ASSIGNMENT-14.1

NAME :- G.Jeevan

ROLL NO :- 2403A52055

BATCH N0 :- 03

Task 1: Create a Responsive Web Page Layout

Instructions:

• Design a basic web page layout with a header, main content area, and footer using HTML and CSS.
• Use AI to assist in generating responsive CSS for different screen sizes.
• Ensure the layout is clean and visually organized**.**

CODE:-

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Responsive Web Page</title>
<style>
 *{margin:0;padding:0;box-sizing:border-box;}
 body{font-family:Arial,sans-serif;}
 header,footer{background:#0077b6;color:white;text-align:center;padding:15px;}
 nav ul{list-style:none;display:flex;justify-content:center;gap:20px;margin-top:10px;}
 nav a{color:white;text-decoration:none;font-weight:bold;}
 main{text-align:center;padding:20px;}
 main img{width:100%;max-width:500px;border-radius:10px;margin-top:10px;}
 footer{font-size:0.9em;}
 @media(max-width:768px){
```

```
 nav ul{flex-direction:column;gap:10px;}
  }
</style>
</head>
<body>

<header>
  <h1>My Responsive Page</h1>
  <nav>
    <ul>
      <li><a       href="#">Home</a></li>
      <li><a       href="#">About</a></li>
      <li><a    href="#">Services</a></li>
      <li><a href="#">Contact</a></li>
    </ul>
  </nav>
</header>

<main>
  <h2>Welcome!</h2>
  <p>This is a simple responsive web page using HTML and CSS.</p>
  <img src="https://via.placeholder.com/500x250" alt="Sample Image">
</main>

<footer>
  <p>&copy; 2025 My Website | Contact: info@example.com</p>
</footer>

</body>
</html>
```

OUTPUT -

OBSERVATION :

- The HTML document defines the structure of the web page.
- The <head> section includes the page title, character encoding, and CSS styling.
- The <meta viewport> tag makes the page adjust to mobile and tablet screens.
- The <header> contains the website title and navigation links.
- The navigation menu helps users move between different sections.
- The <main> section holds the main content like headings, text, and an image.
- The <footer> displays copyright and contact information.
- CSS styles are written inside the <style> tag to give color, spacing, and alignment.
- The layout uses the flexbox property to arrange navigation links in a line.
- The background color, text color, and padding make the design clean and readable.
- The image in the main section is centered and adjusts to screen width.
- Media queries in CSS make the layout responsive for tablets and mobiles.
- On small screens, navigation links appear vertically for better readability.
- The overall design ensures that the web page looks good on all screen sizes.
- The page is simple, responsive, and visually organized with basic HTML and CSS.

Task 2: **Interactive Button with JavaScript**

Instructions:

- The code is clean and well-commented Create a button on a web page.
- Use AI to generate JavaScript code that displays an alert message when the button is clicked.
- Ensure the code is clean and well-commented

CODE :

```
<!DOCTYPEhtml>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Interactive Button</title>
 <style>
  /* Simple styling for the button */
  body {
   font-family: Arial, sans-serif;
   text-align: center;
   margin-top: 100px;
  }
  button {
   background-color: #4CAF50;
   color: white;
   border: none;
   padding: 12px 24px;
   font-size: 16px;
   border-radius: 8px;
   cursor: pointer;
  }
  button:hover {
   background-color: #45a049;
  }
```

```
 </style>
</head>
<body>
  <!--Button Element -->
  <button id="alertButton">Click Me!</button>

  <script>
    //Get the button element by its ID
    const button = document.getElementById("alertButton");

    //Add a click event listener to the button
    button.addEventListener("click", function() {
      //Display an alert message when button is clicked
      alert("Hello! You clicked the button!");
    });
  </script>
</body>
</html>
```

OUTPUT :

EXPLANATION :

- HTML structure created with <button> element.
- CSS styles the page and button for a clean look.
- JavaScript selects the button using its id.
- An event listener waits for a "click" action.
- When clicked, an alert message is shown to the user.
- Code is properly commented and organized for readability.

Task 3: Formwith Validation

Instructions:

- Design a contact form with fields: Name, Email, Message.
- Use AI to generate JavaScript code for form validation (e.g., non-empty fields, valid email format).
- Add inline error messages if input is invalid.

CODE :

