# Lab - Week 1\_A

PART A

(PART A: TO BE REFERRED BY STUDENTS)

# HTML Concepts

## Document Structure

* **<!DOCTYPE html>**: Declares the document type as HTML5.
* **<html>**: The root element of the HTML document.
* **<head>**: Contains metadata, links to stylesheets, and other head elements.
* **<body>**: Contains the visible page content.

## <meta> Tags:

* + <meta charset="UTF-8">: Specifies character encoding.
  + <meta name="viewport" content="width=device-width, initial-scale=1.0">:

Ensures responsive scaling for mobile devices.

## Forms and Input Fields

* **<form>**: Groups form elements for user input.
* **<input>**: Defines input types (text, number, email, etc.).
* **<select> & <option>**: Dropdown menus.
* **<button>**: Clickable elements, often for submitting forms.

## Semantic Elements

* **<header>**: Defines introductory content or navigation.
* **<main>**: Represents the primary content area.
* **<footer>**: Contains footer information.
* **<label>**: Links text labels to form inputs using the for attribute.

## Text and Containers

* **<p>**: Paragraphs.
* **<h1> to <h6>**: Hierarchical headings.
* **<div>**: Block container for layout.
* **<span>**: Inline container for grouping.

## Links and Resources

* **<link>**: Links external resources like CSS.
* **<script>**: Embeds or links JavaScript files.

# CSS Concepts

## Selectors

* **Element Selector**: Targets elements by name (e.g., h1 {}).
* **Class Selector**: Targets elements by class name (.class {}).
* **ID Selector**: Targets a specific element (#id {}).
* **Combinators**: Select elements based on their relationships:
  + Descendant: div p
  + Child: div > p
  + Sibling: div + p, div ~ p.

## Box Model

* Components: margin, border, padding, content.
* Defines spacing and element boundaries.

## Positioning and Layout

* **position**: Controls element placement (static, relative, absolute, fixed, sticky).
* **display**: Defines layout type (block, inline, flex, grid).
* Flexbox/Grid: For advanced responsive layouts.

## Typography

* **Font Settings**: font-family, font-size, font-weight.
* **Spacing**: line-height, letter-spacing.

## Colors and Backgrounds

* **Colors**: color, background-color.
* **Backgrounds**: background-image, gradients (linear-gradient, radial-gradient).

## Borders and Shadows

* **Borders**: border, border-radius.
* **Shadows**: box-shadow.

## Responsive Design

* **Media Queries**: Adapt styles for different screen sizes (@media).
* Flexible units: %, em, rem, vh, vw.

## Transitions and Animations

* **Transitions**: Smooth style changes (transition).
* **Animations**: Define keyframes (@keyframes).

# JavaScript Concepts

## DOM Manipulation

* **Selecting Elements**: document.getElementById, document.querySelector.
* **Updating Content**: .innerHTML, .textContent, .value.
* **Adding/Removing Elements**: .appendChild, .removeChild.
* **Handling Classes**: .classList.add, .classList.toggle.

## Events

* **Add Listeners**: addEventListener.
* **Common Events**: click, submit, input.
* **Event Properties**: event.target, event.preventDefault.

## Data Types and Variables

* Primitives: string, number, boolean.
* Reference Types: array, object.
* Declarations: let, const, var.

## Control Structures

* Conditional Statements: if, else, else if.
* Loops: for, while, do-while, forEach.

## Functions

* **Declarations**: function funcName() {}.
* **Arrow Functions**: const func = () => {}.

## Parameters and Return Values.

**String Manipulation**

* Methods: .toUpperCase(), .trim(), .replace().
* Template Literals: `${variable}`.

# Code Snippets:

## Document Structure

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>HTML Document</title>

</head>

<body>

<h1>Welcome to My Website</h1>

<p>This is a sample webpage.</p>

</body>

</html>

## Forms and Input Fields

Forms are used to collect user input with various input types like text, numbers, and dropdowns.

<form action="/submit" method="post">

<label for="name">Name:</label>

<input type="text" id="name" name="name" placeholder="Enter your name">

<label for="age">Age:</label>

<input type="number" id="age" name="age" placeholder="Enter your age">

<label for="gender">Gender:</label>

<select id="gender" name="gender">

<option value="male">Male</option>

<option value="female">Female</option>

</select>

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

</form>

## Semantic Elements

These provide meaningful tags that improve accessibility and SEO.

<header>

<h1>Website Header</h1>

<nav>

<a href="#home">Home</a>

<a href="#about">About</a>

</nav>

</header>

<main>

<p>Main content goes here.</p>

</main>

<footer>

<p>&copy; 2024 My Website</p>

</footer>

## CSS Concepts Selectors

Selectors target specific HTML elements for styling.

/\* Element selector \*/ h1 {

color: blue;

}

/\* Class selector \*/

.container {

background-color: lightgray; padding: 10px;

}

/\* ID selector \*/ #unique {

font-size: 20px;

}

HTML for this CSS:

<div id="unique" class="container">

<h1>Hello World</h1>

</div>

## Box Model

Every element consists of margin, border, padding, and content.

.box {

margin: 20px; padding: 15px;

border: 2px solid black; width: 200px;

height: 100px;

background-color: lightblue;

}

HTML for this CSS:

<div class="box">Box Model Example</div>

## Flexbox

Flexbox simplifies layout creation with properties for alignment and spacing.

.flex-container { display: flex;

justify-content: space-around; align-items: center;

}

.flex-item {

background-color: lightcoral; padding: 20px;

margin: 10px; width: 100px; text-align: center;

}

HTML for this CSS:

<div class="flex-container">

<div class="flex-item">1</div>

<div class="flex-item">2</div>

<div class="flex-item">3</div>

</div>

# JavaScript Concepts

DOM Manipulation

JavaScript can dynamically modify the structure and content of a webpage.

