AI-ASSITED CODING

LAB:14.1

ROLLNO:2403A52084

BATCH:04

## TASK01:

PROMPT: **Design a responsive HTML/CSS web page with a header (navigation links), main section (placeholder text and image), and footer (copyright/contact info). Use AI-generated CSS to ensure clean layout and adaptability across desktop, tablet, and mobile screens.**

## CODE:

## HTML:

# CSS:

```css
/* Reset and base styles */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: Arial, sans-serif;
  background-color: #14d51e; /* Light gray background */
  color: #333; /* Dark text */
}

/* Header styles */
header {
  background-color: #2c3e50; /* Dark blue-gray */
  color: rgb(214, 22, 22);
  padding: 1rem 2rem;
}

nav {
  display: flex;
  justify-content: space-between;
  align-items: center;
  flex-wrap: wrap;
}

nav h1 {
  font-size: 1.8rem;
}

.nav-links {
  list-style: none;
  display: flex;
  gap: 1rem;
```

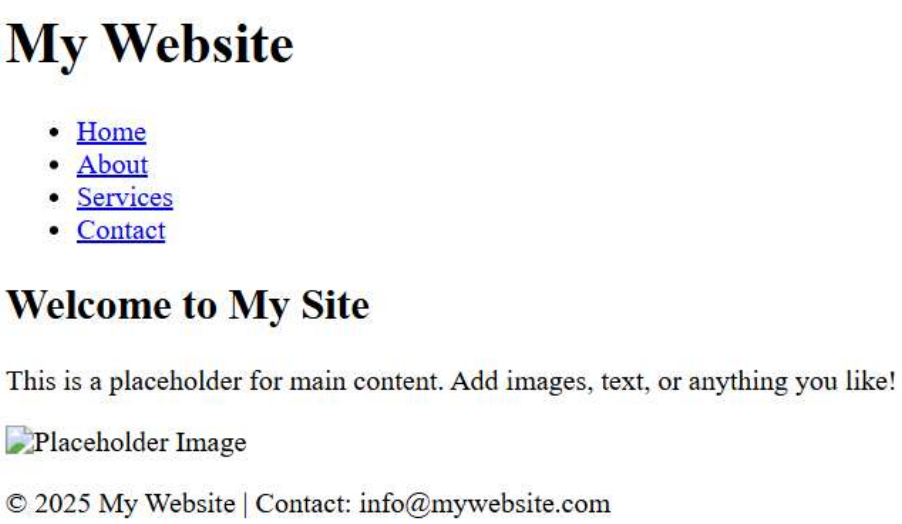
```css
.nav-links {
}

.nav-links a {
  color: #139fc2; /* Light gray text */
  text-decoration: none;
  font-weight: bold;
}

.nav-links a:hover {
  color: #00bcd4; /* Cyan on hover */
}

/* Main content styles */
main {
  padding: 2rem;
  text-align: center;
}

.hero h2 {
  font-size: 2rem;
  margin-bottom: 1rem;
  color: #2c3e50; /* Match header color */
}

.hero p {
  font-size: 1.1rem;
  margin-bottom: 1rem;
}

.hero img {
  max-width: 100%;
  height: auto;
  border-radius: 8px;
}
```

```css
/* Footer styles */
footer {
  background-color: #34495e; /* slightly lighter than header */
  color: white;
  text-align: center;
  padding: 1rem;
  font-size: 0.9rem;
}

/* Responsive styles */
@media (max-width: 768px) {
  nav {
    flex-direction: column;
    align-items: flex-start;
  }

  .nav-links {
    flex-direction: column;
    width: 100%;
  }

  .nav-links li {
    margin: 0.5rem 0;
  }

  main {
    padding: 1rem;
  }

  .hero h2 {
    font-size: 1.5rem;
  }
}
```

1 > Task1 > # index.css > {} @media (max-width: 480px)

```css
@media (max-width: 768px) {
}

@media (max-width: 480px) {
  header {
    padding: 1rem;
  }

  nav h1 {
    font-size: 1.5rem;
  }

  .hero h2 {
    font-size: 1.2rem;
  }

  .hero p {
    font-size: 1rem;
  }

  footer {
    font-size: 0.8rem;
    padding: 0.8rem;
  }
}
```

## OUTPUT:




## OBSERVATION:

The code creates a clean, responsive web page layout using HTML and CSS with a header, main content, and footer. It adapts seamlessly across desktop, tablet, and mobile screens using media queries. Navigation links and placeholder content ensure a visually organized structure.

