**LAB ASSIGNMENT (Internet and Web Programming)**

**NAME: AYSHATH AFLA**

**REGISTRATION NO: 21BCY10133**

1. **Analyze the existing IRCTC website and improve the website using HTML**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Train Ticket Booking</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<header>

<h1>Train Ticket Booking</h1>

</header>

<main>

<form action="#" method="post" class="booking-form">

<label for="train\_number">Train Number:</label>

<input type="text" id="train\_number" name="train\_number" required>

<label for="from\_station">From:</label>

<input type="text" id="from\_station" name="from\_station" required>

<label for="to\_station">To:</label>

<input type="text" id="to\_station" name="to\_station" required>

<label for="departure\_date">Departure Date:</label>

<input type="date" id="departure\_date" name="departure\_date" required>

<label for="passenger\_name">Passenger Name:</label>

<input type="text" id="passenger\_name" name="passenger\_name" required>

<label for="seat\_class">Seat Class:</label>

<select id="seat\_class" name="seat\_class" required>

<option value="economy">Economy</option>

<option value="business">Business</option>

<option value="first\_class">First Class</option>

</select>

<input type="submit" value="Book Now">

</form>

</main>

<footer>

<p>&copy; 2023 Train Ticket Booking</p>

</footer>

</body>

</html>

/\* Basic Reset \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

/\* Body styles \*/

body {

font-family: Arial, sans-serif;

line-height: 1.6;

}

/\* Header styles \*/

header {

background-color: #333;

color: #fff;

text-align: center;

padding: 1rem 0;

}

header h1 {

margin: 0;

}

/\* Main styles \*/

main {

padding: 2rem;

}

.booking-form label,

.booking-form input,

.booking-form select {

display: block;

margin-bottom: 1rem;

width: 100%;

}

.booking-form input[type="submit"] {

width: auto;

padding: 0.5rem 1rem;

background-color: #333;

color: #fff;

border: none;

cursor: pointer;

transition: background-color 0.3s ease;

}

.booking-form input[type="submit"]:hover {

background-color: #555;

}

/\* Footer styles \*/

footer {

text-align: center;

background-color: #333;

color: #fff;

padding: 1rem 0;

position: fixed;

bottom: 0;

width: 100%;

}

**2.Design an English alphabet chart such that on clicking the alphabet the appropriate example must be displayed using HTML client-side image mapping**

<!DOCTYPE html>

<html>

<head>

<title>Alphabet Chart</title>

<style>

#alphabet\_chart {

position: relative;

width: 500px;

height: 500px;

}

.example {

position: absolute;

background-color: rgba(255, 255, 255, 0.8);

border: 1px solid black;

padding: 5px;

display: none;

}

</style>

<script>

window.onload = function() {

let alphabet\_chart = document.getElementById('alphabet\_chart');

alphabet\_chart.addEventListener('click', function(event) {

let target = event.target;

if (target.tagName === 'AREA') {

let letter = target.getAttribute('letter');

let example = document.getElementById('example\_' + letter);

if (example.style.display === 'none') {

example.style.display = 'block';

} else {

example.style.display = 'none';

}

}

});

}

</script>

</head>

<body>

<h2>Alphabet Chart</h2>

<div id="alphabet\_chart">

<img src="alphabet\_chart.jpg" alt="Alphabet Chart" usemap="#alphabet\_map">

<map name="alphabet\_map">

<area shape="rect" coords="0,0,50,50" letter="A" href="#A">

<area shape="rect" coords="50,0,100,50" letter="B" href="#B">

<!-- Add other alphabet areas here -->

</map>

<!-- Add examples for each alphabet letter here -->

<div class="example" id="example\_A">A for Apple</div>

<div class="example" id="example\_B">B for Ball</div>

<!-- Add other examples here -->

</div>

</body>

</html>

**3.Design the online periodic table as follows using CSS**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Periodic Table</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<div class="periodic-table">

<div class="element" id="hydrogen">

<span class="symbol">H</span>

<span class="name">Hydrogen</span>

<span class="number">1</span>

</div>

<!-- Add other elements similarly -->

<!-- Example: -->

<!-- <div class="element" id="helium">

<span class="symbol">He</span>

<span class="name">Helium</span>

<span class="number">2</span>

</div> -->

<!-- Continue adding elements for the periodic table -->

</div>

</body>

</html>

/\* Resetting default margin and padding \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

}

.periodic-table {

display: grid;

grid-template-columns: repeat(18, 1fr);

gap: 1px;

max-width: 800px;

border: 1px solid #ccc;

background-color: #fff;