// Select an element and change its content document.querySelector('h1').textContent = 'Updated Heading';

HTML for this script:

<h1>Original Heading</h1>

<script src="script.js"></script>

## Events

Events enable interaction by responding to user actions.

document.querySelector('button').addEventListener('click', () => { alert('Button clicked!');

});

HTML for this script:

<button>Click Me</button>

<script src="script.js"></script>

## APIs and Fetch

Use the fetch function to retrieve data from a server or API. fetch('https://jsonplaceholder.typicode.com/posts')

.then(response => response.json())

.then(data => console.log(data))

.catch(error => console.error('Error:', error));

## Responsive Design

Use media queries to adapt layouts for different screen sizes. @media (max-width: 600px) {

.responsive-box {

background-color: lightgreen;

}

}

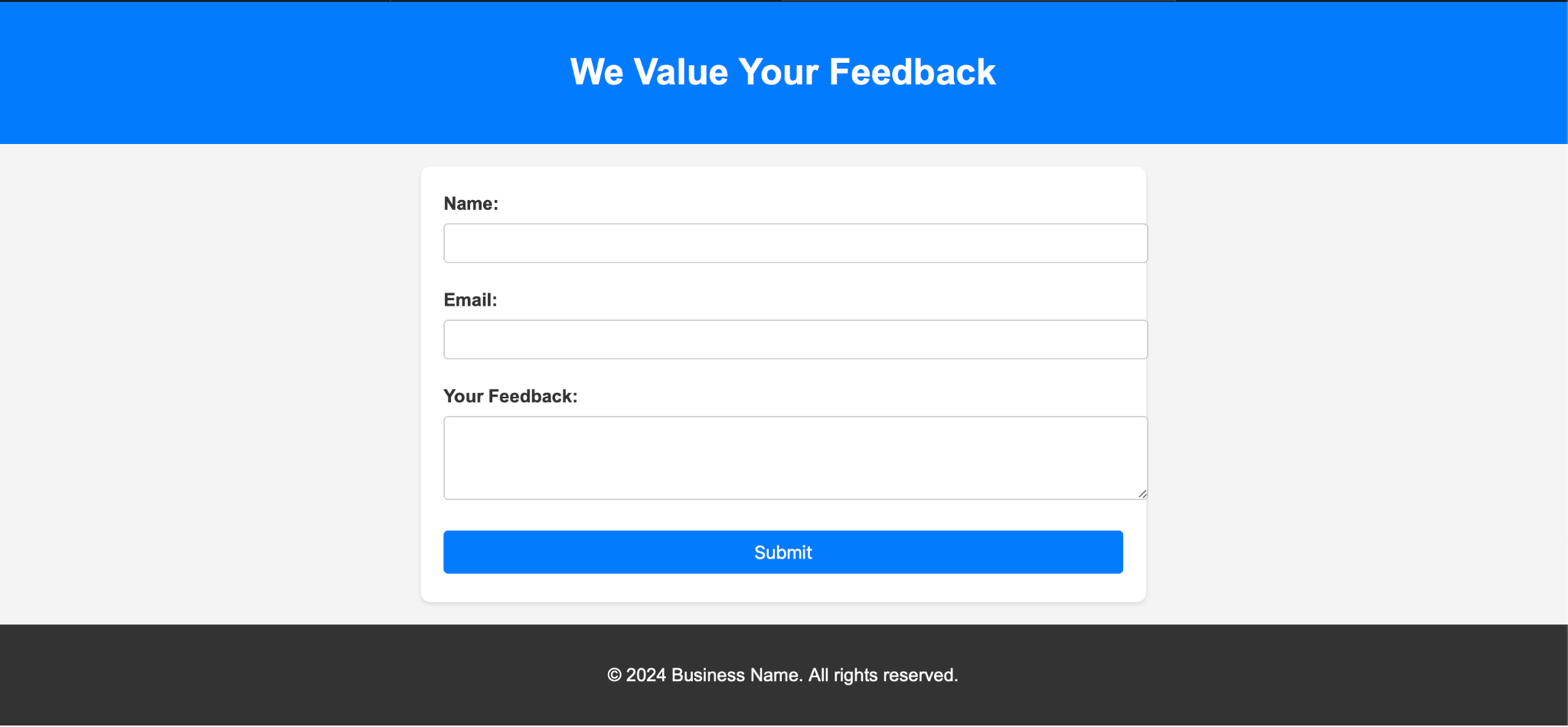
HTML for this CSS:

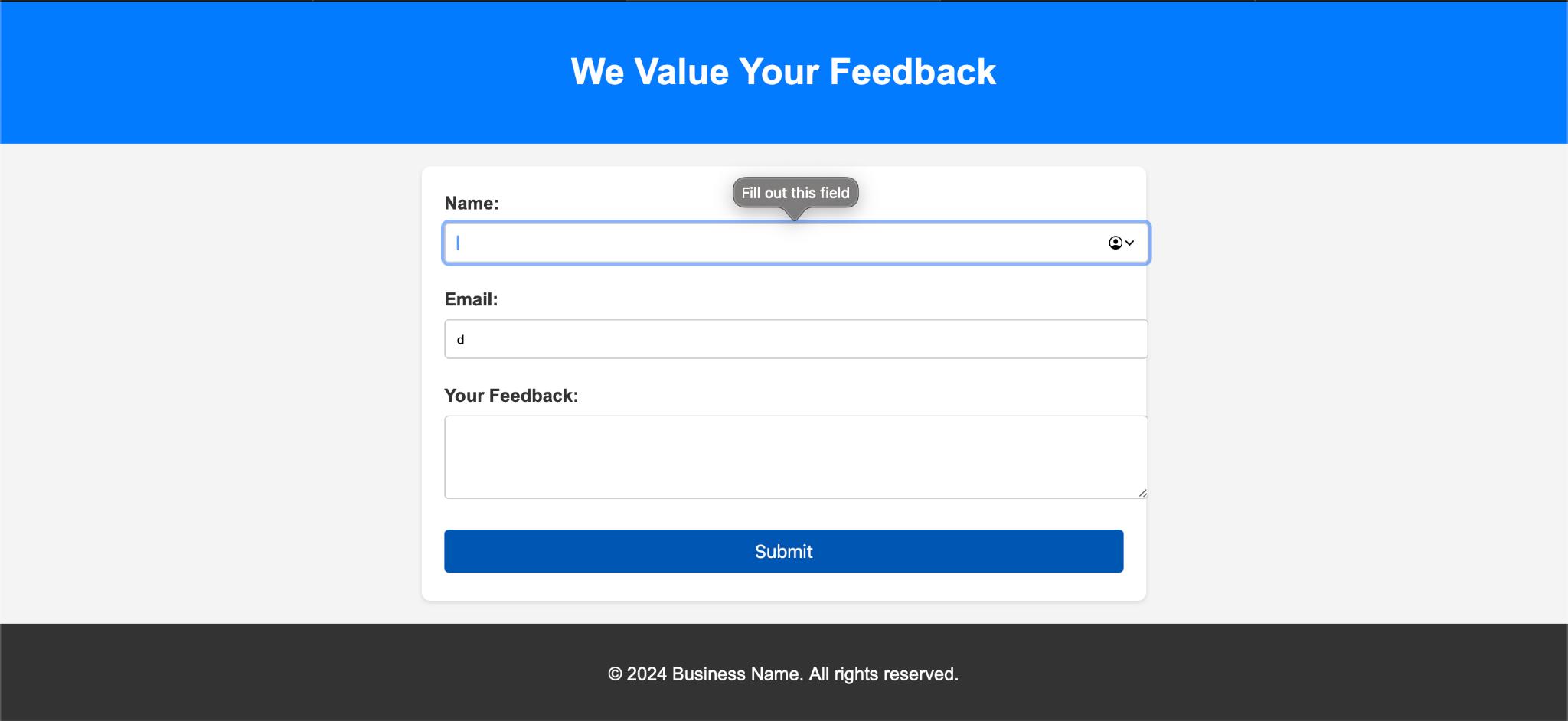
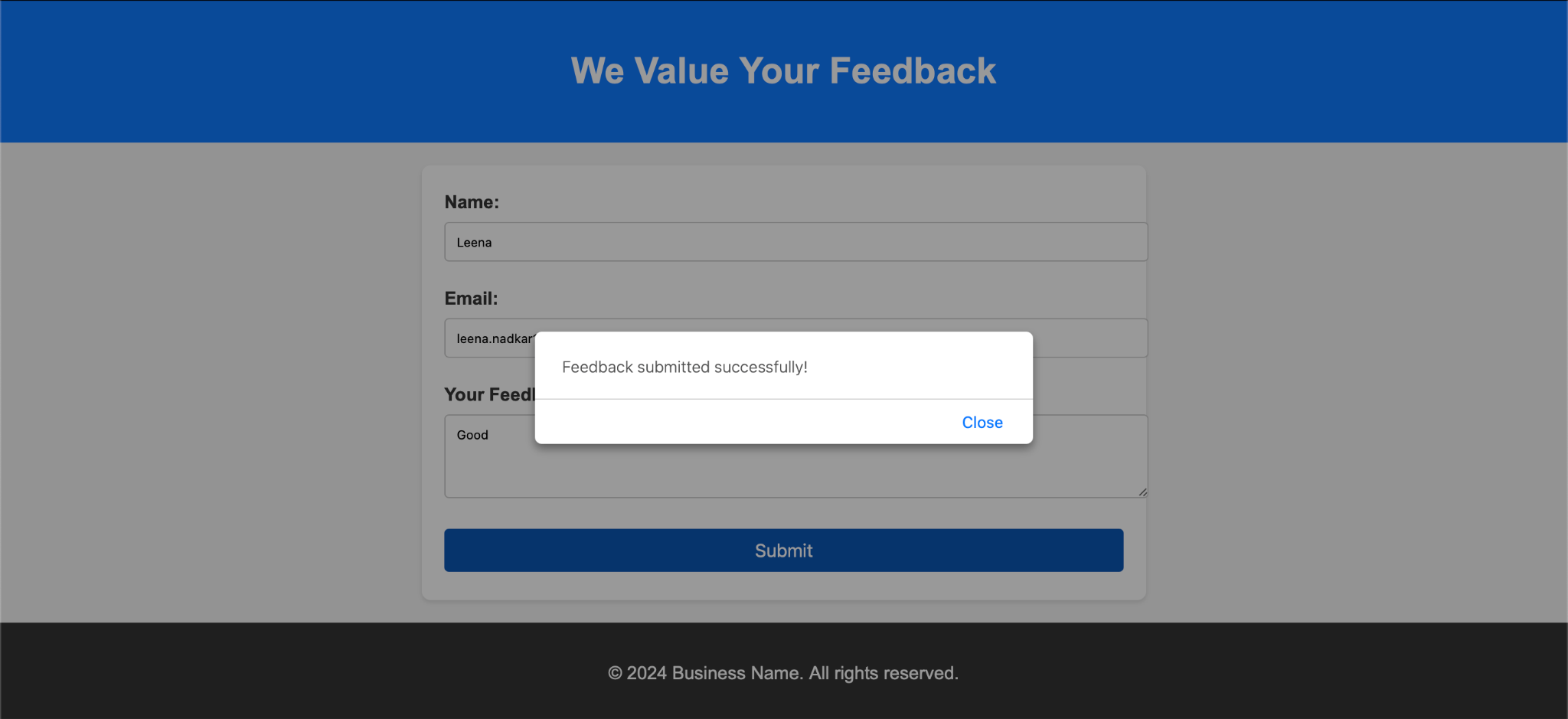
<div class="responsive-box">Resize the browser to see the effect.</div>