## TASK2:

## PROMPT:

Create a simple web page with a button. When the button is clicked, use JavaScript to display an alert message saying “Button clicked!”. Make sure the code is clean, responsive, and well-commented.

## CODE:

C:\Users\NIKHITHA\OneDrive\Desktop\AI\Lab14.1

```
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Interactive Button</title>
  <style>
    /* Basic styling for the button */
    #alertButton {
      padding: 10px 20px;
      font-size: 16px;
      background-color: #4CAF50;
      color: white;
      border: none;
      border-radius: 5px;
      cursor: pointer;
    }

    #alertButton: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() {
      // Show an alert message when the button is clicked
      alert('Button clicked!');
    });
  </script>

</body>
</html>
```

## OUTPUT:


## OBSERVATION:

The code creates a simple interactive button using HTML, CSS, and JavaScript. When clicked, the button triggers a cleanly written alert function that displays “Button clicked!”—demonstrating basic DOM manipulation and event handling. Styling ensures the button is visually appealing and responsive to hover.

## TASK3:

## PROMPT:

Create a contact form with fields for Name, Email, and Message. Use JavaScript to validate that all fields are filled and the email is in a valid format. Show inline error messages for invalid input and display “Form submitted successfully” when the form is valid.

# CODE:

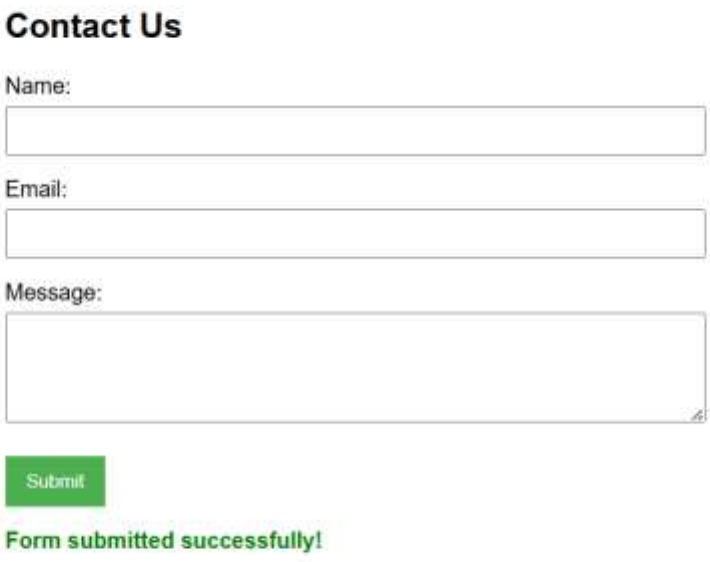
```
task3 > listbutton.html > html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Contact Form</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      padding: 20px;
      max-width: 500px;
      margin: auto;
    }

    label {
      display: block;
      margin-top: 15px;
    }

    input, textarea {
      width: 100%;
      padding: 8px;
      margin-top: 5px;
      box-sizing: border-box;
    }

    .error {
      color: red;
      font-size: 0.9em;
    }

    .success {
      color: green;
      font-weight: bold;
      margin-top: 15px;
    }

    button {
      margin-top: 20px;
      padding: 10px 15px;
      background-color: #4CAF50;
      color: white;
```

```
      border: none;
      cursor: pointer;
    }

    button:hover {
      background-color: #45a049;
    }
  </style>
</head>
<body>

  <h2>Contact Us</h2>
  <form id="contactForm">
    <label>Name:
      <input type="text" id="name" />
      <span class="error" id="nameError"></span>
    </label>

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

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

    <button type="submit">Submit</button>
    <div class="success" id="successMessage"></div>
  </form>

  <script>
    // Form validation logic
    document.getElementById('contactForm').addEventListener('submit', function(e) {
      e.preventDefault(); // Prevent form submission

      // Clear previous messages
```

```
      // Clear previous messages
      document.getElementById('nameError').textContent = '';
      document.getElementById('emailError').textContent = '';
      document.getElementById('messageError').textContent = '';
      document.getElementById('successMessage').textContent = '';

      // Get input values
      const name = document.getElementById('name').value.trim();
      const email = document.getElementById('email').value.trim();
      const message = document.getElementById('message').value.trim();

      let isValid = true;

      // Validate Name
      if (name === '') {
        document.getElementById('nameError').textContent = 'Name is required.';
        isValid = false;
      }

      // Validate Email
      const emailPattern = /^[^@\s]+@[^@\s]+\.[^@\s]+$/;
      if (email === '') {
        document.getElementById('emailError').textContent = 'Email is required.';
        isValid = false;
      } else if (!emailPattern.test(email)) {
        document.getElementById('emailError').textContent = 'Enter a valid email.';
        isValid = false;
      }

      // Validate Message
      if (message === '') {
        document.getElementById('messageError').textContent = 'Message is required.';
        isValid = false;
      }

      // Show success message if valid
      if (isValid) {
```

```
      isValid = false;
    } else if (!emailPattern.test(email)) {
      document.getElementById('emailError').textContent = 'Enter a valid email.';
      isValid = false;
    }

    // Validate Message
    if (message === '') {
      document.getElementById('messageError').textContent = 'Message is required.';
      isValid = false;
    }

    // Show success message if valid
    if (isValid) {
      document.getElementById('successMessage').textContent = 'Form submitted successfully!'
      document.getElementById('contactForm').reset();
    }
  });
</script>

</body>
</html>
```

