CPRO 1102 B

Assignment- 3

By Pranav Talwar
& Manjot Singh
& Bhagatbir Singh

Table of Contents

1. Project Title

"My Fitness Companion"

2. Timer Feature

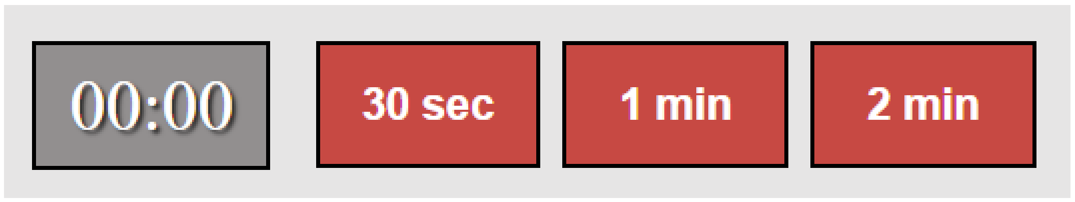


Figure 1 TIMER APP

2.1 Purpose

The timer feature is essential for users to time their exercise routines, breaks, or any health-related activities. It adds value by encouraging users to maintain structured sessions for better productivity and adherence to schedules.

2.2 Functionality

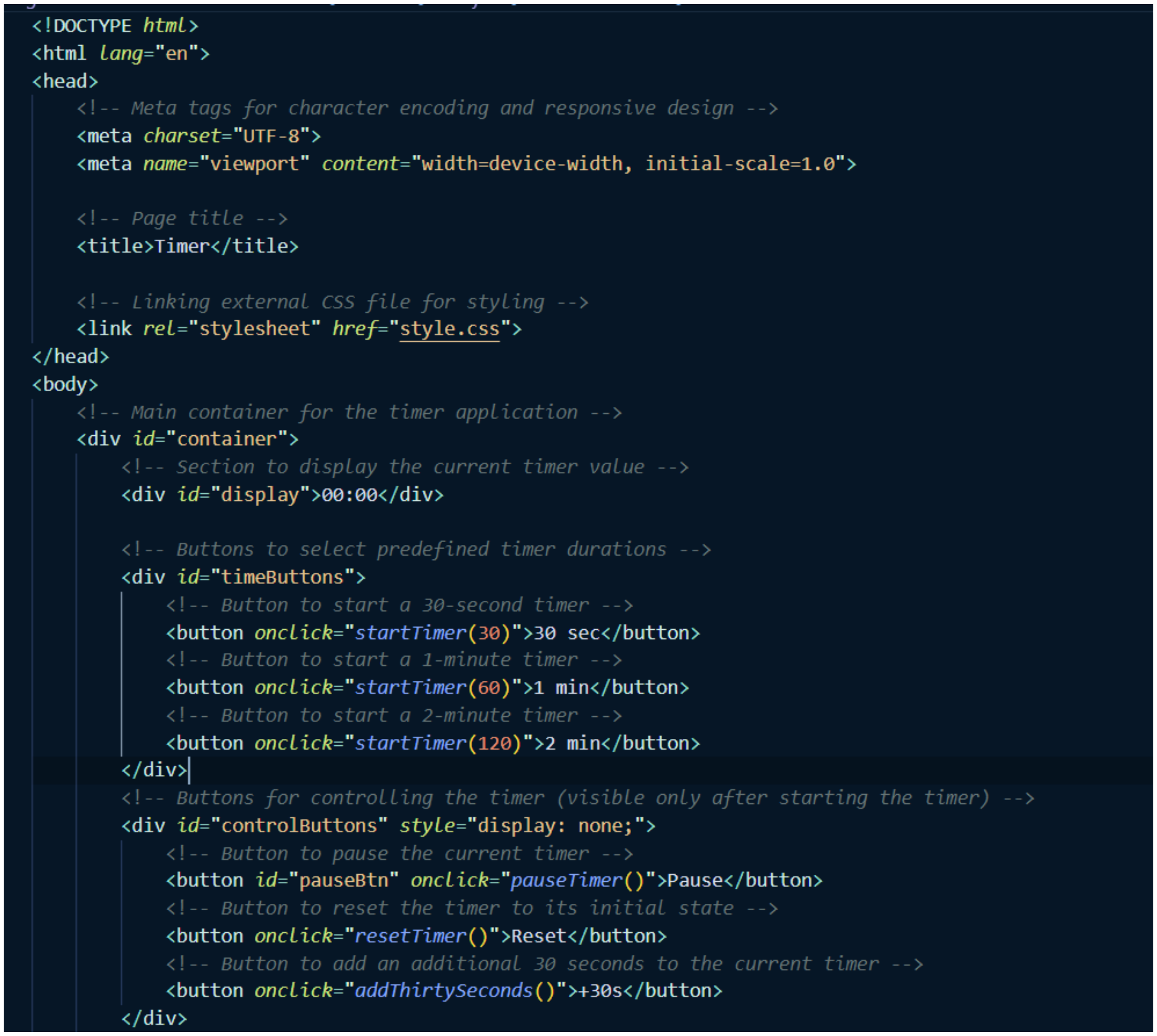
The timer allows users to select preset durations (e.g., 30 seconds, 60 seconds, 2 minutes) and start, pause, resume, or reset the countdown. Upon clicking a duration button, control buttons (start, pause, resume, reset) appear dynamically. Clicking reset hides the control buttons and shows the duration buttons again.

2.3 Integration

- **HTML:** Timer buttons and controls will be included within the user interface.
- **CSS:** Styling will ensure buttons are visually distinct and responsive.
- **JavaScript/jQuery:** Handles button visibility toggles, countdown logic, and user interaction.

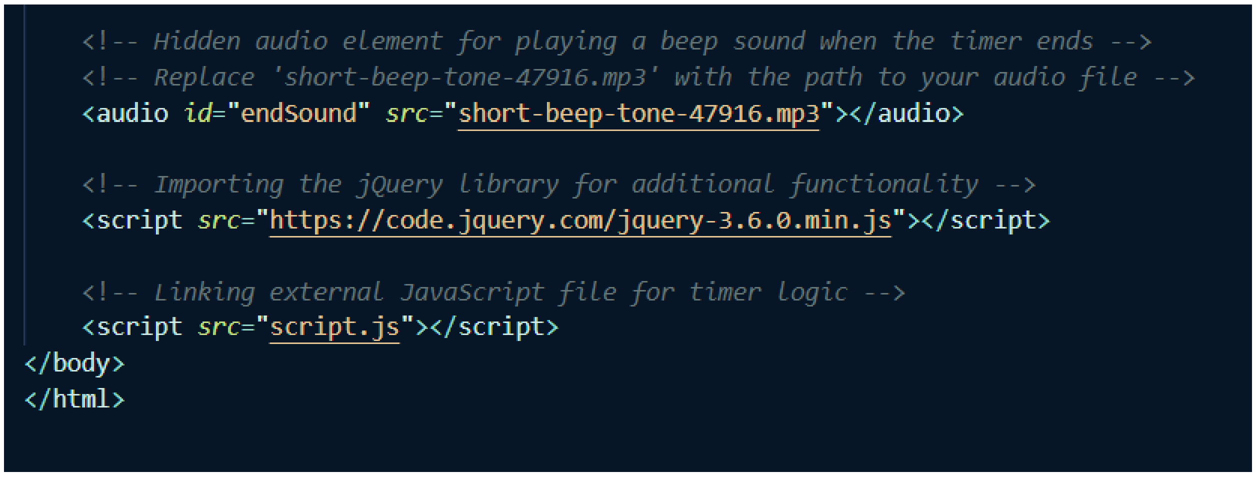
2.4 Implementation

2.4.1 HTML Structure



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <!-- Meta tags for character encoding and responsive design -->
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- Page title -->
    <title>Timer</title>

    <!-- Linking external CSS file for styling -->
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <!-- Main container for the timer application -->
    <div id="container">
        <!-- Section to display the current timer value -->
        <div id="display">00:00</div>

        <!-- Buttons to select predefined timer durations -->
        <div id="timeButtons">
            <!-- Button to start a 30-second timer -->
            <button onclick="startTimer(30)">30 sec</button>
            <!-- Button to start a 1-minute timer -->
            <button onclick="startTimer(60)">1 min</button>
            <!-- Button to start a 2-minute timer -->
            <button onclick="startTimer(120)">2 min</button>
        </div>
        <!-- Buttons for controlling the timer (visible only after starting the timer) -->
        <div id="controlButtons" style="display: none;">
            <!-- Button to pause the current timer -->
            <button id="pauseBtn" onclick="pauseTimer()">Pause</button>
            <!-- Button to reset the timer to its initial state -->
            <button onclick="resetTimer()">Reset</button>
            <!-- Button to add an additional 30 seconds to the current timer -->
            <button onclick="addThirtySeconds()">+30s</button>
        </div>
```

Figure 2 timer html code



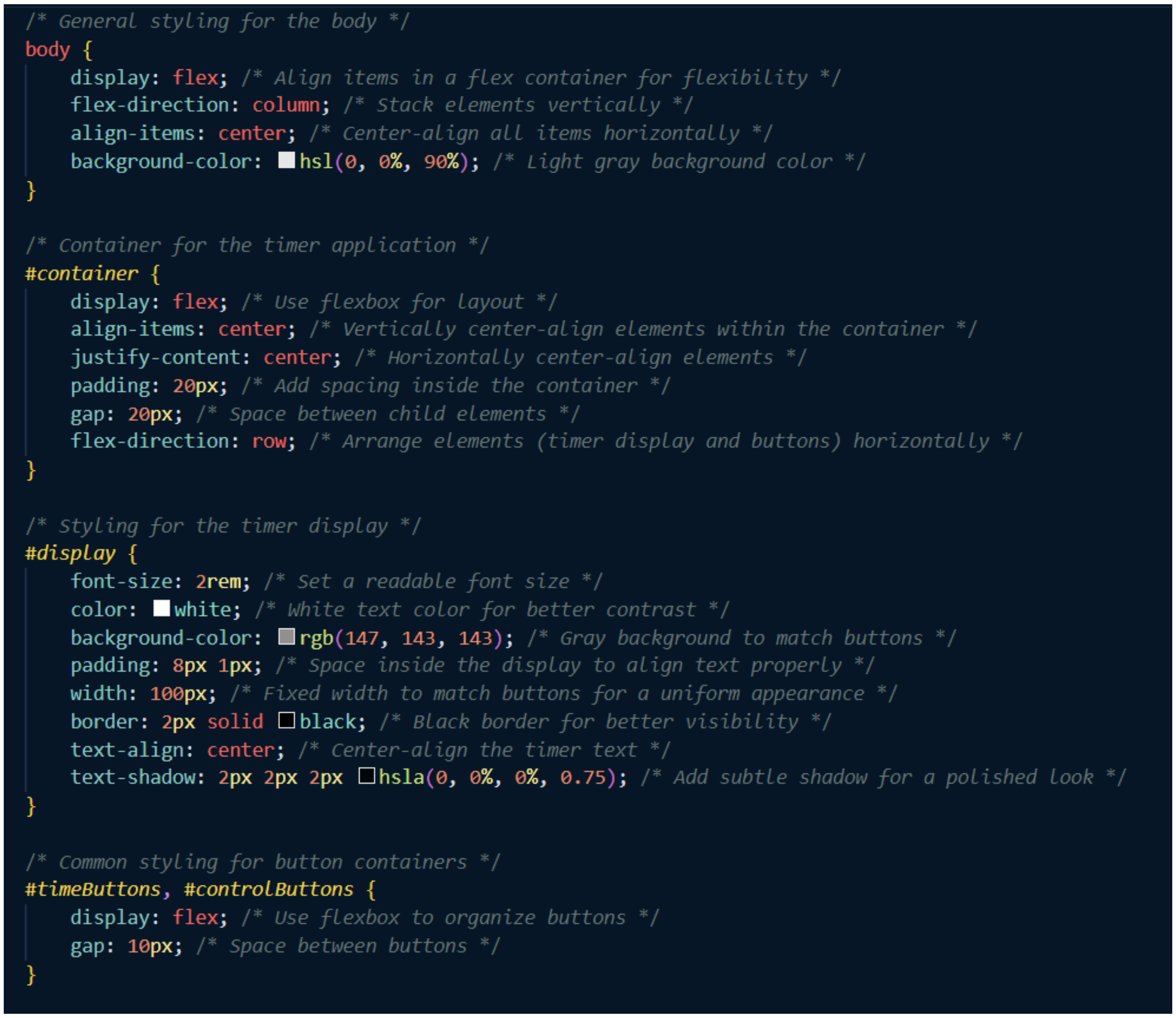
```html
    <!-- Hidden audio element for playing a beep sound when the timer ends -->
    <!-- Replace 'short-beep-tone-47916.mp3' with the path to your audio file -->
    <audio id="endSound" src="short-beep-tone-47916.mp3"></audio>