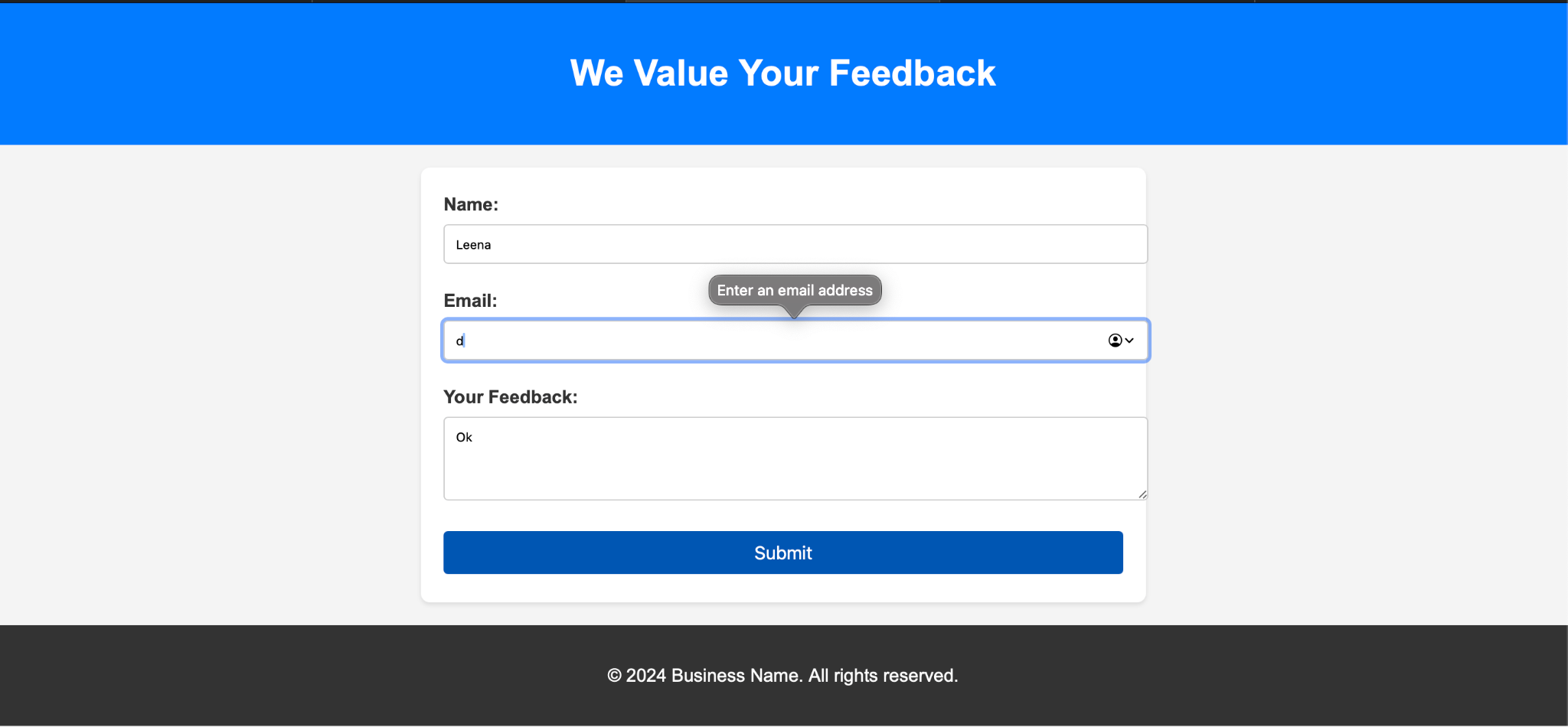
PART B

|  |  |
| --- | --- |
| **Roll No.: A073** | **Name: Aryan Srivastava** |
| **Prog/Yr/Sem: MCA/FY/sem II** | **Batch: B3** |
| **Date of Experiment: 02-01-2025** | **Date of Submission: 02-01-2025** |

## Task 1:

Imagine a business that wants to collect customer feedback through a simple web-based form. The webpage will include a form where users can enter their name, email, feedback message, and submit the information. It also provides real-time validation to ensure the form is properly filled out.



Requirements:

1. **Responsive Design**: The form adapts to different screen sizes.
2. **Real-Time Validation**: Ensures that all fields are properly filled before submission.
3. **User Feedback**: Alerts the user upon successful submission.

**CODE**

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Feedback Form</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <div class="header">

        <h1>We Value Your Feedback</h1>

    </div>

    <div class="form-container">

        <form id="feedbackForm">

            <label>Name:</label>

            <input type="text" id="name" name="name">

            <span class="error" id="nameError"></span>

            <label>Email:</label>

            <input type="email" id="email" name="email">

            <span class="error" id="emailError"></span>

            <label>Your Feedback:</label>

            <textarea id="feedback" name="feedback" rows="4"></textarea>

            <span class="error" id="feedbackError"></span>

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

        </form>

    </div>

    <div class="modal" id="successModal">

        <div class="modal-content">

            <span class="close-button" id="closeModal"></span>

            <h2>Thank You!</h2>

            <p>Your feedback has been submitted successfully.</p>

        </div>

    </div>

    <footer>

        <p>2024 Business Name. All rights reserved.</p>

    </footer>

    <script src="script.js"></script>

</body>

</html>

**Styles.css**

body {

    margin: 0;

    font-family: Arial, sans-serif;

    background-color: #f2f2f2;

  }

  .header {

    background-color: #007bff;

    color: white;

    text-align: center;

    padding: 20px;

  }

  .form-container {

    max-width: 600px;

    margin: 40px auto;

    padding: 20px;

    background-color: white;

    border-radius: 8px;

  }

  form {

    display: flex;

    flex-direction: column;

  }

  label {

    margin-bottom: 8px;

    font-weight: bold;

  }

  input, textarea {

    margin-bottom: 20px;

    padding: 10px;

    font-size: 16px;

    border: 1px solid #ccc;

    border-radius: 4px;

  }

  button {

    padding: 10px;

    font-size: 16px;

    color: white;

    background-color: #007bff;

    border: none;

    border-radius: 4px;

  }

  button:hover {

    background-color: #0056b3;

  }

  footer {

    text-align: center;

    padding: 10px;

    background-color: #333;

    color: white;

    position: fixed;

    bottom: 0;

    width: 100%;

  }

  .error {

    color: red;

    font-size: 14px;

    margin-bottom: 10px;

    display: none;

}

.modal {

    display: none;

    position: fixed;

    z-index: 1000;

    left: 0;

    top: 0;

    width: 100%;

    height: 100%;

    overflow: auto;

    background-color: rgba(0, 0, 0, 0.5);

}

.modal-content {

    background-color: white;

    margin: 15% auto;

    padding: 20px;

    border-radius: 8px;

    max-width: 400px;

    text-align: center;

}

**Script.js**

document.getElementById("feedbackForm").addEventListener("submit", function (event) {

    event.preventDefault();

    document.getElementById("nameError").style.display = "none";

    document.getElementById("emailError").style.display = "none";

    document.getElementById("feedbackError").style.display = "none";

    const name = document.getElementById("name").value.trim();

    const email = document.getElementById("email").value.trim();

    const feedback = document.getElementById("feedback").value.trim();

    let isValid = true;

    if (name === "") {

        document.getElementById("nameError").textContent = "Name is required.";

        document.getElementById("nameError").style.display = "block";

        isValid = false;

    }

    if (email === "") {

        document.getElementById("emailError").textContent = "Email is required.";

        document.getElementById("emailError").style.display = "block";

        isValid = false;

    } else if (!/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email)) {

        document.getElementById("emailError").textContent = "Enter a valid email address.";

        document.getElementById("emailError").style.display = "block";

        isValid = false;

    }

    if (feedback === "") {

        document.getElementById("feedbackError").textContent = "Feedback is required.";

        document.getElementById("feedbackError").style.display = "block";

        isValid = false;

    }

    if (isValid) {

        document.getElementById("successModal").style.display = "block";

    }

});

document.getElementById("closeModal").addEventListener("click", function () {

    document.getElementById("successModal").style.display = "none";

});

window.addEventListener("click", function (event) {

    const modal = document.getElementById("successModal");

