Web Programming Technology Lab Content

In Web Programming Technology Front-End Portion is built by using some languages which are discussed below:

- HTML: HTML stands for Hyper Text Markup Language. It is used to design the front-end portion of web pages using a markup language.HTML is a combination of Hypertext and Markup Language. Hypertext defines the link between the web pages. The Markup language is used to define the link between the text documentation within the tag which defines the structure of web pages.
- CSS: Cascading Style Sheet fondly refers to CSS is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page.
- JavaScript: JavaScript is a famous scripting language used to create magic on the sites to make the site interactive to the user. It is used to enhancing the functionality of a website to running cool games and web-based software.

To understand the basic concept of HTML, CSS & JavaScript We have some questions as below:

HTML

Basic Structure of HTML:

<!DOCTYPE html> Tells browser you are using HTML5

<html> Root of an HTML document

<head> Container for metadata

<title>My First Page</title> Page title

</head>

<body> Contains all data rendered by the browser

<p>Hello world</p> Paragraph tag

</body>

</html>

How To add Paragraph in HTML File:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
      // Syntax for adding paragraph using paragraph tag
       <p> Hello Guyss!! Welcome to web technologies world </p>

</body>
</html>
```

How To add Image In HTML file:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
      //Syntax for adding image using img tag.
      // To adjust the size of image Height or width properties are used.
    <img src="sample1.jpeg" width ="500"alt="">
</body>
</html>
```

How To add Heading in HTML File:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
      // HTML headings are defined with H1 to H6 tags
      // H1 defines the most important heading while H6 defines the least
important heading.
    <h1>Welcome Guyss</h1>
    <h2>Welcome Guyss</h2>
    <h3>Welcome Guyss</h3>
    <h4>Welcome Guyss</h4>

</body>
</html>
```

How To add Link Inside HTML File:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    // The HTML anchor tech defines the hyperlink.
    Click to connect to cdac <a href="https://cdac.in/">Click Here</a>
</body>
</html>
```

How To Add Entities In HTML File:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    //Entity names or entity numbers can be used to display reserved HTML
characters
    &copy; C-DAC 2024. All Rights Reserved <br>
    &dollar; 20
</body>
</html>
```

How To Add Emojis In HTML File:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
      //Use Predefined Values Of Emojis
      &#128508;
      &#128514;
</body>
</html>
```

How To Add Audio Files To HTML File:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
   //The HTML <audio> element is used to play an audio file on a web page
    <audio src="audio.mp3" controls autoplay></audio>
</body>
</html>
```

How To Add Video Files To HTML File:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    //The HTML <video> element is used to play an video file on a web page
    <video src="video.mp4" controls autoplay width="1000"></video>
</body>
</html>
```

How To Add Table Inside The HTML File:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
      // The HTML <table> element is used to add table on a web page
      // Each table row is defined by <tr> and </tr> tag
      // Each table cell is defined by <td> and </td> tag
      // Defines a header cell in a table by using <th> and </th>

    <table border="2">
        <tr><th>First Name</th><th>Last Name</th><th>Organization</th></tr>
        <tr><td>Ashish</td><td>Mishra</td><td>UBI</td></tr>
        <tr><td colspan="2">Amit</td><td>Visa</td></tr>
        <tr><td>Anoop</td><td>Pandey</td><td rowspan="3">C-DAC</td></tr>
        <tr><td>Nilesh</td><td>Sharma</td></tr>
        <tr><td>Abhishek</td><td>Raghav</td></tr>
    </table>
</body>
</html>
```

How To Add Lists In HTML File:

```
 <!DOCTYPE html>

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
      //An ordered list start with <ol> tag.
    <ol>
        <b>Fruits:</b>

      //Each list item starts with <li> tags
        <li>Mango</li>
        <li>Apple</li>
        <li>Grape</li>
    </ol>

      //An unordered list start with <ul> tag.
    <ul>
        <b>Vegies:</b>

      //Each list item starts with <li> tags
        <li>Onion</li>
        <li>Carrot</li>
        <li>Spinach</li>
        <li>Coriander</li>
    </ul>
<!---List by using other notations-->
    <ol type="I">
        <b>Fruits:</b>
        <li>Mango</li>
        <li>Apple</li>
        <li>Grape</li>
    </ol>

    <ul type="square">
        <b>Vegies:</b>
        <li>Onion</li>
        <li>Carrot</li>
        <li>Spinach</li>
        <li>Coriander</li>
    </ul>
</body>
</html>
```

ASSIGNMENT:

Que: 1 How To Create Form In HTML File:

```
!DOCTYPE html>

<html>
  <body>
    <h2>Text input fields</h2>
    <form> //To create form
//The label Tech defines a label for several elements.
      <label for="fname">First name:</label>
      <br/>
// The <input> tag specifies an input field where the user can enter data
      <input
        type="text"
        id="fname"
        name="fname"
        placeholder="Enter first name"
      /><br />
      <label for="lname">Last name:</label><br />
      <input
        type="text"
        id="lname"
        name="lname"
        placeholder="Enter last name"
      /><br />
      <input type="submit" />
    </form>
  </body>
</html>
```

BOOTSTRAP ASSIGNMENT:

```
<!doctype html>
<html lang="en">

<head>
  <title>Assignment1</title>
  <!-- Required meta tags for proper rendering and touch zooming -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-
to-fit=no">

  <!-- Bootstrap CSS v5.2.1 -->
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet"
    integrity="sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
crossorigin="anonymous">
</head>

<body class="bg-secondary">

  <!-- Navigation bar -->
  <nav class="navbar navbar-expand-sm navbar-light bg-light">
    <div class="container">
      <!-- Brand logo -->
      <a class="navbar-brand" href="#"><img src="Logo(1)2.png" alt="Logo"></a>
      <!-- Toggler/collapsible Button -->
      <button class="navbar-toggler d-lg-none" type="button" data-bs-
toggle="collapse" data-bs-target="#collapsibleNavId" aria-
controls="collapsibleNavId"
        aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <!-- Navbar links -->
      <div class="collapse navbar-collapse" id="collapsibleNavId">
        <ul class="navbar-nav me-auto mt-2 mt-lg-0">
          <li class="nav-item">
            <a class="nav-link active" href="#" aria-current="page">About Us
<span class="visually-hidden">(current)</span></a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">Our Courses</a>
          </li>
          <li class="nav-item">
            <a class="nav-link link-underline-primary active" href="#">Free
Resources</a>
```

```
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">Online Community</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">IELTS</a>
          </li>
        </ul>

        <!-- Register button -->
        <button class="btn btn-primary mx-4 border border rounded-5"
type="submit">Register</button>
      </div>
    </div>
  </nav>

  <div class="container">
    <!-- Header section with headline, sub-headline, and image -->
    <div class="row bg-light my-3">
      <div class="col-md-8">
        <h1>ESL Blogs - Headline</h1>
        <p>Sub Head Line</p>
      </div>
      <div class="col-md-4">
        <img class="mx-auto" width="368" height="250" src="182.png" alt="Blog
Image">
      </div>
    </div>

    <!-- Section title -->
    <h5>Most Popular Blogs</h5>

    <!-- Blog cards section -->
    <div class="row row-cols-1 row-cols-md-3 g-4">
      <!-- Blog card 1 -->
      <div class="col">
        <div class="card rounded-4">
          <img class="card-img-top" src="grammar.png" alt="Grammar Image">
          <div class="card-body">
            <h4 class="card-title">How English Grammar Can Help You Everyday
Life!</h4>
            <p class="card-text">GRAMMAR</p>
          </div>
        </div>
      </div>
      <!-- Blog card 2 -->
      <div class="col">
        <div class="card rounded-4">
```

```
          <img class="card-img-top" src="grammar(3).png" alt="Common Mistakes
Image">
          <div class="card-body">
            <h4 class="card-title">Avoiding Common Mistakes in English!</h4>
            <p class="card-text">GRAMMAR</p>
          </div>
        </div>
      </div>
      <!-- Blog card 3 -->
      <div class="col">
        <div class="card rounded-4">
          <img class="card-img-top" src="grammar(2).png" alt="Verb Tenses
Image">
          <div class="card-body">
            <h4 class="card-title">The Basics of Verb Tenses in an Informative
Guide!</h4>
            <p class="card-text">VOCABULARY</p>
          </div>
        </div>
      </div>
      <!-- Blog card 4 -->
      <div class="col">
        <div class="card rounded-4">
          <img class="card-img-top" src="vocabulary.png" alt="Vocabulary
Image">
          <div class="card-body">
            <h4 class="card-title">Top 7 Things You Can Do Today To Get An
Amazing Vocabulary</h4>
            <p class="card-text">VOCABULARY</p>
          </div>
        </div>
      </div>
      <!-- Blog card 5 -->
      <div class="col">
        <div class="card rounded-4">
          <img class="card-img-top" src="spokenenglish.png" alt="Spoken
English Image">
          <div class="card-body">
            <h4 class="card-title">Importance of Spoken Professional
English!</h4>
            <p class="card-text">SPOKEN ENGLISH</p>
          </div>
        </div>
      </div>
      <!-- Blog card 6 -->
      <div class="col">
        <div class="card rounded-4">
```

```
          <img class="card-img-top" src="pronuciation.png" alt="Pronunciation
Image">
          <div class="card-body">
            <h4 class="card-title">How to Avoid Annoying Pronunciation Errors
in English!</h4>
            <p class="card-text">PRONUNCIATION</p>
          </div>
        </div>
      </div>
    </div>

