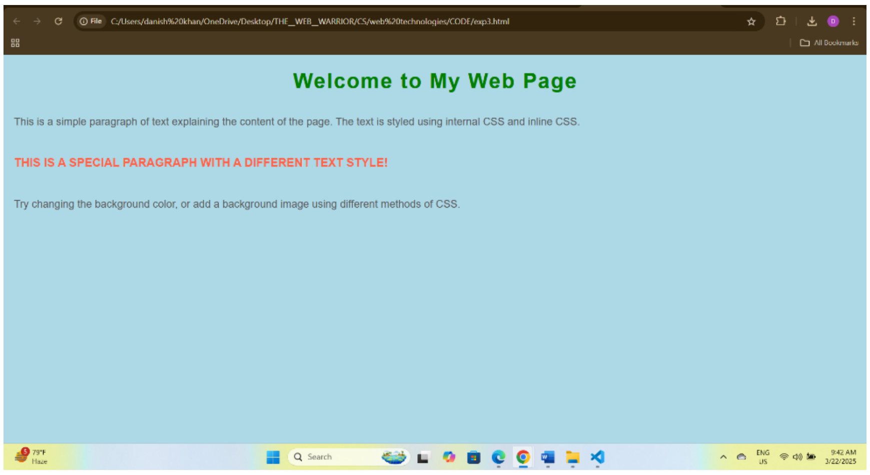
```
            color: darkblue;
            text-align: center;
            letter-spacing: 2px;
        }

        p {
            font-size: 18px;
            color: #333;
            line-height: 1.6;
            text-align: justify;
            padding: 10px;
        }

        /* Specific class for the paragraph text */
        .special-text {
            font-size: 20px;
            color: #ff6347; /* Tomato color */
            text-transform: uppercase;
            font-weight: bold;
        }
    </style>
    <!-- External CSS (Link to external stylesheet) -->
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <!-- Inline CSS -->
    <h1 style="color: green; text-align: center; font-family: Arial, sans-serif;">Welcome to My Web Page</h1>
    <p>This is a simple paragraph of text explaining the content of the page. The text is styled using internal CSS and inline CSS.</p>

    <!-- Using the class defined in internal CSS -->

    <p class="special-text">This is a special paragraph with a different text style!</p>

    <p>Try changing the background color, or add a background image using different methods of CSS.</p>

</body>

</html>
```

EXTERNAL CSS CODE:

STYLES.CSS

```
/* External CSS example */

body {

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

}

h1 {

    color: navy;

}

p {

    color: #555;

}
```

Output:



C:/Users/danish%20khan/OneDrive/Desktop/THE_WEB_WARRIOR/CS/web%20technologies/CODE/exp3.html

Welcome to My Web Page

This is a simple paragraph of text explaining the content of the page. The text is styled using internal CSS and inline CSS.

THIS IS A SPECIAL PARAGRAPH WITH A DIFFERENT TEXT STYLE!

Try changing the background color, or add a background image using different methods of CSS.

Learning Outcomes:

1. Understanding the difference between inline, internal, and external CSS.
2. Ability to customize various text styles and backgrounds using CSS.
3. Ability to structure a webpage's design using different CSS methods.
4. Gaining practical experience in styling text and page elements for better readability and user experience.

Course Outcomes:

1. Develop the skills to apply styles to web pages using CSS.
2. Gain hands-on experience in organizing CSS code for better maintainability.
3. Master basic layout and design techniques in web development.
4. Learn how to use various CSS properties for enhancing the user interface of websites.

Conclusion:

This project demonstrated the application of **inline**, **internal**, and **external CSS** to style a simple webpage. By changing the background color, adding text styling, and experimenting with different methods of applying CSS, students can enhance the visual appeal of a webpage. These skills are essential for building visually engaging websites.

Viva Questions:

1. What is the difference between inline, internal, and external CSS?
2. How can you apply CSS to an element using the style attribute?
3. What is the purpose of the font-family property in CSS?
4. How can you change the background of a webpage using CSS?
5. What are the advantages and disadvantages of using inline CSS over external CSS?

6. Explain how you can use the background-image property to set an image as the background.
7. What is letter-spacing, and how does it affect text styling in CSS?
8. How would you link an external CSS file to an HTML page?
9. What is the difference between text-align: center and text-align: justify?

For Faculty use:

Correction Parameters	Formative Assessment[40%]	Timely Completion of Practical[40%]	Attendance Learning Attitude[20%]

Experiment 4: Interactive Features with JavaScript

Aim:

Write a script that:

a) Displays a welcome message using a popup box when the page loads.

b) Implements a simple form validation to ensure all required fields are filled and a valid email address is entered.

Tools Used:

- **Text Editor**: Any text editor (e.g., Visual Studio Code, Sublime Text, Notepad++) to write HTML and JavaScript code.
- **Web Browser**: Google Chrome, Mozilla Firefox, or any modern browser to view the output.

Learning Objectives:

1. Understand how to use **JavaScript** to interact with users.
2. Learn how to display **popup messages** using the alert() function.
3. Understand how to implement **form validation** with JavaScript.
4. Learn how to validate user input to ensure data integrity (e.g., checking for required fields and a valid email format).

Theory:

1. JavaScript Alerts:

The alert() function in JavaScript is used to display a popup message to the user. It takes a string as an argument and shows it in a small modal box.

```
alert("Your welcome message");
```

2. Form Validation:

Form validation is the process of ensuring that the user provides the correct input in a form before submitting it. Common validations include:

- Checking if all required fields are filled.
- Validating email format.
- Using regular expressions (RegEx) to verify formats like email addresses.

A basic email validation pattern looks like this:

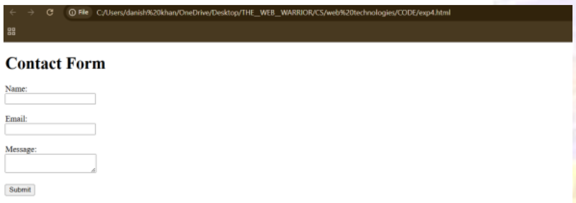
```
/^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/
```

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Form Validation and Welcome Popup</title>
  <script>
    // Function to show the welcome popup when the page loads
    window.onload = function() {
      alert("Welcome to our website!");
    }
    // Form validation function
    function validateForm() {
      // Get the form values
      var name = document.getElementById("name").value;
      var email = document.getElementById("email").value;
```

```
            var message = document.getElementById("message").value;

            // Validate name (must not be empty)

            if (name == "") {

                alert("Name is required!");

                return false;

            }

            // Validate email (must be in a valid email format)

            var emailPattern = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/;

            if (email == "") {

                alert("Email is required!");

                return false;