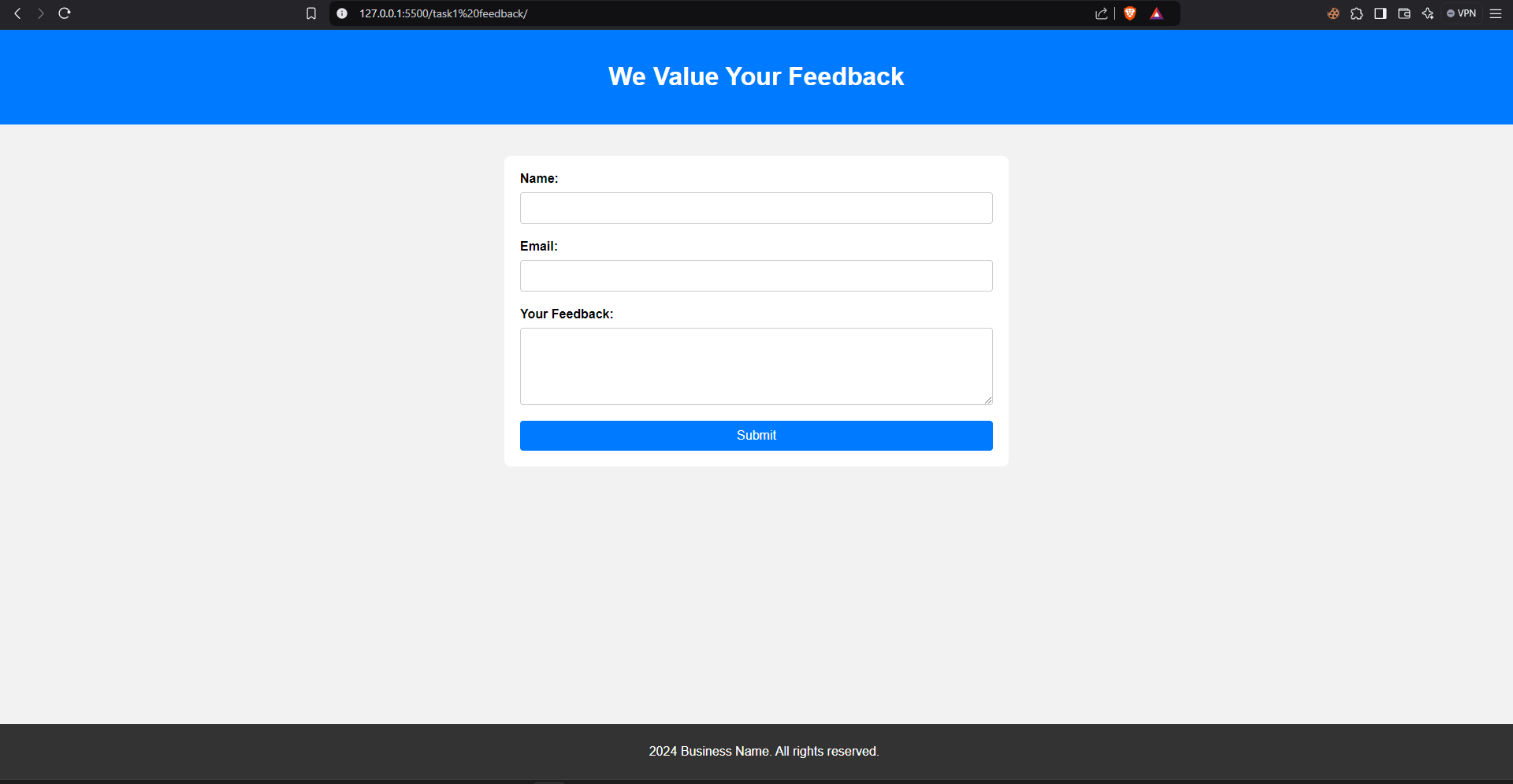
    if (event.target === modal) {

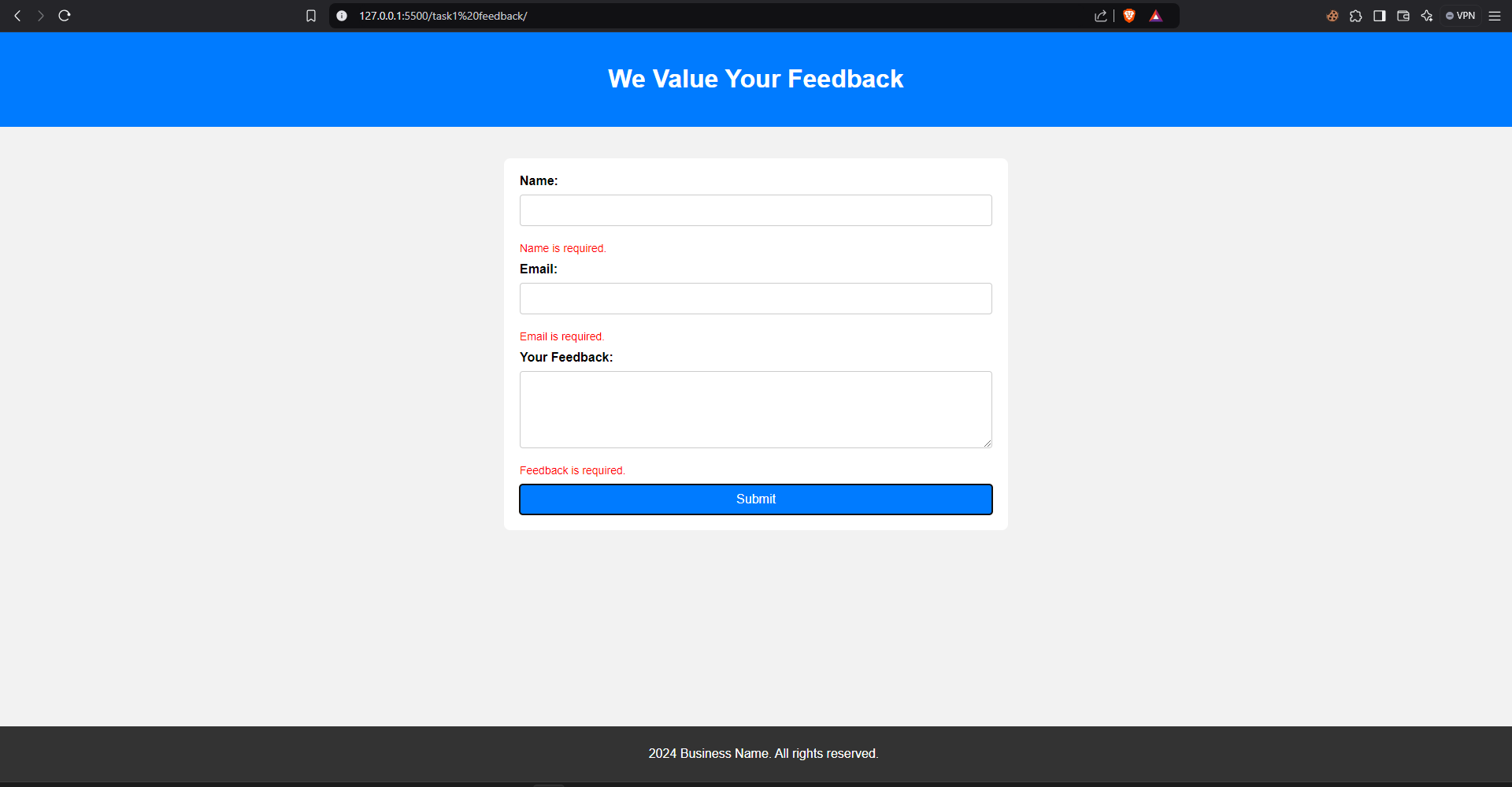
        modal.style.display = "none";