    <!-- Promotional section for level test -->
    <div class="row bg-light my-3 rounded-4">
      <div class="col-md-4">
        <img class="mx-auto" width="300" height="250"
src="Taketheleveltest.png" alt="Level Test Image">
      </div>
      <div class="col-md-8 my-4 mx-auto">
        <h3>Take the level test and understand your English knowledge</h3>
        <p>Takes just 10 minutes and we will share a customized report</p>
        <button class="btn btn-outline-primary border rounded-4"
type="submit">Take Test Now</button>
      </div>
    </div>
  </div>

  <!-- Bootstrap JS -->
  <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js
"
    integrity="sha384-
I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r"
crossorigin="anonymous">
  </script>

  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.min.js"
    integrity="sha384-
BBtl+eGJRgqQAUMxJ7pMwbEyER4l1g+015P+16Ep7Q9Q+zqX6gSbd85u4mG4QzX+"
crossorigin="anonymous">
  </script>
</body>

</html>
```

ASSIGNMENT 2:

Que2:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Assignmet-2</title>
    <style>
        body {
            margin: 0;
            font-family: Arial, sans-serif;
            display: flex;
            flex-direction: column;
            height: 100vh;
        }
        header {
            background-color: #f8f8f8;
            padding: 20px;
            display: flex;
            justify-content: center;
            align-items: center;
            box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
        }
        form {
            display: flex;
            justify-content: center;
            align-items: center;
        }
        input[type="text"] {
            padding: 10px;
            margin-right: 10px;
            width: 300px;
            border: 1px solid #ccc;
            border-radius: 4px;
        }
        button {
            padding: 10px 20px;
            border: none;
            background-color: #28a745;
            color: white;
            border-radius: 4px;
            cursor: pointer;
        }
        button:hover {
            background-color: #218838;
        }
        #skills-container {
            flex: 1;
            padding: 20px;
            display: flex;
            flex-wrap: wrap; /* Allows items to wrap to the next line */
            align-items: flex-start; /* Aligns items at the top */
            overflow-y: auto;
            background-color: #e8f4fc; /* Highlighting background color */
            border: 2px solid #bb1313; /* Highlighting border color */
            border-radius: 8px;
            margin: 20px;
        }
        .badge {
            background-color: #333;
            color: white;
            padding: 10px 20px; /* Adjust padding for better visibility */
            border-radius: 20px;
            display: inline-block;
            margin: 5px;
            font-size: 14px;
            position: relative;
        }
        .badge a {
            color: rgb(207, 19, 19);
            text-decoration: none;
            margin-left: 10px;
            margin-right: -5px; /* Slight adjustment for better alignment */
            position: absolute;
            right: 10px;
            top: 50%;
            transform: translateY(-50%);
        }
        .badge a:hover {
            color: red;
        }
    </style>
</head>
<body>
    <header>
        <form id="skill-form">
            <h2>Skills</h2>
            <input type="text" id="skill-input" placeholder="Enter a skill"
required>
            <button type="submit">Add Skill</button>
        </form>
    </header>
    <section id="skills-container">
```

```
        <!-- Skills will be appended here -->
    </section>

    <script>
        document.addEventListener('DOMContentLoaded', () => {
            const skillForm = document.getElementById('skill-form');
            const skillInput = document.getElementById('skill-input');
            const skillsContainer = document.getElementById('skills-
container');

            skillForm.addEventListener('submit', (e) => {
                e.preventDefault();
                addSkill(skillInput.value);
                skillInput.value = '';
            });

            function addSkill(skill) {
                const span = document.createElement('span');
                span.className = 'badge';
                span.textContent = skill;

                const deleteBtn = document.createElement('a');
                deleteBtn.href = '#';
                deleteBtn.innerHTML = '✕';
                deleteBtn.addEventListener('click', (e) => {
                    e.preventDefault();
                    skillsContainer.removeChild(span);
                });

                span.appendChild(deleteBtn);
                skillsContainer.appendChild(span);
            }
        });
    </script>
</body>
</html>
```

ASSIGNMENT 3: NODE JS

Q1:Write a JS program to create array of string and perform the following:

1. Object DE structuring
2. Iteration
3. Define an arrow function to print an element
4. Demonstrate the spread operator

```
//Object Distruction
const object=["Mike","Joe","Tyson","Jane","Mary","James","Dom"];
for(const a of object)
{
    console.log(a);
}
console.log("==========================");

//Itration
const a="Hello Guyss";
for(const b of a)
{
     console.log(b);
}
console.log("==========================");

//Arrow Function

let day = object =>
{
    return object
}
console.log(day(object));
console.log("==========================");


//Spread Operator

console.log(...object);
```

