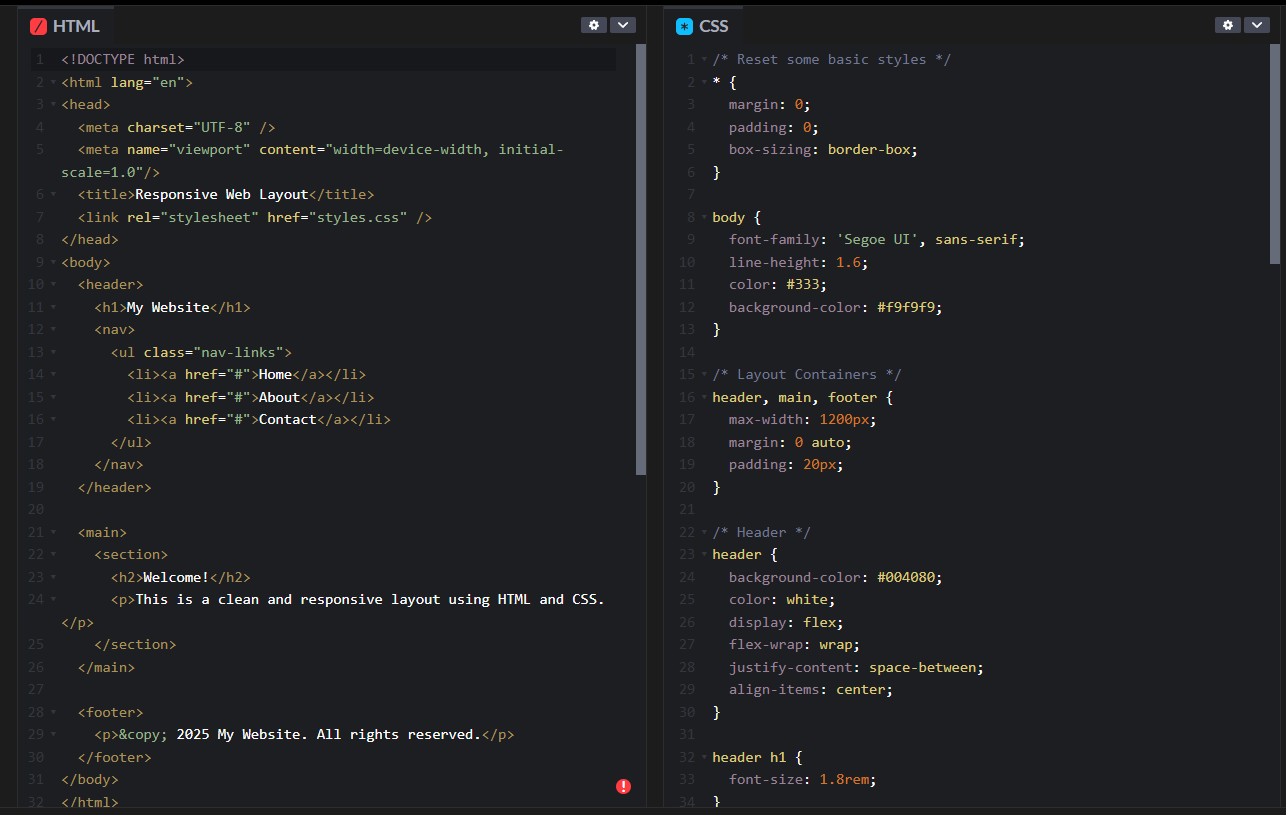
# LAB ASSIGNMENT-14

