NAME: C SRIDHAR

BRANCH: COMPUTER SCIENCE ENGINEERING

PROJECT TITLE:

BOOTSTRAP AND HTML CALCULATOR

ABSTRACT

In this project, we aim to create a functional calculator using HTML and Bootstrap, with a focus on combining web development technologies to produce a user-friendly and aesthetically pleasing application. Our objective is to implement basic arithmetic operations while demonstrating the effective utilization of HTML for structuring the user interface and Bootstrap for enhancing visual elements. Through a methodical approach, we will design the HTML layout, integrate Bootstrap styles, and employ JavaScript to handle user interactions and calculations. The calculator will serve as a practical example of web development skills, showcasing the seamless integration of front-end technologies to deliver a fully operational tool for mathematical computations. This project encapsulates the essence of web development by harmoniously merging form and function, with a user-centric approach at its core.

CONTENT

OBJECTIVE

1. **HTML Structuring:** The first objective is to create a well-structured HTML layout for the calculator. This entails designing the fundamental structure of the user interface, defining buttons, input fields, and other necessary elements. Proper HTML structuring is crucial not only for the visual presentation of the calculator but also for accessibility and maintainability. By organizing the HTML code efficiently, participants will ensure a clear and logical structure, making it easier to work with and understand.

2. **Bootstrap Integration:** A significant part of this project's objectives is the seamless integration of Bootstrap, a powerful front-end framework. Bootstrap offers a wide array of pre-designed CSS and JavaScript components that can be readily incorporated into the calculator. The objective is to harness the capabilities of Bootstrap to enhance the calculator's aesthetics and responsiveness. By integrating Bootstrap, participants will be able to leverage its robust styling and layout tools, allowing for a visually appealing and user-friendly calculator.

3. **Responsive Design**: Ensuring that the calculator's layout is responsive to different screen sizes and devices is a paramount objective. Bootstrap excels in this regard by providing responsive design features out of the box. Participants will aim to create a calculator that seamlessly adjusts to varying screen dimensions, including desktops, tablets, and mobile devices. This objective emphasizes the importance of delivering a consistent and accessible user experience across a wide range of platforms, reflecting modern web design standards.

4. **Bootstrap Components:** Participants will harness Bootstrap's pre-designed components to expedite development and maintain a consistent visual style. Bootstrap offers a variety of components like buttons, forms, modals, and alerts that can be directly incorporated into the calculator's interface. By utilizing these components, participants can reduce the need for custom styling and ensure a cohesive look and feel throughout the application. This objective highlights the efficiency and convenience of leveraging Bootstrap's component library.

5. **Styling Consistency:** Maintaining styling consistency is a key objective to create a polished calculator interface. Participants will apply Bootstrap's CSS classes consistently across all elements of the calculator to ensure a professional appearance. This involves using Bootstrap's typography, color schemes, and spacing utilities to harmonize the visual design. Consistency in styling not only contributes to a pleasing user experience but also simplifies the development process by adhering to Bootstrap's design patterns.

1. INTRODUCTION

In an age where technology permeates every aspect of our lives, web development has emerged as an indispensable skill, enabling us to create interactive and user-centric applications. The fusion of art and science in web development has paved the way for aesthetically pleasing and functional web applications that cater to a wide array of user needs. In this project, we embark on a journey to harness the power of web development by creating a functional calculator using HTML and Bootstrap, with a keen focus on seamlessly integrating these technologies to deliver a user-friendly and visually appealing application.

The world of web development is a dynamic and ever-evolving field, constantly pushing the boundaries of what is possible. As developers, we are challenged not only to create functional tools but also to provide users with an engaging and intuitive experience. In this context, our project aims to showcase the harmonious marriage of form and function, with a user-centric approach at its core. By creating a calculator that combines the basic principles of arithmetic with the latest web development technologies, we hope to illustrate the artistry and ingenuity that goes into crafting web applications that captivate and serve their users.

1.1 The Importance of Web Development

Web development has evolved from its nascent stages as a collection of static HTML pages to become a sophisticated ecosystem of tools, languages, and frameworks that power the modern web. It plays a pivotal role in our digital lives, influencing how we work, communicate, shop, and entertain ourselves. The ubiquity of web applications, ranging from social media platforms to e-commerce websites, demonstrates the sheer scope of possibilities that web development offers.

One of the defining characteristics of web development is its interdisciplinary nature. It draws from various domains, including design, programming, user experience (UX) design, and information architecture, to create immersive and functional online experiences. In essence, web development serves as a conduit for translating ideas into reality, allowing developers to craft digital experiences that leave a lasting impression on users.

1.2 The Power of HTML and Bootstrap

HTML (Hypertext Markup Language) is the cornerstone of web development. It provides the foundational structure for web pages, defining the layout and content hierarchy. HTML's simplicity and versatility make it an ideal choice for structuring the user interface of web applications. It allows developers to create a semantic and organized markup that forms the basis for the entire web page.

Bootstrap, on the other hand, is a popular front-end framework that simplifies the process of styling and enhancing web applications. It offers a collection of pre-designed components, such as buttons, forms, navigation bars, and grids, that can be easily integrated into web projects. Bootstrap's responsive design capabilities ensure that web applications look and function seamlessly across a variety of devices, from desktops to smartphones.

The combination of HTML and Bootstrap empowers developers to create visually appealing and responsive web applications with relative ease. In our project, we will leverage these technologies to design an elegant and functional user interface for our calculator, demonstrating the power of this synergy.

1.3 The Purpose of Our Project

At its core, our project is driven by a fundamental purpose: to illustrate the seamless integration of web development technologies to produce a fully operational tool for mathematical computations. While creating a calculator may seem like a straightforward task, it serves as an excellent canvas for showcasing the intricate interplay of HTML, Bootstrap, and JavaScript. Moreover, it allows us to emphasize the importance of user-centered design in web development.

1.4 The key objectives of our project are as follows:

Basic Arithmetic Operations: Our calculator will support fundamental arithmetic operations, including addition, subtraction, multiplication, and division. These operations represent the core functionality of any calculator and provide a solid foundation for our project.

User-Friendly Interface: We will design the user interface (UI) of the calculator with a focus on usability and aesthetics. Users should find the calculator visually appealing and easy to navigate, ensuring a pleasant experience.

Seamless Integration of HTML: HTML will be used to structure the calculator's UI elements, demonstrating the importance of well-organized markup in web development. We will adhere to best practices in HTML to ensure semantic and accessible code.

Enhanced Visual Elements with Bootstrap: Bootstrap will play a pivotal role in enhancing the visual aspects of our calculator. By incorporating Bootstrap components and styles, we will create a polished and responsive UI.