    <!-- Importing the jQuery library for additional functionality -->
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

    <!-- Linking external JavaScript file for timer logic -->
    <script src="script.js"></script>
</body>
</html>
```

Figure 3 timer html code 2

Key Points:

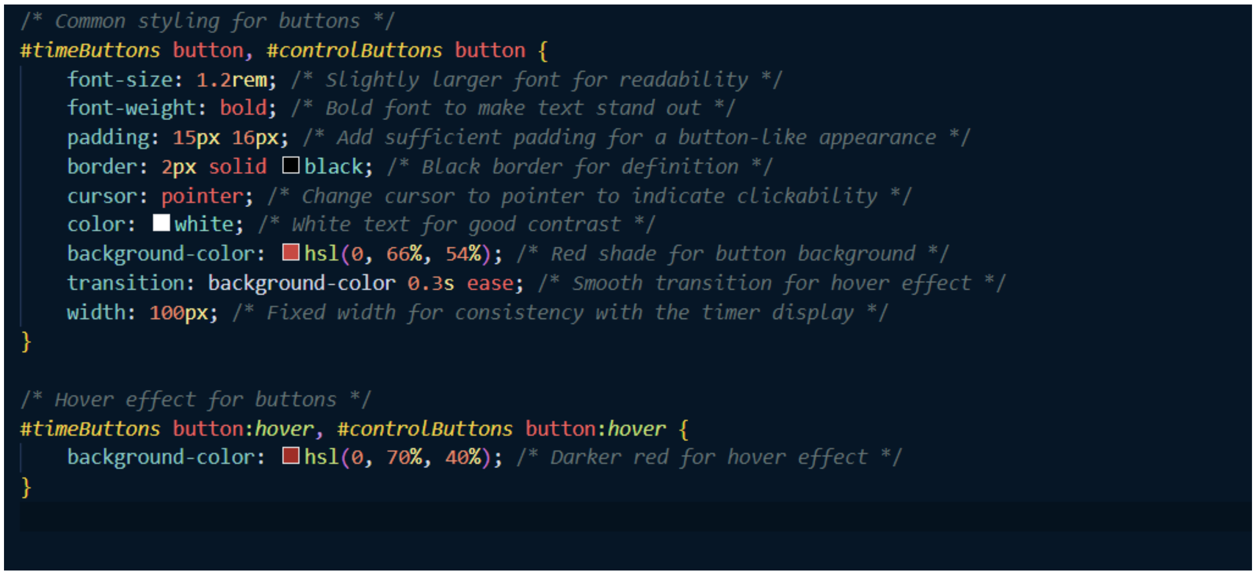
1. Contains a container for the timer application layout, including:
   - **Timer Display:** Displays the countdown time dynamically.
   - **Time Selection Buttons:** Allows users to select a predefined time (30 seconds, 1 minute, 2 minutes).
   - **Control Buttons:** Enables actions like pausing, resetting, and adding time dynamically.
2. Includes an `<audio>` element for the end sound.

2.4.2 CSS Styles



```css
/* General styling for the body */
body {
    display: flex; /* Align items in a flex container for flexibility */
    flex-direction: column; /* Stack elements vertically */
    align-items: center; /* Center-align all items horizontally */
    background-color: hsl(0, 0%, 90%); /* Light gray background color */
}

/* Container for the timer application */
#container {
    display: flex; /* Use flexbox for layout */
    align-items: center; /* Vertically center-align elements within the container */
    justify-content: center; /* Horizontally center-align elements */
    padding: 20px; /* Add spacing inside the container */
    gap: 20px; /* Space between child elements */
    flex-direction: row; /* Arrange elements (timer display and buttons) horizontally */
}

/* Styling for the timer display */
#display {
    font-size: 2rem; /* Set a readable font size */
    color: white; /* White text color for better contrast */
    background-color: rgb(147, 143, 143); /* Gray background to match buttons */
    padding: 8px 1px; /* Space inside the display to align text properly */
    width: 100px; /* Fixed width to match buttons for a uniform appearance */
    border: 2px solid black; /* Black border for better visibility */
    text-align: center; /* Center-align the timer text */
    text-shadow: 2px 2px 2px hsla(0, 0%, 0%, 0.75); /* Add subtle shadow for a polished look */
}

/* Common styling for button containers */
#timeButtons, #controlButtons {
    display: flex; /* Use flexbox to organize buttons */
    gap: 10px; /* Space between buttons */
}
```

Figure 4 CSS code for timer app



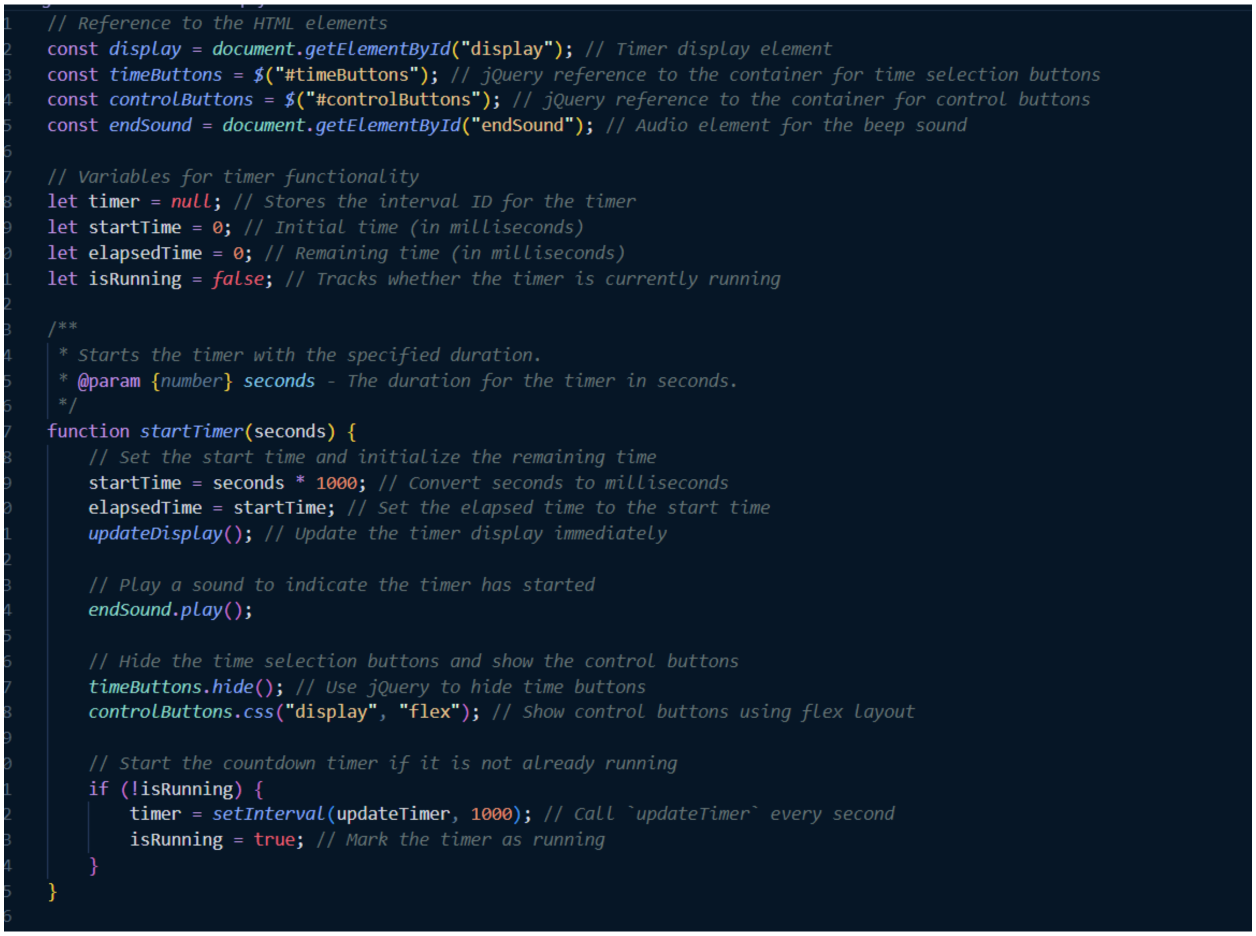
```css
/* Common styling for buttons */
#timeButtons button, #controlButtons button {
    font-size: 1.2rem; /* Slightly larger font for readability */
    font-weight: bold; /* Bold font to make text stand out */
    padding: 15px 16px; /* Add sufficient padding for a button-like appearance */
    border: 2px solid black; /* Black border for definition */
    cursor: pointer; /* Change cursor to pointer to indicate clickability */
    color: white; /* White text for good contrast */
    background-color: hsl(0, 66%, 54%); /* Red shade for button background */
    transition: background-color 0.3s ease; /* Smooth transition for hover effect */
    width: 100px; /* Fixed width for consistency with the timer display */
}

/* Hover effect for buttons */
#timeButtons button:hover, #controlButtons button:hover {
    background-color: hsl(0, 70%, 40%); /* Darker red for hover effect */
}
```

Figure 5 CSS code for timer app 2

Key Points:

1. **Background Design:**
   - A light gray (`hsl(0, 0%, 90%)`) background provides a clean interface.
2. **Timer Display Styling:**
   - Centralized text with a gray background and shadow for readability.
   - Font size, width, and padding match the buttons for consistency.
3. **Button Styling:**
   - Consistent gradient background (`hsl(0, 66%, 54%)`) with hover effects for interactivity.
   - Buttons are responsive with fixed widths and bold text for accessibility.
4. **Responsive Design:**
   - Ensures the application layout adjusts for various devices.

2.4.3 jQuery Functionality



```
// Reference to the HTML elements
const display = document.getElementById("display"); // Timer display element
const timeButtons = $("#timeButtons"); // jQuery reference to the container for time selection buttons
const controlButtons = $("#controlButtons"); // jQuery reference to the container for control buttons
const endSound = document.getElementById("endSound"); // Audio element for the beep sound

// Variables for timer functionality
let timer = null; // Stores the interval ID for the timer
let startTime = 0; // Initial time (in milliseconds)
let elapsedTime = 0; // Remaining time (in milliseconds)
let isRunning = false; // Tracks whether the timer is currently running

/**
 * Starts the timer with the specified duration.
 * @param {number} seconds - The duration for the timer in seconds.
 */
function startTimer(seconds) {
    // Set the start time and initialize the remaining time
    startTime = seconds * 1000; // Convert seconds to milliseconds
    elapsedTime = startTime; // Set the elapsed time to the start time
    updateDisplay(); // Update the timer display immediately

    // Play a sound to indicate the timer has started
    endSound.play();

    // Hide the time selection buttons and show the control buttons
    timeButtons.hide(); // Use jQuery to hide time buttons
    controlButtons.css("display", "flex"); // Show control buttons using flex layout