            } else if (!email.match(emailPattern)) {

                alert("Please enter a valid email address!");

                return false;

            }

            // Validate message (must not be empty)

            if (message == "") {

                alert("Message is required!");

                return false;

            }

            // If all validations pass, return true to submit the form

            alert("Form submitted successfully!");

            return true;

        }

    </script>

</head>

<body>

    <h1>Contact Form</h1>

    <form onsubmit="return validateForm()">

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

        <input type="text" id="name" name="name" required><br><br>
        <label for="email">Email:</label><br>
        <input type="email" id="email" name="email" required><br><br>
        <label for="message">Message:</label><br>
        <textarea id="message" name="message" required></textarea><br><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

Output:



```
C:/Users/danish%20khan/OneDrive/Desktop/THE_WEB_WARRIOR/CS/web%20technologies/CODE/exp4.html

Contact Form
Name:
Email:
Message:
Submit
```

Learning Outcomes:

1. Understand how to use JavaScript for creating interactive user interfaces.
2. Learn how to display **popup messages** using the alert() function.
3. Gain experience in implementing **form validation** with JavaScript to check required fields and validate data like email addresses.
4. Improve your ability to manage user interactions on a webpage using JavaScript.

Course Outcomes:

1. Ability to implement basic interactive features (alerts, form validation) in web pages.
2. Understanding how JavaScript works with HTML to enhance user experience.

3. Develop skills for validating user input and ensuring data integrity before submission.
4. Learn to enhance webpages with practical, real-world JavaScript solutions.

Conclusion:

This project demonstrated how to use JavaScript to create interactive features for web pages. By learning how to display welcome messages using popups and validate form inputs, students gain valuable skills for improving the functionality and usability of websites. JavaScript is an essential tool for adding interactivity to a webpage, and this exercise forms the foundation for more advanced web development topics.

Viva Questions:

1. What is the purpose of the window.onload function in JavaScript?
2. How does the alert() function work in JavaScript?
3. What is the role of form validation in web development?
4. Can you explain how the onsubmit attribute in the form is used with JavaScript?
5. How do you validate an email address using a regular expression in JavaScript?
6. Why is it important to check for required fields in a form before submitting?
7. What will happen if the user leaves one of the required fields empty in this form?
8. What are some other ways you can validate form inputs in JavaScript besides using alert()?

For Faculty use:

Correction Parameters	Formative Assessment[40%]	Timely Completion of Practical[40%]	Attendance Learning Attitude[20%]

Experiment 5: Work with Multimedia and Browser Objects

Aim:

a) Embed an audio or video file in your web page with playback controls.

b) Create a script to display the current date and time on the page.

c) Create a script to use the browser objects to display the browser and operating system details.

Tools Used:

- **Text Editor**: Any text editor (e.g., Visual Studio Code, Sublime Text, Notepad++) to write the HTML and JavaScript code.
- **Web Browser**: Google Chrome, Mozilla Firefox, or any modern browser to view the output.

Learning Objectives:

1. Learn how to embed **audio and video** content into a webpage with playback controls.
2. Understand how to use JavaScript to display **dynamic content**, such as the current date and time.
3. Use JavaScript to retrieve **browser details** (e.g., browser name, version) and **operating system** information.

Theory:

1. Embedding Multimedia (Audio/Video):

- The <audio> and <video> tags are used in HTML to embed multimedia content into a webpage.
- You can include controls (play, pause, volume, etc.) by adding the controls attribute.

Example for audio:

```
<audio controls>
    <source src="audiofile.mp3" type="audio/mp3">
    Your browser does not support the audio element.
</audio>
```

Example for video:

```
<video controls width="600">
    <source src="video.mp4" type="video/mp4">
    Your browser does not support the video tag.
</video>
```

2. Displaying Date and Time:

- You can use JavaScript's Date() object to get the current date and time.
- The setInterval() method can be used to update the time every second.

Example:

```
var now = new Date();
document.getElementById("time").innerHTML = now.toLocaleString();
```

3. Browser Objects:

- The navigator object is used to retrieve information about the browser (name, version, platform).

Example:

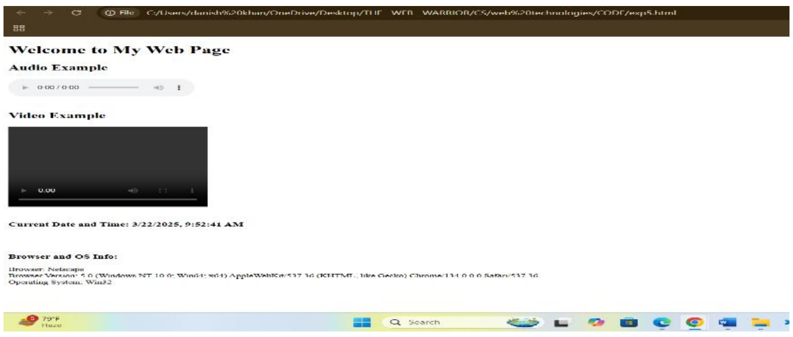
```
navigator.userAgent
navigator.platform
```

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Web Page with Audio, Date & Browser Info</title>
  <script>
    // Display current date and time
    function displayDateTime() {
      var currentDate = new Date();
      var formattedDate = currentDate.toLocaleString(); // Display date and time in local format
      document.getElementById("current-date-time").innerHTML = "Current Date and Time: " + formattedDate;
    }
    // Display browser and OS details
    function displayBrowserInfo() {
      var browserInfo = "Browser: " + navigator.appName + "<br>" +
                        "Browser Version: " + navigator.appVersion + "<br>" +
                        "Operating System: " + navigator.platform;
      document.getElementById("browser-info").innerHTML = browserInfo;
    }
    // Call the functions when the page loads
    window.onload = function() {
      displayDateTime();
      displayBrowserInfo();
    }
  </script>
</head>
<body>
```

```
    <h1>Welcome to My Web Page</h1>
    <!-- Embed an audio file with playback controls -->
    <h2>Audio Example</h2>
    <audio controls>
        <source src="your-audio-file.mp3" type="audio/mp3">
        Your browser does not support the audio element.
    </audio>
    <br><br>
    <!-- Embed a video file with playback controls -->
    <h2>Video Example</h2>
    <video width="320" height="240" controls>
        <source src="your-video-file.mp4" type="video/mp4">
        Your browser does not support the video element.
    </video>
    <br><br>
    <!-- Display the current date and time -->
    <h3 id="current-date-time"></h3>
    <br><br>
    <!-- Display browser and OS info -->
    <h3>Browser and OS Info:</h3>
    <div id="browser-info"></div>
</body>
</html>
```

Output:



C:/Users/danish%20khan/OneDrive/Desktop/THE WEB WARRIOR/CS/web%20technologies/CODE/exp5.html