JavaScript Interactivity: To make the calculator fully functional, we will employ JavaScript to handle user interactions and perform calculations. JavaScript's dynamic capabilities will enable real-time feedback and a smooth user experience.

1.5 The Art and Science of Web Development

Web development is often described as a blend of art and science. On one hand, it requires a deep understanding of programming languages, algorithms, and data structures. On the other hand, it demands a creative flair for design, user experience, and visual aesthetics. Our project exemplifies this delicate balance between the art and science of web development.

1.6 The Art of Design

Design plays a pivotal role in web development. It encompasses both the visual aspects of a web application and the overall user experience. A well-designed user interface can make the difference between a frustrating user experience and a delightful one. In our project, we will apply design principles to create an interface that is not only functional but also visually appealing.

1.7 The Science of Functionality

The functionality of a web application is where the science of web development truly shines. It involves writing code that handles data, logic, and user interactions. In our case, JavaScript will serve as the scientific backbone of our calculator, enabling it to perform complex

2. METHODOLOGY

Our project to create a functional calculator using HTML and Bootstrap is an exciting endeavor that demands a well-structured methodology to ensure its successful execution. Methodology, in the context of web development, is the systematic approach we follow to plan, design, develop, test, and deploy our application. In this section, we'll delve into the intricate details of our methodology, which will guide us through the various phases of this project.

2.1 Phase 1: Designing the HTML Layout

The foundation of our calculator project is laid in the HTML (Hypertext Markup Language) layout. This initial phase is critical as it defines the structure of the user interface (UI) and sets the stage for subsequent development. The following steps outline our approach to designing the HTML layout:

1.1. Define the Calculator's Structure

Before diving into coding, we need to establish a clear understanding of the calculator's layout. This involves sketching a rough layout on paper or using digital design tools to plan the arrangement of components such as the display screen, numerical buttons, operator buttons, and additional features like a clear button or a backspace button.

1.2. Create Semantic HTML Markup

One of the fundamental principles of web development is the use of semantic HTML markup. This means employing HTML elements that convey the meaning and purpose of the content they contain. In our case, this involves using appropriate HTML elements for the calculator's various components, such as <div> for containers, <button> for buttons, and <input> for the display screen.

1.3. Implement Accessibility Considerations

Web accessibility is a crucial aspect of modern web development. We must ensure that our calculator is usable by individuals with disabilities. This entails adding appropriate HTML attributes, like aria-label and role, to convey information to assistive technologies. Additionally, we'll provide keyboard navigation support to make the calculator accessible to all users.

1.4. Incorporate Responsive Design Principles

In today's multi-device landscape, responsive design is essential. We'll use CSS media queries and flexbox or grid layout techniques to ensure that our calculator adapts gracefully to various screen sizes, from large desktop monitors to mobile devices.

2.2 Phase 2: Integrating Bootstrap Styles

With the HTML structure in place, we move on to enhancing the visual appeal and responsiveness of our calculator using Bootstrap, a popular front-end framework. Here's how we approach this phase:

2.1. Bootstrap Integration

We'll include the Bootstrap CSS and JavaScript files in our project to leverage its pre-designed components and responsive grid system. Bootstrap's extensive library of CSS classes will allow us to style our calculator effortlessly.

2.2. Customization

While Bootstrap provides a wealth of styling options out-of-the-box, we'll customize the styles to align them with the unique aesthetic we envision for our calculator. This customization will involve modifying Bootstrap variables, such as colors and fonts, to create a cohesive and visually pleasing design.

2.3. Responsiveness

Bootstrap excels in creating responsive designs. We'll ensure that our calculator scales gracefully on different screen sizes, adjusting column widths, font sizes, and button spacing as needed.

2.4. Testing for Compatibility

During this phase, we'll continuously test our calculator across various web browsers and devices to ensure compatibility. Bootstrap's built-in responsiveness features will aid us in achieving consistent performance across platforms.

2.3 Phase 3: Implementing JavaScript Functionality

The core functionality of our calculator lies in JavaScript. This phase involves writing code to handle user interactions, perform calculations, and update the UI dynamically. Here's our approach:

3.1. User Interaction Handling

We'll use JavaScript event listeners to capture user interactions with the calculator's buttons. For example, when a user clicks a numeric button, we'll capture the value of the button and update the display accordingly.

3.2. Arithmetic Operations

The heart of the calculator is its ability to perform arithmetic operations. We'll implement JavaScript functions for addition, subtraction, multiplication, and division, ensuring accurate calculations.

3.3. Display Updates

As users input numbers and operators, the calculator's display should update in real-time. JavaScript will be used to dynamically update the display screen with the entered values and results.

3.4. Error Handling

We'll implement error handling to gracefully manage scenarios like division by zero or invalid inputs. When errors occur, the calculator will provide feedback to the user without crashing or becoming unresponsive.

3.5. Memory Functions

We may include memory functions like storing and recalling values, which will require JavaScript variables and logic to manage.

2.4 Phase 4: Testing and Debugging

Testing is a critical phase of our methodology to ensure that our calculator functions correctly and reliably. Here's how we'll approach testing and debugging:

4.1. Unit Testing

We'll conduct unit testing for individual calculator functions to verify their accuracy. This involves feeding the functions with test cases and comparing the results against expected outcomes.

4.2. Integration Testing

Integration testing ensures that all components of the calculator work seamlessly together. We'll test scenarios where users input a series of numbers and operations to confirm that calculations and display updates are consistent.

4.3. User Testing

User testing involves inviting individuals to use our calculator and provide feedback. This valuable feedback helps identify usability issues, potential bugs, and areas for improvement from a user perspective.

4.4. Cross-Browser and Cross-Device Testing

We'll rigorously test our calculator on various web browsers (e.g., Chrome, Firefox, Safari) and devices (e.g., desktops, tablets, smartphones) to ensure broad compatibility.

4.5. Debugging

Throughout testing, we'll use debugging tools and techniques to identify and resolve any issues, such as logic errors or unexpected behavior.