    // Start the countdown timer if it is not already running
    if (!isRunning) {
        timer = setInterval(updateTimer, 1000); // Call `updateTimer` every second
        isRunning = true; // Mark the timer as running
    }
}
```

Figure 6 Javascript code for timer app 1

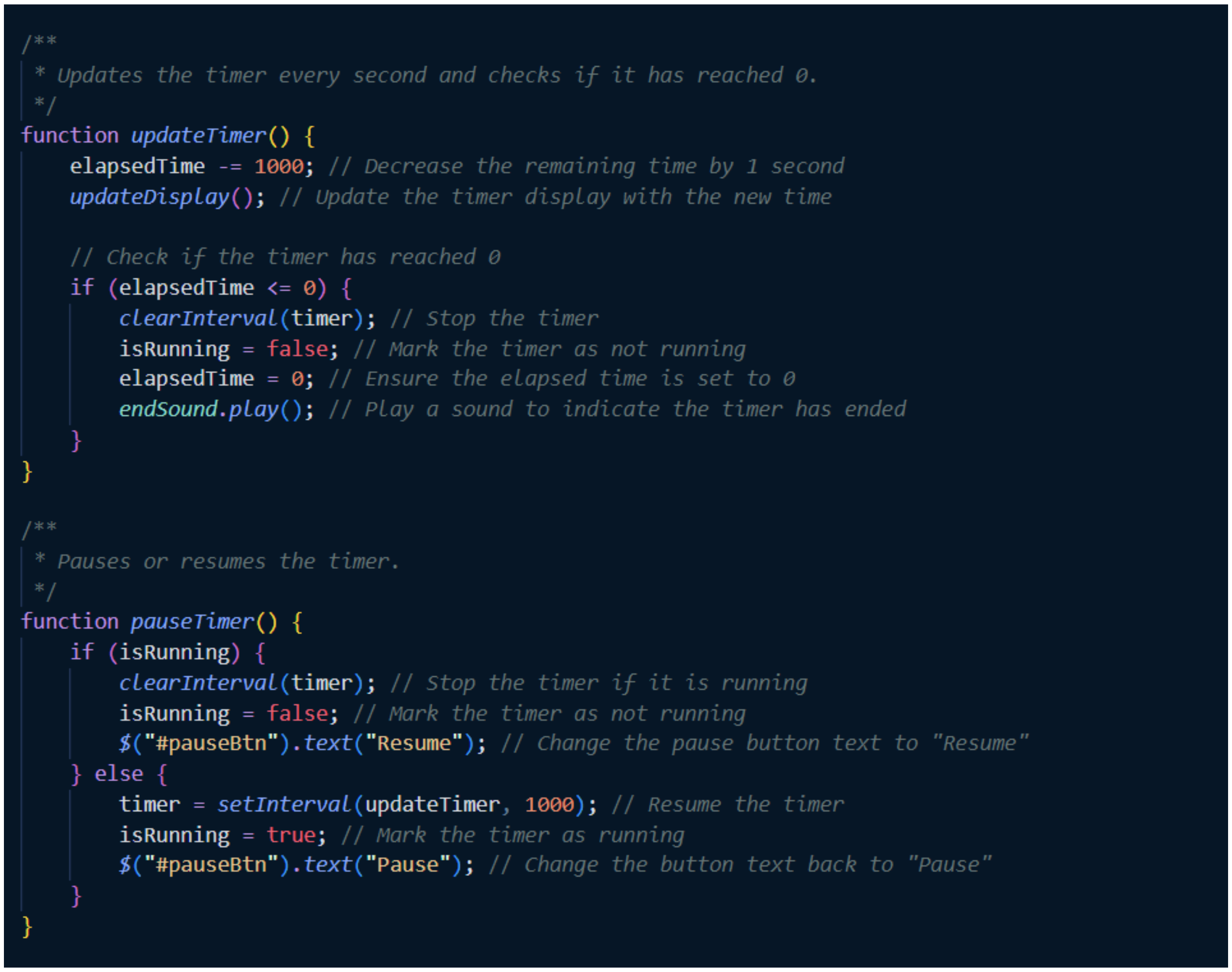
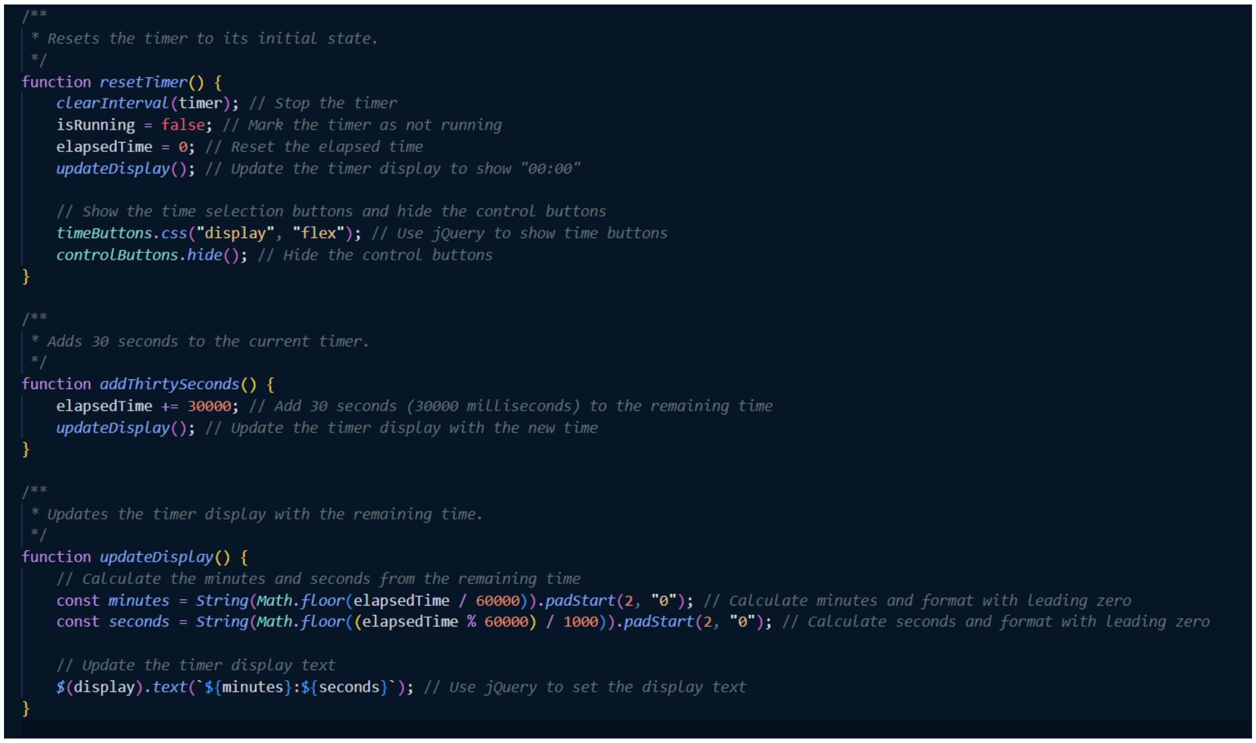

```
/**
 * Updates the timer every second and checks if it has reached 0.
 */
function updateTimer() {
    elapsedTime -= 1000; // Decrease the remaining time by 1 second
    updateDisplay(); // Update the timer display with the new time

    // Check if the timer has reached 0
    if (elapsedTime <= 0) {
        clearInterval(timer); // Stop the timer
        isRunning = false; // Mark the timer as not running
        elapsedTime = 0; // Ensure the elapsed time is set to 0
        endSound.play(); // Play a sound to indicate the timer has ended
    }
}

/**
 * Pauses or resumes the timer.
 */
function pauseTimer() {
    if (isRunning) {
        clearInterval(timer); // Stop the timer if it is running
        isRunning = false; // Mark the timer as not running
        $("#pauseBtn").text("Resume"); // Change the pause button text to "Resume"
    } else {
        timer = setInterval(updateTimer, 1000); // Resume the timer
        isRunning = true; // Mark the timer as running
        $("#pauseBtn").text("Pause"); // Change the button text back to "Pause"
    }
}
```

Figure 7 Javascript code for timer app 2



```
/**
 * Resets the timer to its initial state.
 */
function resetTimer() {
    clearInterval(timer); // Stop the timer
    isRunning = false; // Mark the timer as not running
    elapsedTime = 0; // Reset the elapsed time
    updateDisplay(); // Update the timer display to show "00:00"

    // Show the time selection buttons and hide the control buttons
    timeButtons.css("display", "flex"); // Use jQuery to show time buttons
    controlButtons.hide(); // Hide the control buttons
}

/**
 * Adds 30 seconds to the current timer.
 */
function addThirtySeconds() {
    elapsedTime += 30000; // Add 30 seconds (30000 milliseconds) to the remaining time
    updateDisplay(); // Update the timer display with the new time
}

/**
 * Updates the timer display with the remaining time.
 */
function updateDisplay() {
    // Calculate the minutes and seconds from the remaining time
    const minutes = String(Math.floor(elapsedTime / 60000)).padStart(2, "0"); // Calculate minutes and format with leading zero
    const seconds = String(Math.floor((elapsedTime % 60000) / 1000)).padStart(2, "0"); // Calculate seconds and format with leading zero