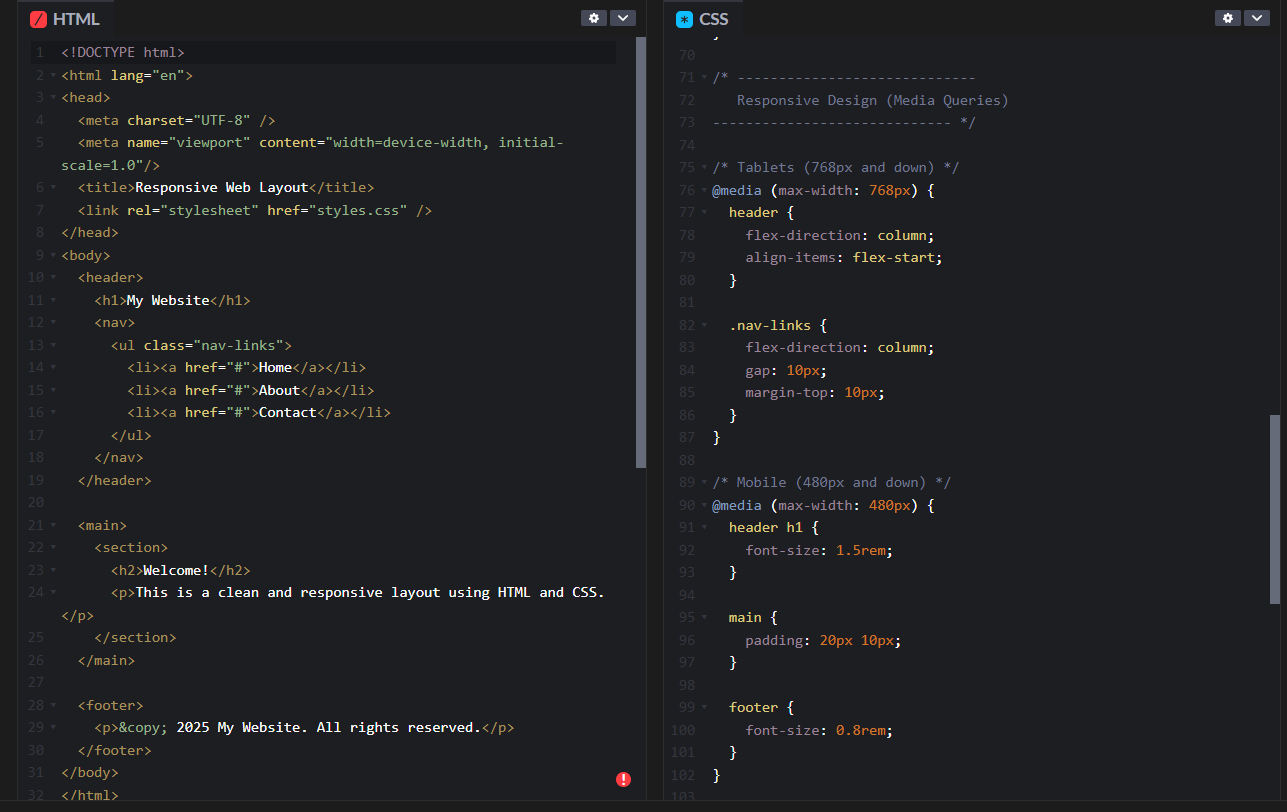
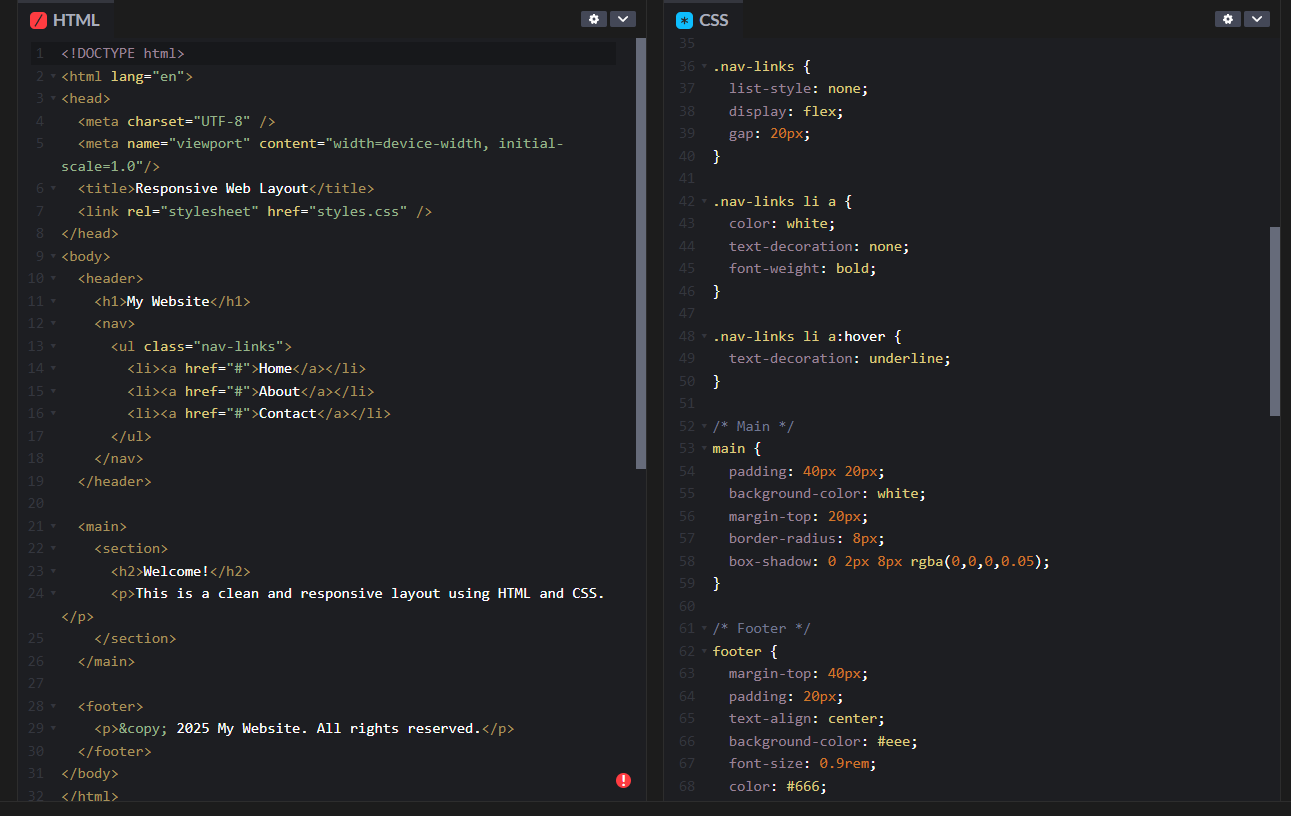
TASK-1:

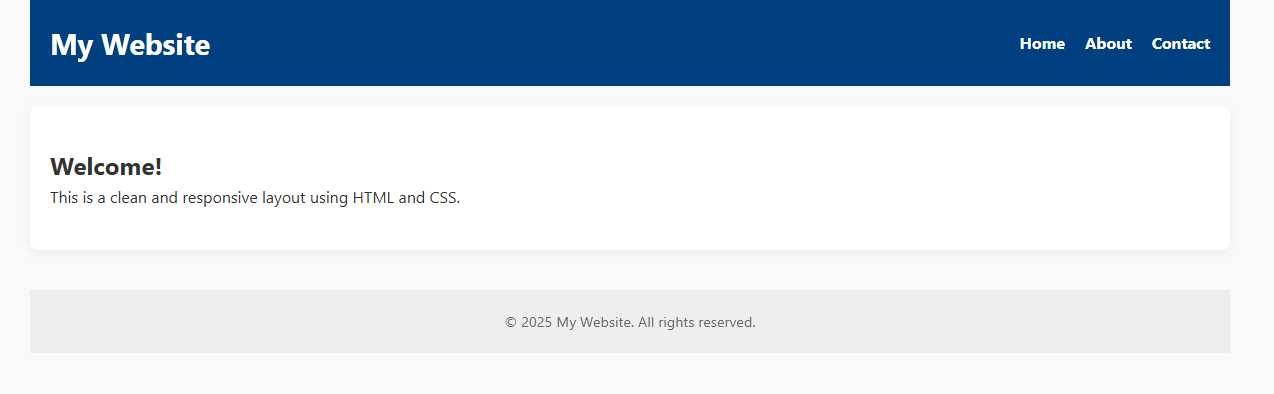
Prompt:

Design a basic web page layout with a header, main content area, and footer using HTML and CSS. Generate responsive CSS for different screen sizes. Ensure the layout is clean and visually organized.

Code and Output:







Code Explanation:

**HTML:**

* Uses semantic tags: <header>, <main>, and <footer>.
* <header> includes site title and navigation menu.
* <main> contains the page content inside a <section>.
* <footer> shows copyright.
* .container class centers and limits the width of content.

**CSS:**

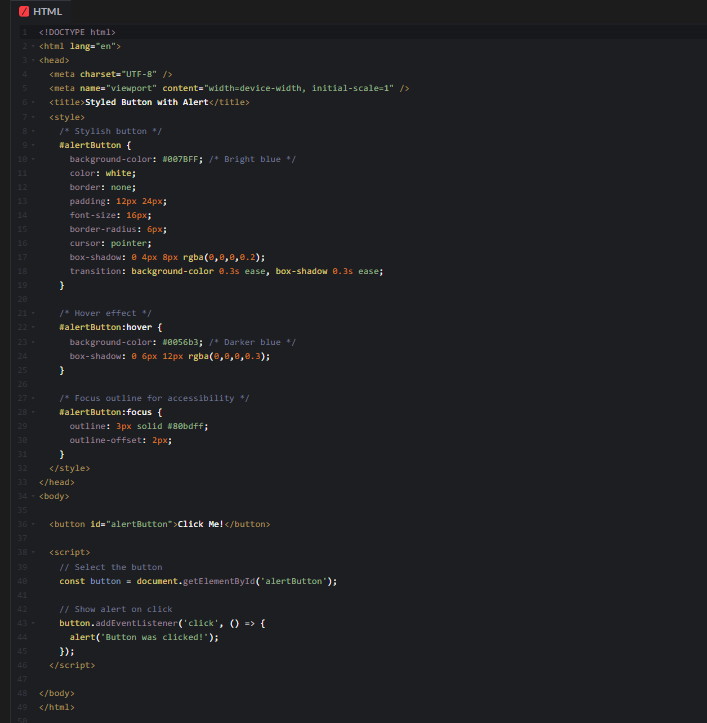
* Resets default margins/padding and applies a clean font.
* Uses Flexbox for horizontal nav layout.
* Adds spacing and background colors for visual clarity.
* Responsive via media queries:
  + Below 768px: nav items stack, text centers.
  + Below 480px: smaller font, tighter spacing.