```
<!DOCTYPEhtml>
<html lang="en">
<head>

 <metacharset="UTF-8">
 <metaname="viewport" content="width=device-width, initial-scale=1.0">
 <title>Contact Form with Validation</title>
 <style>
  /*Basic page styling */
  body {
   font-family: Arial, sans-serif;
   margin: 50px;
   background-color: #f9f9f9;
  }

  h2{
   text-align: center;
  }

  form {
   max-width: 400px;
   margin: auto;
   background: #fff;
   padding: 20px;
   border-radius: 10px;
   box-shadow: 0 0 10px rgba(0,0,0,0.1);
```

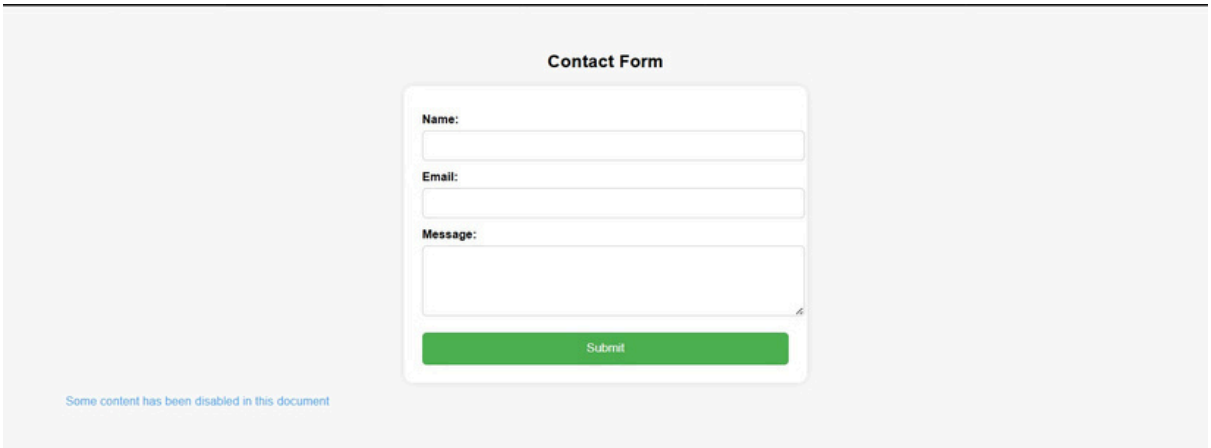
```
}

label {
  display:        block;
  margin-top:   10px;
  font-weight: bold;
}

input, textarea {
  width: 100%;
  padding: 8px;
  margin-top: 5px;
  border: 1px solid #ccc;
  border-radius: 5px;
}

button {
  margin-top: 15px;
  padding: 10px 15px;
  background-color: #4CAF50;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  width: 100%;
}

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

/*Error message styling */
.error {
```

```
    color: red;
    font-size: 13px;
   }

   /*Success message */
   .success {
    text-align: center;
    color: green;
    font-weight: bold;
   }
  </style>
</head>
<body>

  <h2>Contact Form</h2>

  <form id="contactForm">
   <label>Name:</label>
   <input type="text" id="name">
   <span class="error" id="nameError"></span>

   <label>Email:</label>
   <input type="text" id="email">
   <span class="error" id="emailError"></span>

   <label>Message:</label>
   <textarea id="message" rows="4"></textarea>
   <span class="error" id="messageError"></span>

   <button type="submit">Submit</button>
  </form>

  <p class="success" id="successMsg"></p>
```

```
<script>
  //Get form elements
  const form = document.getElementById("contactForm");
  const name = document.getElementById("name");
  const email = document.getElementById("email");
  const message = document.getElementById("message");

  const nameError = document.getElementById("nameError");
  const emailError = document.getElementById("emailError");
  const messageError = document.getElementById("messageError");
  const successMsg = document.getElementById("successMsg");

  //Function to validate email format using regex
  function isValidEmail(email) {
    return /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email);
  }

  //Validate form on submit
  form.addEventListener("submit", function(e) {
    e.preventDefault(); // Prevent page reload

    //Reset error messages
    nameError.textContent = "";
    emailError.textContent = "";
    messageError.textContent = "";
    successMsg.textContent = "";

    letisValid = true;

    //Check name
    if(name.value.trim() === "") {
      nameError.textContent = "Name is required";
```

```
      isValid = false;
    }

    //Check email
    if(email.value.trim() === "") {
      emailError.textContent = "Email is required";
      isValid = false;
    }else if (!isValidEmail(email.value.trim())) {
      emailError.textContent = "Invalid email format";
      isValid = false;
    }