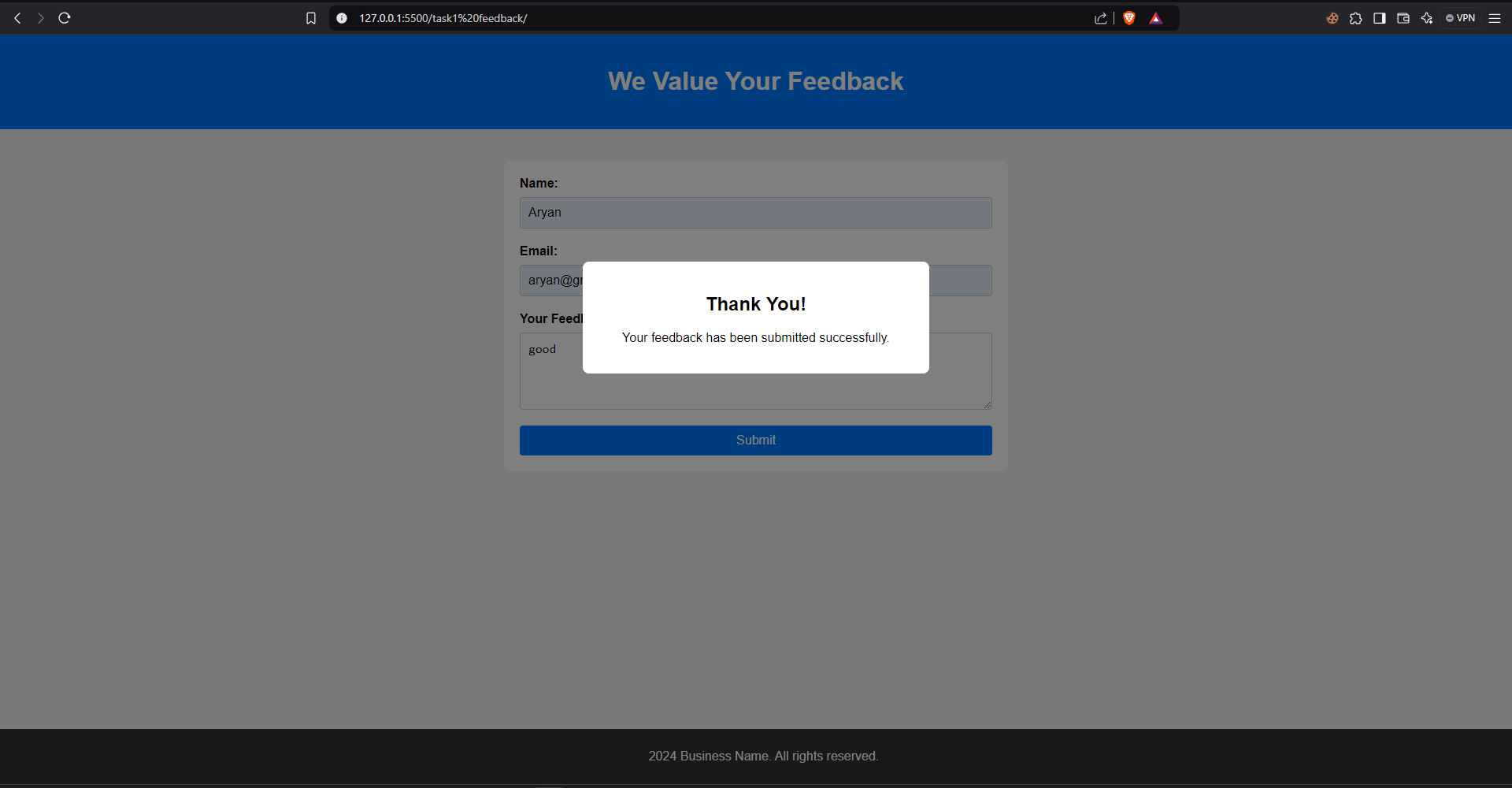
    }

});