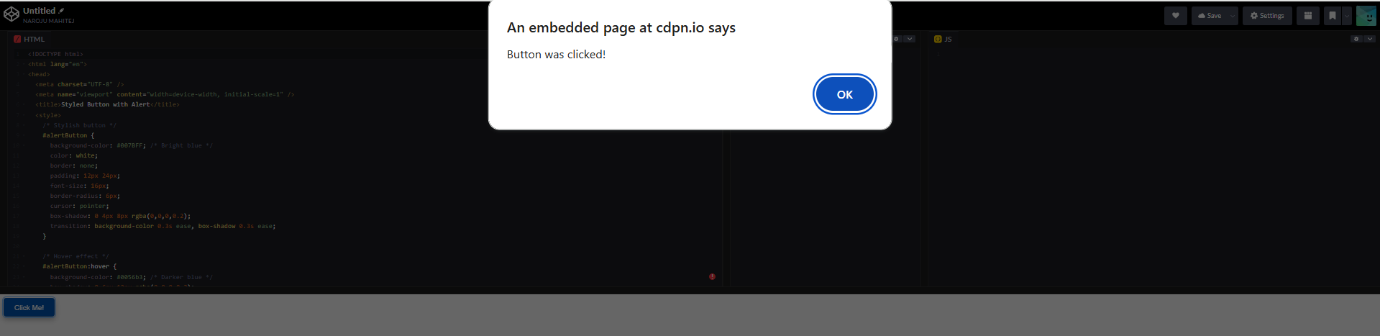
TASK-2:

Prompt:

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

Code and Output:





Code Explanation:

**HTML:**

* A single <button> element with id="alertButton" so we can target it with CSS and JS.

**CSS (inside <style>):**

* **Button base style** (#alertButton):
  + Blue background (#007BFF), white text.
  + No border, padding for size, and rounded corners (border-radius: 6px).
  + Cursor changes to pointer on hover to indicate it’s clickable.
  + Box shadow adds subtle depth.
  + Transition smooths color and shadow changes.
* **Hover state** (#alertButton:hover):
  + Darker blue background (#0056b3).
  + Larger, more pronounced shadow for a “lifted” effect.
* **Focus state** (#alertButton:focus):
  + Visible outline to improve keyboard accessibility, making it clear which element is focused.

**JavaScript (inside <script>):**

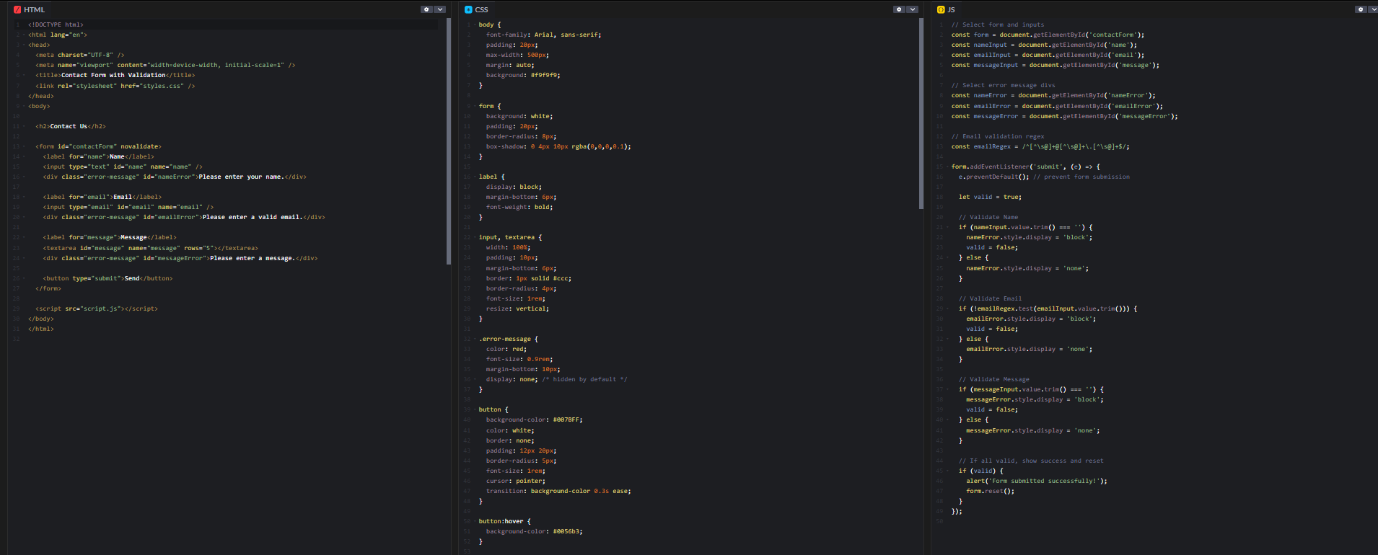
* Selects the button with getElementById.
* Adds a click event listener.
* When clicked, triggers an alert with the message: **"Button was clicked!"**