```
Welcome to My Web Page
Audio Example
Video Example
Current Date and Time: 3/22/2025, 9:52:41 AM

Browser and OS Info:
Browser: Netscape
Browser Version: 5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36
Operating System: Win32
```

Learning Outcomes:

1. Multimedia Integration: Learn how to embed audio and video content in a webpage using HTML5 <audio> and <video> tags.
2. JavaScript Date and Time Handling: Understand how to retrieve and display dynamic content (current date and time) using JavaScript.
3. Using Browser Objects: Learn how to retrieve and display information about the user's browser and operating system using the navigator object.

Course Outcomes:

1. Develop practical skills in embedding multimedia content on web pages.
2. Gain proficiency in JavaScript for dynamic content handling and browser interaction.
3. Learn how to enhance user experience by displaying relevant browser and system information.
4. Understand and apply the fundamentals of multimedia and dynamic data handling in web development.

Conclusion:

This project provided an introduction to working with multimedia elements (audio and video) and using JavaScript to display dynamic content like date, time, and browser details. These techniques are essential for creating interactive and responsive web applications that cater to diverse user environments.

Viva Questions:

1. How do you embed an audio file in an HTML webpage?
2. What is the purpose of the controls attribute in the <audio> and <video> tags?
3. How can you dynamically update the time on a webpage using JavaScript?
4. What is the use of the navigator.userAgent and navigator.platform in JavaScript?

5. How does the setInterval() method work in JavaScript?
6. Can you explain how the Date() object in JavaScript helps display dynamic information on a webpage?
7. What would happen if the browser does not support the audio or video tag?
8. How can you make the page display the current time in a different format?

For Faculty use:

Correction Parameters	Formative Assessment[40%]	Timely Completion of Practical[40%]	Attendance Learning Attitude[20%]

Experiment 6: Create and Validate an XML Document

Aim:

a)Design an XML document to store data about books (e.g., title, author, genre, and price).
b) Write a DTD (either internal or external) to define the structure of the XML document and validate it against the DTD.

Tools Used:

- **Text Editor**: Any text editor (e.g., Visual Studio Code, Sublime Text, Notepad++) to write the XML and DTD files.
- **Web Browser**: Modern web browsers (Google Chrome, Mozilla Firefox, etc.) to test XML validation.
- **XML Validator**: Use an online XML validator tool (like XML Validation) to validate the XML document against the DTD.

Learning Objectives:

1. Learn how to design an **XML document** to represent data in a structured way.
2. Understand how to create a **DTD** to define the structure and rules for XML data.
3. Learn how to **validate** an XML document against a DTD to ensure it conforms to the structure.

Theory:

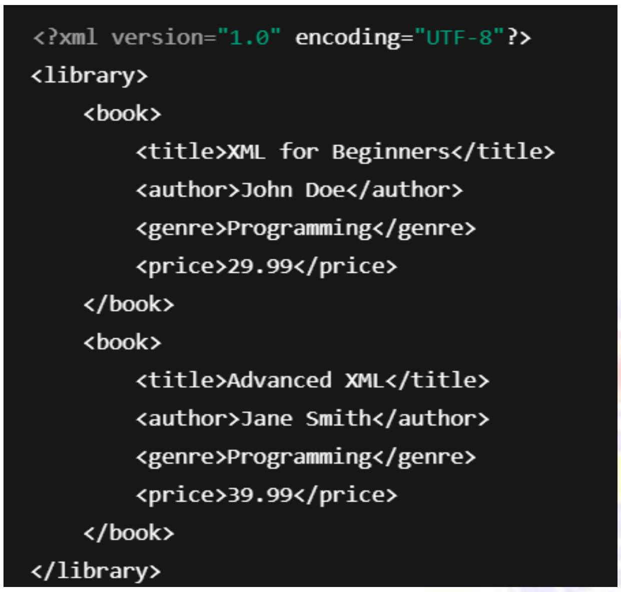
1. XML (Extensible Markup Language):

XML is a markup language used to store and transport data. It uses a tag-based syntax similar to HTML but is designed for describing data rather than presenting it.

- **XML Document Structure**:
    - An XML document must have a single root element.

- Elements are enclosed in opening and closing tags.
- The XML document must be well-formed (e.g., no missing closing tags, properly nested elements).

Example of an XML document representing books:



```
<?xml version="1.0" encoding="UTF-8"?>
<library>
    <book>
        <title>XML for Beginners</title>
        <author>John Doe</author>
        <genre>Programming</genre>
        <price>29.99</price>
    </book>
    <book>
        <title>Advanced XML</title>
        <author>Jane Smith</author>
        <genre>Programming</genre>
        <price>39.99</price>
    </book>
</library>
```

2. DTD (Document Type Definition):

A DTD defines the structure of an XML document. It specifies the allowed elements, their attributes, and their relationships. It can be defined internally (within the XML file) or externally (in a separate file).

- **DTD Structure**:
  - It defines **element types** (like <book>, <title>, etc.).
  - Specifies the order of elements.
  - Defines attributes for elements.
  - Provides **validation rules** for the document.

3. Validating XML against DTD:

- To validate an XML document against a DTD, you can either:
  1. Include the DTD within the XML document using the <!DOCTYPE> declaration.

2. Reference an external DTD file using a SYSTEM or PUBLIC identifier.

Code:

XML Document to Store Data About Books:

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
  <book>
    <title>Harry Potter and the Sorcerer's Stone</title>
    <author>J.K. Rowling</author>
    <genre>Fantasy</genre>
    <price>19.99</price>
  </book>
  <book>
    <title>The Great Gatsby</title>
    <author>F. Scott Fitzgerald</author>
    <genre>Fiction</genre>
    <price>10.99</price>
  </book>
  <book>
    <title>1984</title>
    <author>George Orwell</author>
    <genre>Dystopian</genre>
    <price>14.99</price>
  </book>
</books>
```

