Scenario-based assignments that integrate HTML, CSS, and JavaScript for a Java Full Stack Development course:

Objective:

- ☐ **The given assignments offer a practical and real-world context for students to apply their HTML, CSS, and JavaScript skills.**
- ☐ **They focus on building functional applications while allowing students to exercise their creativity and problem-solving abilities.**

Time:60 mins

Instructions:

1. **Solve the given questions.**
2. **Take screenshot of the code and the executed output**
3. **Upload the above in your respective folder**

Assignment 1: Personal Portfolio Website

Scenario: As a developer, you've been tasked to create a personal portfolio website to showcase your skills and projects.

Requirements:

1. HTML: Create a homepage with sections for About, Projects, and Contact.
2. CSS: Apply styles to ensure the website is visually appealing and responsive.
3. JavaScript: Implement a contact form with form validation for email and required fields.

Coding:

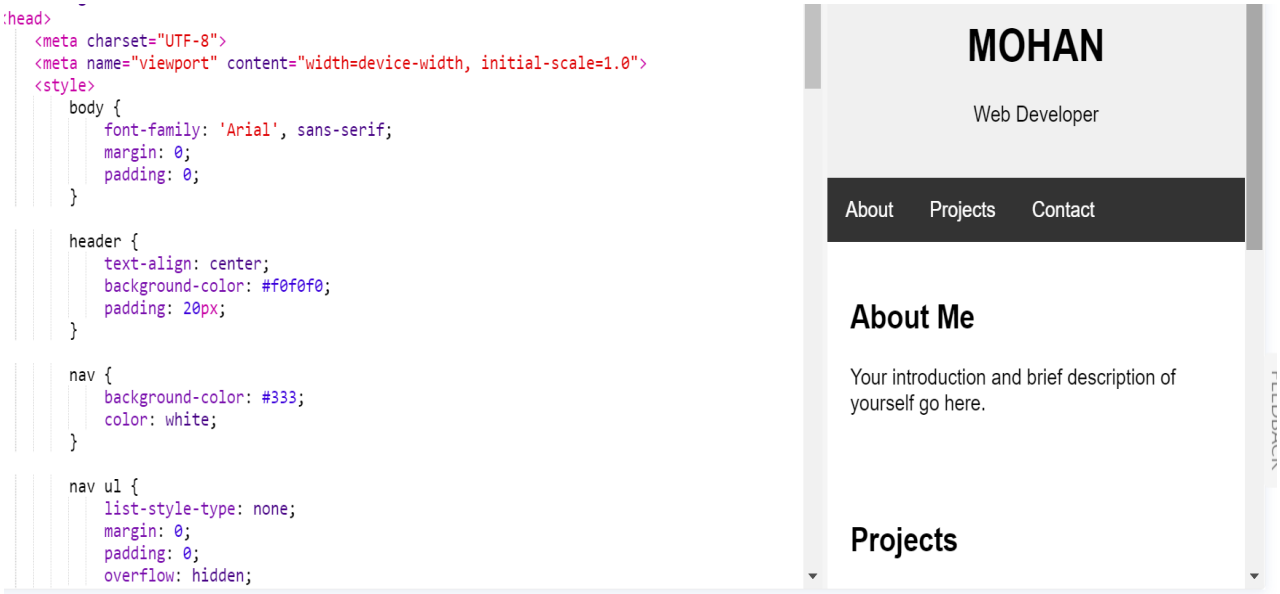
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        body {
            font-family: 'Arial', sans-serif;
            margin: 0;
            padding: 0;
        }

        header {
            text-align: center;
            background-color: #f0f0f0;
            padding: 20px;
        }

        nav {
            background-color: #333;
            color: white;
        }

        nav ul {
```

```
    list-style-type: none;
    margin: 0;
    padding: 0;
    overflow: hidden;
}

nav li {
    float: left;
}

nav a {
    display: block;
    color: white;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}

nav a:hover {
    background-color: #ddd;
    color: black;
}

section {
    padding: 20px;
}

form {
    max-width: 600px;
    margin: auto;
}

label {
    display: block;
    margin-bottom: 8px;
}

input,
textarea {
    width: 100%;
    padding: 10px;
    margin-bottom: 15px;
    border: 1px solid #ccc;
    border-radius: 4px;
    box-sizing: border-box;
}

input[type="submit"] {
    background-color: #4caf50;
    color: white;
    cursor: pointer;
}

input[type="submit"]:hover {
    background-color: #45a049;
}
```

```
        footer {
            text-align: center;
            padding: 10px;
            background-color: #333;
            color: white;
        }
    </style>
    <title>Your Name - Portfolio</title>
</head>
<body>
    <header>
        <h1>Your Name</h1>
        <p>Web Developer</p>
    </header>