    //Check message
    if(message.value.trim() === "") {
      messageError.textContent = "Message cannot be empty";
      isValid = false;
    }

    //If all valid
    if(isValid) {
      successMsg.textContent = "Form submitted successfully!";
      form.reset(); // Clear form fields
    }
   });
  </script>
</body>
</html>
```

OUTPUT :



EXPLANATION :

- HTML form created with fields: Name, Email, Message.
- CSS adds a simple, clean layout and styles for errors and success messages.
- JavaScript validates each field on form submission.
- Email validation uses a regular expression (regex).
- Inline error messages appear below invalid fields.
- If all fields are valid, a success message is displayed, and the form reset.

Task 4: **Dynamic Content Generation**

Instructions:

• Create a list of items (e.g., product names) using HTML.
• Use AI-generated JavaScript to dynamically add or remove items from the list when a button is clicked.

CODE:

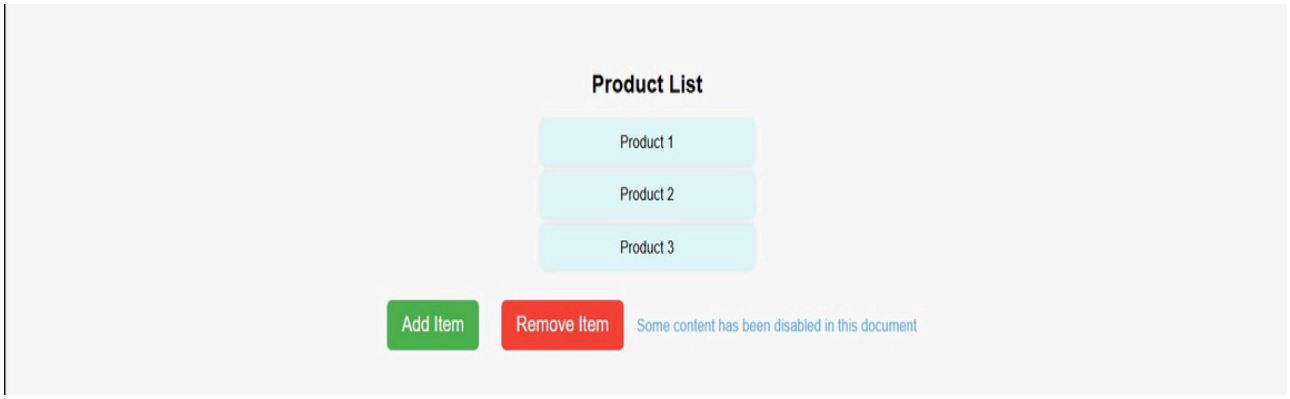
```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Dynamic List Example</title>
 <style>
```

```
/*Page styling */
body {
  font-family: Arial, sans-serif;
  text-align: center;
  margin-top: 50px;
  background-color: #f9f9f9;
}

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

li{
  background-color: #e0f7fa;
  margin: 5px auto;
  padding: 10px;
  width: 200px;
  border-radius: 5px;
  box-shadow: 0 0 5px rgba(0,0,0,0.1);
}

button {
  margin: 10px;
  padding: 10px 15px;
  font-size: 16px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

#addBtn {
  background-color: #4CAF50;
```

```
    color: white;
   }

   #removeBtn {
    background-color: #f44336;
    color: white;
   }

   button:hover {
    opacity: 0.9;
   }
  </style>
</head>
<body>

  <h2>Product List</h2>

  <!-- List of items -->
  <ul id="itemList">
   <li>Product  1</li>
   <li>Product  2</li>
   <li>Product 3</li>
  </ul>

  <!-- Buttons to add or remove items -->
  <button id="addBtn">Add Item</button>
  <button id="removeBtn">Remove Item</button>

  <script>
   // Get references to HTML elements
   const itemList = document.getElementById("itemList");
   const addBtn = document.getElementById("addBtn");
   const removeBtn = document.getElementById("removeBtn");
```

```
  //Counter to keep track of added items
  letitemCount = 3;

  //Function to add a new item dynamically
  addBtn.addEventListener("click", function() {
   itemCount++; // Increase count
   const newItem = document.createElement("li"); // Create new list item
   newItem.textContent = "Product " + itemCount; // Add text to new item
   itemList.appendChild(newItem); // Add to list
  });