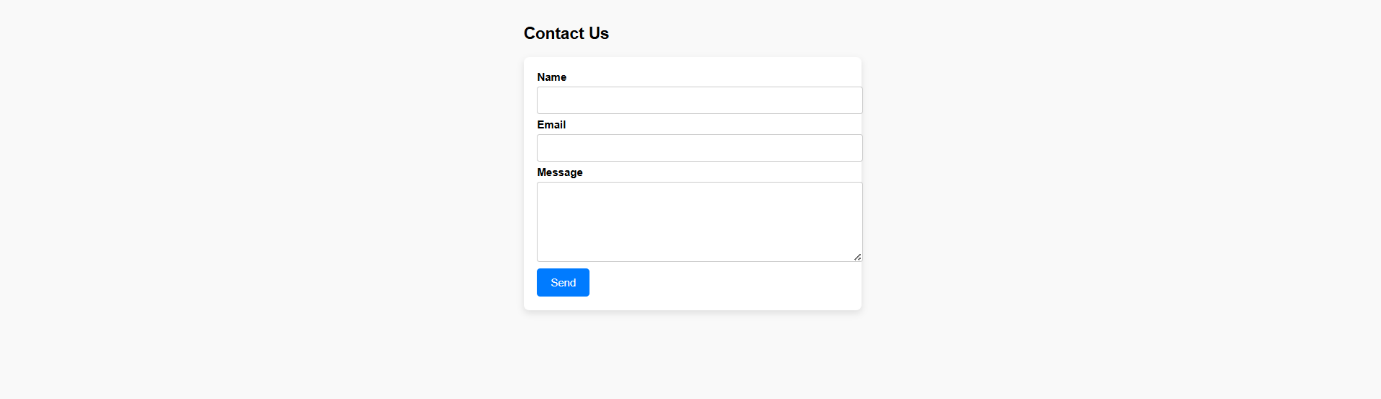
TASK-3:

Prompt:

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

Code and Output:





Code Explanation:

**1. HTML (index.html)**

* Defines the page structure with a simple **contact form**.
* Form fields: **Name**, **Email**, and **Message**.
* Each input has an associated hidden **error message div** shown only if validation fails.
* Includes <link> to CSS and <script> to JavaScript files.

**2. CSS (styles.css)**

* Styles the page for clean, readable layout.
* Sets font, spacing, and container width for a centered form.
* Inputs and textarea styled with padding, border, and rounded corners.
* Error messages styled in red and initially hidden (display: none).
* Button styled with blue background, white text, rounded corners, and hover effect.