    <nav>
        <ul>
            <li><a href="#about">About</a></li>
            <li><a href="#projects">Projects</a></li>
            <li><a href="#contact">Contact</a></li>
        </ul>
    </nav>

    <section id="about">
        <h2>About Me</h2>
        <p>Your introduction and brief description of yourself go here.</p>
    </section>

    <section id="projects">
        <h2>Projects</h2>
        <!-- Add your project details and links here -->
    </section>

    <section id="contact">
        <h2>Contact Me</h2>
        <form id="contactForm" onsubmit="submitForm(event)">
            <label for="name">Name:</label>
            <input type="text" id="name" name="name" required>

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

            <label for="message">Message:</label>
            <textarea id="message" name="message" rows="4" required></textarea>

            <input type="submit" value="Submit">
        </form>
    </section>

    <footer>
        <p>&copy; 2023 Your Name. All rights reserved.</p>
    </footer>

    <script>
        function submitForm(event) {
```

```
        event.preventDefault();

        // Simple email validation
        const email = document.getElementById("email").value;
        if (!validateEmail(email)) {
            alert("Please enter a valid email address.");
            return;
        }

        // You can add additional validation or submit the form data as needed

        alert("Form submitted successfully!");
        // Here you can send the form data to your server or perform any other action
    }

    function validateEmail(email) {
        const regex = /\S+@\S+\.\S+/;
        return regex.test(email);
    }
  </script>
</body>
</html>
```

Output:



Assignment 2: Interactive Quiz Application

Scenario: You are building an interactive quiz application for users to test their knowledge.

Requirements:

1. HTML: Design the structure for the quiz with questions and multiple-choice answers.
2. CSS: Style the quiz interface to make it visually appealing.
3. JavaScript: Implement functionality to display questions one at a time, handle user answers, and show the final score.

Coding:

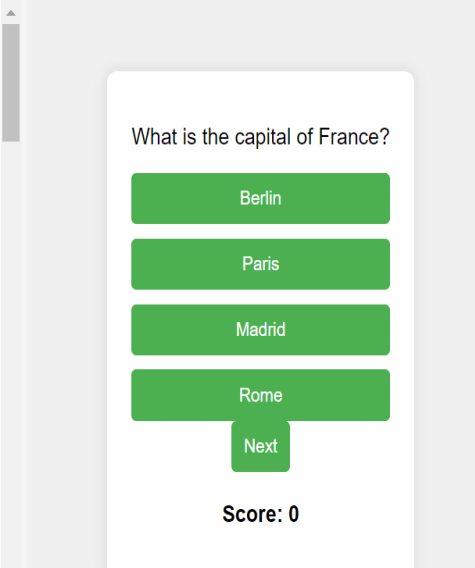
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        body {
            font-family: 'Arial', sans-serif;
            margin: 0;
            padding: 0;
            display: flex;
            align-items: center;
            justify-content: center;
            height: 100vh;
            background-color: #f0f0f0;
        }

        #quiz-container {
            background-color: #fff;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            text-align: center;
        }

        .btn-container {
            display: flex;
            flex-direction: column;
            gap: 10px;
        }

        .btn {
            padding: 10px;
            background-color: #4caf50;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }

        .btn:hover {
            background-color: #45a049;
        }

        #score {
            margin-top: 20px;
            font-weight: bold;
        }
    </style>
    <title>Interactive Quiz</title>
</head>
<body>
    <div id="quiz-container">
        <div id="question-container">
            <p id="question-text"></p>
```

```
    </div>
    <div id="answer-buttons" class="btn-container">
        <!-- Answer buttons will be dynamically added here -->
    </div>
    <button id="next-button" class="btn" onclick="nextQuestion()">Next</button>
    <p id="score">Score: 0</p>
</div>

<script>
    const questions = [
        {
            question: "What is the capital of France?",
            answers: [
                { text: "Berlin", correct: false },
                { text: "Paris", correct: true },
                { text: "Madrid", correct: false },
                { text: "Rome", correct: false }
            ]
        },
        {
            question: "Which planet is known as the Red Planet?",
            answers: [
                { text: "Earth", correct: false },
                { text: "Mars", correct: true },
                { text: "Venus", correct: false },
                { text: "Jupiter", correct: false }
            ]
        },
        // Add more questions as needed
    ];

    let currentQuestionIndex = 0;
    let score = 0;

    const questionTextElement = document.getElementById("question-text");