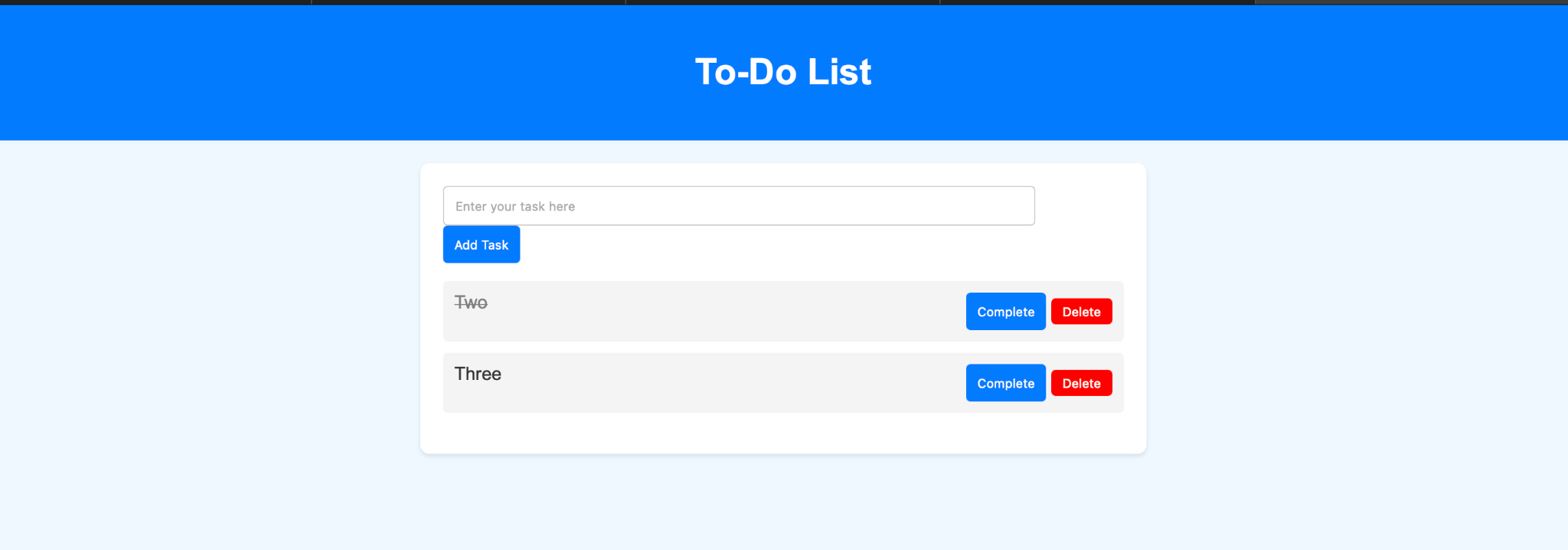
**OUTPUT**







## Task 2:

Imagine a scenario where users need a simple webpage to manage their daily tasks. They can add tasks, mark them as complete, and delete them.

**CODE**

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>To-Do List</title>

  <link rel="stylesheet" href="styles.css">

</head>

<body>

  <div class="header">

    <h1>To-Do List</h1>

  </div>

  <div class="todo-container">

    <div class="input-container">

      <input type="text" id="taskInput" placeholder="Enter your task here">

      <button id="addTaskButton">Add Task</button>

    </div>

    <ul id="taskList"></ul>

  </div>

  <script src="script.js"></script>

</body>

</html>

**Styles.css**

body {

    margin: 0;

    font-family: Arial, sans-serif;

    background-color: #f0f8ff;

  }

  .header {

    background-color: #007bff;

    color: white;

    text-align: center;

    padding: 20px;

  }

  .todo-container {

    max-width: 500px;

    margin: 40px auto;

    padding: 20px;

    background-color: white;

    border-radius: 8px;

    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);

  }

  .input-container {

    display: flex;

    gap: 10px;

  }

  #taskInput {

    flex: 1;

    padding: 10px;

    font-size: 16px;

    border: 1px solid #ccc;

    border-radius: 4px;

  }

  #addTaskButton {

    padding: 10px 20px;

    font-size: 16px;

    color: white;

    background-color: #007bff;

    border: none;

    border-radius: 4px;

    cursor: pointer;

  }

  #addTaskButton:hover {

    background-color: #0056b3;

  }

  #taskList {

    list-style: none;

    padding: 0;

    margin: 20px 0 0;

  }

  .task-item {

    display: flex;

    justify-content: space-between;

    align-items: center;

    padding: 10px;

    margin-bottom: 10px;

    background-color: #f9f9f9;

    border-radius: 4px;

    border: 1px solid #ccc;

  }

  .complete-btn, .delete-btn {

    padding: 5px 10px;

    font-size: 14px;

    border: none;

    border-radius: 4px;

    cursor: pointer;

  }

  .complete-btn {

    background-color: #007bff;

    color: white;

  }

  .complete-btn:hover {

    background-color: #0056b3;

  }

  .delete-btn {

    background-color: #ff4d4d;

    color: white;

  }

  .delete-btn:hover {

    background-color: #cc0000;

  }

  .completed {

    text-decoration: line-through;

    color: #888;

  }

**Script.js**

document.addEventListener("DOMContentLoaded", () => {

    const taskInput = document.getElementById("taskInput");

    const addTaskButton = document.getElementById("addTaskButton");

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

    addTaskButton.addEventListener("click", () => {

      const taskText = taskInput.value.trim();

      if (taskText === "") {

        alert("Please enter a task.");

        return;

      }

      const taskItem = document.createElement("li");

      taskItem.classList.add("task-item");

      const taskContent = document.createElement("span");

      taskContent.textContent = taskText;

      const completeButton = document.createElement("button");

      completeButton.textContent = "Complete";

      completeButton.classList.add("complete-btn");

      const deleteButton = document.createElement("button");

      deleteButton.textContent = "Delete";

      deleteButton.classList.add("delete-btn");

      taskItem.appendChild(taskContent);

      taskItem.appendChild(completeButton);

      taskItem.appendChild(deleteButton);

      taskList.appendChild(taskItem);

      taskInput.value = "";

      completeButton.addEventListener("click", () => {

        taskContent.classList.toggle("completed");

      });

      deleteButton.addEventListener("click", () => {

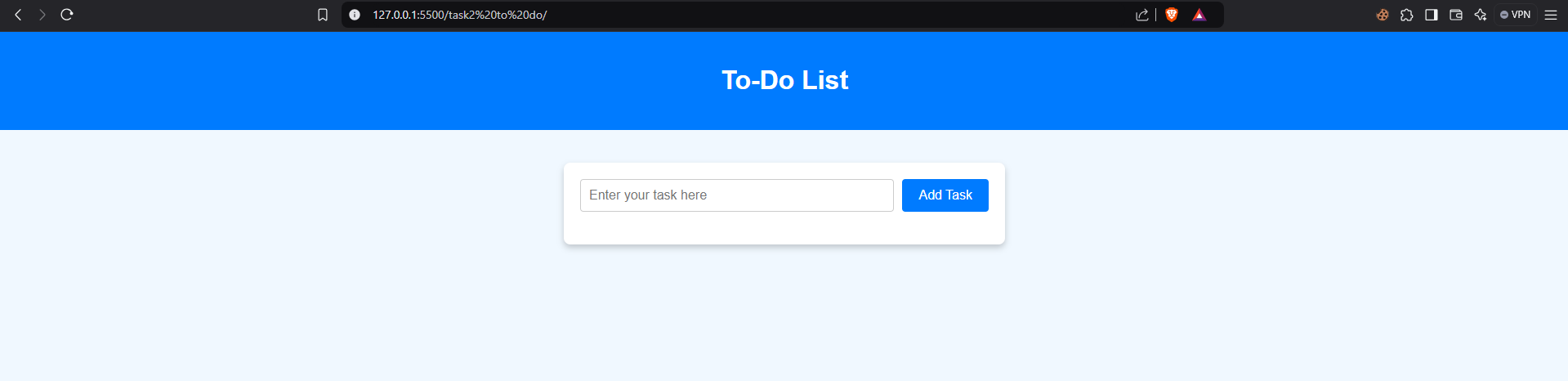
        taskList.removeChild(taskItem);