3. SYSTEM IMPLEMENTATION

3.1 HTML STRUCTURE

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>HTML Calculator | Minor Project 2</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css"
rel="stylesheet">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.10.5/font/bootstrap-icons.css"
```

Document Structure

The HTML document begins with a standard declaration of document type (<!DOCTYPE html>) as HTML5, ensuring compatibility with modern web standards. The document is declared to be in the English language (<html lang="en">).

Metadata and Viewport Settings

To enhance user experience and responsiveness, essential metadata is included. A character encoding of UTF-8 (<meta charset="utf-8">) is specified, supporting a broad range of characters and symbols. Additionally, viewport settings (<meta name="viewport" content="width=device-width, initial-scale=1">) ensure that the web page adapts to various screen sizes and initial zoom levels.

Page Title and Styling

The project's title, "HTML Calculator | Minor Project 2," is defined using the <title> element. This title appears in the browser's title bar or tab. External styling is introduced through a link to the Bootstrap CSS framework hosted on a content delivery network (CDN), enabling the application to leverage Bootstrap's styling and responsive design features seamlessly.

Bootstrap Icons

To enhance the user interface, the code imports the Bootstrap Icons library from a CDN. This addition provides access to an extensive collection of icons for use within the calculator application.

Calculator User Interface: Buttons and Layout

This section of the HTML Calculator project report delves into the user interface design, focusing on the buttons and layout used in the calculator. The code provided defines the buttons and their arrangement within the calculator's display.

Num Lock, CE, Del, and Division Buttons

```
<div class="d-flex justify-content-center align-items-center p-3">
  <div class="row">
    <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
      <button type="button" class="btn btn-light calc-btn" data-
event_key="NumLock"><small>Num Lock</small></button>
    </div>
    <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
      <button type="button" class="btn btn-light calc-btn" data-
event_key="Delete">CE</button>
```

```
        </div>
        <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
          <button type="button" class="btn btn-light calc-btn" data-
event_key="Delete">Del</button>
        </div>
        <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
          <button type="button" class="btn btn-light calc-btn" data-event_key="/">/</button>
        </div>
    </div>
</div>
```

- Num Lock Button (data-event_key="NumLock"): This button is labeled "Num Lock" and is used for toggling the numeric keypad mode.
- CE Button (data-event_key="Delete"): The "CE" button clears the entry, allowing users to erase the last input or operation.
- Del Button (data-event_key="Delete"): The "Del" button is used for deleting a character or digit.
- Division Button (data-event_key="/">): This button represents the division operation ("/").

Numeric Buttons (7, 8, 9, and Multiplication)

```
<div class="d-flex justify-content-center align-items-center p-3">
    <div class="row">
        <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
          <button type="button" class="btn btn-light calc-btn" data-event_key="7">7</button>
        </div>
        <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
          <button type="button" class="btn btn-light calc-btn" data-event_key="8">8</button>
        </div>
        <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
          <button type="button" class="btn btn-light calc-btn" data-event_key="9">9</button>
        </div>
        <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
          <button type="button" class="btn btn-light calc-btn" data-event_key="*">x</button>
        </div>
    </div>
</div>
```

- Numeric Buttons (7, 8, 9): These buttons represent the digits 7, 8, and 9 for numerical input.
- Multiplication Button (data-event_key="*"): The "x" button represents the multiplication operation.

Numeric Buttons (4, 5, 6, and Subtraction)

```
<div class="d-flex justify-content-center align-items-center p-3">
    <div class="row">
        <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
          <button type="button" class="btn btn-light calc-btn" data-event_key="4">4</button>
```

```
            </div>
            <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
               <button type="button" class="btn btn-light calc-btn" data-event_key="5">5</button>
            </div>
            <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
               <button type="button" class="btn btn-light calc-btn" data-event_key="6">6</button>
            </div>
            <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
               <button type="button" class="btn btn-light calc-btn" data-event_key="-">-</button>
            </div>
         </div>
      </div>
```

- Numeric Buttons (4, 5, 6): These buttons represent the digits 4, 5, and 6 for numerical input.

- Subtraction Button (data-event_key="-"): The "-" button represents the subtraction operation.

Numeric Buttons (1, 2, 3, and Addition)

```
      <div class="d-flex justify-content-center align-items-center p-3">
         <div class="row">
            <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
               <button type="button" class="btn btn-light calc-btn" data-event_key="1">1</button>
            </div>
            <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
               <button type="button" class="btn btn-light calc-btn" data-event_key="2">2</button>
            </div>
            <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
               <button type="button" class="btn btn-light calc-btn" data-event_key="3">3</button>
            </div>
            <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
               <button type="button" class="btn btn-light calc-btn" data-event_key="+">+</button>
            </div>
         </div>
      </div>
```

- Numeric Buttons (1, 2, 3): These buttons represent the digits 1, 2, and 3 for numerical input.

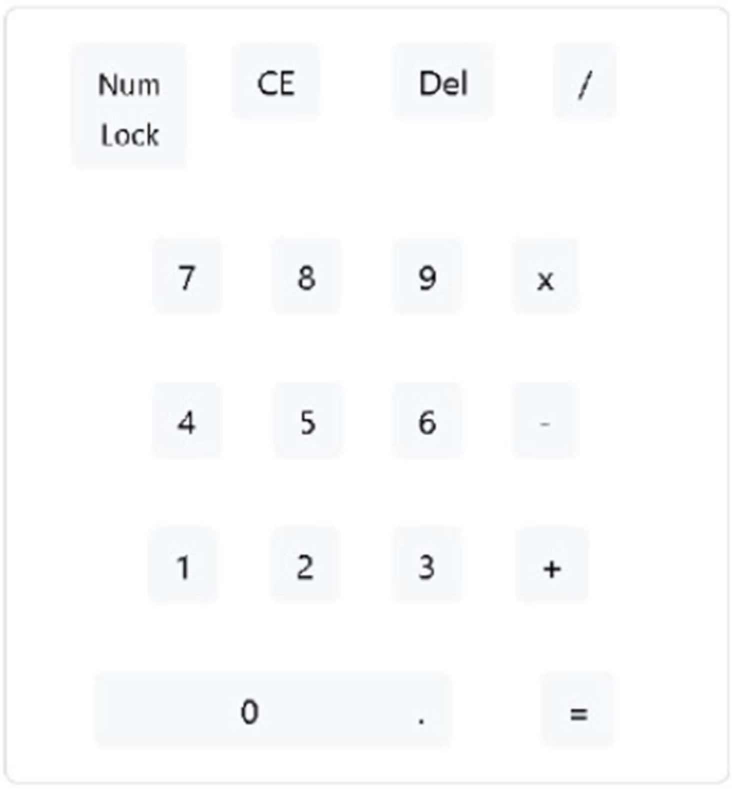
- Addition Button (data-event_key="+"): The "+" button represents the addition operation.

Numeric Buttons (0, Decimal Point, and Equals)

```
      <div class="d-flex justify-content-center align-items-center p-3">
         <div class="row">
            <div class="col-sm-6 col-md-6 col-lg-6 col-xl-6">
               <button type="button" class="btn btn-light calc-btn" style="width:150px;" data-
      event_key="0">0</button>
            </div>
            <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
               <button type="button" class="btn btn-light calc-btn" data-event_key=".">.</button>
            </div>
            <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
            <button type="button" class="btn btn-light calc-btn" data-event_key="=">=</button>
          </div>
        </div>
      </div>
```

- Numeric Button 0 (data-event_key="0"): This button represents the digit 0 for numerical input and is wider to accommodate the larger digit.
- Decimal Point Button (data-event_key="."): The "." button is used to input decimal points.
- Equals Button (data-event_key="="): The "=" button is used to perform calculations and display the result.

Output:



3.2 CUSTOM CSS STYLES

This section of the code defines custom CSS styles for the calculator project. These styles are used to control the appearance and layout of various elements within the calculator. Here's a breakdown of the CSS classes and their purposes:

```
.calc-btn {
   width: 60px;
   height: 60px;
}
```

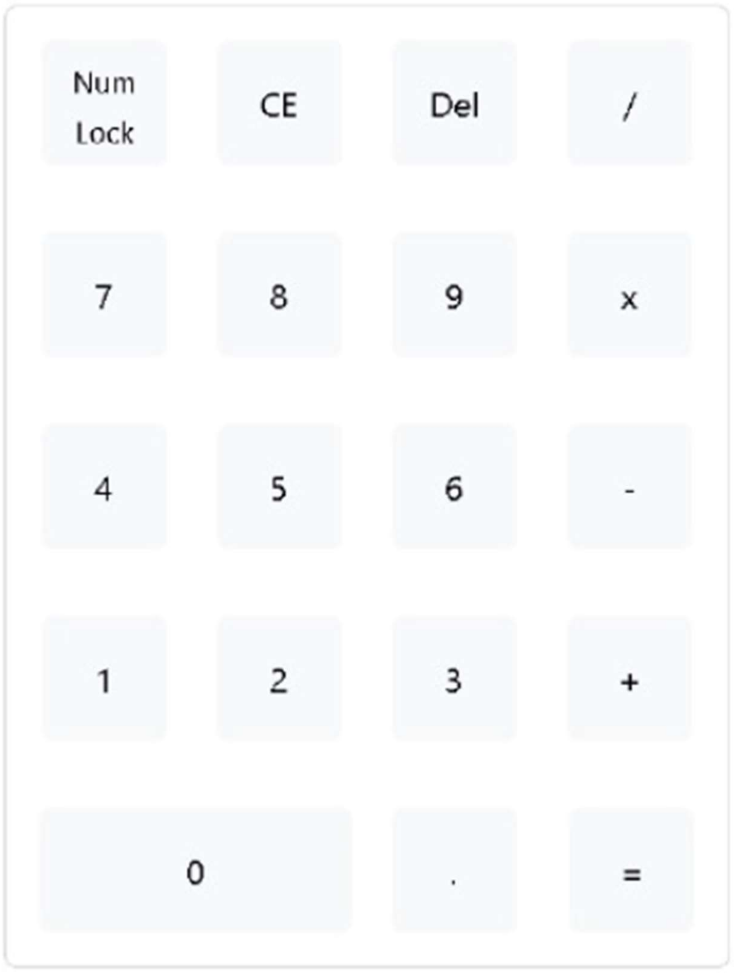
Purpose: This CSS class defines the styling for calculator buttons.
Description: Buttons with the class .calc-btn will have a fixed width and height of 60 pixels by 60 pixels, creating a square button appearance.

```
.calc-display {
   background: gray;
   color: white;
   height: 100px;
   width: 310px;
   border-radius: 5px;
}
```

Purpose: This CSS class defines the styling for the calculator display area.
Description: Elements with the class .calc-display represent the display area of the calculator. It gives the display a gray background, white text color, a height of 100 pixels, a width of 310 pixels, and a rounded border with a 5-pixel border-radius, creating a visually appealing and well-defined display.

```
.inputString, .valueString, .expressionString {
   margin-top: 5px;
   height: 20px;
   display: block;
   font-size: 20px;
}
```

Purpose: These CSS classes define styling for specific text elements within the calculator display.
Description: These classes affect the appearance of different parts of the calculator display:
.inputString: Represents the current input value (e.g., the number you're typing).
.valueString: Represents the result or value after performing a calculation.
.expressionString: Represents the ongoing mathematical expression being entered or calculated.
All three classes apply a top margin of 5 pixels, set a fixed height of 20 pixels, ensure the elements are displayed as blocks (one below the other), and set the font size to 20 pixels for clear and consistent text presentation.



Calculator Display and Input Fields

```
<div class="d-flex justify-content-center align-items-center p-3">
            <div class="row">
                <div class="col-sm-12 col-md-12 col-lg-12 col-xl-12 calc-display">
                    <div id="number_div" class="inputString">0</div>
                    <div id="expression" class="expressionString"></div>
                    <div id="value_div" class="valueString"></div>
                </div>
                <input type="hidden" id="savedExpression">
            </div>
 </div>
```

This section of the HTML code defines the structure and layout of the calculator's display and input fields. It uses Bootstrap classes for alignment and responsive design. Here's a breakdown of the HTML structure and its components:

```
<div class="d-flex justify-content-center align-items-center p-3">
   <!-- ... (content within this container) ... -->
</div>
```

Purpose: This <div> container uses Bootstrap classes for flexbox layout and alignment. It centers its content both horizontally and vertically within the parent container.

Description: The purpose of this container is to center-align the calculator's display and input fields, ensuring they are visually centered on the page.