  //Function to remove the last item dynamically
  removeBtn.addEventListener("click", function() {
   if(itemList.lastElementChild) { // Check if list has items
    itemList.removeChild(itemList.lastElementChild); // Remove last item
    itemCount--; // Decrease count
   }else {
    alert("No more items to remove!"); // Show alert if list empty
   }
  });
 </script>
</body>
</html>
```

OUTPUT:



EXPLANATION:

- Created an initial list (<ul>) with three product items.
- Added two buttons — Add Item and Remove Item.
- JavaScript increases the counter and adds a new <li> when Add Item is clicked.
- Remove Item deletes the last <li> element if the list is not empty.
- The list updates instantly without reloading the page.
- Clean, commented, and beginner-friendly implementation.

Task 5: **Styled Modal Popup**

Instructions:

•Use AI to generate a modal popup that opens when a button is clicked.
•Style the modal using CSS with a semi-transparent overlay.
•Include a close button that hides the modal.

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>

 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Styled Modal Popup</title>
 <style>
  /*Basic page setup */
  body {
   font-family: Arial, sans-serif;
   text-align: center;
   background-color: #f4f4f4;
   margin-top: 100px;
  }

  /*Button styling */
```

```
#openModalBtn {
  padding: 10px 20px;
  background-color: #4CAF50;
  color: white;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  font-size: 16px;
}

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

/*Modal background overlay (hidden by default) */
.modal {
  display: none; /* Hidden until triggered */ position: fixed; z-index: 1;
  /* Stay on top */ left: 0; top: 0; width: 100%; height: 100%; overflow:
  auto; background-color: rgba(0, 0, 0, 0.5); /* Semi-transparent
  overlay */









}

/*Modal content box */
.modal-content {
  background-color: #fff;
  margin: 15% auto;
  padding: 20px;
  border-radius: 10px;
```

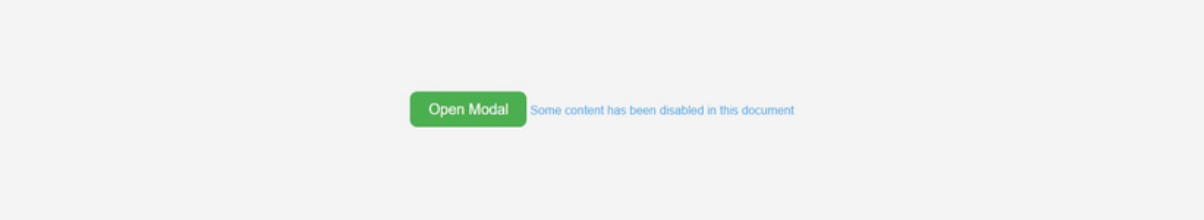
```
    width:80%;
    max-width: 400px;
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.3);
    position: relative;
    animation: fadeIn 0.3s;
   }

   /*Closebutton styling */
   .close {
    position: absolute;
    top:10px;
    right:15px;
    font-size: 22px;
    font-weight: bold;
    color:#333;
    cursor:pointer;
   }

   .close:hover {
    color:red;
   }

   /*Fade-in animation */
   @keyframes fadeIn {
    from{opacity: 0; transform: scale(0.9); }
    to{opacity: 1; transform: scale(1); }
   }
  </style>
</head>
<body>

  <!-- Button to open the modal -->
  <button id="openModalBtn">Open Modal</button>
```

```
<!--Modalstructure -->
<divid="myModal" class="modal">
 <divclass="modal-content">
  <spanclass="close">&times;</span>
  <h2>Welcome!</h2>
  <p>Thisis a styled modal popup window. Click “X” or outside the box to close it.</p>
 </div>
</div>

<script>
 //Getelements
 constmodal = document.getElementById("myModal");
 constopenBtn = document.getElementById("openModalBtn");
 constcloseBtn = document.querySelector(".close");

 //Whenthe "Open Modal" button is clicked
 openBtn.addEventListener("click", function() {
  modal.style.display = "block";
 });

 //Whenthe "X" button is clicked
 closeBtn.addEventListener("click", function() {
  modal.style.display = "none";
 });

 //Closemodal when clicking outside the content area
 window.addEventListener("click", function(event) {
  if(event.target === modal) {
   modal.style.display = "none";
  }
 });
</script>
```

</body>

</html>

OUTPUT:



EXPLANATION:

- A button labeled “Open Modal” is created.
- Modal box and semi-transparent overlay are styled using CSS.
- JavaScript shows the modal when the button is clicked.
- Clicking the “X” or outside the modal closes it.
- Smooth fade-in animation makes it visually appealing.
- Page doesn’t reload — all updates happen dynamically.