Q2: Write a program to compute Fibonacci series, prime numbers, square of a number and to check whether a string is palindrome or not

```
//Fibonacci series:-
function fibonacci(n) {
  let a=0,b=1,temp;
  while(n>0)
  {
    temp=a+b;
    a=b;
    b=temp;
    n--;
    console.log(a);
  }
  return a;
}
console.log( "fibonacci(10): "+fibonacci(10));
console.log("===========================");

//Square of a number:-
function square(num) {
  return num * num;
}
console.log(square(5));
console.log("===========================");

//Prime number:-
function isPrime(num) {
  if (num <= 1) {
    return false;
  }
  for (let i = 2; i <= Math.sqrt(num); i++) {
    if (num % i === 0) {
      return false;
    }
  }
  return true;
}
console.log(isPrime(7));
console.log("===========================");
```

```javascript
//table of a number
function table(num) {
  for (let i = 1; i <= 10; i++) {
    console.log(`${num} x ${i} = ${num * i}`);
  }
}
table(5);
console.log("===========================");


//palindrome number
function isPalindrome(num) {
  const str = num.toString();
  for (let i = 0, j = str.length - 1; i < j; i++, j--) {
    if (str[i] !== str[j]) {
      return false;
    }
  }
  return true;
}
console.log(isPalindrome(121));
```

Que3:Demonstrate with a JS program the following methods:

1. includes
2. startsWith
3. endsWith
4. entries
5. form
6. find
7. findIndex
8. Math.trunc

```
//includes()
let text = "Hello Guys!!! Welcome into the coding world";
console.log(text.includes("Welcome"));
console.log("-----------------------------");

//Startswith()
console.log(text.startsWith("Hello"));
console.log("------------------------------");

//Endswith()
console.log(text.endsWith("world"));
console.log("-----------------------------");

//entries()

let names =["Ritesh","Harsh","Abhijeet","Saurabh","Rajnish"];
const n= names.entries();
for(let x of n)
{
    console.log(x);
}

console.log("------------------------------");

//form()
console.log(Array.from("RITESH"));
console.log("-------------------------------");

//find()
const xyz =[10,20,30,40,50];
// console.log(xyz.find(a  =>  a>20));
let x1= xyz.find(myFunction);

function myFunction(value)
{
    return value>20;
}
console.log(x1);
console.log("-------------------------------");

//findIndex()

 const Numbers=[2,4,6,8,10];
```

```
// console.log(Numbers.findIndex(a => a>6));
 let index=Numbers.findIndex(myFunction);
function myFunction(value)
{
    return value>6;
}
console.log(index);
// console.log("---------------------------------");

//Math Functions
console.log(Math.trunc(5.6));
console.log("---------------------------------");
console.log(Math.sign(-4));
console.log("---------------------------------");
console.log(Math.cbrt(64));
console.log("---------------------------------");
console.log(Math.log2(100));
console.log("---------------------------------");
console.log(Math.log10(25));
console.log("---------------------------------");
```

ASSIGNMENT 4: (DAY2)

Que1: Use random function to generate random number and divide a number by this random number. If the random number is =1 throw the error message saying Random number is 1, division not allowed. Otherwise perform the division. Create a Promise to handle the asynchronous operation.

```javascript
// Function that returns a Promise which divides the given number by a random
number
function divideByRandomNumber(num) {
    // Return a new Promise
    return new Promise((resolve, reject) => {
        // Generate a random number between 1 and 11 (exclusive of 11)
        const randomNum = (Math.random() * 10) + 1;
        console.log(randomNum); // To display the random number

        // Check if the random number is exactly 1
        if (randomNum === 1) {
            // If random number is 1, reject the Promise with an error message
            reject("Random number is 1, division not allowed....");
        } else {
            // Otherwise, calculate the result of the division
            const result = num / randomNum;
            // Resolve the Promise with the result
            resolve(result);
        }
    });
}

// Call the function and handle the Promise
divideByRandomNumber(10)
    .then(result => console.log(`Result is ${result}`)) // If resolved, log
the result
    .catch(error => console.log(error)); // If rejected, log the error
```

Que2:Create a file and perform the following operation:

1. Write to file
2. Append to file
3. Overwrite to file
4. Rename the file
5. Delete the file

```
// Import the 'fs' (File System) module
var fs = require('fs');

// Create a file
fs.open('mynewfile1.txt', 'w', function(err, file) {
    if (err) throw err; // If an error occurs, throw the error
    console.log('File is Created!'); // Log message indicating the file has
been created
});

// Write inside file
fs.writeFile('mynewfile1.txt', 'Hello content!', function(err) {
    if (err) throw err; // If an error occurs, throw the error
    console.log('Saved!'); // Log message indicating the content has been
saved
});

// Update the file by appending content to it
fs.appendFile('mynewfile1.txt', 'Welcome to the coding world!', function(err)
{
    if (err) throw err; // If an error occurs, throw the error
    console.log('Updated!'); // Log message indicating the file has been
updated with appended content
});

// Override the file content with new content
fs.writeFile('mynewfile1.txt', 'This is updated content!', function(err) {
    if (err) throw err; // If an error occurs, throw the error
    console.log('Updated!'); // Log message indicating the file content has
been overridden with new content
});

// Rename the file
fs.rename('mynewfile1.txt', 'mynewfile2.txt', function(err) {
    if (err) throw err; // If an error occurs, throw the error
    console.log('Renamed!'); // Log message indicating the file has been
renamed
});

// Delete the file
fs.unlink('mynewfile2.txt', function(err) {
    if (err) throw err; // If an error occurs, throw the error
    console.log('Deleted!'); // Log message indicating the file has been
deleted
});
```

Que3:Write a Node.js program to demonstrate the following:

1. setTimeout
2. setInterval
3. setImmediate

```
// Demonstrating setTimeout
console.log("start"); // Log "start" to the console

setTimeout(() => {
    console.log("inside timeout"); // Log "inside timeout" after 2000
milliseconds (2 seconds)
}, 2000);

console.log("end"); // Log "end" to the console immediately
console.log("-----------------------------"); // Log a separator for clarity

// Demonstrating setInterval
console.log("started"); // Log "started" to the console

setInterval(() => {
    console.log("inside interval"); // Log "inside interval" every 1000
milliseconds (1 second)
}, 1000);

console.log("ended"); // Log "ended" to the console immediately
console.log("-----------------------------"); // Log a separator for clarity

// Demonstrating setImmediate
console.log("Starting"); // Log "Starting" to the console

setImmediate((A) => {
    console.log(`Immediate: ${A}`); // Log "Immediate: 1000" immediately after
I/O events
}, 1000);
```

```
console.log("ending"); // Log "ending" to the console immediately
console.log("-----------------------------"); // Log a separator for clarity
```

ASSIGNMENT 5: (DAY3)

Que1:Using express, demonstrate get(), post(), put(), delete() in Node js

```
// Import the express module
const express = require('express');

// Create an instance of an Express application
const app = express();

// Import the 'path' and 'body-parser' modules
var path = require('path');
var bodyparser = require('body-parser');

// Middleware to parse incoming request bodies in a middleware before your
handlers, available under the req.body property
app.use(bodyparser.json());

// Route to handle GET requests to the root URL
app.get('/', (req, res) => {
    res.send('Hello!!'); // Send a response "Hello!!" when the root URL is
accessed
});

// Route to handle POST requests to the /post URL
app.post('/post', (req, res) => {
    console.log('Post Request'); // Log "Post Request" to the console
    console.log(req.body); // Log the request body to the console
    res.send('Data is Posted'); // Send a response "Data is Posted"
});

// Route to handle PUT requests to the /put URL
app.put('/put', (req, res) => {
    console.log('Put Request'); // Log "Put Request" to the console
    console.log(req.body); // Log the request body to the console
    res.send('Data is updated'); // Send a response "Data is updated"
});

// Route to handle DELETE requests to the /delete URL
app.delete('/delete', (req, res) => {
    console.log('Post Deleted'); // Log "Post Deleted" to the console
    console.log(req.body); // Log the request body to the console
    res.send('Data is Deleted'); // Send a response "Data is Deleted"
});

// Define the port to listen on, defaulting to 3000 if process.env.port is not
set
const port = process.env.port || 3000;

// Start the server and listen on the defined port
app.listen(port, () => {
    console.log(`Server is running on port ${port}`); // Log a message when
the server starts
});
```

Que2: Create a simple HTML page and then build the same application in node.js

HTML FILE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Demo</title>
</head>
<body>
    <h1>Welcome to JavaScript</h1>
    <p>Hello Guyss we are here to learn node js </p>
</body>
</html>
```

JS FILE:

```
// Import the express module
var express = require('express');

// Create an instance of an Express application
var app = express();

// Import the 'path' module
var path = require('path');
```

```
// Route to handle GET requests to the root URL
app.get('/', function(req, res) {
    // Send the index.html file as a response
    res.sendFile(path.join(__dirname + '/index.html'));
});

// Start the server and listen on port 5000
var server = app.listen(5000, function() {
    // Log a message when the server starts
    console.log('Express app running at http://127.0.0.1:5000/');
});
```

Que3: Create a TO-DO List using express, body-parser in node.js

```
// Import required modules
const express = require('express');
const bodyParser = require('body-parser');

// Create an instance of Express application
const app = express();

// Define port number
const port = 3000;

// Middleware to parse incoming request bodies as JSON
app.use(bodyParser.json());