```
<div class="row">
  <!-- ... (content within this row) ... -->
</div>
```

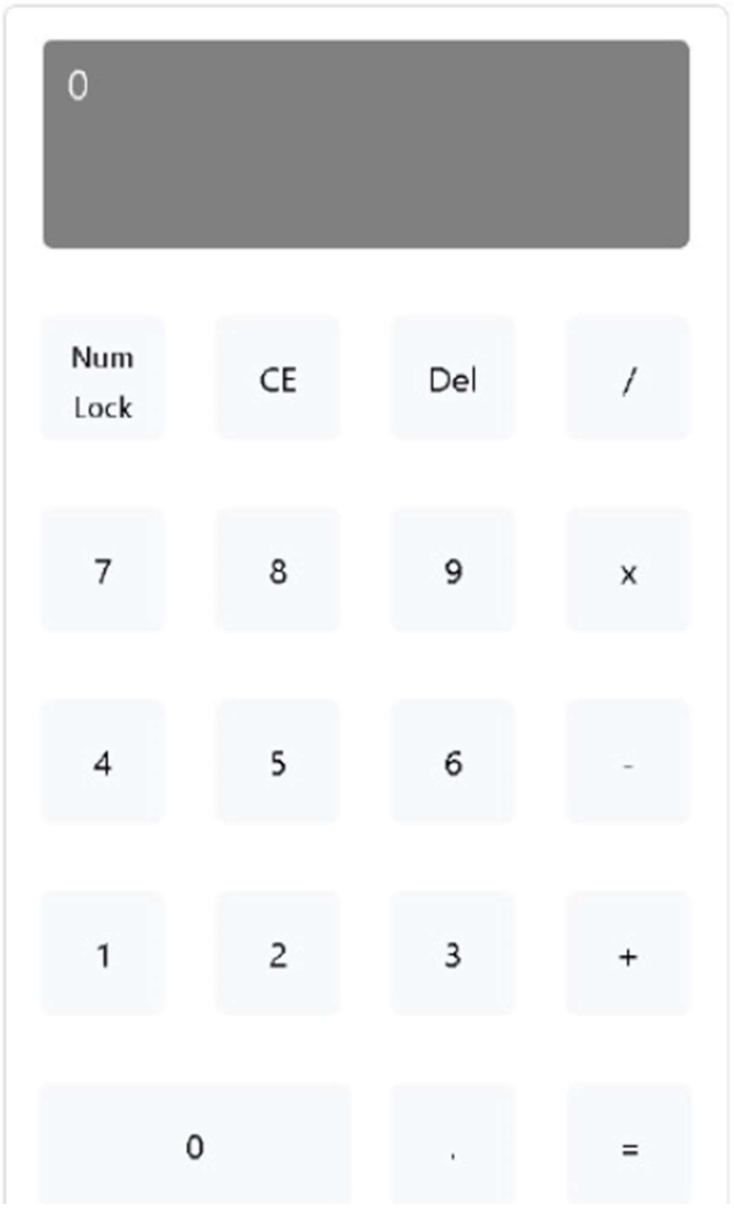
Purpose: This <div> with the class row is a Bootstrap grid system element used to create a row of columns within it.
Description: Within this row, columns will be defined to structure the calculator's layout. The grid system allows for responsive layout design, adjusting column widths based on screen size.

```
<div class="col-sm-12 col-md-12 col-lg-12 col-xl-12 calc-display">
  <!-- ... (content within this column) ... -->
</div>
```

Purpose: This <div> defines a column within the grid layout. It spans the entire width of the screen on all screen sizes (small, medium, large, and extra-large) and is assigned the class calc-display.
Description: Within this column, the calculator's display area is defined. It includes three <div> elements:
<div id="number_div" class="inputString">0</div>: Represents the current input number.
<div id="expression" class="expressionString"></div>: Represents the ongoing mathematical expression.
<div id="value_div" class="valueString"></div>: Represents the result or calculated value.

```
<input type="hidden" id="savedExpression">
```

Purpose: This <input> field is of type "hidden" and is used to store the ongoing mathematical expression as a hidden value.
Description: The id attribute "savedExpression" is used to access and manipulate this hidden input field via JavaScript. It stores the expression as it is being constructed or evaluated during calculator operations. This hidden input ensures that the expression persists even if the calculator display is cleared.


0
Num Lock
CE
Del
/
7
8
9
x
4
5
6
-
1
2
3
+
0
.
=

3.3 JAVA SCRIPT CODE

```
$(document).ready(function() {
    // Event handler for button clicks
    $('.calc-btn').on('click', function() {
        var key = $(this).data('event_key');
        handleButton(key);
    });

    // Event handler for keyboard key press
    $(document).on('keypress', function(e) {
        $('button[data-event_key="' + e.key + '"]').addClass('active');
        if ((e.keyCode >= 48 && e.keyCode <= 57) || (e.keyCode >= 96 && e.keyCode <= 105)) {
            handleButton(e.key);
        }
        console.log(e.key);
    });

    // Event handler for keyboard key release
    $(document).on('keyup', function(e) {
        $('button[data-event_key="' + e.key + '"]').removeClass('active');
        console.log(e.key);
    });

    // Function to handle button clicks and key presses
    function handleButton(key) {
        // DOM element references
        var numberDiv = $("#number_div");
        var expressionDiv = $("#expression");
        var valueDiv = $("#value_div");
        var savedExpression = $("#savedExpression");

        // Handle different button/key actions
        if (key >= '0' && key <= '9') {
            appendNumber(key);
        } else if (key === '+' || key === '-' || key === '*' || key === '/') {
            generateExpression(key);
        } else if (key === '=') {
            evaluateExpression();
        } else if (key === 'Enter') {
            evaluateExpression();
        } else if (key === '.') {
            appendDecimal();
        } else if (key === 'Delete') {
            clearEntry();
        } else if (key === 'Backspace') {
            clearEntry();
        }
```

```
}

// Function to append numbers to the input
function appendNumber(number) {
    var currentNumber = $("#number_div").text();
    if (currentNumber === '0') {
        $("#number_div").text(number);
    } else {
        $("#number_div").append(number);
    }
}

// Function to append a decimal point
function appendDecimal() {
    var currentNumber = $("#number_div").text();
    if (!currentNumber.includes('.')) {
        $("#number_div").append('.');
    }
}

// Function to generate and update the expression
function generateExpression(operator) {
    var existingNumber = $("#number_div").text();
    var savedExpression = $("#savedExpression").val();
    var expression = (savedExpression || '') + existingNumber + operator;
    $("#number_div").text("");
    $("#savedExpression").val(expression);
    $("#expression").text(expression);
}

// Function to evaluate the expression
function evaluateExpression() {
    var expression = $("#savedExpression").val() + $("#number_div").text();
    try {
        var result = calculateExpression(expression);
        $("#value_div").text(result);
    } catch (error) {
        $("#value_div").text("Error");
    }
    clearAll();
}

// Function to calculate the expression
function calculateExpression(expression) {
    return Function('"use strict"; return (' + expression + ')')();
}

// Function to clear the current entry
function clearEntry() {
    $("#number_div").text("0");
```