    const answerButtonsElement = document.getElementById("answer-buttons");
    const nextButton = document.getElementById("next-button");
    const scoreElement = document.getElementById("score");

    function startQuiz() {
        currentQuestionIndex = 0;
        score = 0;
        showQuestion(questions[currentQuestionIndex]);
    }

    function showQuestion(question) {
        questionTextElement.innerText = question.question;
        answerButtonsElement.innerHTML = "";

        question.answers.forEach(answer => {
            const button = document.createElement("button");
            button.innerText = answer.text;
            button.classList.add("btn");
            button.addEventListener("click", () => selectAnswer(answer));
            answerButtonsElement.appendChild(button);
        });
```

```
        }

        function selectAnswer(answer) {
            if (answer.correct) {
                score++;
            }

            if (currentQuestionIndex < questions.length - 1) {
                currentQuestionIndex++;
                showQuestion(questions[currentQuestionIndex]);
            } else {
                showScore();
            }
        }

        function showScore() {
            questionTextElement.innerText = "Quiz Completed!";
            answerButtonsElement.innerHTML = "";
            nextButton.style.display = "none";
            scoreElement.innerText = `Your Score: ${score}/${questions.length}`;
        }

        function nextQuestion() {
            currentQuestionIndex++;
            showQuestion(questions[currentQuestionIndex]);
            nextButton.style.display = "block";
        }

        // Start the quiz when the page loads
        startQuiz();
    </script>
</body>
</html>
```

Output:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        body {
            font-family: 'Arial', sans-serif;
            margin: 0;
            padding: 0;
            display: flex;
            align-items: center;
            justify-content: center;
            height: 100vh;
            background-color: #f0f0f0;
        }

        #quiz-container {
            background-color: #fff;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            text-align: center;
        }
```



Assignment 3: E-commerce Product Page

Scenario: Develop an e-commerce product page for an online store.

Requirements:

1. HTML: Create a product display page showing details, images, and a section for customer reviews.
2. CSS: Style the product page and implement a responsive layout.
3. JavaScript: Add a rating system where users can rate products and display the average rating.

Coding:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="styles.css">
    <title>Product Page</title>
</head>
<body>
    <div class="product-container">
        <div class="product-images">
            <!-- Product images go here -->
            <img src="product-image1.jpg" alt="Product Image 1">
            <img src="product-image2.jpg" alt="Product Image 2">
            <img src="product-image3.jpg" alt="Product Image 3">
        </div>
        <div class="product-details">
            <h1>Product Name</h1>
            <p>Description: Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
            <p>Price: $99.99</p>

            <div class="rating-container">
                <p>Rating: <span id="average-rating">0</span></p>
                <div id="rating-stars" class="rating-stars">
                    <!-- Rating stars go here -->
                    <span class="star" onclick="rateProduct(1)">&#9733;</span>
                    <span class="star" onclick="rateProduct(2)">&#9733;</span>
                    <span class="star" onclick="rateProduct(3)">&#9733;</span>
                    <span class="star" onclick="rateProduct(4)">&#9733;</span>
                    <span class="star" onclick="rateProduct(5)">&#9733;</span>
                </div>
            </div>
        </div>
        <div class="customer-reviews">
            <h2>Customer Reviews</h2>
            <div id="reviews-list">
                <!-- Customer reviews go here -->
            </div>
            <textarea id="review-input" placeholder="Write a review..."></textarea>
            <button onclick="submitReview()">Submit Review</button>
        </div>
    </div>
```

```
    <script src="script.js"></script>
</body>
</html>
```

