HI Guys!

SET - I

1) Create a webpage with HTML describing your department use paragraph and list tags.

```
<!DOCTYPE html>
<html>
<head>
  <title>CSE Department</title>
</head>
<body>
  <h1>Welcome to the CSE Department</h1>
  <p>The Computer Science and Engineering (CSE) department focuses on
developing the technical and problem-solving skills of students.</p>

  <h2>Core Subjects:</h2>
  <ul>
    <li>Data Structures</li>
    <li>Operating Systems</li>
    <li>Computer Networks</li>
    <li>Machine Learning</li>
  </ul>

  <h2>Highlights:</h2>
  <ol>
    <li>Modern Labs</li>
    <li>Expert Faculty</li>
    <li>Industry Projects</li>
    <li>Hackathons & Workshops</li>
  </ol>
</body>
</html>
```

2) Develop a web application to control over different layouts.

```
<!DOCTYPE html>
<html>
<head>
  <title>Layout Control</title>
  <style>
    body { font-family: Arial; margin: 0; }
    .navbar { background: #333; color: white; padding: 10px; }
    .navbar a { color: white; margin: 10px; text-decoration: none; }
    .btns { margin: 15px; }
    .content { display: flex; gap: 15px; flex-wrap: wrap; padding: 10px; }
    .section { background: #eee; padding: 20px; width: 30%; }
    .list .section { width: 100%; }
  </style>
</head>
<body>

  <div class="navbar">
    <strong>My Website</strong>
    <a href="#">Home</a>
    <a href="#">Services</a>
  </div>
```

```
 <div class="btns">
  <button onclick="grid()">Grid View</button>
  <button onclick="list()">List View</button>
 </div>

 <div class="content" id="layout">
  <div class="section">Section 1: Web Development</div>
  <div class="section">Section 2: App Design</div>
  <div class="section">Section 3: Cloud Services</div>
 </div>

 <script>
  function grid() { layout.classList.remove("list"); }
  function list() { layout.classList.add("list"); }
  const layout = document.getElementById("layout");
 </script>

</body>
</html>
```

SET – II

1) Apply various colors to suitable distinguish key words, also apply font styling like italics,underline and two other fonts to words you find appropriate, also use header tags. (using html layout and links)

```
<!DOCTYPE html>
<html>
<head>
```

```
  <title>Styled Page</title>
  <style>
    .keyword { color: darkblue; font-weight: bold; }
    .highlight { color: green; font-family: 'Courier New'; }
    .important { color: maroon; font-family: 'Georgia'; }
  </style>
</head>
<body>

  <h1>HTML Styling Example</h1>

  <h2>Welcome to Web Development</h2>

  <p>
    HTML stands for <span class="keyword">HyperText Markup Language</span>.
    It is used to <i>design</i> the structure of web pages and
    <u>organize</u> content effectively.
  </p>

  <p>
    Styling is done using <span class="highlight">CSS (Cascading Style
Sheets)</span>,
    and layout is enhanced using tags like <span
class="important">&lt;div&gt;</span> and <span
class="important">&lt;section&gt;</span>.
  </p>
```

```
};

  <p>Learn more at <a href="https://www.w3schools.com"
target="_blank">W3Schools</a>.</p>

</body>
</html>
```

2) Data binding using Ajax.

```
<!DOCTYPE html>
<html>
<head>
  <title>AJAX Capital Lookup</title>
</head>
<body>

  <h3>Enter Country:</h3>
  <input id="country">
  <button onclick="loadData()">Show Capital</button>
  <p>Capital: <span id="capital"></span></p>

  <script>
    function loadData() {
      let c = document.getElementById("country").value;
      let xhr = new XMLHttpRequest();
      xhr.onload = function() {
        let data = JSON.parse(this.responseText);
        document.getElementById("capital").innerText = data[0].capital;
      };
      xhr.open("GET", "https://restcountries.com/v3.1/name/" + c + "?fullText=true",
true);

      xhr.send();

    }

  </script>


</body>

</html>
```