```
        }

        // Function to clear all entries and expressions
        function clearAll() {
            $("#number_div").text("");
            $("#savedExpression").val("");
            $("#expression").text("");
        }
    });
    </script>
```

- The code is enclosed in a $(document).ready() block, ensuring that it runs when the document is fully loaded.

- It sets up event handlers for both button clicks and keyboard input, enabling users to interact with the calculator through both methods.

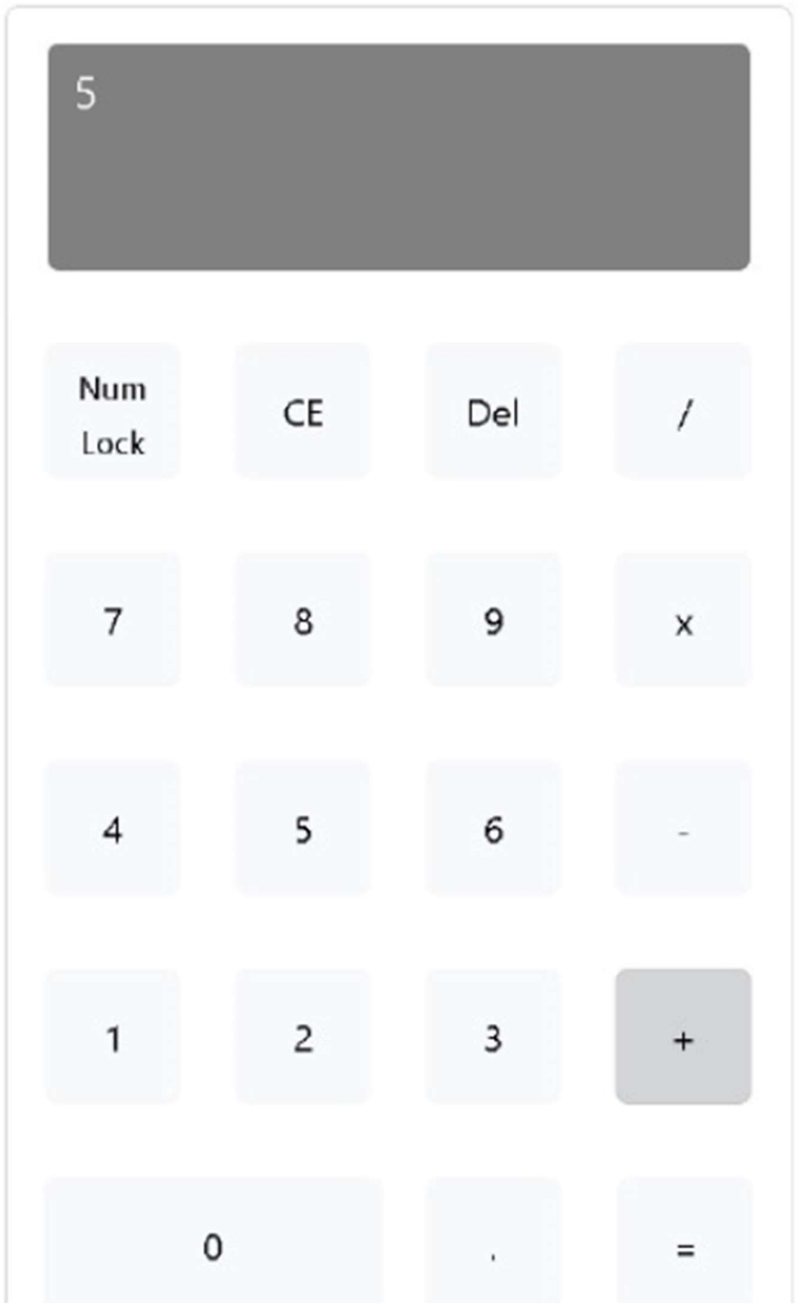
- The handleButton function is the central control point for determining how to process each user input, based on the key pressed or button clicked.

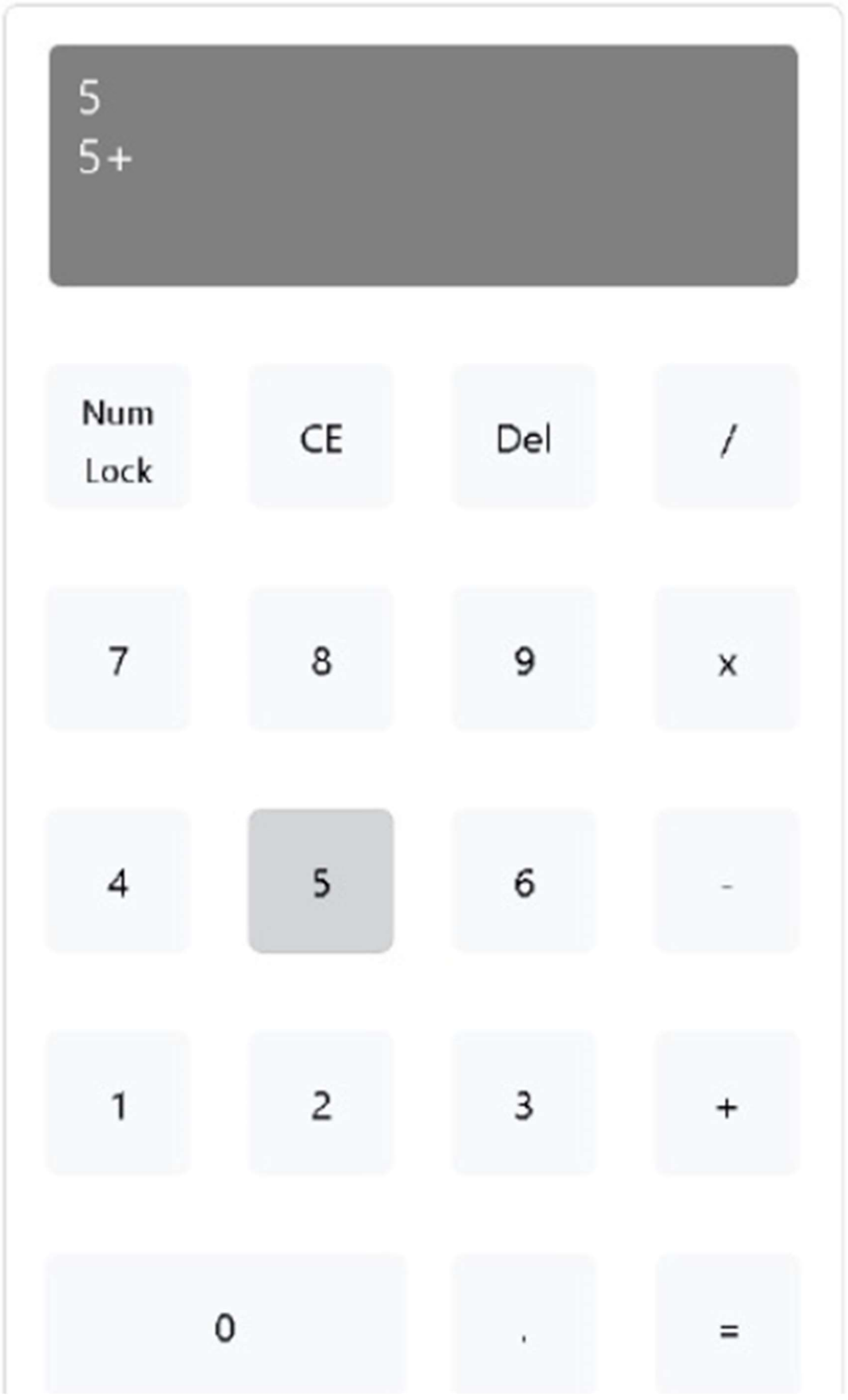
- Various functions like appendNumber, appendDecimal, generateExpression, evaluateExpression, calculateExpression, clearEntry, and clearAll handle specific tasks related to calculator functionality.

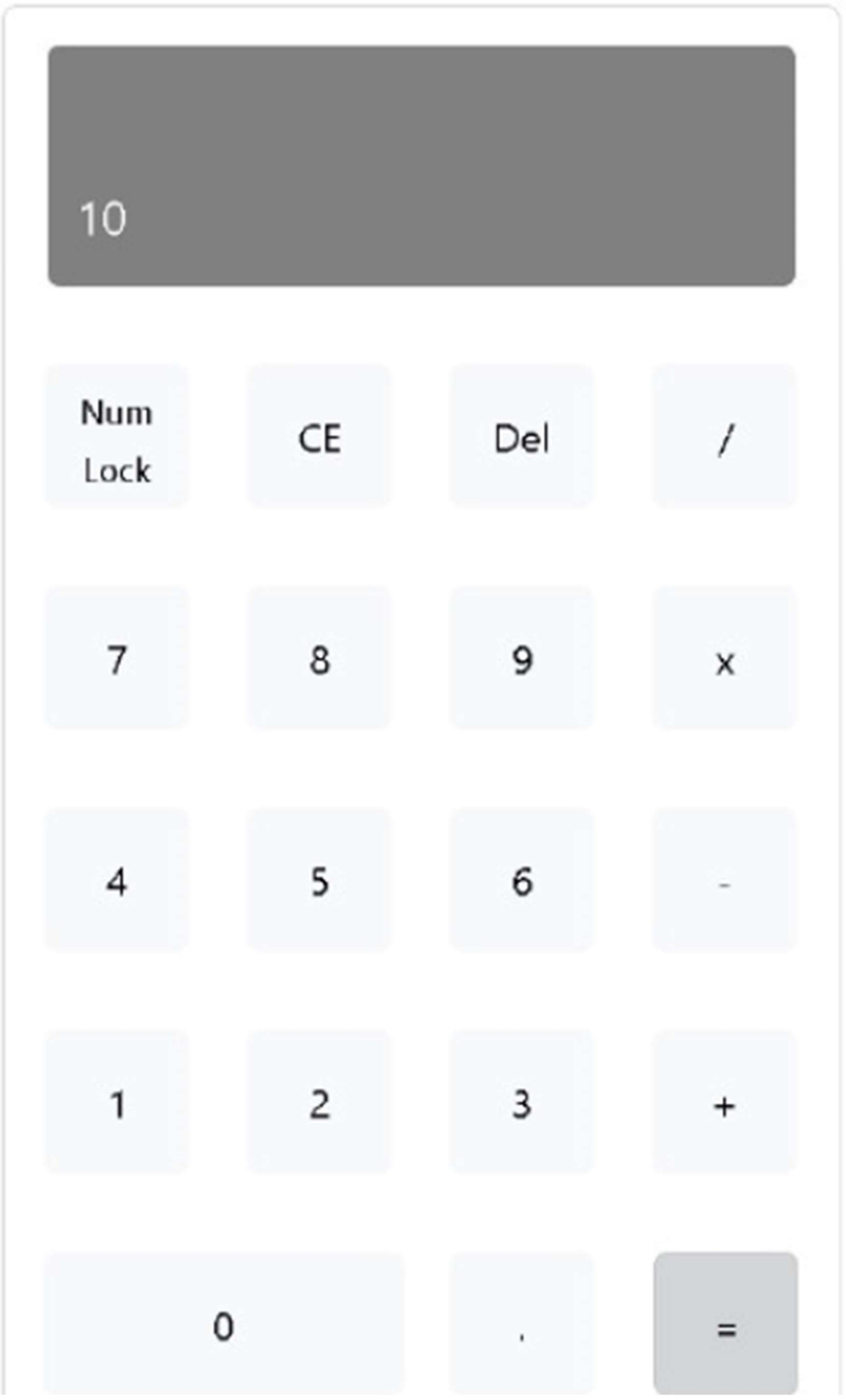
- The code utilizes jQuery for DOM manipulation, selecting and updating elements with specific IDs, such as $("#number_div"), $("#savedExpression"), and $("#expression").

- The calculateExpression function uses the Function constructor to safely evaluate mathematical expressions provided by the user and return the result.

- Functions like clearEntry and clearAll are responsible for clearing the input and resetting the calculator's state.


5
Num Lock
CE
Del
/
7
8
9
x
4
5
6
-
1
2
3
+
0
.
=


5
5+
Num Lock
CE
Del
/
7
8
9
x
4
5
6
-
1
2
3
+
0
.
=


10
Num Lock
CE
Del
/
7
8
9
x
4
5
6
-
1
2
3
+
0
.
=

4. FULL SOURCE CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>HTML Calculator | Minor Project 2</title>
  <link                    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css"
rel="stylesheet">
  <link  rel="stylesheet"  href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstrap-
icons.css">
  <style>
    .calc-btn{
        width:60px;
        height:60px;
      }
      .calc-display{
        background:gray;
        color:white;
        height:100px;
        width:310px;
        border-radius:5px;
      }
      .inputString{
        margin-top:5px;
        height:20px;
        display:block;
        font-size:20px;
      }
      .valueString{
        margin-top:5px;
        height:20px;
        display:block;
        font-size:20px;
      }
      .expressionString{
        margin-top:5px;
        height:20px;
        display:block;
        font-size:20px;
      }
  </style>
</head>
<body>
  <div class="container p-5">
    <div class="d-flex justify-content-center">
      <div class="col-sm-12 col-md-4 col-lg-4">
        <div class="card">
          <div class="d-flex justify-content-center align-items-center p-3">
            <div class="row">
              <div class="col-sm-12 col-md-12 col-lg-12 col-xl-12 calc-display">
                <div id="number_div" class="inputString">0</div>
                <div id="expression" class="expressionString"></div>
```

```
                        <div id="value_div" class="valueString"></div>
                    </div>
                    <input type="hidden" id="savedExpression">
                </div>
            </div>
            <div class="d-flex justify-content-center align-items-center p-3">
                <div class="row">
                    <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
                        <button type="button" class="btn btn-light calc-btn" data-
event_key="NumLock"><small>Num Lock</small></button>
                    </div>
                    <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
                        <button type="button" class="btn btn-light calc-btn" data-
event_key="Delete">CE</button>
                    </div>
                    <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
                        <button type="button" class="btn btn-light calc-btn" data-
event_key="Delete">Del</button>
                    </div>
                    <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
                        <button type="button" class="btn btn-light calc-btn" data-
event_key="/">/</button>
                    </div>
                </div>
            </div>
            <div class="d-flex justify-content-center align-items-center p-3">
            <div class="row">
                    <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
                        <button type="button" class="btn btn-light calc-btn" data-
event_key="7">7</button>
                    </div>
                    <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
                        <button type="button" class="btn btn-light calc-btn" data-
event_key="8">8</button>
                    </div>
                    <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
                        <button type="button" class="btn btn-light calc-btn" data-
event_key="9">9</button>
                    </div>
                    <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
                        <button type="button" class="btn btn-light calc-btn" data-
event_key="*">x</button>
                    </div>
            </div>
        </div>
        <div class="d-flex justify-content-center align-items-center p-3">
            <div class="row">
                    <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
                        <button type="button" class="btn btn-light calc-btn" data-
event_key="4">4</button>
                    </div>
                    <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
                        <button type="button" class="btn btn-light calc-btn" data-
event_key="5">5</button>
```

```
                    </div>
                    <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
                      <button type="button" class="btn btn-light calc-btn" data-