Internal DTD (Embedded within the XML Document):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE books [
  <!ELEMENT books (book+)>
  <!ELEMENT book (title, author, genre, price)>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT author (#PCDATA)>
  <!ELEMENT genre (#PCDATA)>
  <!ELEMENT price (#PCDATA)>
]>
<books>
  <book>
    <title>Harry Potter and the Sorcerer's Stone</title>
    <author>J.K. Rowling</author>
    <genre>Fantasy</genre>
    <price>19.99</price>
  </book>
  <book>
    <title>The Great Gatsby</title>
    <author>F. Scott Fitzgerald</author>
    <genre>Fiction</genre>
    <price>10.99</price>
  </book>
  <book>
    <title>1984</title>
    <author>George Orwell</author>
    <genre>Dystopian</genre>
    <price>14.99</price>
  </book>
</books>
```

External DTD (Separate File - books.dtd):

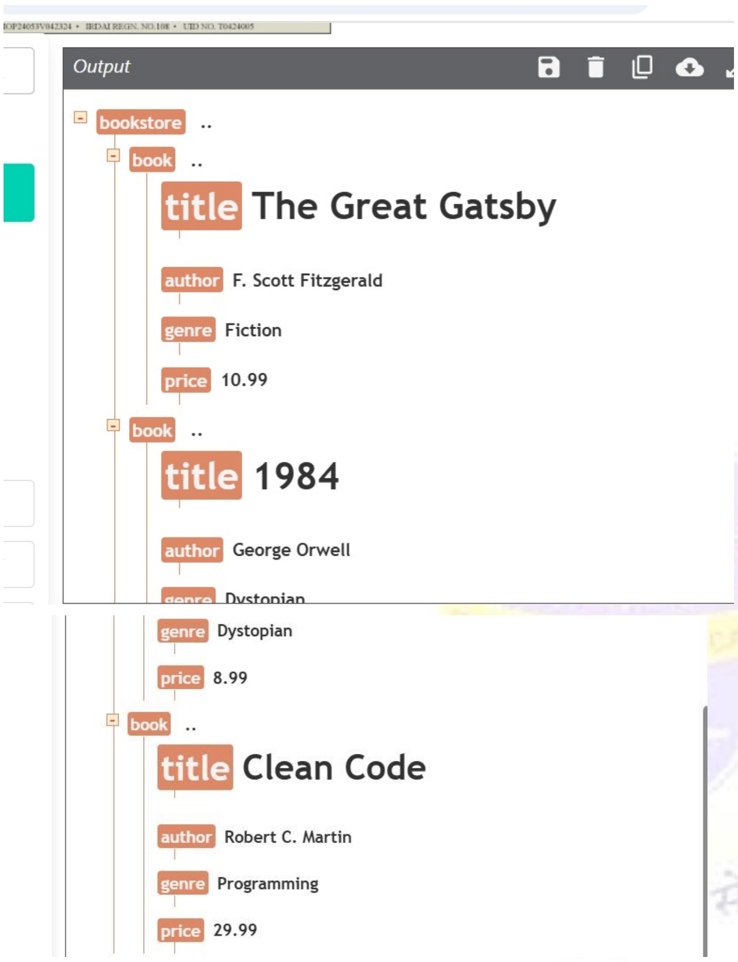
```
<!ELEMENT books (book+)>
<!ELEMENT book (title, author, genre, price)>
```

```
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT genre (#PCDATA)>
<!ELEMENT price (#PCDATA)>
```

XML Document Referencing the External DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE books SYSTEM "books.dtd">
<books>
  <book>
    <title>Harry Potter and the Sorcerer's Stone</title>
    <author>J.K. Rowling</author>
    <genre>Fantasy</genre>
    <price>19.99</price>
  </book>
  <book>
    <title>The Great Gatsby</title>
    <author>F. Scott Fitzgerald</author>
    <genre>Fiction</genre>
    <price>10.99</price>
  </book>
  <book>
    <title>1984</title>
    <author>George Orwell</author>
    <genre>Dystopian</genre>
    <price>14.99</price>
  </book>
</books>
```

Output:



```
Output
- bookstore ..
  - book ..
      title The Great Gatsby
      author F. Scott Fitzgerald
      genre Fiction
      price 10.99
  - book ..
      title 1984
      author George Orwell
      genre Dystopian
      price 8.99
  - book ..
      title Clean Code
      author Robert C. Martin
      genre Programming
      price 29.99
```

Learning Outcomes:

1. Gain an understanding of the XML document structure and how to store and organize data effectively.
2. Learn how to create and implement a DTD to define and validate XML document structure.
3. Learn how to ensure that the data in the XML document adheres to the required structure through validation.
4. Understand how to integrate internal and external DTDs in XML documents.

Course Outcomes:

1. Ability to create and manipulate **XML documents** for data storage and transmission.
2. Knowledge of **DTDs** and how to define the structure of XML data.
3. Understand the importance of **validating XML** documents to ensure data integrity and consistency.
4. Ability to **integrate DTDs** with XML documents for structured data management in real-world applications.

Conclusion:

This exercise has demonstrated how to create an XML document to represent data about books and how to use a DTD to define and validate the structure of that document. Validating XML ensures that the document is correctly structured and the data is consistent with the expected format. Mastery of XML and DTDs is crucial for any application that deals with structured data, as it enables data storage, transmission, and validation.

Viva Questions:

1. What is the difference between XML and HTML?
2. What does the <!DOCTYPE> declaration do in an XML document?
3. How does a DTD define the structure of an XML document?
4. What would happen if an XML document doesn't adhere to the DTD structure?
5. Can an XML document be validated without a DTD? If so, how?
6. What is the significance of the #PCDATA keyword in a DTD?
7. How can you reference an external DTD in an XML document?
8. What is the purpose of defining a DTD for an XML document?
9. Can you validate an XML document using other methods besides a DTD (e.g., XML Schema)?

For Faculty use:

Correction Parameters	Formative Assessment[40%]	Timely Completion of Practical[40%]	Attendance Learning Attitude[20%]

Experiment 7: Transform XML with XSLT

Aim:

a)Create an XSLT file to transform the XML document of books into an HTML table.

Tools Used:

- **Text Editor**: Any text editor (e.g., Visual Studio Code, Sublime Text, Notepad++) to write the XML, XSLT, and HTML files.
- **Web Browser**: Modern web browsers (Google Chrome, Mozilla Firefox, etc.) to test XML and XSLT transformation.
- **XSLT Processor**: A browser that supports XSLT or an online XSLT processor.

Learning Objectives:

1. Learn how to **transform XML data** into a human-readable format using **XSLT**.
2. Understand the **separation of concerns**: separating data from its presentation layer.
3. Learn the basic syntax and functions of **XSLT** for transforming XML to other formats (HTML in this case).
4. Learn how to display XML data in a structured format (HTML table) using XSLT.

Theory:

1. XML (Extensible Markup Language):

XML is used to store and transport data. It provides a way to describe structured information using a set of rules to define tags and their relationships.

2. XSLT (Extensible Stylesheet Language Transformations):

XSLT is a language used for transforming XML documents into other formats, such as HTML, text, or even other XML documents. It uses **XSLT stylesheets** to define how XML elements should be displayed.

- **Key Components of XSLT**:
  - **<xsl:template>**: Defines how to match and transform XML elements.
  - **<xsl:value-of>**: Extracts the value of an XML element or attribute.
  - **<xsl:for-each>**: Iterates over XML elements.
  - **<xsl:apply-templates>**: Applies templates to child elements.