}

.element {

position: relative;

background-color: #fff;

border: 1px solid #ccc;

padding: 10px;

text-align: center;

cursor: pointer;

}

.element:hover {

background-color: #f0f0f0;

}

.symbol {

font-size: 1.5em;

font-weight: bold;

}

.name {

font-size: 1em;

margin-top: 5px;

}

.number {

font-size: 0.8em;

position: absolute;

top: 5px;

right: 5px;

}

**4. Validate the ISBN number of a given book using regular expressions.**

**CODE:**

<!DOCTYPE html>

<html>

<head>

<title>ISBN Validation</title>

<script>

function validateISBN() {

var isbn = document.getElementById('isbn').value;

// Remove any non-digit characters from the input

isbn = isbn.replace(/[^\dX]/gi, '');

if (/^(\d{9}[0-9X])$/.test(isbn)) {

alert('ISBN number is valid.');

} else {

alert('ISBN number is invalid.');

}

}

</script>

</head>

<body>

<h1>ISBN Validation</h1>

<p>Enter the ISBN number of the book:</p>

<input type="text" id="isbn" name="isbn">

<button onclick="validateISBN()">Validate ISBN</button>

</body>

</html>

**5. Develop tic-tac-toe game using JavaScript.**

**CODE:**

<!DOCTYPE html>

<html>

<head>

<title>Tic-Tac-Toe Game</title>

<style>

body {

font-family: Arial, sans-serif;

}

#game-board {

display: grid;

grid-template-columns: repeat(3, 100px);

grid-template-rows: repeat(3, 100px);

grid-gap: 1px;

margin: 20px auto;

}

.cell {

display: flex;

justify-content: center;

align-items: center;

width: 100px;

height: 100px;

border: 1px solid #333;

font-size: 24px;

cursor: pointer;

}

</style>

</head>

<body>

<h1>Tic-Tac-Toe Game</h1>

<div id="game-board"></div>

<button onclick="resetGame()">Reset Game</button>

<script>

const gameBoard = document.getElementById('game-board');

const winningCombinations = [

[0, 1, 2],

[3, 4, 5],

[6, 7, 8],

[0, 3, 6],

[1, 4, 7],

[2, 5, 8],

[0, 4, 8],

[2, 4, 6]

];

let gameState = [];

let currentPlayer = 'X';

function createBoard() {

for (let i = 0; i < 9; i++) {

const cell = document.createElement('div');

cell.classList.add('cell');

cell.setAttribute('id', `cell-${i}`);

cell.setAttribute('onclick', `makeMove(${i})`);

gameBoard.appendChild(cell);

}

}

function makeMove(cellId) {

if (gameState[cellId] || checkWin()) {

return;

}

gameState[cellId] = currentPlayer;

document.getElementById(`cell-${cellId}`).textContent = currentPlayer;

currentPlayer = currentPlayer === 'X' ? 'O' : 'X';

}

function checkWin() {

for (const combination of winningCombinations) {

const [a, b, c] = combination;

if (gameState[a] && gameState[a] === gameState[b] && gameState[a] === gameState[c]) {

alert(`Player ${gameState[a]} wins!`);

return true;

}

}

return false;

}

function resetGame() {

gameState = [];

currentPlayer = 'X';

gameBoard.innerHTML = '';

createBoard();

}

createBoard();

</script>

</body>

</html>

**6. Develop an online application to find the transpose of the given matrix. Obtain the number elements from the user based on the number of rows and columns using JavaScript.**

**CODE:**

<!DOCTYPE html>

<html>

<head>

<title>Transpose of a Matrix</title>

<style>

body {

font-family: Arial, sans-serif;

}

#matrix-input {

display: grid;

grid-template-columns: repeat(auto-fill, 50px);

grid-template-rows: repeat(auto-fill, 50px);

grid-gap: 5px;

margin: 20px auto;

}

.input-cell {

width: 50px;

height: 50px;

padding: 5px;

border: 1px solid #333;

text-align: center;

}

</style>

</head>

<body>

<h1>Transpose of a Matrix</h1>

<form id="matrix-form">

<div id="matrix-input"></div>

<button type="submit">Calculate Transpose</button>

</form>

<p id="transpose-output"></p>

<script>

document.getElementById('matrix-form').addEventListener('submit', function(event) {

event.preventDefault();

const numRows = parseInt(document.getElementById('numRows').value);

const numCols = parseInt(document.getElementById('numCols').value);

const matrixInput = document.getElementById('matrix-input');

matrixInput.innerHTML = '';

for (let i = 0; i < numRows \* numCols; i++) {

const inputCell = document.createElement('input');

inputCell.classList.add('input-cell');

inputCell.setAttribute('id', `input-cell-${i}`);

matrixInput.appendChild(inputCell);