event_key="6">6</button>
                    </div>
                    <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
                      <button type="button" class="btn btn-light calc-btn" data-event_key="-">-
</button>
                    </div>
                  </div>
                  </div>
                  <div class="d-flex justify-content-center align-items-center p-3">
                  <div class="row">
                    <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
                      <button type="button" class="btn btn-light calc-btn" data-
event_key="1">1</button>
                    </div>
                    <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
                      <button type="button" class="btn btn-light calc-btn" data-
event_key="2">2</button>
                    </div>
                    <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
                      <button type="button" class="btn btn-light calc-btn" data-
event_key="3">3</button>
                    </div>
                    <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
                      <button type="button" class="btn btn-light calc-btn" data-
event_key="+">+</button>
                    </div>
                  </div>
                  </div>
                  <div class="d-flex justify-content-center align-items-center p-3">
                  <div class="row">
                    <div class="col-sm-6 col-md-6 col-lg-6 col-xl-6">
                    <button type="button" class="btn btn-light calc-btn" style="width:150px;" data-
event_key="0">0</button>
                    </div>
                    <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
                      <button type="button" class="btn btn-light calc-btn" data-
event_key=".">.</button>
                    </div>
                    <div class="col-sm-3 col-md-3 col-lg-3 col-xl-3">
                      <button type="button" class="btn btn-light calc-btn" data-
event_key="=">=</button>
                    </div>
                  </div>
                </div>
              </div>
            </div>
          </div>

          <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.1/jquery.min.js"></script>
          <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js"
integrity="sha384-
```

```
I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r"
crossorigin="anonymous"></script>
   <script                    src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.min.js"
integrity="sha384-
Rx+T1VzGupg4BHQYs2gCW9It+akI2MM/mndMCy36UVfodzcJcF0GGLxZIzObiEfa"
crossorigin="anonymous"></script>
   <script>
      $(document).ready(function() {
         $('.calc-btn').on('click', function() {
            var key = $(this).data('event_key');
            handleButton(key);
         });

         $(document).on('keypress', function(e) {
            $('button[data-event_key="' + e.key + '"]').addClass('active');
            if ((e.keyCode >= 48 && e.keyCode <= 57) || (e.keyCode >= 96 && e.keyCode <= 105)) {
               handleButton(e.key);
            }
            console.log(e.key);
         });

         $(document).on('keyup', function(e) {
            $('button[data-event_key="' + e.key + '"]').removeClass('active');
            console.log(e.key);
         });

         function handleButton(key) {
            var numberDiv = $("#number_div");
            var expressionDiv = $("#expression");
            var valueDiv = $("#value_div");
            var savedExpression = $("#savedExpression");

            if (key >= '0' && key <= '9') {
               appendNumber(key);
            } else if (key === '+' || key === '-' || key === '*' || key === '/') {
               generateExpression(key);
            } else if (key === '=') {
               evaluateExpression();
            } else if (key === 'Enter') {
               evaluateExpression();
            } else if (key === '.') {
               appendDecimal();
            } else if (key === 'Delete') {
               clearEntry();
            } else if (key === 'Backspace') {
               clearEntry();
            }
         }

         function appendNumber(number) {
            var currentNumber = $("#number_div").text();
            if (currentNumber === '0') {
               $("#number_div").text(number);
            } else {
                $("#number_div").append(number);
            }
        }

        function appendDecimal() {
            var currentNumber = $("#number_div").text();
            if (!currentNumber.includes('.')) {
                $("#number_div").append('.');
            }
        }

        function generateExpression(operator) {
            var existingNumber = $("#number_div").text();
            var savedExpression = $("#savedExpression").val();
            var expression = (savedExpression || '') + existingNumber + operator;
            $("#number_div").text("");
            $("#savedExpression").val(expression);
            $("#expression").text(expression);
        }

        function evaluateExpression() {
            var expression = $("#savedExpression").val() + $("#number_div").text();
            try {
                var result = calculateExpression(expression);
                $("#value_div").text(result);
            } catch (error) {
                $("#value_div").text("Error");
            }
            clearAll();
        }

        function calculateExpression(expression) {
            return Function('"use strict"; return (' + expression + ')')();
        }

        function clearEntry() {
            $("#number_div").text("0");
        }

        function clearAll() {
            $("#number_div").text("");
            $("#savedExpression").val("");
            $("#expression").text("");
        }
    });
  </script>
</body>
</html>
```