3. XSLT Process:

- The **XML document** provides the raw data.
- The **XSLT file** defines the rules for transforming that data into a desired output (HTML table).
- The **browser** or an **XSLT processor** applies the XSLT to the XML document, producing the desired output (HTML).

Code:

XML Document (books.xml):

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
  <book>
    <title>Harry Potter and the Sorcerer's Stone</title>
    <author>J.K. Rowling</author>
    <genre>Fantasy</genre>
    <price>19.99</price>
  </book>
  <book>
    <title>The Great Gatsby</title>
    <author>F. Scott Fitzgerald</author>
    <genre>Fiction</genre>
    <price>10.99</price>
  </book>
  <book>
    <title>1984</title>
    <author>George Orwell</author>
    <genre>Dystopian</genre>
    <price>14.99</price>
  </book>
</books>
```

XSLT File (books.xsl):

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- XSLT File to transform XML data into HTML table -->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <!-- Template to match the root element "books" -->
  <xsl:template match="/books">
    <html>
      <head>
        <title>Book List</title>
        <style>
          table {
            width: 100%;
            border-collapse: collapse;
          }
          th, td {
            padding: 8px;
            text-align: left;
            border: 1px solid #ddd;
          }
          th {
            background-color: #f2f2f2;
          }
```

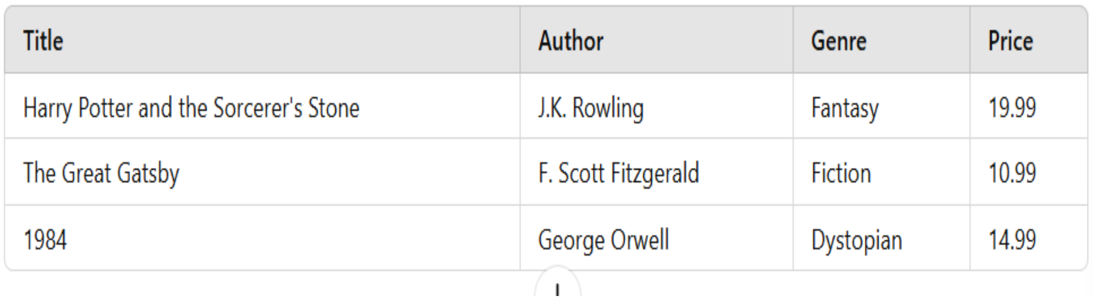
```
            </style>
          </head>
          <body>
            <h1>Book List</h1>
            <table>
              <tr>
                <th>Title</th>
                <th>Author</th>
                <th>Genre</th>
                <th>Price</th>
              </tr>
              <!-- Apply template to each book element -->
              <xsl:apply-templates select="book" />
            </table>
          </body>
        </html>
      </xsl:template>
      <!-- Template to match each "book" element and extract data -->
      <xsl:template match="book">
        <tr>
          <td><xsl:value-of select="title" /></td>
          <td><xsl:value-of select="author" /></td>
          <td><xsl:value-of select="genre" /></td>
          <td><xsl:value-of select="price" /></td>
        </tr>
      </xsl:template>
    </xsl:stylesheet>
```