SET - III

1) Create links on the words e.g. “Wi-Fi” and “LAN” to link them to Wikipedia pages.

```
<!DOCTYPE html>

<html>

<head>

  <title>Link Example</title>

</head>

<body>


  <p>

    Modern internet access methods include

    <a href="https://en.wikipedia.org/wiki/Wi-Fi" target="_blank">Wi-Fi</a>

    and

    <a href="https://en.wikipedia.org/wiki/Local_area_network"
target="_blank">LAN</a>.

  </p>


</body>
```

```
</html>
```

2) Develop a web application to control over different layouts.

```
<!DOCTYPE html>
<html>
<head>
  <title>Layout Control</title>
  <style>
    body { font-family: Arial; margin: 0; }
    .navbar { background: #333; color: white; padding: 10px; }
    .navbar a { color: white; margin: 10px; text-decoration: none; }
    .btns { margin: 15px; }
    .content { display: flex; gap: 15px; flex-wrap: wrap; padding: 10px; }
    .section { background: #eee; padding: 20px; width: 30%; }
    .list .section { width: 100%; }
  </style>
</head>
<body>

  <div class="navbar">
    <strong>My Website</strong>
    <a href="#">Home</a>
    <a href="#">Services</a>
  </div>

  <div class="btns">
    <button onclick="grid()">Grid View</button>
    <button onclick="list()">List View</button>
  </div>

  <div class="content" id="layout">
    <div class="section">Section 1: Web Development</div>
    <div class="section">Section 2: App Design</div>
    <div class="section">Section 3: Cloud Services</div>
  </div>

  <script>
    function grid() { layout.classList.remove("list"); }
    function list() { layout.classList.add("list"); }
    const layout = document.getElementById("layout");
  </script>

</body>
</html>
```

SET - IV

1) Building Interfaces Using Javascript

a. Set up the Folder Structure.

b. Write the Model code and initialize the application.

c. Implement the list objects and use cases.

d. Implement the create object use case.

e. Implement the update object use case.

```
<!DOCTYPE html>
<html>
```

```
<head>
 <title>Student Management App</title>
</head>
<body>

 <h2>Student Management</h2>
 <form id="student-form">
  <input type="text" id="name" placeholder="Enter name" required>
  <button type="submit">Add</button>
 </form>

 <ul id="student-list"></ul>

 <script>
  let students = [
   { id: 1, name: 'Alice' },
   { id: 2, name: 'Bob' }
  ];

  function displayStudents() {
   let list = document.getElementById('student-list');
   list.innerHTML = '';
   students.forEach(s => {
    let li = document.createElement('li');
    li.textContent = s.name;
    list.appendChild(li);
    });
   }

   document.getElementById('student-form').onsubmit = function(e) {
    e.preventDefault();
    let name = document.getElementById('name').value.trim();
    if (name) {
     students.push({ id: students.length + 1, name: name });
     document.getElementById('name').value = '';
     displayStudents();
    }
   };

   window.onload = displayStudents;
  </script>

</body>
</html>
```

2) Create a webpage with HTML describing your department use paragraph and list tags.

```
<!DOCTYPE html>
<html>
<head>
  <title>CSE Department</title>
</head>
<body>
```

```
 <h1>Welcome to the CSE Department</h1>

 <p>The Computer Science and Engineering (CSE) department focuses on
developing the technical and problem-solving skills of students.</p>


 <h2>Core Subjects:</h2>

 <ul>

  <li>Data Structures</li>

  <li>Operating Systems</li>

  <li>Computer Networks</li>

  <li>Machine Learning</li>

 </ul>


 <h2>Highlights:</h2>

 <ol>

  <li>Modern Labs</li>

  <li>Expert Faculty</li>

  <li>Industry Projects</li>

  <li>Hackathons & Workshops</li>

 </ol>

</body>

</html>
```