// In-memory database (for demonstration purposes)
let todos = [];
let id = 1;

// Route to create a new to-do item
app.post('/todos', (req, res) => {
    // Create a new to-do item with auto-incremented ID, title from request
body, and default completed status
    const todo = {
        id: id++,
        title: req.body.title,
        completed: true  // For demonstration, setting default completion
status to true
    };

    todos.push(todo); // Add the new to-do item to the in-memory database
    res.status(201).json(todo); // Respond with status 201 (Created) and JSON
representation of the new to-do item
});
```

```
// Route to get all to-do items
app.get('/todos', (req, res) => {
    res.json(todos); // Respond with JSON array of all to-do items
});

// Route to get a specific to-do item by ID
app.get('/todos/:id', (req, res) => {
    // Find the to-do item with the specified ID in the in-memory database
    const todo = todos.find(t => t.id === parseInt(req.params.id));

    // If no matching to-do item is found, respond with status 404 (Not Found)
    if (!todo) return res.status(404).send('To-do item not found');

    res.json(todo); // Respond with JSON representation of the found to-do
item
});

// Route to update a specific to-do item by ID
app.put('/todos/:id', (req, res) => {
    // Find the to-do item with the specified ID in the in-memory database
    const todo = todos.find(t => t.id === parseInt(req.params.id));

    // If no matching to-do item is found, respond with status 404 (Not Found)
    if (!todo) return res.status(404).send('To-do item not found');

    // Update the title and completion status of the found to-do item based on
request body
    todo.title = req.body.title;
    todo.completed = req.body.completed;

    res.json(todo); // Respond with JSON representation of the updated to-do
item
});

// Route to delete a specific to-do item by ID
app.delete('/todos/:id', (req, res) => {
    // Find the index of the to-do item with the specified ID in the in-memory
database
    const todoIndex = todos.findIndex(t => t.id === parseInt(req.params.id));

    // If no matching to-do item index is found, respond with status 404 (Not
Found)
    if (todoIndex === -1) return res.status(404).send('To-do item not found');

    // Remove the found to-do item from the in-memory database and capture the
deleted item
    const deletedTodo = todos.splice(todoIndex, 1);
    res.json(deletedTodo[0]); // Respond with JSON representation of the
deleted to-do item
});

// Start the server and listen on the defined port
app.listen(port, () => {
    console.log(`To-do app listening at http://localhost:${port}`);
});
```

ASSIGNMENT 6: (DAY4)

Que:1 Print Hello World in app.js

Que2: Function and Class Component with simple Hello Message.

Que3:Function and Class Component with Hello with properties(props.name, this,props,name)

Que4:Function and lass component with state variables: useSate(0), this.state{count:0}

Que5:Alert message onClick(Clicked by and Clicked)

Que6:Simple forms using class and function component

Que7:Use existing HTML code and render it with react

Greeting.js

```
import React from 'react';

// Define a functional component named Greeting
export default function Greeting() {
  return (
    // Return JSX content: a heading element displaying a greeting message
    <h1>Hello From function</h1>
  );
}
```

Message.js

```
import React, { Component } from 'react';

// Define a class-based component named Message that extends Component
export default class Message extends Component {
  // Render method required by React for class-based components
  render() {
    return (
      // Return JSX content: a heading element displaying a message
      <h1>Hello From Class</h1>
    );
  }
}
```

Demo.js

```
import React from 'react';

// Define a functional component named Demo that accepts props as an argument
export default function Demo(props) {
  // Return JSX content: a heading element displaying a personalized greeting
  return <h1>Hello, {props.name}!</h1>;
}
```

Trail.js

```
import React, { Component } from 'react';

// Define a class-based component named Trial that extends Component
export default class Trial extends Component {
  // Render method required by React for class-based components
  render() {
    // Return JSX content: a heading element displaying a personalized
greeting
    return <h1>Hello, {this.props.name}!</h1>;
  }
}
```

Counter.js

```
import React, { useState } from 'react';

// Define a functional component named Counter
export default function Counter() {
  // Define state variables using the useState hook
  const [count, setCount] = useState(0);

  // Function to increment the count state
  const incrementCount = () => {
    setCount(count + 1); // Update count state by incrementing it by 1
  };

  // Function to decrement the count state, with a check to prevent count from
going below 0
  const decrementCount = () => {
    if (count === 0) return; // Prevent decrementing below 0
    setCount(count - 1); // Update count state by decrementing it by 1
  };

  // JSX rendering
  return (
    <div>
      <h1>Count: {count}</h1> {/* Display the current value of count */}
      <button onClick={incrementCount}>Increment</button> {/* Button to
increment count */}
      <button onClick={decrementCount}>Decrement</button> {/* Button to
decrement count */}
    </div>
  );
}
```

ClickEvent.js

```
import React from 'react';