# OUTPUT:



## OBSERVATION:

The code builds a contact form with client-side validation using JavaScript. It checks for empty fields and a valid email format, displaying inline error messages when inputs are invalid. On successful validation, it shows a confirmation message and resets the form.

## TASK4:

## PROMPT:

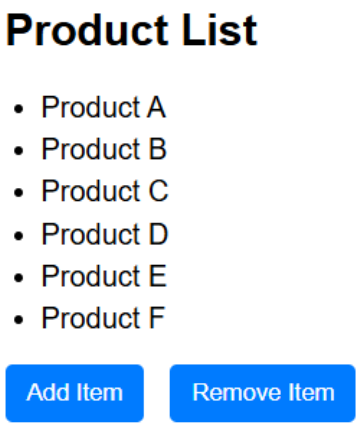
Create a web page with a list of items and two buttons: “Add Item” and “Remove Item”. Use JavaScript to dynamically add a new item to the list or remove the last item when the respective button is clicked. Ensure updates happen without reloading the page.

## CODE:

```
> task4 > list.html > html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Dynamic List</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      padding: 20px;
      max-width: 500px;
      margin: auto;
    }

    ul {
      padding-left: 20px;
    }

    li {
      margin-bottom: 5px;
    }

    button {
      margin-right: 10px;
      padding: 8px 12px;
      background-color: #007BFF;
      color: white;
      border: none;
      border-radius: 4px;
      cursor: pointer;
    }

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

  <h2>Product List</h2>
  <ul id="productList">
    <li>Product A</li>
```

```
<ul id="productList">
    <li>Product B</li>
    <li>Product C</li>
</ul>

<button onclick="addItem()">Add Item</button>
<button onclick="removeItem()">Remove Item</button>

<script>
    let itemCount = 3; // Initial number of items

    // Function to add a new item
    function addItem() {
        itemCount++;
        const ul = document.getElementById('productList');
        const li = document.createElement('li');
        li.textContent = `Product ${String.fromCharCode(64 + itemCount)}`;
        ul.appendChild(li);
    }

    // Function to remove the last item
    function removeItem() {
        const ul = document.getElementById('productList');
        if (ul.lastElementChild) {
            ul.removeChild(ul.lastElementChild);
            itemCount--;
        }
    }
</script>

</body>
</html>
```

# OUTPUT:




# OBSERVATION:

The code dynamically manages a product list using JavaScript, allowing users to add or remove items without

reloading the page. It uses DOM manipulation to append new

 elements and remove the last one, demonstrating real-time interactivity and clean event handling.

## TASK5:

## PROMPT:

Create a web page with a button labeled “Open Modal”. When clicked, it should display a centered modal popup with content. Style the modal with a semi-transparent overlay and include a close “X” button. The modal should close when the “X” or overlay area is clicked.