styles.css:

```
css
Copy code
body {
    font-family: 'Arial', sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f0f0f0;
}

.product-container {
    display: flex;
    max-width: 800px;
    margin: 20px auto;
    background-color: #fff;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

.product-images {
    flex: 1;
    padding: 20px;
}

.product-images img {
    width: 100%;
    margin-bottom: 10px;
}

.product-details {
    flex: 1;
    padding: 20px;
}

.rating-container {
    margin-top: 20px;
}

.rating-stars {
    font-size: 24px;
    cursor: pointer;
}

.star {
    color: #ffd700;
}

.customer-reviews {
    flex: 1;
    padding: 20px;
}
```

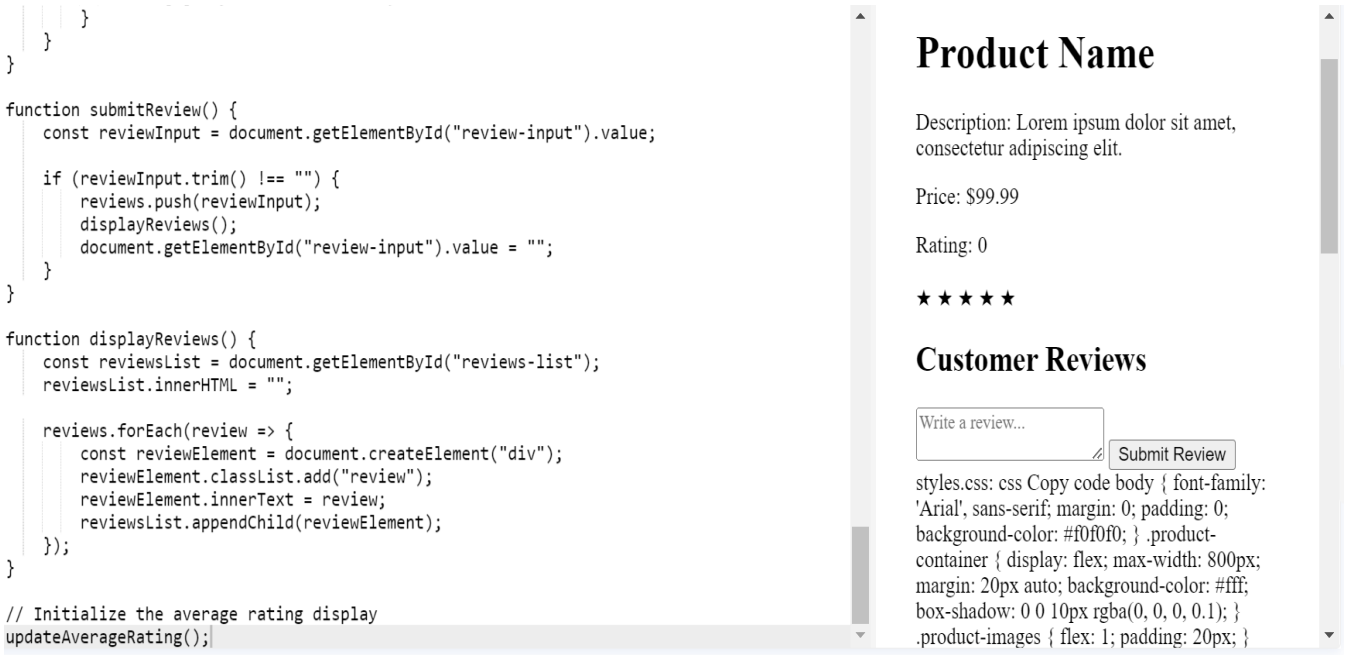
```
textarea {
    width: 100%;
    padding: 10px;
    margin-bottom: 10px;
}

button {
    padding: 10px;
    background-color: #4caf50;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}

button:hover {
    background-color: #45a049;
}
```