// Define a functional component named ClickEvent that accepts props as an
argument
export default function ClickEvent(props) {
    // Define a function named click that shows an alert when the button is
clicked
    const click = () => {
        alert("Clicked By " + props.name); // Display an alert with a message
including the value of props.name
    }

    // JSX rendering
    return (
        <button onClick={click}>Click the button</button> // Button element
triggering the click function when clicked
    );
}
```

Form.js

```
import React from "react";

// Define a functional component named Form
export default function Form() {
  // JSX rendering
  return (
    <div>
      <form>
        {/* Input field for entering first name */}
        <label>
          Enter Your FirstName:
          <input type="text" />
        </label>
        <br></br>

        {/* Input field for entering last name */}
        <label>
          Enter Your LastName:
          <input type="text" />
        </label>
        <br></br>

        {/* Input field for entering date of birth */}
        <label>
```

```
          Enter Your Dob:
          <input type="date" />
        </label>
        <br></br>

        {/* Input field for entering education */}
        <label>
          Enter Your Education:
          <input type="text" />
        </label>
        <br></br>

        {/* Submit button for submitting the form */}
        <input type="submit" />
      </form>
    </div>
  );
}
```

FormClass.js

```
import React, { Component } from 'react';

// Define a class component named FormClass
export default class FormClass extends Component {
    constructor(props) {
        super(props);
        // Initialize state with name and email fields
        this.state = {
            name: "",
            email: ""
        };
    }

    render() {
        return (
            <div>
                <form>
                    {/* Input field for entering name */}
                    <label>Name</label>
                    <input
                        type="text"
                        value={this.state.name}
                        onChange={(e) => this.setState({ name:
e.target.value })}
                    />
                    {/* Input field for entering email */}
                    <label>Email</label>
                    <input
                        type="text"
                        value={this.state.email}
                        onChange={(e) => this.setState({ email:
e.target.value })}
                    />

                    {/* Submit button */}
                    <button>Submit</button>
                </form>

                {/* Displaying current state values */}
                <p>Name: {this.state.name}</p>
                <p>Email: {this.state.email}</p>
            </div>
        );
    }
}
```

HTML.js

```
import React from "react";

// Define a functional component named Html
export default function Html() {
  // JSX rendering
  return (
    <div>
      {/* Heading */}
      <h1>HTML</h1>

      {/* Paragraph */}
      <p>This is a login page</p>

      {/* Form element */}
      <form>
        {/* Email input field */}
        Email:
        <input type="email" name="email" />
        <br />

        {/* Password input field */}
        Password:
        <input type="password" name="Password" />
        <br />
        {/* Submit button */}
        <input type="submit" value="Submit" />
      </form>
    </div>
  );
}
```

App.js

```
// Importing CSS file for styling
import './App.css';

// Importing various components from their respective files
import ClickEvent from './components/ClickEvent';
import Counter from './components/Counter';
import Demo from './components/Demo';
import Form from './components/Form';
import FormClass from './components/FormClass';
import Greeting from './components/Greeting';
import Html from './components/Html';
import Message from './components/Message';
import Trial from './components/Trial';

// Define the main App component
function App() {
  return (
    <div className="App"> {/* Main container div with 'App' class */}
      { /* JSX Comment: Hello World!!! */ }
      <p>Hello World!!!</p>

      {/* Rendering various imported components */}
      <Greeting/> {/* Renders the Greeting component */}
      <Message/> {/* Renders the Message component */}

      {/* Passing props to Demo and Trial components */}
      <Demo name="Allen"/> {/* Renders Demo component with name prop set to
"Allen" */}
      <Trial name="Jane"/> {/* Renders Trial component with name prop set to
"Jane" */}

      <Counter/> {/* Renders the Counter component */}

      <ClickEvent name="Ritesh"/> {/* Renders ClickEvent component with name
prop set to "Ritesh" */}

      <Form/> {/* Renders the Form component */}
      <FormClass/> {/* Renders the FormClass component */}

      <Html/> {/* Renders the Html component */}
    </div>
  );
}

export default App; // Exporting the App component as the default export
```

ASSIGNMENT 7: (DAY5)

Que1:Develop a web application using Node as backend and react as frontend. Create a simple application form. Connect to Db and store the entered data

Backend/Server.js

```
// Importing required dependencies
const express = require('express'); // Express framework for Node.js
const bodyParser = require('body-parser'); // Middleware to parse JSON request
bodies
const mysql = require('mysql'); // MySQL client for Node.js
const cors = require('cors'); // Middleware for handling Cross-Origin Resource
Sharing

// Instantiating Express app and defining PORT number
const app = express();
const PORT = 3000;

// Using cors middleware to enable CORS for all routes
app.use(cors());

// Using bodyParser middleware to parse JSON bodies of incoming requests
app.use(bodyParser.json());