    // Update the timer display text
    $(display).text(`${minutes}:${seconds}`); // Use jQuery to set the display text
}
```

Figure 8 Javascript code for timer app 2

a. Initialization

- Key DOM elements (`#timeButtons` and `#controlButtons`) are toggled between visible states.
- `#display` is updated dynamically based on elapsed time.

b. Timer Start Functionality

- **Function:** `startTimer(seconds)`
  - Sets the countdown timer based on the selected time in seconds.
  - Hides the time selection buttons and displays control buttons using jQuery.
  - Starts the timer using `setInterval()`.

c. Countdown Logic

- **Function:** `updateTimer()`
  - Decreases elapsed time by 1000 ms every second.

    - Updates the display in `mm:ss` format dynamically.
    - Stops and resets the timer when it reaches zero, playing the end sound.

d. Pause/Resume Functionality

- **Function:** `pauseTimer()`
    - Toggles between pausing and resuming the timer.
    - Updates the pause button text dynamically (e.g., "Pause" -> "Resume").

e. Reset Functionality

- **Function:** `resetTimer()`
    - Stops the timer and resets elapsed time to zero.
    - Restores the time selection buttons and hides the control buttons.

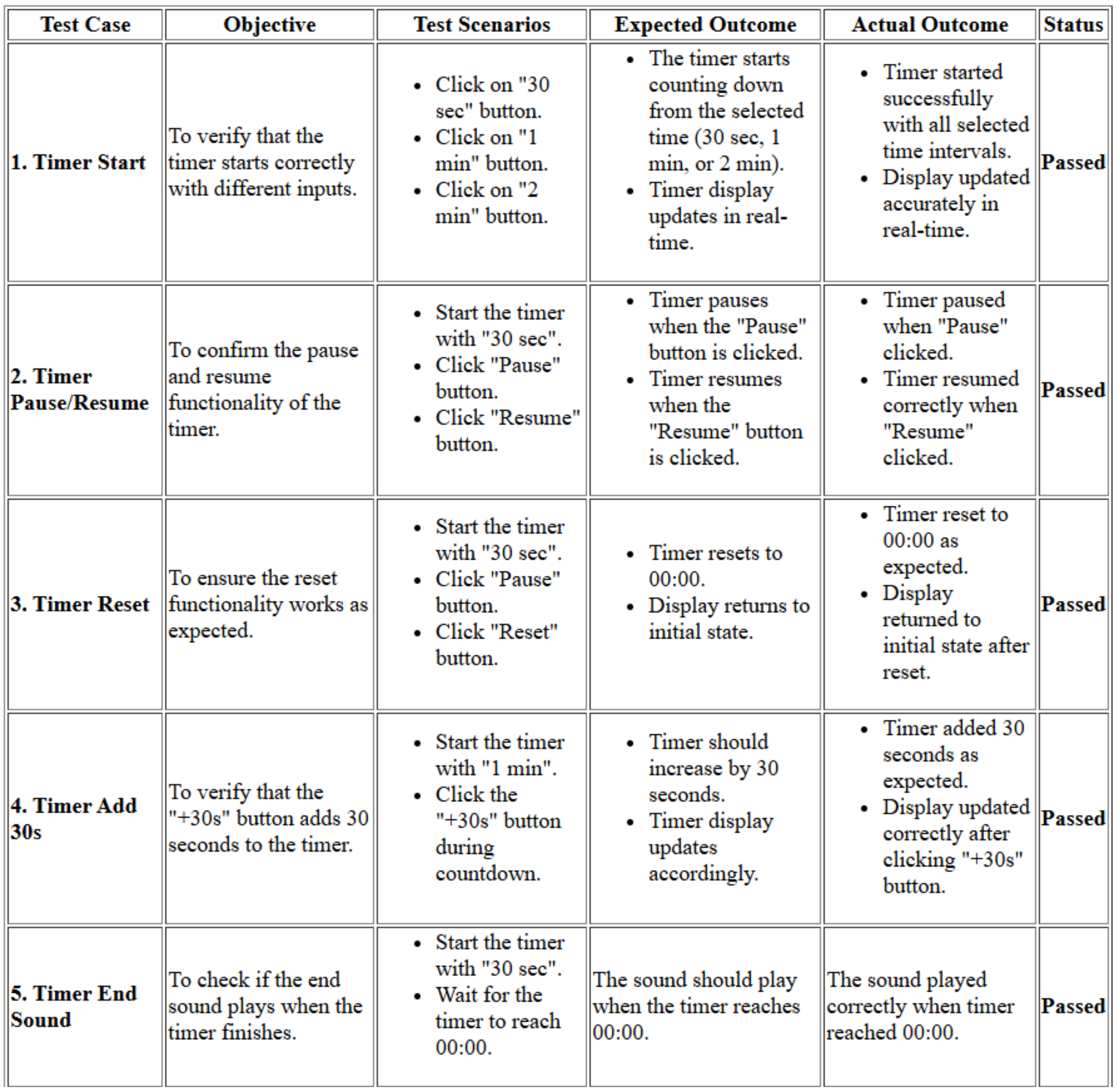
f. Add Time Functionality

- **Function:** `addThirtySeconds()`
    - Adds 30 seconds (30000 ms) to the current elapsed time.
    - Updates the display dynamically.

g. Display Update Logic

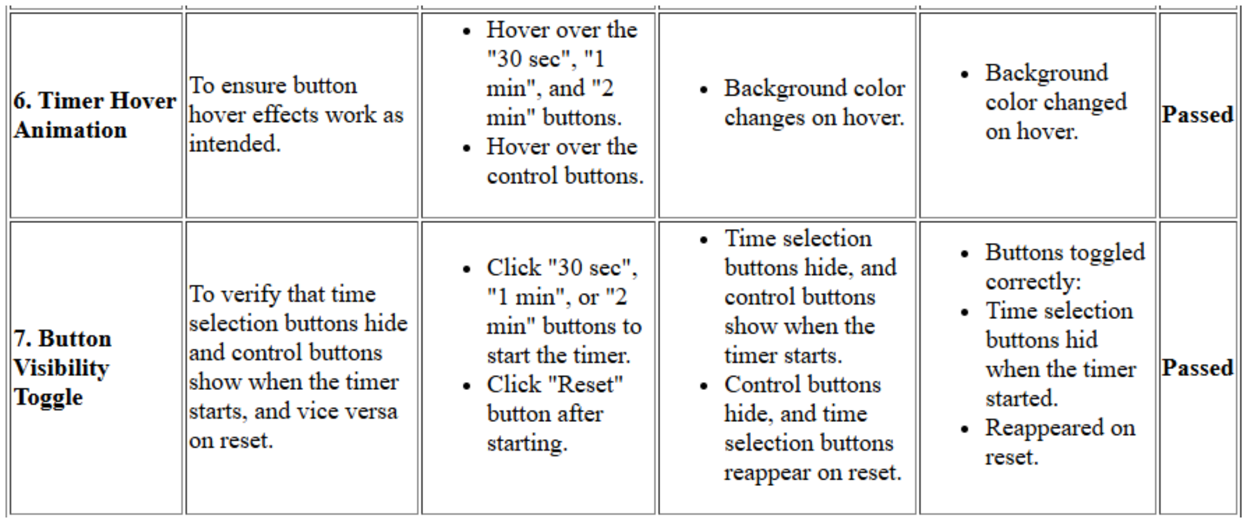
- **Function:** `updateDisplay()`
    - Formats elapsed time as `mm:ss` using `padStart()` for consistency.
    - Updates the timer display dynamically using jQuery.

2.5 Testing Table



Test Case	Objective	Test Scenarios	Expected Outcome	Actual Outcome	Status
1. Timer Start	To verify that the timer starts correctly with different inputs.	• Click on "30 sec" button. • Click on "1 min" button. • Click on "2 min" button.	• The timer starts counting down from the selected time (30 sec, 1 min, or 2 min). • Timer display updates in real-time.	• Timer started successfully with all selected time intervals. • Display updated accurately in real-time.	**Passed**
2. Timer Pause/Resume	To confirm the pause and resume functionality of the timer.	• Start the timer with "30 sec". • Click "Pause" button. • Click "Resume" button.	• Timer pauses when the "Pause" button is clicked. • Timer resumes when the "Resume" button is clicked.	• Timer paused when "Pause" clicked. • Timer resumed correctly when "Resume" clicked.	**Passed**
3. Timer Reset	To ensure the reset functionality works as expected.	• Start the timer with "30 sec". • Click "Pause" button. • Click "Reset" button.	• Timer resets to 00:00. • Display returns to initial state.	• Timer reset to 00:00 as expected. • Display returned to initial state after reset.	**Passed**
4. Timer Add 30s	To verify that the "+30s" button adds 30 seconds to the timer.	• Start the timer with "1 min". • Click the "+30s" button during countdown.	• Timer should increase by 30 seconds. • Timer display updates accordingly.	• Timer added 30 seconds as expected. • Display updated correctly after clicking "+30s" button.	**Passed**
5. Timer End Sound	To check if the end sound plays when the timer finishes.	• Start the timer with "30 sec". • Wait for the timer to reach 00:00.	The sound should play when the timer reaches 00:00.	The sound played correctly when timer reached 00:00.	**Passed**

Figure 9 Timer Testing table 1