script.js:

```
javascript
Copy code
let ratings = [];
let reviews = [];

function rateProduct(stars) {
    ratings.push(stars);
    updateAverageRating();
}

function updateAverageRating() {
    const totalRating = ratings.reduce((acc, rating) => acc + rating, 0);
    const averageRating = totalRating / ratings.length || 0;

    document.getElementById("average-rating").innerText = averageRating.toFixed(1);
    updateRatingStars(averageRating);
}

function updateRatingStars(averageRating) {
    const stars = document.getElementsByClassName("star");

    for (let i = 0; i < stars.length; i++) {
        if (i < averageRating) {
            stars[i].style.color = "#ffd700";
        } else {
            stars[i].style.color = "#ccc";
        }
    }
}

function submitReview() {
    const reviewInput = document.getElementById("review-input").value;

    if (reviewInput.trim() !== "") {
        reviews.push(reviewInput);
        displayReviews();
```

```
        document.getElementById("review-input").value = "";
    }
}

function displayReviews() {
    const reviewsList = document.getElementById("reviews-list");
    reviewsList.innerHTML = "";

    reviews.forEach(review => {
        const reviewElement = document.createElement("div");
        reviewElement.classList.add("review");
        reviewElement.innerText = review;
        reviewsList.appendChild(reviewElement);
    });
}

// Initialize the average rating display
updateAverageRating();
```

Output:



Assignment 4: Weather Forecast Application

Scenario: Build a simple weather forecast application to display current weather conditions.

Requirements:

1. HTML: Design the layout to show current weather details and a search bar for different locations.
2. CSS: Apply styles for weather elements and ensure the layout is user-friendly.
3. JavaScript: Use an API to fetch weather data based on user input and display it on the page.

Coding:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="styles.css">
    <title>Weather Forecast</title>
</head>
<body>
    <div class="weather-container">
        <h1>Weather Forecast</h1>
        <div class="search-container">
            <input type="text" id="location-input" placeholder="Enter location">
            <button onclick="getWeather()">Get Weather</button>
        </div>
        <div id="weather-info">
            <!-- Weather details will be displayed here -->
        </div>
    </div>

    <script src="script.js"></script>
</body>
</html>
```

styles.css:

```
css
Copy code
body {
    font-family: 'Arial', sans-serif;
    margin: 0;
    padding: 0;
    display: flex;
    align-items: center;
    justify-content: center;
    height: 100vh;
    background-color: #f0f0f0;
}

.weather-container {
    background-color: #fff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    text-align: center;
}

.search-container {
    margin-top: 20px;
}

input {
    padding: 10px;
    margin-right: 10px;
    border: 1px solid #ccc;
    border-radius: 4px;
```

```
}

button {
    padding: 10px;
    background-color: #4caf50;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}

button:hover {
    background-color: #45a049;
}

#weather-info {
    margin-top: 20px;
}
```

script.js:

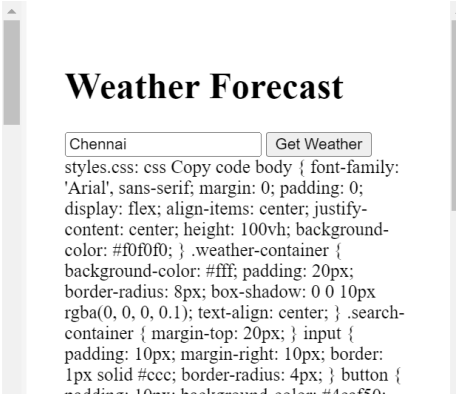
javascript
Copy code

```
async function getWeather() {
    const locationInput = document.getElementById("location-input").value;

    if (locationInput.trim() === "") {
        alert("Please enter a location.");
        return;
    }

    const apiKey = "YOUR_OPENWEATHERMAP_API_KEY"; // Replace with your API key
    const apiUrl =
`https://api.openweathermap.org/data/2.5/weather?q=${locationInput}&appid=${apiKey}&units=m
etric`;

    try {
        const response = await fetch(apiUrl);
        const data = await response.json();

        if (response.ok) {
            displayWeather(data);
        } else {
            alert(`Error: ${data.message}`);
        }
    } catch (error) {
        console.error("Error fetching weather data:", error);
        alert("Error fetching weather data. Please try again later.");
    }
}