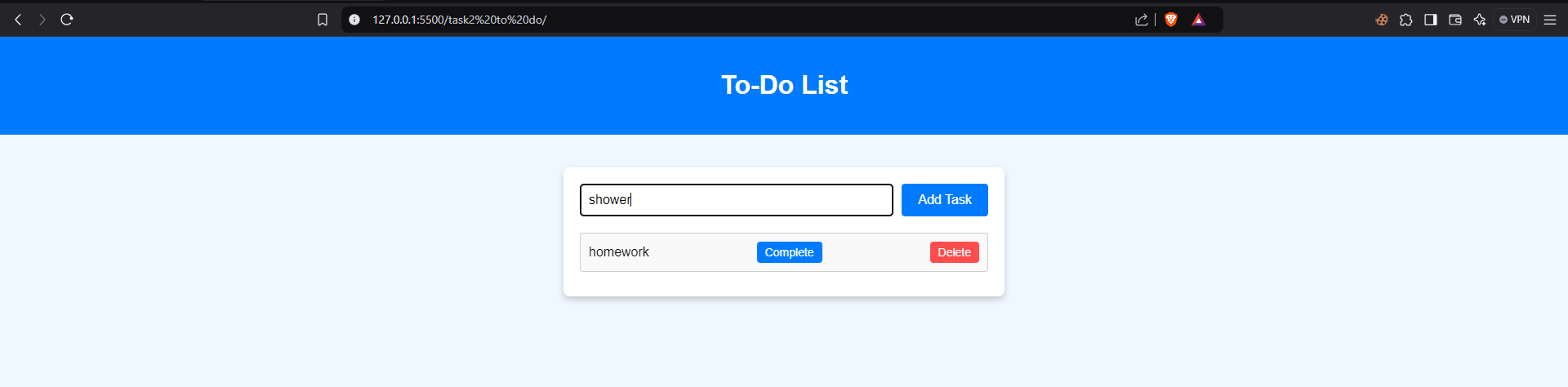
      });

    });

  });

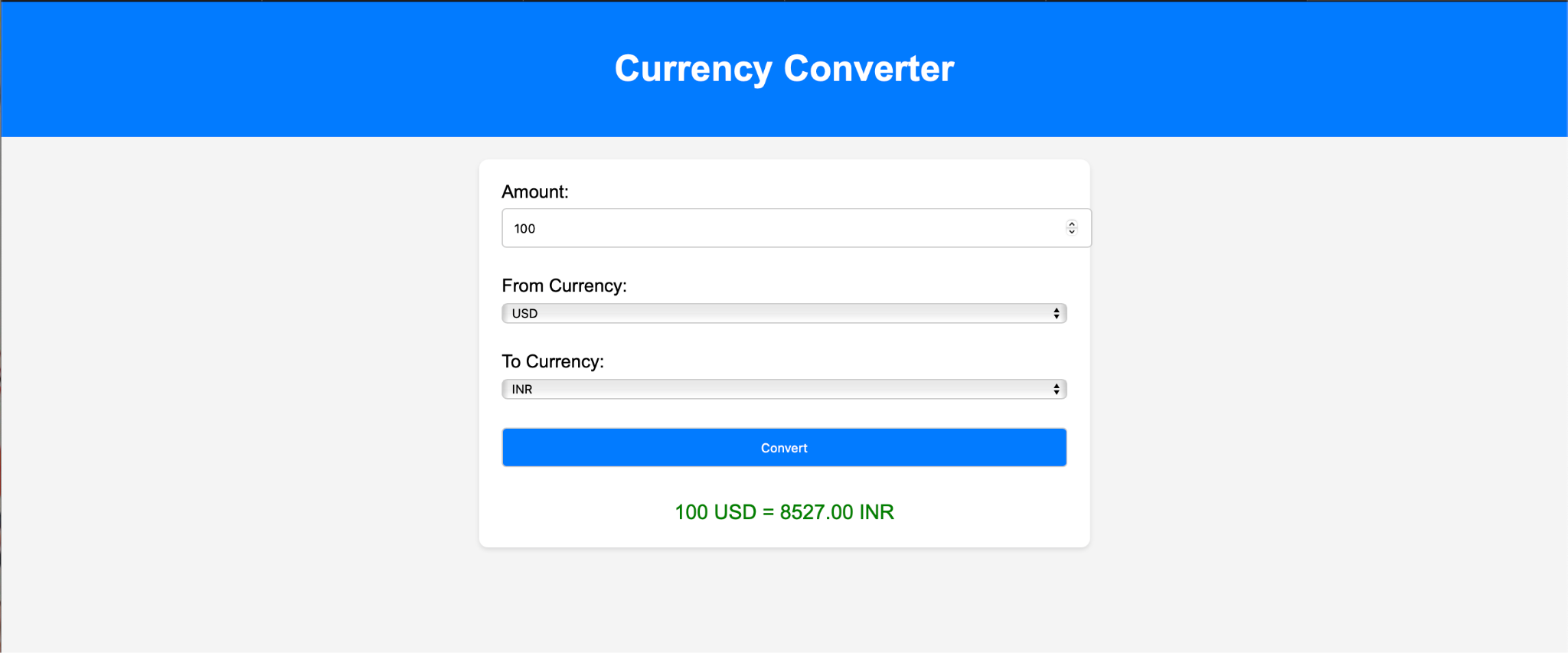
**OUTPUT**

****

****

## Task 3:

**API- https://api.exchangerate-api.com/v4/latest/${fromCurrency}**

This application fetches real-time currency exchange rates using a free API and converts the amount entered by the user.

Features:

## Currency Converter:

* + Uses **Exchange Rate API** for real-time currency rates.
  + Converts between selected currencies and displays results.

## Weather App:

* + Uses **OpenWeather API** to fetch current weather data.
  + Displays temperature, weather description, and city information.

**Colour Code**: #007bff

**Include Learnings and Conclusion in detail.**