**3. JavaScript (script.js)**

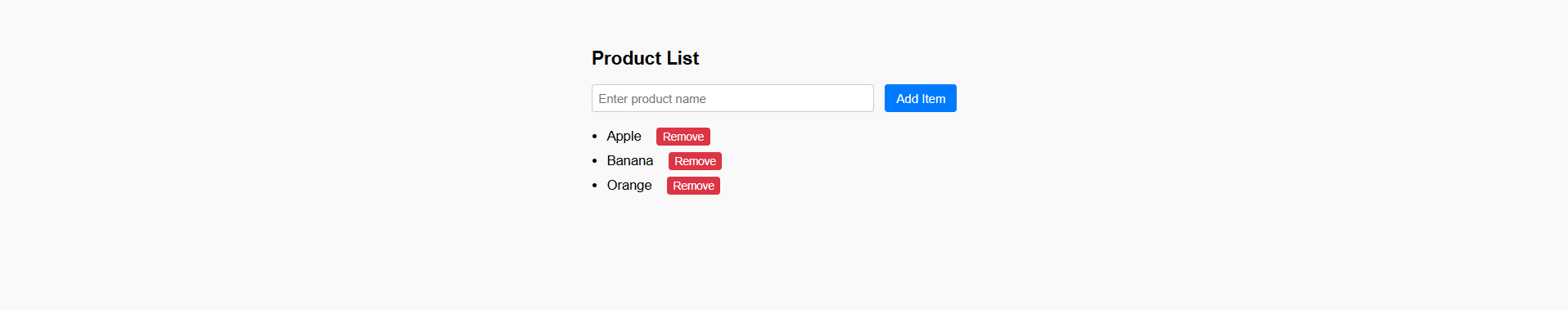
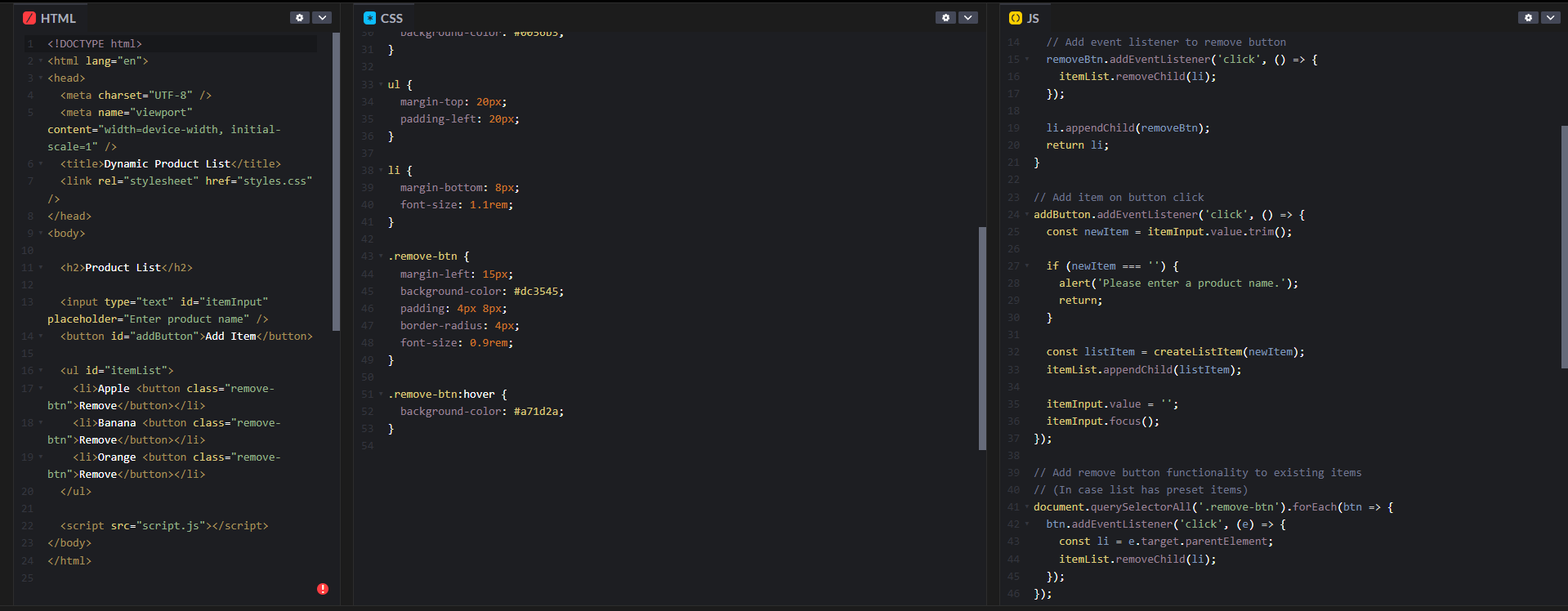
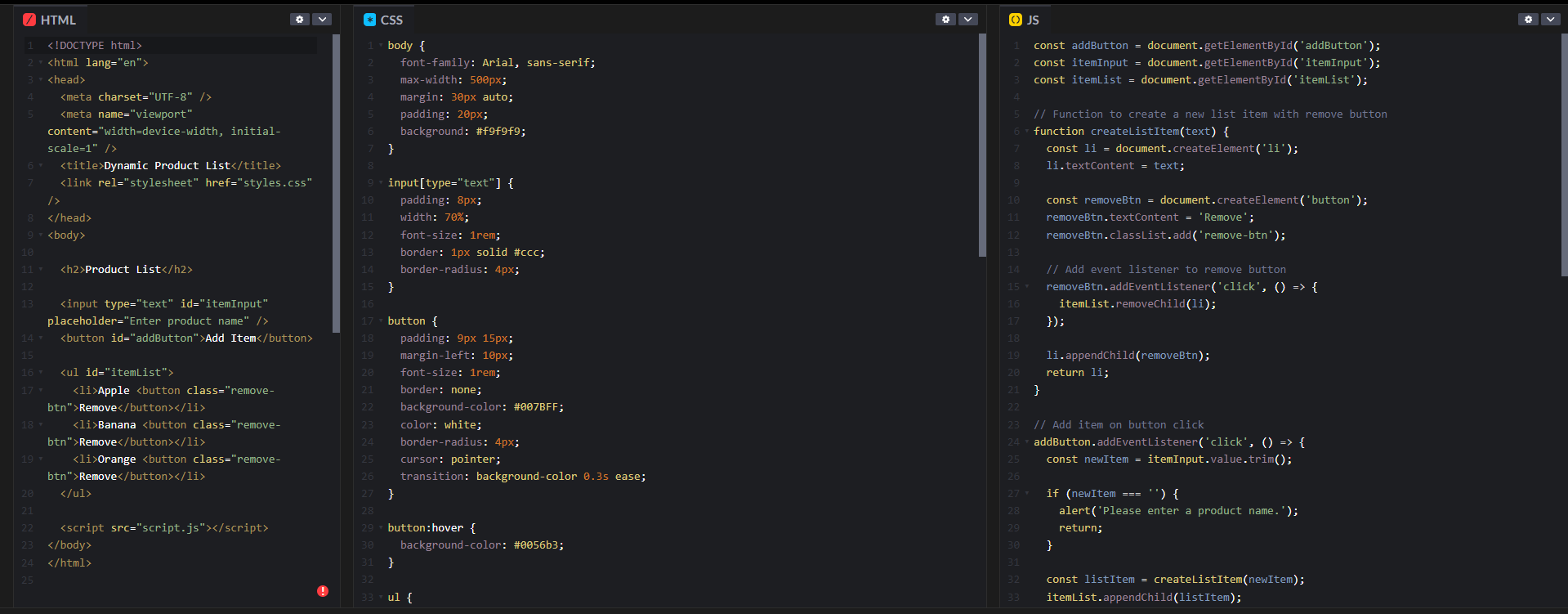
* Selects form, inputs, and error message elements.
* Defines a regular expression to validate email format.
* Listens for form submission (submit event).
* On submit:
  + Prevents default page reload.
  + Checks each field:
    - If empty or invalid email, shows respective error message.
    - Otherwise hides the error.
  + If all fields valid, displays a success alert and resets the form.