SET - V

1) Develop a web application using left menu

```
<!DOCTYPE html>

<html>

<head>
```

```
<title>Left Menu Layout</title>

<style>

 body {

  margin: 0;

  font-family: Arial;

  display: flex;

 }


 .menu {

  width: 200px;

  background-color: #333;

  color: white;

  height: 100vh;

  padding: 20px;

 }


 .menu a {

  display: block;

  color: white;

  text-decoration: none;

  margin: 10px 0;

 }


 .menu a:hover {

  background-color: #444;

  padding-left: 10px;
```

```
    }

    .content {
      flex: 1;
      padding: 20px;
    }
  </style>
</head>
<body>

  <div class="menu">
    <h3>Navigation</h3>
    <a href="#">Home</a>
    <a href="#">About</a>
    <a href="#">Services</a>
    <a href="#">Contact</a>
  </div>

  <div class="content">
    <h2>Welcome to Our Web App</h2>
    <p>This is the main content area. Click on links in the left menu to
navigate.</p>
  </div>

</body>
</html>
```

2) Developing Web Page Styles using JavaScript and CSS.

```
<!DOCTYPE html>
<html>
<head>
 <title>Style Changer</title>
 <style>
  body {
   font-family: Arial;
   background-color: #f0f0f0;
   text-align: center;
   margin-top: 100px;
  }
  h1 { color: #333; }
  p { color: #666; font-size: 18px; }
  button {
   padding: 10px 20px;
   background: #007bff;
   color: white;
   border: none;
   border-radius: 5px;
  }
  button:hover { background: #0056b3; }
 </style>
</head>
<body>
```

```
 <h1>Hello, World!</h1>

 <p>This is a simple web page styled with JavaScript and CSS.</p>

 <button onclick="changeStyle()">Change Styles</button>


 <script>

  function changeStyle() {

   document.body.style.backgroundColor = 'yellow';

   document.querySelector('h1').style.color = 'orangered';

   document.querySelector('p').style.fontSize = '20px';

  }

 </script>


</body>

</html>
```

SET – VI

1) Develop Script interactive forms

```
<!DOCTYPE html>

<html>

<head>

 <title>Interactive Form</title>

 <style>

  body { font-family: Arial; background: #f0f0f0; text-align: center; padding: 50px;
}

  input, textarea, button { margin: 10px; padding: 8px; width: 250px; }

  button { background: #007bff; color: white; border: none; border-radius: 5px; }

  button:hover { background: #0056b3; }
  </style>
</head>
<body>

  <h2>Contact Us</h2>
  <form id="myForm">
    <input type="text" id="name" placeholder="Name" required><br>
    <input type="email" id="email" placeholder="Email" required><br>
    <textarea id="message" placeholder="Your Message" rows="3"
required></textarea><br>
    <button type="submit">Submit</button>
  </form>

  <script>
    document.getElementById("myForm").addEventListener("submit", function(e)
{
      e.preventDefault();
      let name = document.getElementById("name").value;
      let email = document.getElementById("email").value;
      let msg = document.getElementById("message").value;
      alert(`Submitted:\nName: ${name}\nEmail: ${email}\nMessage: ${msg}`);
      this.reset();
    });
  </script>

</body>
</html>
```

2) Develop setting to change the theme of entire web Application.

```
<!DOCTYPE html>

<html>

<head>

 <title>Theme Toggle</title>

 <style>

  body { font-family: sans-serif; text-align: center; padding: 50px; transition:
0.3s; }

  .dark { background: #121212; color: #fff; }

  .switch {

   position: relative; display: inline-block; width: 50px; height: 25px;

  }

  .switch input { display: none; }

  .slider {

   position: absolute; top: 0; left: 0; right: 0; bottom: 0;

   background: #ccc; border-radius: 25px; transition: .4s;

  }

  .slider:before {

   content: ""; position: absolute; height: 19px; width: 19px;

   left: 3px; bottom: 3px; background: white; border-radius: 50%; transition: .4s;

  }

  input:checked + .slider { background: #2196F3; }

  input:checked + .slider:before { transform: translateX(25px); }

 </style>

</head>

<body>
```

```
  <h2>Theme Toggle</h2>
  <label class="switch">
    <input type="checkbox" onclick="document.body.classList.toggle('dark')">
    <span class="slider"></span>
  </label>

</body>
</html>
```

SET - VII

1) React Environment setup

a. Setting up development environment.