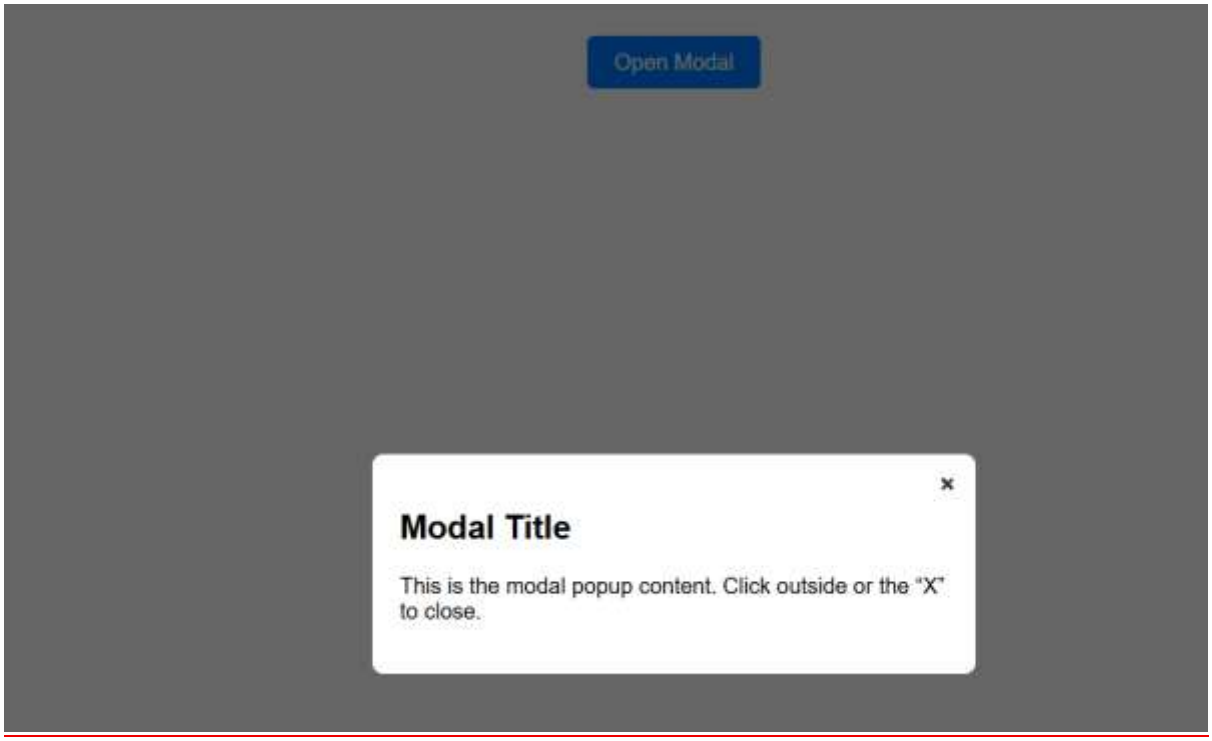
## CODE:

```html
<!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>
    body {
      font-family: Arial, sans-serif;
      padding: 20px;
      text-align: center;
    }

    /* Button styling */
    #openModalBtn {
      padding: 10px 20px;
      font-size: 16px;
      background-color: #007BFF;
      color: white;
      border: none;
      border-radius: 5px;
      cursor: pointer;
    }

    #openModalBtn:hover {
      background-color: #0056b3;
    }

    /* Modal overlay */
    .modal-overlay {
      display: none;
      position: fixed;
      top: 0;
      left: 0;
      width: 100%;
      height: 100%;
      background: rgba(0, 0, 0, 0.6);
      justify-content: center;
      align-items: center;
      z-index: 1000;
    }
```

```
    /* Modal content */
    .modal-content {
      background: white;
      padding: 20px;
      border-radius: 8px;
      position: relative;
      width: 80%;
      max-width: 400px;
      text-align: left;
    }

    /* Close button */
    .close-btn {
      position: absolute;
      top: 10px;
      right: 15px;
      font-size: 20px;
      font-weight: bold;
      color: #333;
      cursor: pointer;
    }

    .close-btn:hover {
      color: red;
    }
  </style>
</head>
<body>

  <button id="openModalBtn">Open Modal</button>

  <!-- Modal Structure -->
  <div class="modal-overlay" id="modalOverlay">
    <div class="modal-content">
      <span class="close-btn" id="closeBtn">&times;</span>
      <h2>Modal Title</h2>
      <p>This is the modal popup content. Click outside or the “X” to close.</p>
    </div>
```

```
</div>

<script>
  const openModalBtn = document.getElementById('openModalBtn');
  const modalOverlay = document.getElementById('modalOverlay');
  const closeBtn = document.getElementById('closeBtn');

  // Show modal
  openModalBtn.addEventListener('click', () => {
    modalOverlay.style.display = 'flex';
  });

  // Hide modal on close button click
  closeBtn.addEventListener('click', () => {
    modalOverlay.style.display = 'none';
  });

  // Hide modal on overlay click
  modalOverlay.addEventListener('click', (e) => {
    if (e.target === modalOverlay) {
      modalOverlay.style.display = 'none';
    }
  });
</script>

/body>
/html>
```

# OUTPUT:




## OBSERVATION:

The code implements a styled modal popup that appears when a button is clicked and disappears when the user clicks the close “X” or the overlay. It uses CSS for a semi-transparent background and JavaScript for toggling visibility, ensuring a smooth and user-friendly interaction.