}

const calculateButton = document.createElement('button');

calculateButton.textContent = 'Calculate Transpose';

calculateButton.setAttribute('onclick', 'calculateTranspose()');

matrixInput.appendChild(calculateButton);

});

function calculateTranspose() {

const numRows = parseInt(document.getElementById('numRows').value);

const numCols = parseInt(document.getElementById('numCols').value);

const matrix = [];

for (let i = 0; i < numRows \* numCols; i++) {

matrix.push(parseInt(document.getElementById(`input-cell-${i}`).value));

}

const transpose = [];

for (let i = 0; i < numCols; i++) {

const row = [];

for (let j = 0; j < numRows; j++) {

row.push(matrix[j \* numCols + i]);

}

transpose.push(row);

}

const transposeOutput = document.getElementById('transpose-output');

transposeOutput.innerHTML = 'Transpose:<br>';

for (const row of transpose) {

for (const num of row) {

transposeOutput.innerHTML += `${num} `;

}

transposeOutput.innerHTML += '<br>';

}

}

</script>

</body>

</html>

**7. Create a hospital registration form and validate the fields using JavaScript.**

**CODE:**

<!DOCTYPE html>

<html>

<head>

<title>Hospital Registration Form</title>

<style>

body {

font-family: Arial, sans-serif;

}

form {

width: 300px;

margin: 0 auto;

}

form input {

width: 100%;

padding: 10px;

margin: 10px 0;

}

form button {

padding: 10px;

background-color: #007BFF;

color: white;

cursor: pointer;

}

form button:hover {

background-color: #0056b3;

}

form p {

color: red;

margin-top: 10px;

}

</style>

</head>

<body>

<h1>Hospital Registration Form</h1>

<form id="hospital-form">

<input type="text" id="name" placeholder="Hospital Name" required>

<input type="text" id="address" placeholder="Address" required>

<input type="text" id="city" placeholder="City" required>

<input type="text" id="state" placeholder="State" required>

<input type="text" id="country" placeholder="Country" required>

<input type="text" id="phone" placeholder="Phone Number" required>

<input type="email" id="email" placeholder="Email" required>

<button type="submit">Register</button>

</form>

<p id="form-message"></p>

<script>

document.getElementById('hospital-form').addEventListener('submit', function(event) {

event.preventDefault();

const name = document.getElementById('name').value;

const address = document.getElementById('address').value;

const city = document.getElementById('city').value;

const state = document.getElementById('state').value;

const country = document.getElementById('country').value;

const phone = document.getElementById('phone').value;

const email = document.getElementById('email').value;

if (name && address && city && state && country && phone && email) {

const formMessage = document.getElementById('form-message');

formMessage.textContent = 'Hospital registered successfully!';

formMessage.style.color = 'green';

}

});

</script>

</body>

</html>

**8. Develop a shopping cart application using PHP sessions.**

**CODE:**

<?php

session\_start();

?>

<!DOCTYPE html>

<html>

<head>

<title>Shopping Cart</title>

</head>

<body>

<h1>Shopping Cart</h1>

<form method="post" action="add\_to\_cart.php">

<input type="hidden" name="product\_code" value="P001">

<input type="text" name="quantity" value="1">

<input type="submit" value="Add to Cart">

</form>

<?php

if(isset($\_SESSION["cart"])) {

$total = 0;

echo '<table border="1"><tr><th>Product Code</th><th>Quantity</th><th>Total</th></tr>';

foreach($\_SESSION["cart"] as $product\_code => $quantity) {

$total += $quantity;

echo "<tr><td>$product\_code</td><td>$quantity</td><td>$total</td></tr>";

}

echo '</table>';

}

?>

</body>

</html>

**CODE:**

This is the **add\_to\_cart.php** file that processes the form submission and adds the selected product to the cart.

php

<?php

session\_start();

if(!isset($\_SESSION["cart"])) {

$\_SESSION["cart"] = array();

}

if(isset($\_POST["product\_code"])) {

$product\_code = $\_POST["product\_code"];

$quantity = $\_POST["quantity"];

if(!isset($\_SESSION["cart"][$product\_code])) {

$\_SESSION["cart"][$product\_code] = 0;

}

$\_SESSION["cart"][$product\_code] += $quantity;

header('Location: cart.php');

}

?>

**9. Develop an online quiz application using PHP where the questions and answers are maintained in database.**

**CODE:**

**Connect to the database using PHP.**

<?php

$servername = "localhost";

$username = "username";

$password = "password";

$dbname = "quiz\_db";