Setting up a React development environment involves installing Node.js, the Node Package

Manager (npm), and creating a new React application using Create React App (CRA).

Step 1: Install Node.js and npm

Go to the official Node.js website.

Step 2: Verify Installation

```
node -v
npm -v
```

Step 3: Install Create React App

```
npm install -g create-react-app
```

Step 4: Create a New React Application

```
create-react-app my-app
cd my-app
```

Step 5: Start the Development Server

npm start

b. Integration with Existing Apps.

c. Running on Device.

d. Debugging

e. Testing

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>React Integration Example</title>
</head>
<body>
  <h1>My Existing Application</h1>
  <div id="react-root"></div>

  <!-- React and ReactDOM via CDN -->
  <script src="https://unpkg.com/react@18/umd/react.development.js"
crossorigin></script>
  <script src="https://unpkg.com/react-dom@18/umd/react-
dom.development.js" crossorigin></script>

  <!-- Babel for JSX transformation -->
  <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

  <!-- React Component -->
  <script type="text/babel" src="app.js"></script>
</body>
</html>
```

app.js

```
// Using React 18+
const root = ReactDOM.createRoot(document.getElementById('react-root'));

class App extends React.Component {
  render() {
    return (
      <div>
        <h2>Hello, React!</h2>
        <p>This is a React component integrated into an existing HTML application.</p>
      </div>
    );
  }
}

root.render(<App />);
```

2) Programming With React

a. Basics Interactive examples.

b. Function Components and Class Components

c. React Native Fundamental, Handling Text Input,

a. Basics Interactive examples.

IN TERMINAL

```
npx create-react-app my-app

cd my-app

npm start
```

app.js

```
import React from 'react';
function App() {
return (
<div>
<h1>Hello, World!</h1>
<p>This is a first React example.</p>
</div>
);
}
export default App;
```

b. Function Components and Class Components

functioncomponent.jsx

```
import React, { useState } from 'react';
const FunctionComponent = () => {
const [count, setCount] = useState(0);
const increment = () => {
setCount(count + 1);
};
const decrement = () => {
setCount(count - 1);
};
return (
<div>
<h2>Function Component Counter</h2>
<p>Count: {count}</p>
<button onClick={increment}>Increment</button>
<button onClick={decrement}>Decrement</button>
</div>
);
};
export default FunctionComponent;
```

classcomponent.jsx

```
import React, { Component } from 'react';
class ClassComponent extends Component {
constructor(props) {
super(props);
this.state = {
count: 0,
};
}
increment = () => {
this.setState({ count: this.state.count + 1 });
};
decrement = () => {
this.setState({ count: this.state.count - 1 });
};
render() {
return (
<div>
<h2>Class Component Counter</h2>
<p>Count: {this.state.count}</p>
<button onClick={this.increment}>Increment</button>
<button onClick={this.decrement}>Decrement</button>
</div>
);
}
}
export default ClassComponent;
```

App.jsx

```
import React from 'react';
import FunctionComponent from './functioncomponent';
import ClassComponent from './classcomponent';
const App = () => {
return (
<div>
<h1>Function vs Class Components Example</h1>
<FunctionComponent />
<ClassComponent />
</div>
);
};
export default App;
```

IN TERMINAL: npm start

SET - VIII

1) React Environment setup

Repeated!

2) Programming With React

a. Basics Interactive examples.

b. Using a scroll View, using List View.

C. Platform Specific Code.

a. Basics Interactive examples.

IN TERMINAL

```
npx create-react-app my-app
cd my-app
npm start
```

app.js

```
import React from 'react';
function App() {
return (
<div>
<h1>Hello, World!</h1>