Linking the XML and XSLT:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<?xml-stylesheet type="text/xsl" href="books.xsl"?>
<books>
  <book>
    <title>Harry Potter and the Sorcerer's Stone</title>
    <author>J.K. Rowling</author>
    <genre>Fantasy</genre>
    <price>19.99</price>
  </book>
  <book>
    <title>The Great Gatsby</title>
    <author>F. Scott Fitzgerald</author>
    <genre>Fiction</genre>
    <price>10.99</price>
  </book>
  <book>
    <title>1984</title>
    <author>George Orwell</author>
    <genre>Dystopian</genre>
    <price>14.99</price>
  </book>
</books>
```

Output:



Title	Author	Genre	Price
Harry Potter and the Sorcerer's Stone	J.K. Rowling	Fantasy	19.99
The Great Gatsby	F. Scott Fitzgerald	Fiction	10.99
1984	George Orwell	Dystopian	14.99

Learning Outcomes:

1. **Understanding XSLT**: Learn how to use **XSLT** to transform an XML document into a different format (HTML).
2. **Separation of Concerns**: Understand how XSLT separates **data** (XML) from its **presentation** (HTML).
3. **Using XSLT Templates**: Learn how to use **XSLT templates** and functions like <xsl:value-of> and <xsl:for-each> to manipulate XML data and present it in an HTML format.
4. **HTML Table Creation**: Learn how to create structured HTML content (like a table) dynamically using XML data and XSLT.

Course Outcomes:

1. Gain the ability to **transform XML** data into readable formats such as HTML using **XSLT**.
2. Understand **XSLT syntax** and how to apply it in real-world scenarios for data presentation.
3. Learn how to integrate XML with HTML for **dynamic web content generation**.
4. Understand the importance of **data separation** in web development (data layer vs. presentation layer).

Conclusion:

In this exercise, you have learned how to use **XSLT** to transform an **XML document** containing book data into an **HTML table**. This demonstrates the power of XSLT in separating data from presentation, making it a versatile tool in web development. By mastering XSLT, you can dynamically present structured data in various formats based on the needs of the application.

Viva Questions:

1. What is **XSLT**, and how does it differ from XML?

2. How does the <xsl:for-each> tag work in XSLT?
3. What is the purpose of the <xsl:value-of> tag in XSLT?
4. How do you link an XML document with an XSLT stylesheet?
5. What would happen if you do not use the xsl:stylesheet element in an XSLT file?
6. Can XSLT be used to convert XML to formats other than HTML? If so, how?
7. How can you apply **conditional logic** in XSLT to filter data?
8. What is the purpose of separating XML data from HTML presentation using XSLT?

For Faculty use:

Correction Parameters	Formative Assessment[40%]	Timely Completion of Practical[40%]	Attendance Learning Attitude[20%]

Experiment 8: Create an Asynchronous Web Application.

Aim:

Build a web page where: A button fetches the current weather information from a server or mock API using an AJAX request & Display the fetched data dynamically without reloading the page.

Tools Used:

- **HTML5:** For structuring the web page.
- **CSS3**: For styling the page.
- **JavaScript (AJAX):** For making asynchronous requests to fetch data from an API and updating the webpage dynamically.

Learning Objectives:

1. Understand the concept of AJAX (Asynchronous JavaScript and XML).
2. Learn how to fetch data from a server using JavaScript.
3. Learn how to manipulate the DOM to dynamically update a web page with the fetched data without reloading the page.
4. Get hands-on experience with XMLHttpRequest or the Fetch API to perform AJAX requests.

Theory:

- **AJAX (Asynchronous JavaScript and XML)** is a technique that allows web pages to make asynchronous requests to servers, fetch data, and update the page content dynamically without reloading the entire page. It uses JavaScript to make requests and fetch data from the server in the background.

- With AJAX, the client-side application can fetch data from APIs or servers without reloading the web page, providing a smoother and faster user experience.

- **XMLHttpRequest** and **Fetch API** are the two main ways to perform AJAX requests in modern web development. The Fetch API is more modern and promise-based, while XMLHttpRequest is an older approach.

- This is typically used in scenarios like fetching weather data, submitting forms, or loading new content into a page without refreshing the browser.

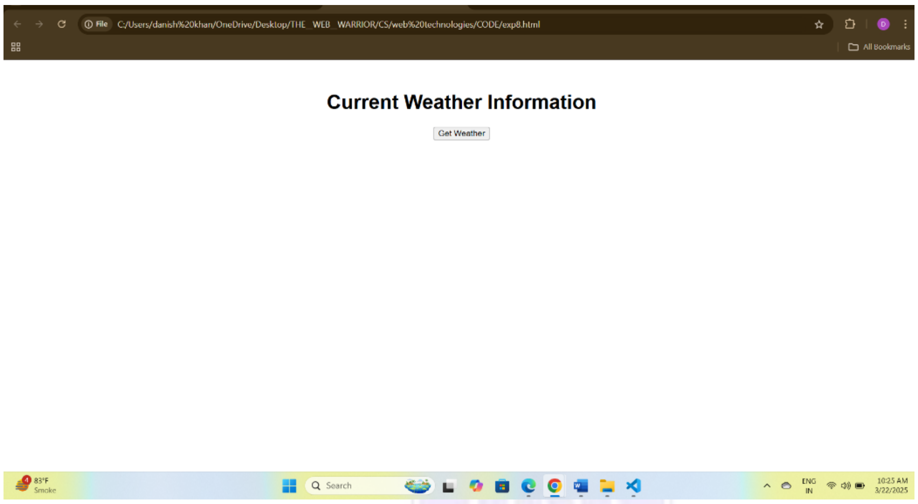
Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Weather Info</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      margin-top: 50px;
    }
    #weather {
      margin-top: 20px;
      padding: 20px;
      background-color: #f2f2f2;
      display: none;
    }
    #error {
      color: red;
    }
  </style>
</head>
<body>
  <h1>Current Weather Information</h1>
  <button id="getWeatherBtn">Get Weather</button>
  <div id="weather">
    <h2>Weather in <span id="city"></span></h2>
```

```
        <p>Temperature: <span id="temp"></span> °C</p>

        <p>Humidity: <span id="humidity"></span>%</p>

        <p>Condition: <span id="condition"></span></p>

    </div>

    <p id="error"></p>

    <script>

        document.getElementById('getWeatherBtn').addEventListener('click', fetchWeather);

        function fetchWeather() {

            const xhr = new XMLHttpRequest();

            const city = 'London';  // You can change this to any city

            // Mock API URL for weather

            const apiURL =
`https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=YOUR_API_KEY&units=metric`;

            xhr.open('GET', apiURL, true);

            xhr.onreadystatechange = function() {

                if (xhr.readyState == 4 && xhr.status == 200) {

                    const data = JSON.parse(xhr.responseText);

                    // Display weather data

                    document.getElementById('city').textContent = data.name;

                    document.getElementById('temp').textContent = data.main.temp;

                    document.getElementById('humidity').textContent = data.main.humidity;

                    document.getElementById('condition').textContent = data.weather[0].description;

                    document.getElementById('weather').style.display = 'block';

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

                } else if (xhr.readyState == 4) {

                    document.getElementById('error').textContent = 'Failed to fetch weather data.';

                    document.getElementById('weather').style.display = 'none';

                }

            };

            xhr.send();
```

```
      }
   </script>
</body>
</html>
```

Output:


C:/Users/danish%20khan/OneDrive/Desktop/THE WEB WARRIOR/CS/web%20technologies/CODE/exp8.html
Current Weather Information
Get Weather

Learning Outcomes:

- Gain a deep understanding of how to send and receive asynchronous requests using AJAX.
- Be able to implement AJAX-based dynamic content fetching and updating.
- Improve proficiency in JavaScript and its interaction with external data sources (APIs).
- Learn how to work with API responses (usually in JSON format) and update HTML dynamically based on the response.

Course Outcomes:

- Students will have a clear understanding of how to use JavaScript for fetching and displaying data dynamically.

- Students will be able to handle asynchronous data requests and integrate them into web applications.
- Students will understand how to use AJAX for building real-time applications such as weather apps, live scores, and more.
- Students will have a basic understanding of API integration and how to manage responses from web services.

Conclusion:

In this task, we demonstrated how to use AJAX to fetch weather information from an external or mock API and display it dynamically on the web page without the need for a full page reload. This technique is fundamental in modern web development, as it allows web pages to function smoothly with real-time data. AJAX is an essential skill for developers who want to build interactive and dynamic web applications. Understanding the Fetch API, how to handle asynchronous requests, and manipulating the DOM is critical for creating engaging user experiences.

Viva Questions:

What is AJAX and how does it work?

What are the advantages of using AJAX in web development?

What is the difference between the XMLHttpRequest and Fetch API?

What type of data format is usually returned by APIs?

For Faculty use:

Correction Parameters	Formative Assessment[40%]	Timely Completion of Practical[40%]	Attendance Learning Attitude[20%]

Experiment 9: Interactive Elements Using AJAX:.

Aim:

Develop a feature where: A dropdown list dynamically updates its options based on another dropdown selection using AJAX.

Tools Used:

- **HTML5:** To structure the web page and create dropdown lists.
- **CSS3:** For basic styling of the page and elements.
- **JavaScript (AJAX):** To make asynchronous requests and update the second dropdown without reloading the page.
- **PHP or a mock API:** For the backend, which provides data to update the second dropdown based on the first dropdown's selection.

Learning Objectives:

1. Understand the concept of AJAX (Asynchronous JavaScript and XML) and how it can be used to fetch data without reloading a page.
2. Learn how to manipulate the DOM (Document Object Model) in JavaScript to dynamically update content based on user input.
3. Explore how to create interactive forms where the content in one dropdown list changes based on the selection in another dropdown.
4. Learn how to structure an API or use a backend service to provide dynamic data to a webpage.
5. Understand client-server interactions using AJAX to fetch and display data in real-time.