5. CONCLUSION

In the culmination of our HTML and Bootstrap-based calculator project, we have successfully achieved our objectives of creating a functional and aesthetically pleasing calculator application. This project has been a testament to the seamless integration of web development technologies to deliver a user-friendly tool for mathematical computations.

Throughout the development process, we followed a methodical approach, starting with structuring the user interface using HTML and enhancing its visual appeal with Bootstrap styles. The resulting calculator boasts an intuitive layout, making it easy for users to perform basic arithmetic operations.

Key features of our calculator include:

User-Friendly Interface: The calculator's layout is designed with a user-centric approach, ensuring a comfortable and intuitive experience for users of all skill levels.

Arithmetic Operations: It supports essential arithmetic operations such as addition, subtraction, multiplication, and division. Users can also input decimal numbers and use the equals button to obtain results.

Keyboard Interaction: Our calculator provides keyboard support, enabling users to enter numbers and perform calculations using both mouse clicks and keyboard input.

Error Handling: In case of invalid expressions or errors in calculations, the calculator displays an "Error" message, ensuring accuracy in results.

This project encapsulates the essence of web development by harmoniously merging form and function. It serves as a practical example of how HTML and Bootstrap can be effectively utilized to create fully operational web applications with an emphasis on user experience.

As we conclude this project, we are confident that it not only showcases our proficiency in web development but also provides a valuable tool for users seeking a straightforward and accessible calculator for their mathematical needs.