// Database connection configuration
const dB = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: 'Venom@2001',
    database: 'form_app'
});

// Connecting to the database
dB.connect((err) => {
    if (err) throw err;
    console.log("Database Connected!");
});

// Route to handle POST requests for registration
app.post('/register', (req, res) => {
    // Destructuring request body to extract form data
    const { name, email, gender, number, address, city, state, pincode } =
req.body;
    console.log(req.body); // Logging the received form data

    // SQL query to insert data into 'user' table
    const sql = "INSERT INTO form_app.user (name, email, gender, number,
address, city, state, pincode) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";

    // Executing the query with data from the request body
    dB.query(sql, [name, email, gender, number, address, city, state,
pincode], (err) => {
        if (err) {
            console.log(err); // Logging any database error
            res.status(500).send("Error Occurred"); // Sending 500 status and
error message on failure
        } else {
            res.status(200).send("Values Inserted"); // Sending 200 status and
success message on successful insertion
        }
    });
});

// Start server and listen on defined PORT
app.listen(PORT, () => {
    console.log(`Server is running on port ${PORT}`); // Logging server
startup message with port number
});
```

Frontend/src/Application_form

```
import React, { useState } from "react";
import axios from "axios"; // Importing axios for making HTTP requests

function ApplicationForm() {
  // State variables to hold form data
  const [name, setName] = useState("");
  const [email, setEmail] = useState("");
  const [gender, setGender] = useState("");
  const [number, setNumber] = useState("");
  const [address, setAddress] = useState("");
  const [city, setCity] = useState("");
  const [state, setState] = useState("");
  const [pincode, setPincode] = useState("");

  // Event handler for form submission
  const handleSubmit = async (e) => {
    e.preventDefault(); // Prevent default form submission

    try {
      // Sending POST request to the server
      await axios.post("http://localhost:3000/register", {
        name: name,
        email: email,
        gender: gender,
        number: number,
        address: address,
        city: city,
        state: state,
        pincode: pincode,
      });

      // Displaying success message and resetting form fields
      alert("Form Submitted Successfully.....");
      setName("");
      setEmail("");
      setGender("");
      setNumber("");
      setAddress("");
      setCity("");
      setState("");
      setPincode("");
    } catch (err) {
      // Handling errors and displaying error message
      console.error(err);
      alert("Something went Wrong!!");
    }
  };

  // JSX for rendering the form
  return (
    <div>
      <form onSubmit={handleSubmit}>
        {/* Name input field */}
        Name:
        <input
          type="text"
          placeholder="Name"
          value={name}
```

```
          onChange={(e) => setName(e.target.value)}
        />
        <br />

        {/* Email input field */}
        Email:
        <input
          type="email"
          placeholder="Email"
          value={email}
          onChange={(e) => setEmail(e.target.value)}
        />
        <br />

        {/* Gender radio buttons */}
        <label>Gender:</label>
        <input
          type="radio"
          name="gender"
          value="Male"
          checked={gender === "Male"}
          onChange={(e) => setGender(e.target.value)}
        />
        Male
        <input
          type="radio"
          name="gender"
          value="Female"
          checked={gender === "Female"}
          onChange={(e) => setGender(e.target.value)}
        />
        Female
        <br />

        {/* Contact number input field */}
        Contact No:
        <input
          type="number"
          placeholder="Contact Number"
          value={number}
          onChange={(e) => setNumber(e.target.value)}
        />
        <br />

        {/* Address textarea */}
        Address:
        <textarea
          placeholder="Address"
          value={address}
          onChange={(e) => setAddress(e.target.value)}
        ></textarea>
        <br />

        {/* City input field */}
        City:
        <input
          type="text"
          placeholder="City"
          value={city}
          onChange={(e) => setCity(e.target.value)}
        />
        <br />

        {/* State input field */}
        State:
        <input
          type="text"
          placeholder="State"
          value={state}
          onChange={(e) => setState(e.target.value)}
        />
        <br />

        {/* Pincode input field */}
        Pincode:
        <input
          type="number"
          placeholder="Pincode"
          value={pincode}
          onChange={(e) => setPincode(e.target.value)}
        />
        <br />

        {/* Submit button */}
        <button type="submit">Submit</button>
      </form>
    </div>
  );
}

export default ApplicationForm;
```

App.js

```
// Importing CSS file for styling
import './App.css';

// Importing the ApplicationForm component from its file
import ApplicationForm from './component/ApplicationForm';

// Importing React and 'component' should be 'Component' (capitalized)
import React, { Component } from 'react';

// Defining the functional component App
function App() {
  return (
    <div>
      {/* Rendering the ApplicationForm component */}
      <ApplicationForm/>
    </div>
  );
}

// Exporting the App component as the default export
export default App;
```