Theory:

1. **AJAX (Asynchronous JavaScript and XML)** is a technique used to send and receive data asynchronously from a web server without refreshing the page. It allows for more dynamic and interactive web applications.
2. **Dropdown Lists** (or select boxes) are commonly used in forms to allow the user to choose from a list of options. With AJAX, it’s possible to update these options dynamically based on previous user input.
3. The process works as follows:
   - The first dropdown sends an AJAX request to a server when a user selects an option.
   - The server responds with data (in the form of JSON or XML).

- The second dropdown is populated with the new options based on the data received.

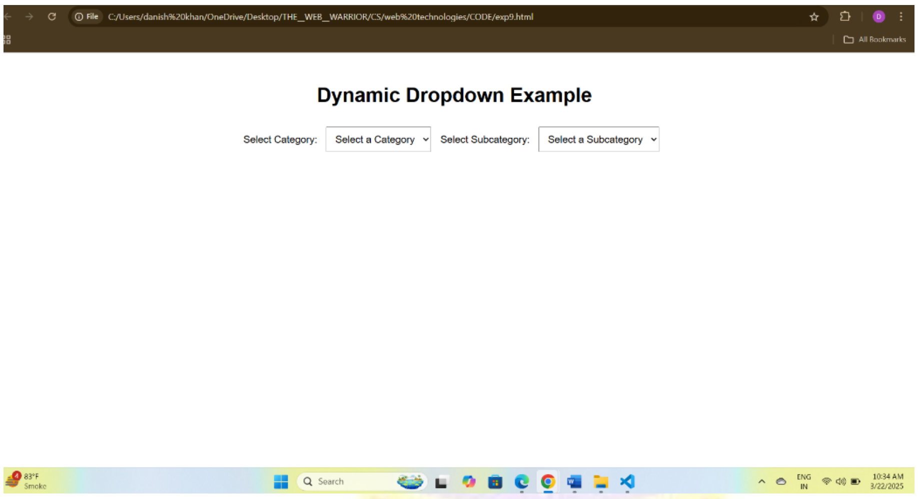
Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Dynamic Dropdown with AJAX</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      text-align: center;

      margin-top: 50px;

    }

    select {

      padding: 10px;

      font-size: 16px;

      margin: 10px;

    }

    #loading {

      display: none;

      font-size: 16px;

      color: #888;

    }

  </style>

</head>

<body>
```

```
<h1>Dynamic Dropdown Example</h1>

<label for="category">Select Category: </label>
<select id="category" onchange="fetchSubcategories()">
  <option value="">Select a Category</option>
  <option value="electronics">Electronics</option>
  <option value="fashion">Fashion</option>
  <option value="books">Books</option>
</select>

<label for="subcategory">Select Subcategory: </label>
<select id="subcategory">
  <option value="">Select a Subcategory</option>
</select>

<p id="loading">Loading...</p>

<script>
  function fetchSubcategories() {
    const category = document.getElementById('category').value;
    const subcategoryDropdown = document.getElementById('subcategory');
    const loadingMessage = document.getElementById('loading');

    if (category) {
      loadingMessage.style.display = 'block';
      subcategoryDropdown.innerHTML = '<option>Loading...</option>';

      const xhr = new XMLHttpRequest();
      xhr.open('GET', `https://api.example.com/subcategories?category=${category}`, true);
      xhr.onload = function () {
        if (xhr.status === 200) {
          const subcategories = JSON.parse(xhr.responseText);
          subcategoryDropdown.innerHTML = '<option value="">Select a Subcategory</option>';

          subcategories.forEach(subcategory => {
            const option = document.createElement('option');
            option.value = subcategory.id;
            option.textContent = subcategory.name;
            subcategoryDropdown.appendChild(option);
          });
        } else {
          subcategoryDropdown.innerHTML = '<option>Error loading subcategories</option>';
        }
        loadingMessage.style.display = 'none';
      };

      xhr.onerror = function () {
        subcategoryDropdown.innerHTML = '<option>Error loading subcategories</option>';
        loadingMessage.style.display = 'none';
      };

      xhr.send();
    } else {
      subcategoryDropdown.innerHTML = '<option value="">Select a Subcategory</option>';
    }
  }
</script>
```

```
</body>

</html>
```

Output:


C:/Users/danish%20khan/OneDrive/Desktop/THE_WEB_WARRIOR/CS/web%20technologies/CODE/exp9.html

Dynamic Dropdown Example

Select Category: Select a Category Select Subcategory: Select a Subcategory

Learning Outcomes:

- Understand the process of making **AJAX requests** to update dropdown lists dynamically based on user selections.
- Be able to implement interactive dropdowns that update based on user input, improving the user experience of forms or web applications.
- Gain familiarity with using **AJAX** to retrieve data from a server and manipulate the webpage content without refreshing the page.
- Learn how to handle **asynchronous data** and update the page in real-time using JavaScript.

Course Outcomes:

1. **Understand AJAX**: Students will understand the fundamentals of making asynchronous requests from a webpage to a server and processing the responses.
2. **Dynamic Forms**: Students will learn how to create dynamic forms where the selection of one dropdown affects the options in another.
3. **API Integration**: Students will know how to structure and query APIs to retrieve dynamic data based on user interaction.

4. **Improved User Interaction**: Students will be able to improve the user experience of web pages by adding dynamic, real-time updates.

.

Conclusion:

This feature demonstrates how to use AJAX to dynamically update a dropdown list based on another dropdown's selection. This type of interactivity is common in modern web applications, especially when dealing with forms, such as selecting a country to populate states or cities, or filtering a product list based on categories. The ability to update options without a page refresh enhances the user experience by providing a smoother and faster interface. AJAX allows web applications to be more responsive and interactive, crucial for building modern web apps.

Viva Questions:

What is AJAX and how does it work?

Explain how you would use AJAX to update a dropdown list based on another dropdown selection.

What is the role of the XMLHttpRequest object or the Fetch API in AJAX?

What data format is typically used when exchanging data between the server and client in AJAX requests?

How can you handle errors in AJAX requests?

What are some use cases for dynamically updating dropdowns with AJAX in web applications?

For Faculty use:

Correction Parameters	Formative Assessment[40%]	Timely Completion of Practical[40%]	Attendance Learning Attitude[20%]

Experiment 10: Build a Simple Web Application with PHP:.

Aim:

Create a web application that:

a) Allows users to fill out a form to submit their name, email, and message.

b) Stores the form data in a database (e.g., MySQL) using PHP.

c) Implements session management to display a personalized greeting for logged-in users.

Tools Used:

1. **HTML/CSS:** For creating and styling the form and page layout.
2. **PHP:** For server-side processing, form handling, and session management.
3. **MySQL:** For storing the form data.
4. **XAMPP/WAMP/MAMP:** Local server setup to run PHP and MySQL.

Learning Objectives:

1. Understand how to create web forms and collect user data.
2. Learn how to use PHP to process and store form data into a MySQL database.
3. Learn how to implement session management for personalized user experiences.
4. Gain knowledge of using SQL queries to insert and retrieve data from a database.
5. Learn the basic concepts of security (e.g., input validation, SQL injection prevention) when handling user data.

Theory:

- **Form Handling**: A form is an essential part of web applications for collecting user input. Using the POST method, the form data is sent to the server for processing.
- **PHP and MySQL**: PHP is used for server-side processing, and MySQL stores the form data in the database. PHP is capable of connecting to MySQL, performing **CRUD** (Create, Read, Update, Delete) operations, and securely handling form inputs.
- **Session Management**: PHP sessions allow you to maintain user-specific data (like login status) across multiple pages. When a user logs in, a session can be created, and data can be retrieved on subsequent pages for personalized content.
- **Security**: To prevent SQL injection and other vulnerabilities, inputs are sanitized and validated using mysqli_real_escape_string() and parameterized queries.

Code:

MySQL Database:

```
CREATE DATABASE contact_form;

USE contact_form;

CREATE TABLE messages (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    message TEXT NOT NULL
);
```

PHP Form Submission and Database Insertion:

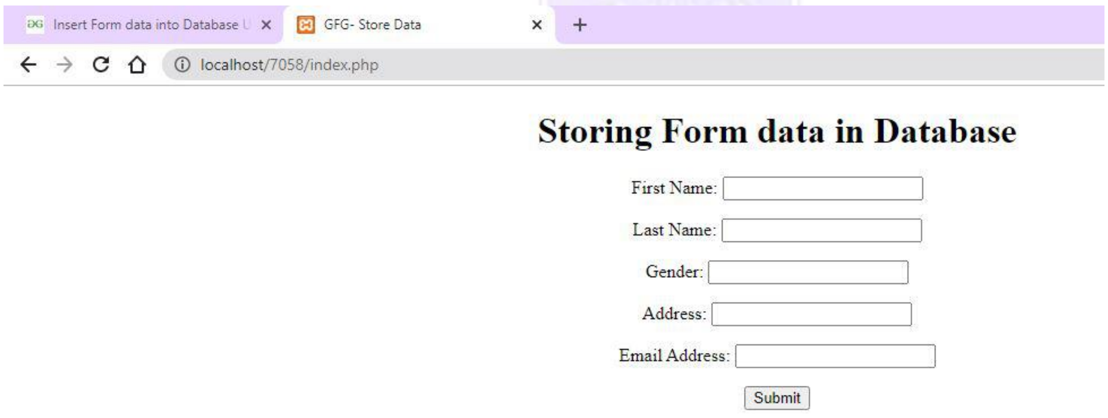
```
<?php
session_start(); // Start the session to manage session variables

// MySQL connection setup
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "contact_form";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
```

```
// Handle form submission
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = mysqli_real_escape_string($conn, $_POST['name']);
    $email = mysqli_real_escape_string($conn, $_POST['email']);
    $message = mysqli_real_escape_string($conn, $_POST['message']);

    // Insert the form data into the database
    $sql = "INSERT INTO messages (name, email, message) VALUES ('$name', '$email', '$message')";

    if ($conn->query($sql) === TRUE) {
        echo "Message sent successfully!";
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }
}

$conn->close();
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Contact Form</title>
</head>
<body>
    <?php
    // Check if the user is logged in
    if (isset($_SESSION['user'])) {
        echo "<h2>Welcome, " . $_SESSION['user'] . "!</h2>";
    }
    ?>

    <h1>Contact Us</h1>
    <form method="POST" action="">
        <label for="name">Name:</label><br>
        <input type="text" name="name" required><br><br>

        <label for="email">Email:</label><br>
        <input type="email" name="email" required><br><br>

        <label for="message">Message:</label><br>
        <textarea name="message" required></textarea><br><br>

        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

PHP Login and Session Handling (for Personalized Greeting):

```
<?php
session_start();

// Simulate user login
if ($_SERVER['REQUEST_METHOD'] == 'POST' && isset($_POST['login'])) {
    $_SESSION['user'] = $_POST['username']; // Store the user's name in the session
}

if ($_SERVER['REQUEST_METHOD'] == 'POST' && isset($_POST['logout'])) {
    session_unset();

    session_destroy();

}


?>


<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Login</title>

</head>

<body>

    <?php

    if (isset($_SESSION['user'])) {

        echo "<h2>Welcome, " . $_SESSION['user'] . "!</h2>";

    } else {

        echo "<h2>Please login</h2>";

    }

    ?>


    <?php if (!isset($_SESSION['user'])): ?>

    <form method="POST" action="">

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

        <input type="text" name="username" required><br><br>


        <input type="submit" name="login" value="Login">

    </form>

    <?php endif; ?>
    <?php if (isset($_SESSION['user'])): ?>
        <form method="POST" action="">
            <input type="submit" name="logout" value="Logout">
        </form>
    <?php endif; ?>
</body>
</html>
```

Output:



Storing Form data in Database

First Name:
Last Name:
Gender:
Address:
Email Address:
Submit

Learning Outcomes:

- Understand how to implement server-side logic using PHP for form submission.
- Learn to connect to a MySQL database and insert user-submitted data.

- Gain experience in managing user sessions to create personalized web applications.
- Understand database interaction (CRUD operations) and how to handle user data securely.
- Learn how to create dynamic web pages with PHP, where the content changes based on user interaction.

Course Outcomes:

1. Understand Web Development: Students will understand the end-to-end process of building a web application, from form creation to database storage and user management.
2. Database Integration: Students will learn how to integrate a MySQL database with a PHP application.
3. Session Management: Students will be able to implement and manage user sessions to deliver personalized experiences.
4. Security in Web Applications: Students will learn how to handle user input securely, preventing vulnerabilities like SQL injection.
5. PHP and MySQL: Students will be able to use PHP and MySQL together to build full-fledged web applications.

Conclusion:

This web application allows users to interact with a form to submit their personal information (name, email, message). The submitted data is securely stored in a MySQL database using PHP. Additionally, session management ensures that logged-in users receive personalized greetings. This application demonstrates the power of PHP and MySQL for server-side web development, along with the importance of security and user interaction in building modern web applications.

Viva Questions:

What is the role of PHP in this web application?

How does session management work in PHP?

What are the best practices for preventing SQL injection in PHP?

Why do we use mysqli_real_escape_string() in PHP?

What is the purpose of the POST method in HTML forms?

How would you implement input validation on the form before storing the data in the database?

For Faculty use:

Correction Parameters	Formative Assessment[40%]	Timely Completion of Practical[40%]	Attendance Learning Attitude[20%]