TASK-4:

Prompt:

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

Code and Output:



Code Explanation:

**1. HTML (index.html):**

* Creates the page structure:
  + An input box to type a product name.
  + An **Add Item** button to add new products.
  + An unordered list (<ul>) with some initial product items.
  + Each product has a **Remove** button next to it.
* Links to the external CSS (styles.css) for styling.
* Links to the external JavaScript (script.js) for interactivity.

**2. CSS (styles.css):**

* Styles the page with:
  + Clean fonts and centered layout.
  + Styled input and buttons with padding, colors, rounded corners.
  + Hover effects on buttons to improve UX.
  + Styled the list and its items.
  + Special styles for **Remove** buttons (red color, smaller size).

**3. JavaScript (script.js):**

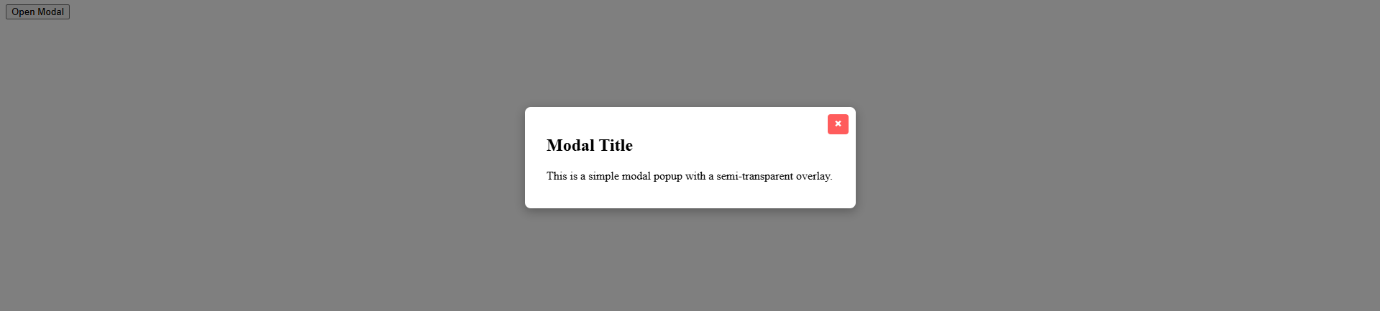
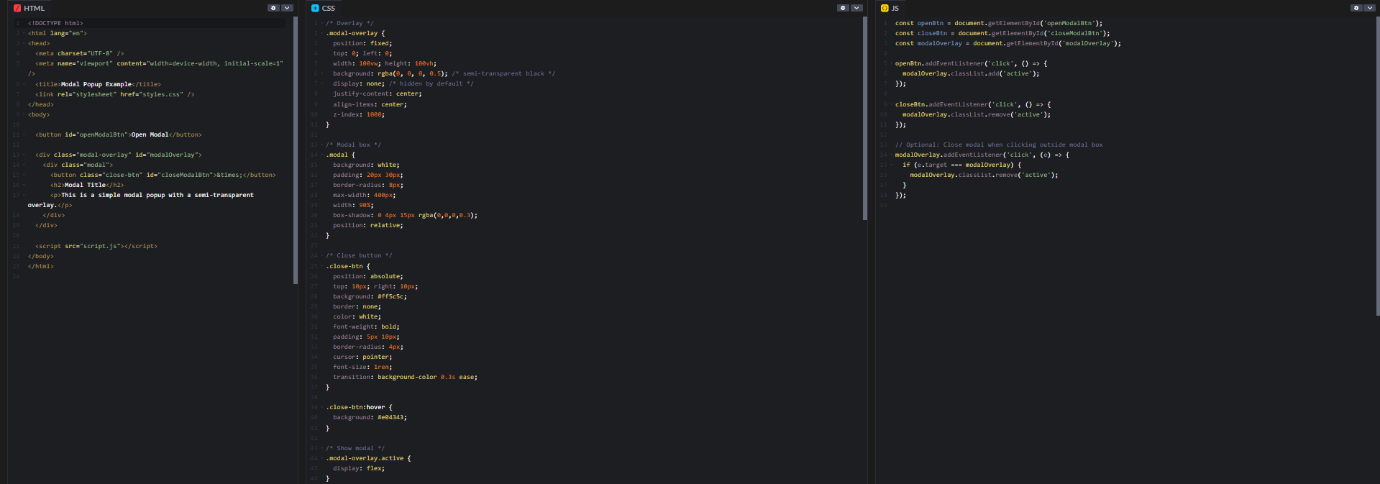
* Selects important DOM elements (addButton, itemInput, itemList).
* Defines a function createListItem(text):
  + Creates a new <li> element with product text.
  + Adds a **Remove** button to it.
  + The remove button deletes the item when clicked.
* Adds a click event to the **Add Item** button:
  + Takes input value, trims whitespace.
  + Alerts if input is empty.
  + Otherwise creates and adds a new list item.
  + Clears and focuses the input for the next entry.
* Adds event listeners to **Remove** buttons of initial list items so they work on page load.

TASK-5:

Prompt:

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 and Output:



Code Explanation:

**1. HTML (index.html)**

* Contains a button labeled **Open Modal**.
* Defines a hidden modal overlay (div.modal-overlay) covering the entire viewport.
* Inside the overlay is the modal box with content and a **close button** (×).
* Links the CSS and JavaScript files.

**2. CSS (styles.css)**

* .modal-overlay:
  + Covers the whole screen with a semi-transparent black background (rgba(0,0,0,0.5)).
  + Hidden by default (display: none).
  + Uses flexbox to center the modal content when visible.
* .modal:
  + White box with padding, rounded corners, and a subtle shadow.
  + Positioned relative to place the close button inside it.
* .close-btn:
  + Positioned top-right inside the modal.
  + Red button with hover effect for better UX.
* .modal-overlay.active:
  + When .active class is added, the overlay becomes visible (display: flex).

**3. JavaScript (script.js)**

* Selects the open button, close button, and modal overlay elements.
* Adds a click event to the **Open Modal** button to add the .active class, showing the modal.
* Adds a click event to the **close button** to remove the .active class, hiding the modal.
* Adds a click event on the overlay itself — if you click outside the modal box (the overlay background), it also hides the modal by removing .active.