6. Timer Hover Animation	To ensure button hover effects work as intended.	• Hover over the "30 sec", "1 min", and "2 min" buttons. • Hover over the control buttons.	• Background color changes on hover.	• Background color changed on hover.	**Passed**
7. Button Visibility Toggle	To verify that time selection buttons hide and control buttons show when the timer starts, and vice versa on reset.	• Click "30 sec", "1 min", or "2 min" buttons to start the timer. • Click "Reset" button after starting.	• Time selection buttons hide, and control buttons show when the timer starts. • Control buttons hide, and time selection buttons reappear on reset.	• Buttons toggled correctly: • Time selection buttons hid when the timer started. • Reappeared on reset.	**Passed**

Figure 10 Timer Testing table 2

2.6 Testing Images

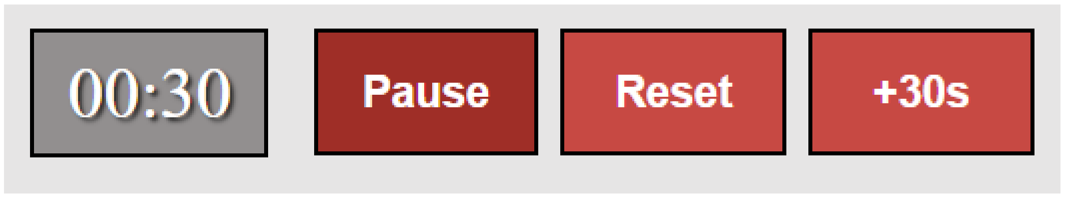


Figure 11 TEST CASE 1.1 when we click on 30 sec

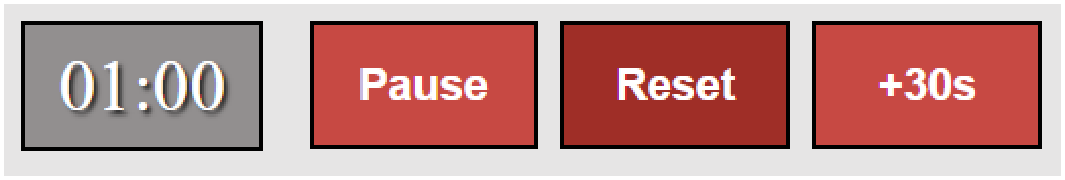


Figure 12 TEST CASE 1.2 when we click on 60 sec

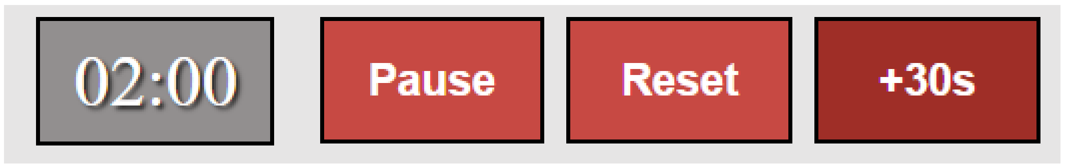


Figure 13 TEST CASE 1.3 when we click on 2 min

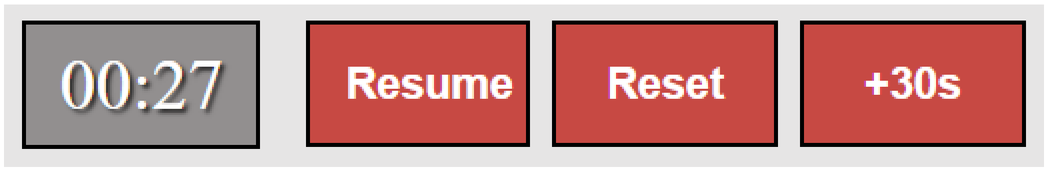


Figure 14 TEST CASE 3 To ensure the reset functionality works as expected.

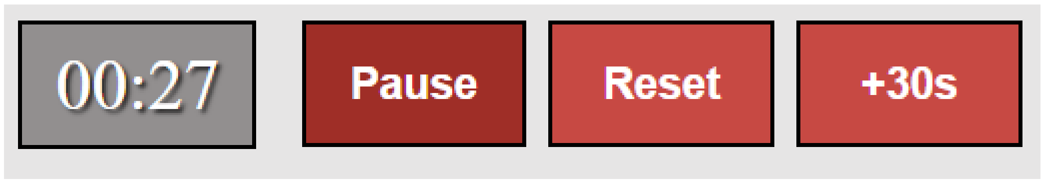


Figure 15 TEST CASE 6 Background color changes on hover.

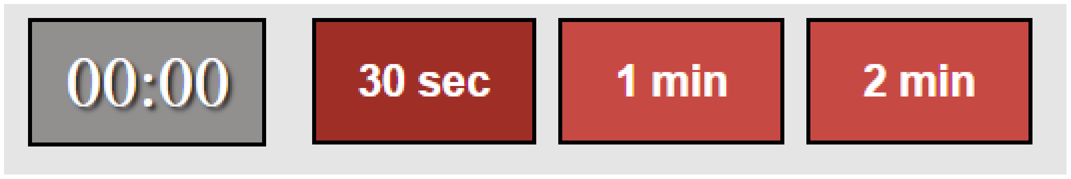


Figure 16 TEST CASE 7.1 Control buttons hide, and time selection buttons reappear on reset

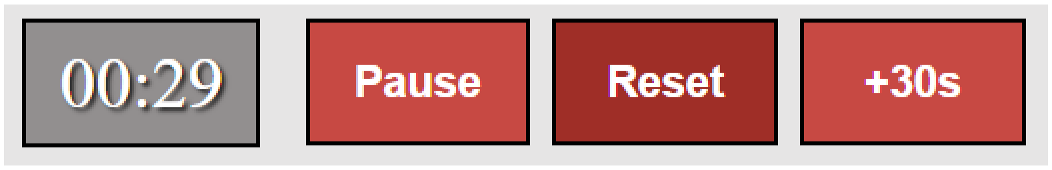


Figure 17 TEST CASE 7.2 Time selection buttons hide, and control buttons show when the timer starts.

3. BMI Calculator Feature

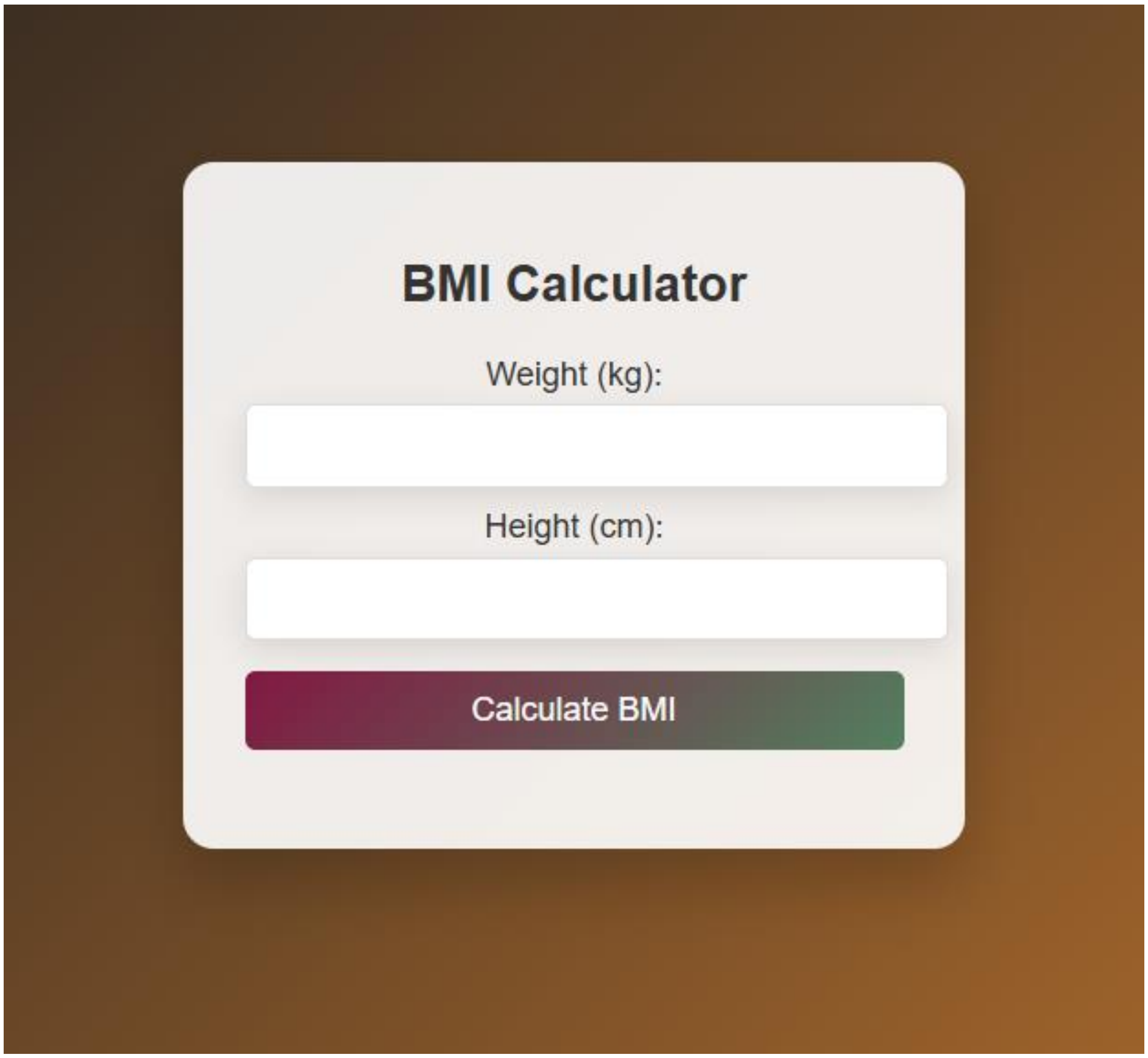


Figure 18 BMI APP

3.1 Purpose

The BMI calculator helps users determine their body mass index based on their height and weight. This feature promotes self-awareness about fitness and health.

3.2 Functionality

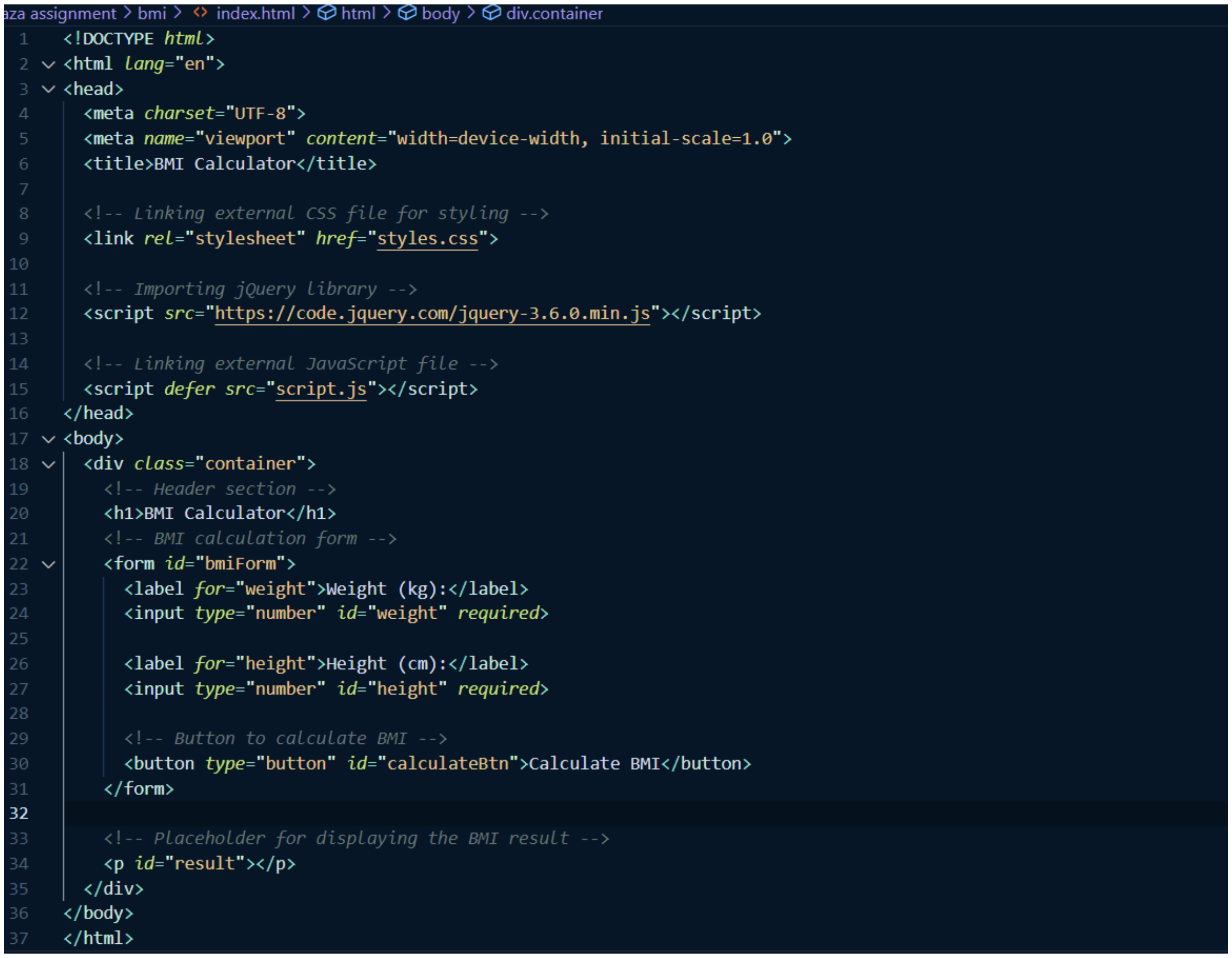
The user inputs their weight and height into dedicated fields, clicks "Calculate," and receives their BMI value along with its category (e.g., Underweight, Normal, Overweight, Obese). Validation ensures only valid numeric inputs are accepted, and error messages are displayed for invalid inputs.

3.3 Integration

- **HTML:** Input fields and a result display section are integrated into the main page.
- **CSS:** Includes hover animations for buttons and a fade-in effect for results.
- **JavaScript/jQuery:** Validates inputs, performs BMI calculation, and dynamically displays the result and corresponding category.

3.4 Implementation

3.4.1 HTML Structure


aza assignment > bmi > index.html > html > body > div.container

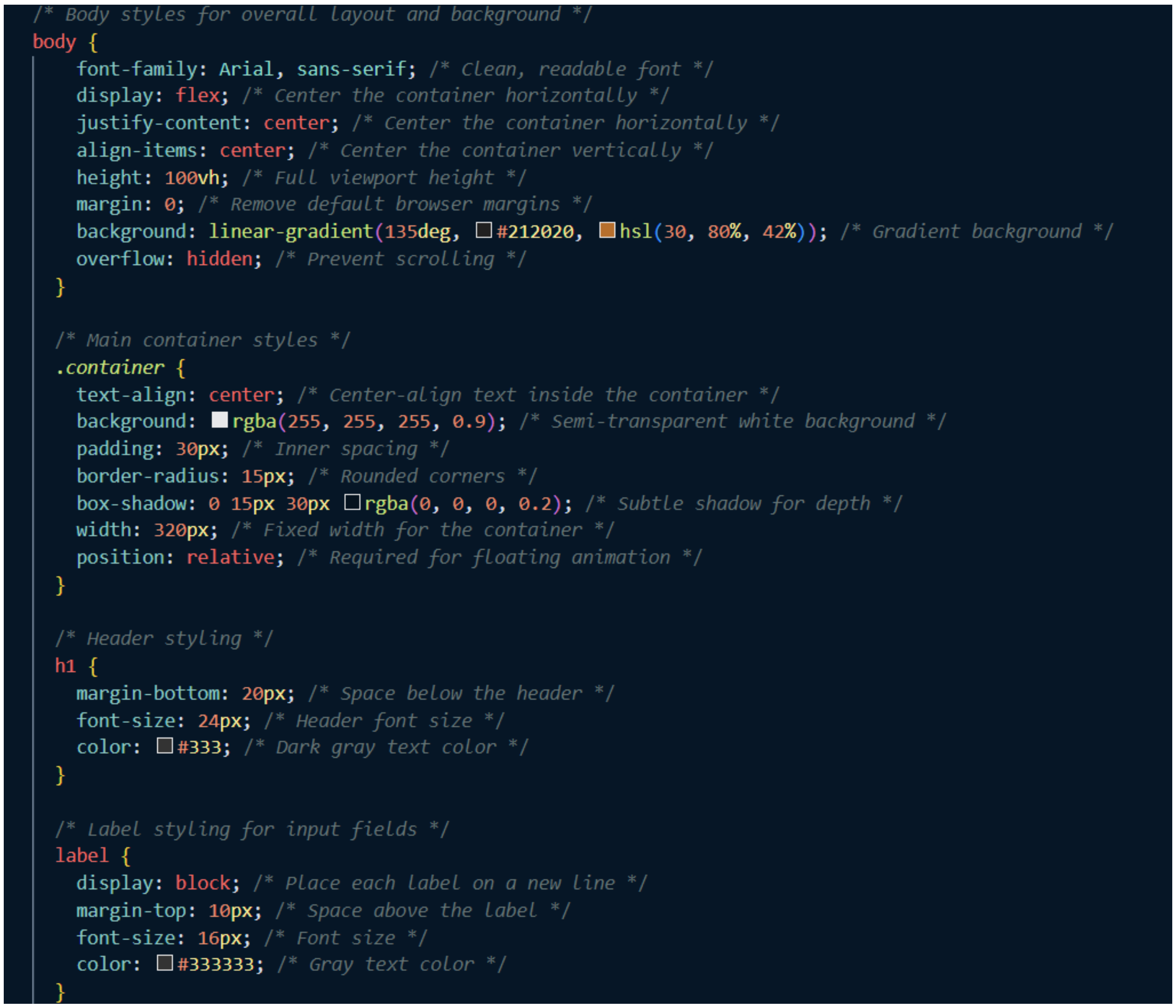
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>BMI Calculator</title>

  <!-- Linking external CSS file for styling -->
  <link rel="stylesheet" href="styles.css">

  <!-- Importing jQuery library -->
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

  <!-- Linking external JavaScript file -->
  <script defer src="script.js"></script>
</head>
<body>
  <div class="container">
    <!-- Header section -->
    <h1>BMI Calculator</h1>
    <!-- BMI calculation form -->
    <form id="bmiForm">
      <label for="weight">Weight (kg):</label>
      <input type="number" id="weight" required>

      <label for="height">Height (cm):</label>
      <input type="number" id="height" required>

      <!-- Button to calculate BMI -->
      <button type="button" id="calculateBtn">Calculate BMI</button>
    </form>

    <!-- Placeholder for displaying the BMI result -->
    <p id="result"></p>
  </div>
</body>
</html>
```