function displayWeather(data) {
    const weatherInfo = document.getElementById("weather-info");
    weatherInfo.innerHTML = "";

    const cityName = data.name;
    const temperature = data.main.temp;
```

```
    const weatherDescription = data.weather[0].description;

    const weatherDetails = document.createElement("div");
    weatherDetails.innerHTML = `
        <h2>${cityName}</h2>
        <p>Temperature: ${temperature}°C</p>
        <p>Weather: ${weatherDescription}</p>
    `;

    weatherInfo.appendChild(weatherDetails);
}
```

Output:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="styles.css">
    <title>Weather Forecast</title>
</head>
<body>
    <div class="weather-container">
        <h1>Weather Forecast</h1>
        <div class="search-container">
            <input type="text" id="location-input" placeholder="Enter location">
            <button onclick="getWeather()">Get Weather</button>
        </div>
        <div id="weather-info">
            <!-- Weather details will be displayed here -->
        </div>
    </div>
```

Assignment 5: Task Management Dashboard

Scenario: Create a task management dashboard for users to add, delete, and organize their tasks.

Requirements:

1. HTML: Design a dashboard with sections for adding tasks, displaying them, and marking them as complete.
2. CSS: Style the dashboard for easy task readability and interaction.
3. JavaScript: Implement functionality to add tasks, mark them complete, and delete tasks from the list.

Coding:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        body {
            font-family: 'Arial', sans-serif;
            margin: 0;
            padding: 0;
            display: flex;
            align-items: center;
            justify-content: center;
            height: 100vh;
```

```
    background-color: #f0f0f0;
}

.task-dashboard {
    background-color: #fff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    text-align: center;
}

.task-input {
    margin-bottom: 20px;
}

input {
    padding: 10px;
    margin-right: 10px;
    border: 1px solid #ccc;
    border-radius: 4px;
}

button {
    padding: 10px;
    background-color: #4caf50;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}

button:hover {
    background-color: #45a049;
}

ul {
    list-style-type: none;
    padding: 0;
}

li {
    display: flex;
    align-items: center;
    justify-content: space-between;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 4px;
    margin-bottom: 10px;
}

.completed-task {
    text-decoration: line-through;
    color: #888;
}

.delete-button {
```

```
            background-color: #ff6347;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }

        .delete-button:hover {
            background-color: #d63434;
        }
    </style>
    <title>Task Management Dashboard</title>
</head>
<body>
    <div class="task-dashboard">
        <div class="task-input">
            <input type="text" id="taskInput" placeholder="Add a new task">
            <button onclick="addTask()">Add Task</button>
        </div>
        <ul id="taskList">
            <!-- Task items will be dynamically added here -->
        </ul>
    </div>

    <script>
        function addTask() {
            const taskInput = document.getElementById("taskInput");
            const taskList = document.getElementById("taskList");

            const taskText = taskInput.value.trim();

            if (taskText !== "") {
                const listItem = document.createElement("li");
                listItem.innerHTML = `
                    <span onclick="toggleComplete(this)">${taskText}</span>
                    <button class="delete-button" onclick="deleteTask(this)">Delete</button>
                `;

                taskList.appendChild(listItem);
                taskInput.value = "";
            }
        }

        function deleteTask(button) {
            const listItem = button.parentNode;
            listItem.remove();
        }

        function toggleComplete(span) {
            span.classList.toggle("completed-task");
        }
    </script>
</body>
</html>
```

Output:

```
        const taskList = document.getElementById("taskList");

        const taskText = taskInput.value.trim();

        if (taskText !== "") {
            const listItem = document.createElement("li");
            listItem.innerHTML = `
                <span onclick="toggleComplete(this)">${taskText}</span>
                <button class="delete-button" onclick="deleteTask(this)">Delete</button
            `;

            taskList.appendChild(listItem);
            taskInput.value = "";
        }
    }

    function deleteTask(button) {
        const listItem = button.parentNode;
        listItem.remove();
    }

    function toggleComplete(span) {
        span.classList.toggle("completed-task");
    }
</script>
```