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

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

?>

**Retrieve questions and answers from the database.**

<?php

$sql = "SELECT id, question, option1, option2, option3, option4, correct\_answer FROM quiz\_questions";

$result = $conn->query($sql);

?>

**Display the questions and answers to the user.**

<?php

if ($result->num\_rows > 0) {

// output data of each row

while($row = $result->fetch\_assoc()) {

echo "Question: " . $row["question"]. "<br>";

echo "Option 1: " . $row["option1"]. "<br>";

echo "Option 2: " . $row["option2"]. "<br>";

echo "Option 3: " . $row["option3"]. "<br>";

echo "Option 4: " . $row["option4"]. "<br><br>";

}

} else {

echo "0 results";

}

?>

**Validate the user's input.**

<?php

if (isset($\_POST['submit'])) {

$correct = 0;

$count = $result->num\_rows;

for ($i = 1; $i <= $count; $i++) {

$answer = $\_POST['question' . $i];

$correct\_answer = $row['correct\_answer']; // assuming you have stored the correct answer in a variable named $correct\_answer

if ($answer == $correct\_answer) {

$correct++;

}

}

// Calculate the score

$score = ($correct / $count) \* 100;

echo "Your score is: " . $score . "%";

}

?>

**Provide feedback to the user based on their performance.**

<?php

if ($score >= 80) {

echo "Congratulations! You have passed the quiz.";

} else {

echo "Sorry, you have not passed the quiz. Please try again.";

}

?>

1. **Maintain student details using XML and validate their order using Schema. Let the first name and last name of the student be grouped. The registration number is an attribute of student element. The root element is student\_details. Student is a complex element that has department, address and phone number. Apply a restriction on the phone number such that is in the format of XXX-XXXX where x is an integer**

<?php

// Create an XML file with the given structure

$xml = <<<XML

<student\_details>

<student registration\_number="1234">

<first\_name>John</first\_name>

<last\_name>Doe</last\_name>

<department>Computer Science</department>

<address>123 Main Street</address>

<phone\_number>123-4567</phone\_number>

</student>

</student\_details>

XML;

// Create a SimpleXMLElement object from the XML file

$student\_details = new SimpleXMLElement($xml);

// Function to validate student details using Schema

function validateStudentDetails($student\_details) {

// Define the XSD schema for validation

$xsd = <<<XSD

<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="student\_details">

<xs:complexType>

<xs:sequence>

<xs:element name="student" maxOccurs="unbounded">

<xs:complexType>

<xs:sequence>

<xs:element name="first\_name" type="xs:string"/>

<xs:element name="last\_name" type="xs:string"/>

<xs:element name="department" type="xs:string"/>

<xs:element name="address" type="xs:string"/>

<xs:element name="phone\_number" type="xs:string"/>

</xs:sequence>

<xs:attribute name="registration\_number" type="xs:integer" use="required"/>

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

XSD;

// Create a DOMDocument object

$dom\_document = new DOMDocument();

$dom\_document->loadXML($student\_details->asXML());

// Create a DOMDocument object from the XSD schema

$dom\_xsd = new DOMDocument();

$dom\_xsd->loadXML($xsd);

// Create a DOMXSDSchema object

$dom\_xs = new DOMXSDSchema($dom\_document);

// Add the schema to the schema collection

$dom\_xs->schemaValidateSource($dom\_xsd);

// Check if the XML document is valid according to the schema

if ($dom\_xs->schemaValidate()) {

return true;

} else {

return false;

}

}

// Call the function to validate the student details

if (validateStudentDetails($student\_details)) {

echo "The student details are valid.";

} else {

echo "The student details are not valid.";

}

?>

**11.Perform the following using PERL: (a) Create an array with 10 elements (b) Print the highest index of the array (c) Assign beyond the end of the array, to an element at index 20 (d) Save the current highest index in a scalar, and print it. (e) Set the array size to 5 elements (index 4) (f) Print the array (g) Set the array size back to the previous size (using the scalar created in (d)). (h) Print the array**

# a) Create an array with 10 elements

my @array = (1) x 10;

# b) Print the highest index of the array

print "Highest index of the array: $#array\n";

# c) Assign beyond the end of the array, to an element at index 20

$array[20] = 20;

# d) Save the current highest index in a scalar, and print it.

my $current\_highest\_index = $#array;

print "Current highest index: $current\_highest\_index\n";

# e) Set the array size to 5 elements (index 4)

$#array = 4;

# f) Print the array

print "Array after resizing to 5 elements: @array\n";

# g) Set the array size back to the previous size (using the scalar created in (d)).

$#array = $current\_highest\_index;

# h) Print the array

print "Array after resizing back to its original size: @array\n";