Figure 19 BMI HTML CODE

Key Points:

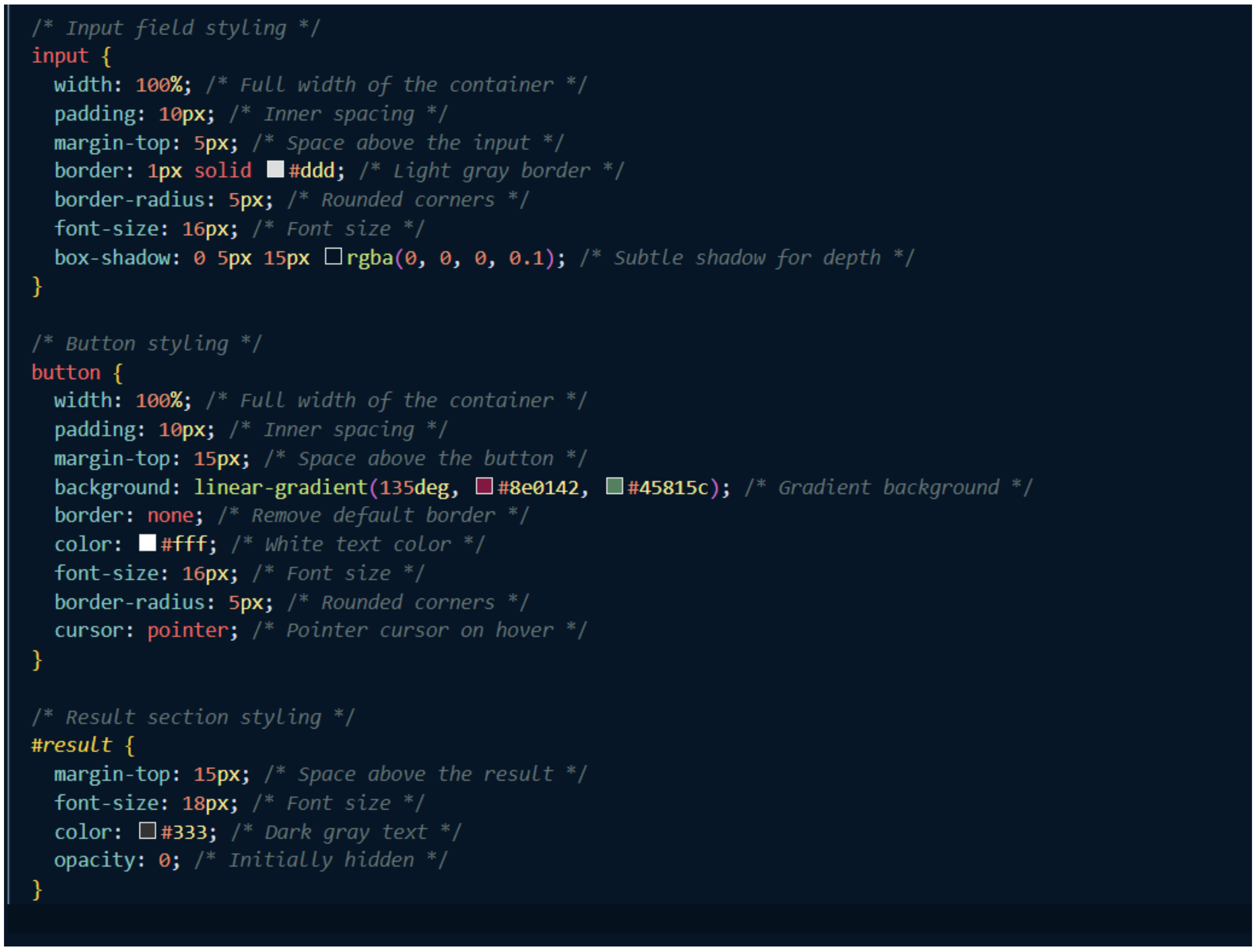
1. A clean and minimalistic structure designed for user inputs, including:
   - **Weight Input Field:** To accept the user's weight in kilograms.
   - **Height Input Field:** To accept the user's height in centimeters.
   - **Calculate BMI Button:** Triggers the calculation functionality.
2. A result display section dynamically updates with the user's BMI and its category.

3.4.2 CSS Styles



```css
/* Body styles for overall layout and background */
body {
    font-family: Arial, sans-serif; /* Clean, readable font */
    display: flex; /* Center the container horizontally */
    justify-content: center; /* Center the container horizontally */
    align-items: center; /* Center the container vertically */
    height: 100vh; /* Full viewport height */
    margin: 0; /* Remove default browser margins */
    background: linear-gradient(135deg, #212020, hsl(30, 80%, 42%)); /* Gradient background */
    overflow: hidden; /* Prevent scrolling */
  }

  /* Main container styles */
  .container {
    text-align: center; /* Center-align text inside the container */
    background: rgba(255, 255, 255, 0.9); /* Semi-transparent white background */
    padding: 30px; /* Inner spacing */
    border-radius: 15px; /* Rounded corners */
    box-shadow: 0 15px 30px rgba(0, 0, 0, 0.2); /* Subtle shadow for depth */
    width: 320px; /* Fixed width for the container */
    position: relative; /* Required for floating animation */
  }

  /* Header styling */
  h1 {
    margin-bottom: 20px; /* Space below the header */
    font-size: 24px; /* Header font size */
    color: #333; /* Dark gray text color */
  }

  /* Label styling for input fields */
  label {
    display: block; /* Place each label on a new line */
    margin-top: 10px; /* Space above the label */
    font-size: 16px; /* Font size */
    color: #333333; /* Gray text color */
  }
```

Figure 20 CSS CODE 1 BMI



```css
/* Input field styling */
input {
  width: 100%; /* Full width of the container */
  padding: 10px; /* Inner spacing */
  margin-top: 5px; /* Space above the input */
  border: 1px solid #ddd; /* Light gray border */
  border-radius: 5px; /* Rounded corners */
  font-size: 16px; /* Font size */
  box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1); /* Subtle shadow for depth */
}

/* Button styling */
button {
  width: 100%; /* Full width of the container */
  padding: 10px; /* Inner spacing */
  margin-top: 15px; /* Space above the button */
  background: linear-gradient(135deg, #8e0142, #45815c); /* Gradient background */
  border: none; /* Remove default border */
  color: #fff; /* White text color */
  font-size: 16px; /* Font size */
  border-radius: 5px; /* Rounded corners */
  cursor: pointer; /* Pointer cursor on hover */
}

/* Result section styling */
#result {
  margin-top: 15px; /* Space above the result */
  font-size: 18px; /* Font size */
  color: #333; /* Dark gray text */
  opacity: 0; /* Initially hidden */
}
```

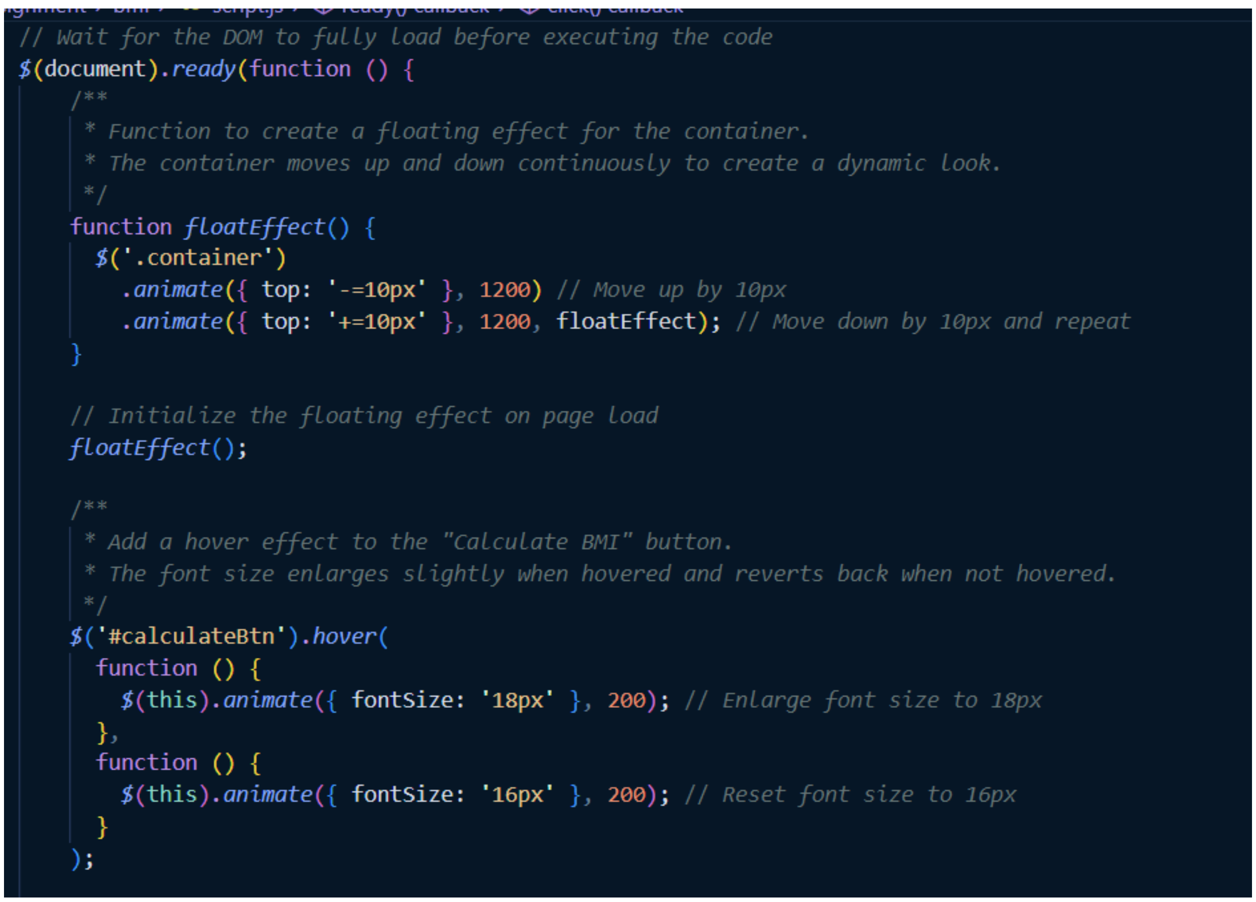
Figure 21 CSS CODE 2 BMI

Key Points:

1. **Background Design:**
   - Gradient colors (`#212020` to `hsl(30, 80%, 42%)`) provide a modern look.
2. **Container Styling:**
   - Centrally aligned and features a shadowed, rounded card for a professional appearance.
3. **Button Styling:**
   - Uses a gradient background (`#8e0142` to `#45815c`) and hover effects to improve interactivity.
4. **Responsive Design:**

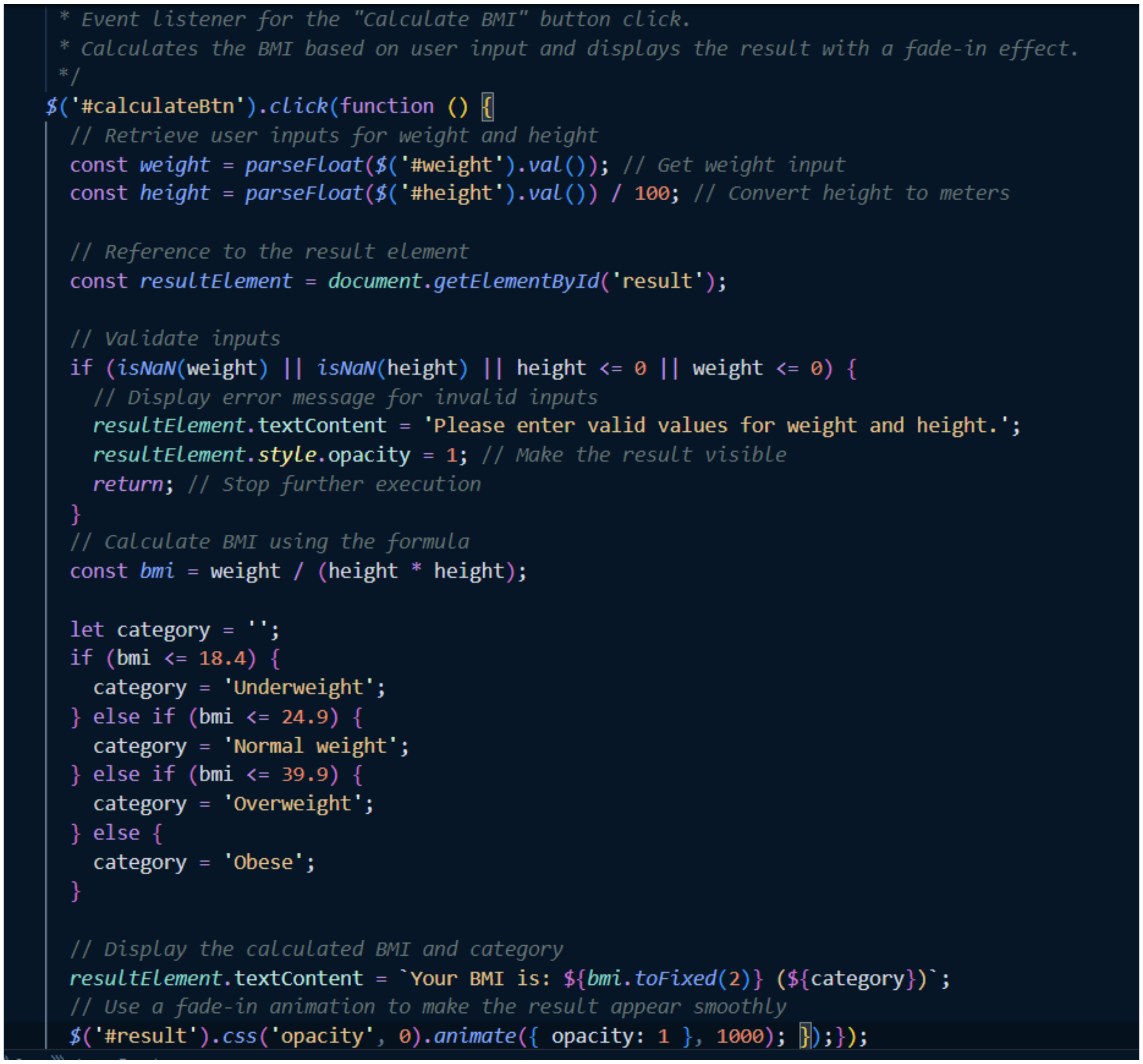
- ○ Ensures compatibility across devices by maintaining proportional widths for input fields and buttons.

3.4.3 jQuery Functionality



```
// Wait for the DOM to fully load before executing the code
$(document).ready(function () {
    /**
     * Function to create a floating effect for the container.
     * The container moves up and down continuously to create a dynamic look.
     */
    function floatEffect() {
      $('.container')
        .animate({ top: '-=10px' }, 1200) // Move up by 10px
        .animate({ top: '+=10px' }, 1200, floatEffect); // Move down by 10px and repeat
    }

    // Initialize the floating effect on page load
    floatEffect();

    /**
     * Add a hover effect to the "Calculate BMI" button.
     * The font size enlarges slightly when hovered and reverts back when not hovered.
     */
    $('#calculateBtn').hover(
      function () {
        $(this).animate({ fontSize: '18px' }, 200); // Enlarge font size to 18px
      },
      function () {
        $(this).animate({ fontSize: '16px' }, 200); // Reset font size to 16px
      }
    );
```

Figure 22 BMI JQUERY CODE 1



```
 * Event listener for the "Calculate BMI" button click.
 * Calculates the BMI based on user input and displays the result with a fade-in effect.
 */
$('#calculateBtn').click(function () {
  // Retrieve user inputs for weight and height
  const weight = parseFloat($('#weight').val()); // Get weight input
  const height = parseFloat($('#height').val()) / 100; // Convert height to meters

  // Reference to the result element
  const resultElement = document.getElementById('result');

  // Validate inputs
  if (isNaN(weight) || isNaN(height) || height <= 0 || weight <= 0) {
    // Display error message for invalid inputs
    resultElement.textContent = 'Please enter valid values for weight and height.';
    resultElement.style.opacity = 1; // Make the result visible
    return; // Stop further execution
  }
  // Calculate BMI using the formula
  const bmi = weight / (height * height);

  let category = '';
  if (bmi <= 18.4) {
    category = 'Underweight';
  } else if (bmi <= 24.9) {
    category = 'Normal weight';
  } else if (bmi <= 39.9) {
    category = 'Overweight';
  } else {
    category = 'Obese';
  }

  // Display the calculated BMI and category
  resultElement.textContent = `Your BMI is: ${bmi.toFixed(2)} (${category})`;
  // Use a fade-in animation to make the result appear smoothly
  $('#result').css('opacity', 0).animate({ opacity: 1 }, 1000); });});
```

Figure 23 BMI JQUERY CODE 2

a. Initialization

- All jQuery functions are encapsulated within the `$(document).ready()` method to ensure DOM readiness.

b. Visual Effects

1. **Floating Animation:**

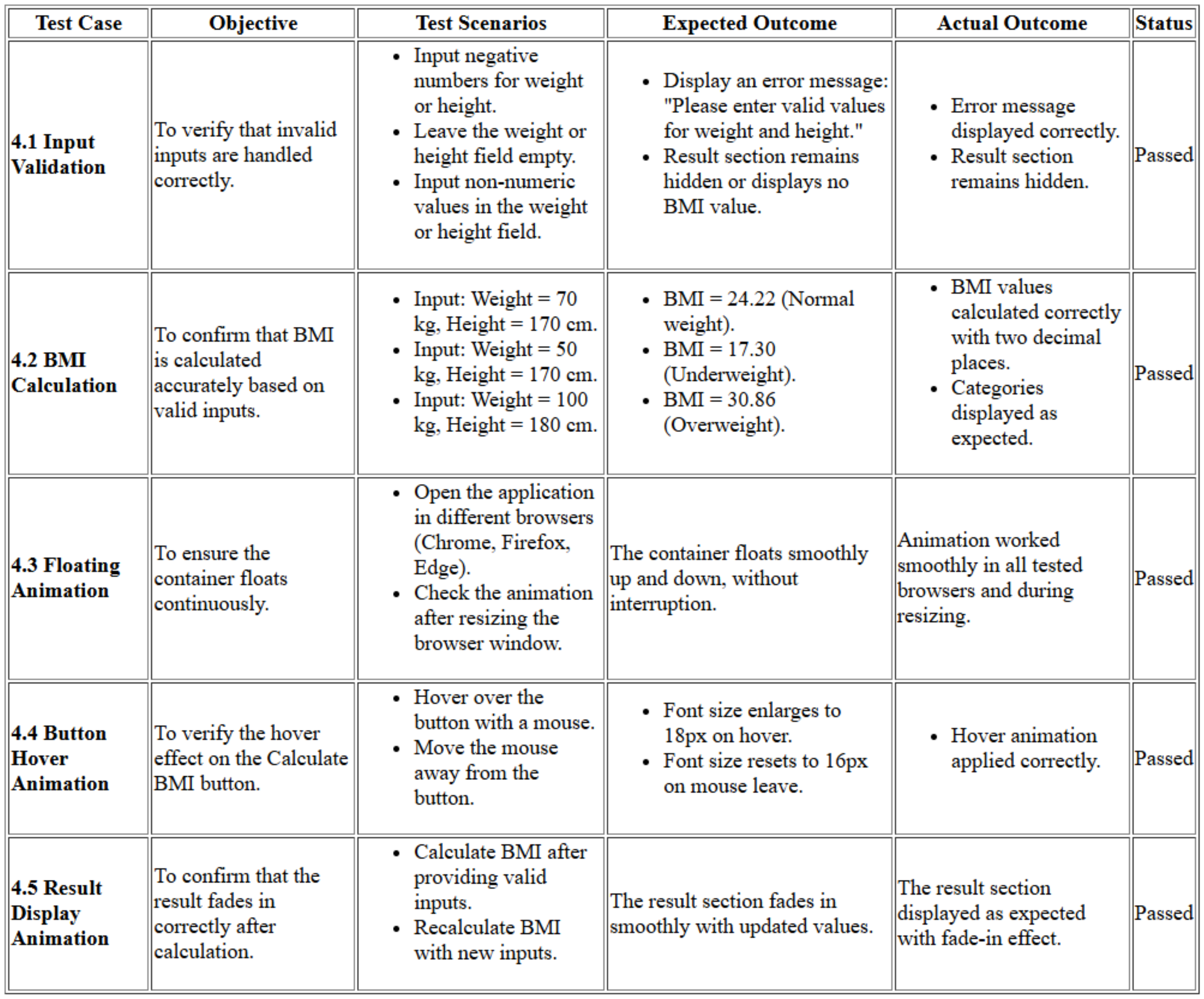
- A subtle floating effect is applied to the container using the `animate()` method:
    - Moves the container up and down continuously for a dynamic appearance.

2. **Button Hover Effect:**
    - The hover effect on the "Calculate BMI" button enlarges the font size slightly:
        - Enlarges the font size to 18px on mouse enter.
        - Resets to 16px on mouse leave.

c. BMI Calculation Logic

1. **Retrieve and Process Inputs:**
    - User inputs for weight and height are retrieved and validated.
2. **Input Validation:**
    - Ensures both fields are non-empty and positive.
    - Displays an error message for invalid inputs.
3. **BMI Calculation Formula:**
    - BMI = weight / (height * height), where height is converted to meters.
4. **BMI Categories:**
    - **Underweight:** BMI ≤ 18.4.
    - **Normal weight:** 18.5 ≤ BMI ≤ 24.9.
    - **Overweight:** 25.0 ≤ BMI ≤ 39.9.
    - **Obese:** BMI ≥ 40.
5. **Result Display:**
    - Dynamically updates the result text with BMI value and category.
    - Applies a fade-in animation using `animate()` for a smooth appearance.