<p>This is a first React example.</p>

</div>

);

}

export default App;
```

b. Using Scroll View, Using List View

Scroll View Simulation in Web:

ScrollList.js

```
import React from 'react';


const ScrollList = () => {
  const items = Array.from({ length: 30 }, (_, i) => `Item ${i + 1}`);


  return (
    <div style={{ height: '200px', overflowY: 'scroll', border: '1px solid #ccc' }}>
      <ul>
        {items.map(item => <li key={item}>{item}</li>)}
      </ul>
    </div>
  );
};


export default ScrollList;
```

App.js (with scroll list)

```
import React from 'react';
import ScrollList from './ScrollList';


function App() {
  return (
    <div>
      <h1>Scroll & List Example</h1>
      <ScrollList />
    </div>
  );
}
export default App;
```

c. Platform-Specific Code (Short Version)

PlatformCheck.js

```
import React from 'react';


const PlatformCheck = () => (
  <p>{navigator.userAgent.includes("Mobi") ? "Mobile View" : "Desktop View"}</p>
);


export default PlatformCheck;
```

In your App.js

```
import PlatformCheck from './PlatformCheck';
<PlatformCheck />
```

How to Run:

```
npx create-react-app my-app

cd my-app

npm start
```

SET - IX

1) Node js modules- npm, functions, modules, installing packages, json

fun.js (Functions, Buffer)

```
function greet(name) {
  return `Hello, ${name}!`;
}
console.log(greet("Jaya"));


const calc = (a, b) => a + b;
console.log(calc(10, 20));


const buffer = Buffer.from('Hello, World!');
console.log(buffer);
console.log(buffer.toString());
```

Run:

```
node fun.js
```

📄 **modules.js (Core Modules, Local Modules, Module Exports)**

```
const fs = require('fs');
fs.readFile('example.txt', 'utf8', (err, data) => {
  if (err) throw err;
  console.log(data);
});
```

```
const os = require('os');
console.log('OS:', os.type(), os.platform(), os.arch());
console.log('Release:', os.release(), 'Host:', os.hostname());
console.log('Version:', os.version(), 'Uptime:', os.uptime());

const path = require('path');
console.log(path.basename('/home/user/dir/file.txt'));
console.log(path.dirname('/home/user/dir/file.txt'));
console.log(path.extname('/home/user/dir/file.txt'));

// Local module methods
exports.add = (a, b) => a + b;
exports.subtract = (a, b) => a - b;

const math = require('./modules');
console.log(math.add(2, 3));
console.log(math.subtract(5, 2));

// Exporting function
module.exports = name => `Hello, ${name}!`;
const greet = require('./modules');
console.log(greet('Bob'));
```

Run After Creating example.txt

```
node modules.js
```

2) File System: Read File, writing a File, opening a File Deleting a File, Writing a file asynchronously and Other I/O Operations.

Synch.js

```
        const fs = require('fs');
// Reading a file asynchronously
fs.readFile('example.txt', 'utf8', (err, data) => {
  if (err) {
    console.error(err);
    return;   }
  console.log(data);
});
// Writing to a file asynchronously
fs.writeFile('example.txt', 'Hello, World!', 'utf8', (err) => {
  if (err) {
    console.error(err);
    return;   }
  console.log('File written successfully');
});


// Opening a file asynchronously
fs.open('example.txt', 'r', (err, fd) => {
  if (err) {
    console.error(err);
    return;
  }
  console.log('File opened successfully');
  fs.close(fd, (err) => {
    if (err) {
      console.error(err);
    }   });
});
// Deleting a file asynchronously
fs.unlink('example.txt', (err) => {
  if (err) {
    console.error(err);
    return;   }
  console.log('File deleted successfully');
});
```

SET - X

1) Node js modules- npm, functions, modules, installing packages, json
2) File System: Read File, writing a File, opening a File Deleting a File, Writing a file asynchronously and Other I/O Operations

QUESTIONS REPEATED PREVIOUSLT..KINDLY CHECK!