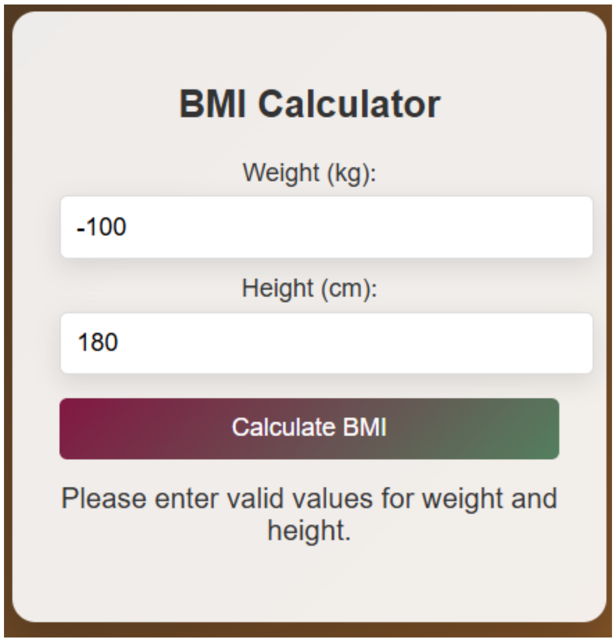
3.5 Testing Table



Test Case	Objective	Test Scenarios	Expected Outcome	Actual Outcome	Status
4.1 Input Validation	To verify that invalid inputs are handled correctly.	• Input negative numbers for weight or height. • Leave the weight or height field empty. • Input non-numeric values in the weight or height field.	• Display an error message: "Please enter valid values for weight and height." • Result section remains hidden or displays no BMI value.	• Error message displayed correctly. • Result section remains hidden.	Passed
4.2 BMI Calculation	To confirm that BMI is calculated accurately based on valid inputs.	• Input: Weight = 70 kg, Height = 170 cm. • Input: Weight = 50 kg, Height = 170 cm. • Input: Weight = 100 kg, Height = 180 cm.	• BMI = 24.22 (Normal weight). • BMI = 17.30 (Underweight). • BMI = 30.86 (Overweight).	• BMI values calculated correctly with two decimal places. • Categories displayed as expected.	Passed
4.3 Floating Animation	To ensure the container floats continuously.	• Open the application in different browsers (Chrome, Firefox, Edge). • Check the animation after resizing the browser window.	The container floats smoothly up and down, without interruption.	Animation worked smoothly in all tested browsers and during resizing.	Passed
4.4 Button Hover Animation	To verify the hover effect on the Calculate BMI button.	• Hover over the button with a mouse. • Move the mouse away from the button.	• Font size enlarges to 18px on hover. • Font size resets to 16px on mouse leave.	• Hover animation applied correctly.	Passed
4.5 Result Display Animation	To confirm that the result fades in correctly after calculation.	• Calculate BMI after providing valid inputs. • Recalculate BMI with new inputs.	The result section fades in smoothly with updated values.	The result section displayed as expected with fade-in effect.	Passed

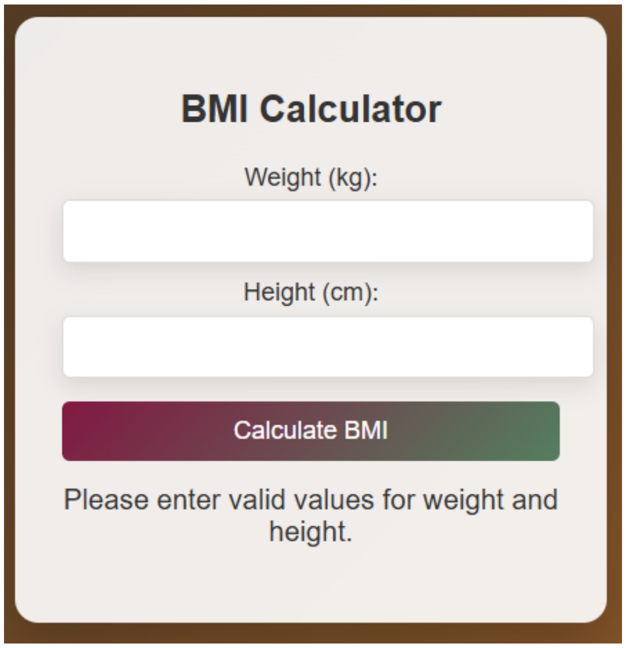
Figure 24 BMI Testing table

3.6 Testing Images



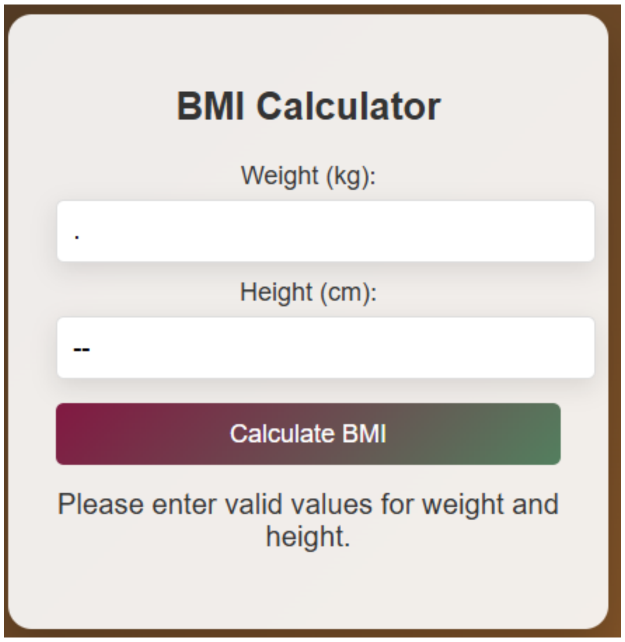
Please enter valid values for weight and height.

Figure 25 TEST CASE 1.1 Input negative numbers for weight or height.



Please enter valid values for weight and height.

Figure 26 TEST CASE 1.2 Leave the weight or height field empty


BMI Calculator

Weight (kg):

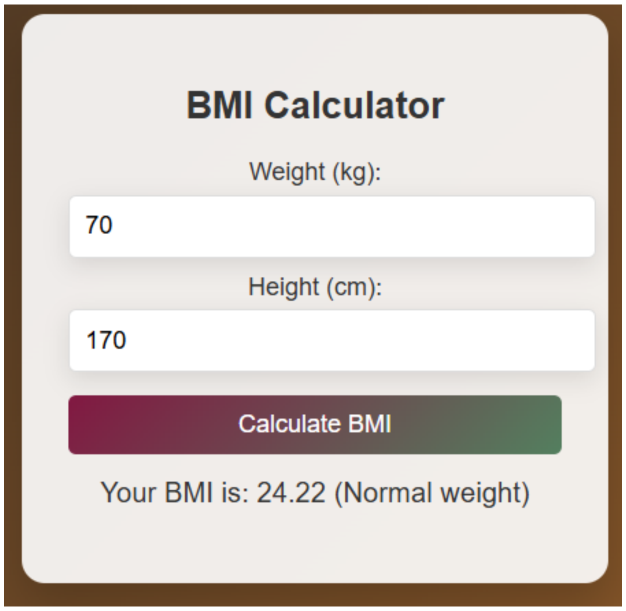
.

Height (cm):

--

Please enter valid values for weight and height.

Figure 27 TEST CASE 1.3 Input non-numeric values in the weight or height field


BMI Calculator

Weight (kg):

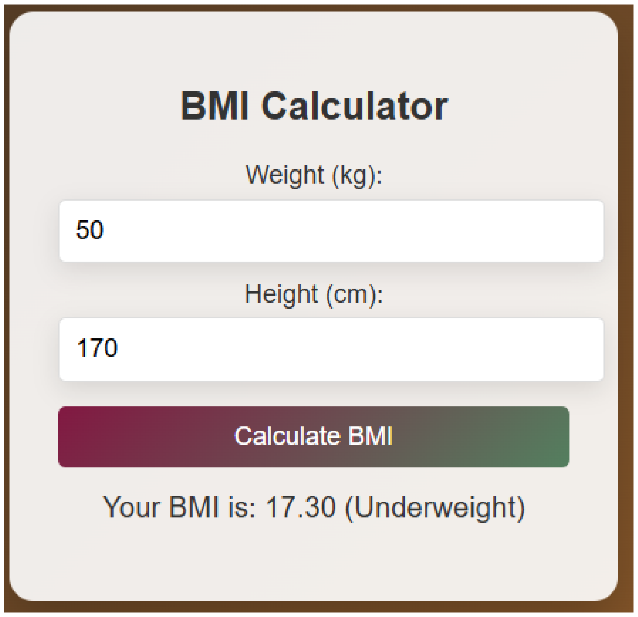
70

Height (cm):

170

Your BMI is: 24.22 (Normal weight)

Figure 28 TEST CASE 2.1 Input: Weight = 70 kg, Height = 170 cm.



BMI Calculator

Weight (kg):

50

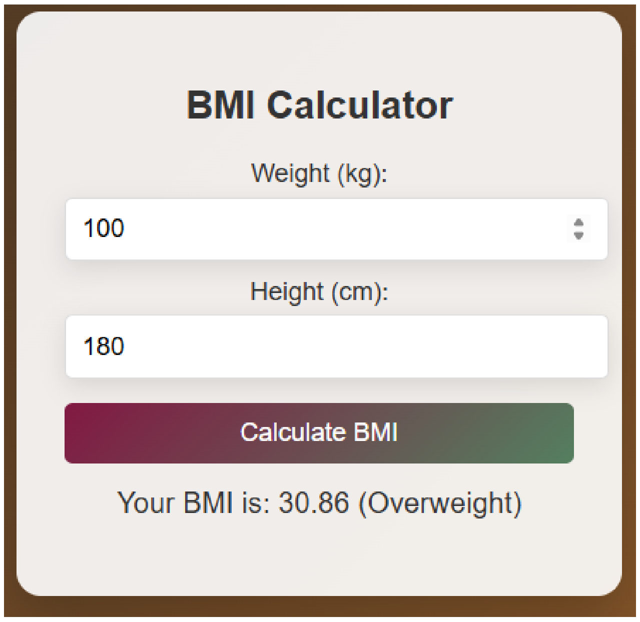
Height (cm):

170

Calculate BMI

Your BMI is: 17.30 (Underweight)

Figure 29 TEST CASE 2.2 Input: Weight = 50 kg, Height = 170 cm.



BMI Calculator

Weight (kg):

100

Height (cm):

180

Calculate BMI

Your BMI is: 30.86 (Overweight)

Figure 30 TEST CASE 2.3 Input: Weight = 100 kg, Height = 180 cm

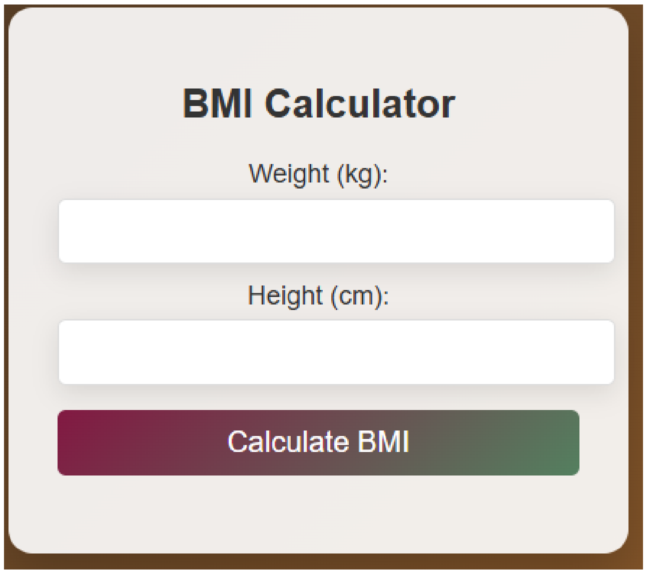


Figure 31 TEST CASE 4 verifying the hover effect on the Calculate BMI button, Font size enlarges to 18px on hover.