Project 2 – Bootstrap and HTML Calculator

- ABSTRACT:

Bootstrap is a popular framework for creating responsive and mobile-friendly web pages. It provides a set of predefined classes and components that can be used to style and layout HTML elements. Bootstrap also has a grid system that allows for easy alignment and positioning of elements on different screen sizes.

One of the applications of Bootstrap is to create a simple HTML calculator that can perform basic arithmetic operations. The calculator can be designed using HTML, CSS, and JavaScript. The HTML code defines the structure and content of the calculator, such as the input field, buttons, and display area. The CSS code applies the Bootstrap classes and custom styles to the HTML elements, such as colors, fonts, borders, and margins. The JavaScript code adds the functionality and logic to the calculator, such as handling user input, performing calculations, and displaying results.

The HTML project on Bootstrap and HTML Calculator demonstrates how to use Bootstrap to create a user-friendly and interactive web page that can perform simple tasks. It also shows how to combine HTML, CSS, and JavaScript to create a dynamic web application that can run on any browser or device.

- OBJECTIVE:

The objective of this project is to design and develop a web-based calculator using Bootstrap and HTML. Bootstrap is a popular framework that provides ready-made components and

styles for creating responsive and modern web pages. HTML is a markup language that defines the structure and content of web pages.

The calculator will have a simple and elegant interface that can perform basic arithmetic operations such as addition, subtraction, multiplication, and division. The calculator can also have some advanced features such as memory functions, percentage, square root, and inverse. The calculator will use Bootstrap's grid system, buttons, and fonts to create a consistent and attractive layout. The calculator will use HTML's input, output, and script elements to handle the user's input and output, and to perform the calculations using JavaScript.

The project will demonstrate the use of Bootstrap and HTML to create a functional and user-friendly web application that can run on any browser and device. The project will also evaluate the performance, usability, and accessibility of the calculator. The project will follow the best practices of web development. So, in a quick recap, these are the objectives of this project:

- To create a web-based calculator that can perform basic arithmetic operations such as addition, subtraction, multiplication, and division.
- To add some advanced features to the calculator such as memory functions, percentage, square root, and inverse.
- To use Bootstrap's grid system, buttons, icons, and fonts to create a consistent and attractive layout for the calculator.
- To use HTML's input, output, and script elements to handle the user's input and output, and to perform the calculations using JavaScript.

- To test the performance, usability, and accessibility of the calculator on various browsers and devices.
- To follow the best practices of web development.

- INTRODUCTION:

A calculator is a device or software that can perform mathematical calculations. Calculators are widely used in various fields such as education, science, engineering, finance, and everyday life. Calculators can range from simple models that can only perform basic arithmetic operations to complex ones that can handle scientific, engineering, or financial functions. Calculators can also vary in their design, interface, and features depending on the user's needs and preferences.

One of the advantages of web-based calculators is that they can be accessed from any browser and device without requiring any installation or download. Web-based calculators can also be customized and enhanced using various web technologies such as HTML, CSS, and JavaScript. HTML is a markup language that defines the structure and content of web pages. CSS is a style sheet language that specifies the appearance and layout of web pages. JavaScript is a scripting language that enables dynamic and interactive features on web pages.

Bootstrap is a popular framework that provides ready-made components and styles for creating responsive and modern web pages. Bootstrap offers a variety of elements such as grids, buttons, icons, fonts, forms, navigation bars, modals, alerts, and more. Bootstrap also uses a mobile-first approach that ensures optimal viewing and user experience across different screen sizes and devices.

The purpose of this project is to design and develop a web-based calculator using Bootstrap and HTML. The project will demonstrate how to use Bootstrap's components and styles to create a simple and user-friendly calculator interface. The project will also show how to use HTML's input, output, and script elements to handle the user's input and output, and to perform the calculations using JavaScript.

- METHODOLOGY:

A possible methodology for creating an HTML Calculator is as follows:

- **Step 1**: Create a basic HTML structure for the calculator using the <html>, <head>, and <body> tags. Include the necessary links to Bootstrap, CSS libraries in the <head> section, and JavaScript, Popper.js and jQuery libraries in the <body> section. This adds functionality and accessibility to our project.
- **Step 2**: Design the layout of the calculator using Bootstrap grid system and CSS classes. Use a element with the class "container p-5" to wrap the calculator elements. Use another element with the class "d-flex justify-content-center" to align the container in the middle of our browser page. Use another element with the class "col-sm-12 col-md-4 col-lg-4". Use another element with the class "card", adding certain CSS properties, to create the calculator template.

  Use another element with the class "d-flex justify-content-center align-items-center p-4" to create the display, and align the contents at the centre of the display of the calculator. Use another element with the class "row" to apply the input, output and current expressions in the display of the calculator. Along with it add element with class "col" to format the display expressions. Now add three

 elements with classes, each for the input expression, current expression and output expression or result. Add a hidden type <input> element to store the current expression for future operations.

Use elements with the class “d-flex justify-content-center align-items-center p-3” to create the buttons, and align them and their symbols at the centre. Now use such elements as much as rows of button is required. Here, I have used five such elements. Use another element with the class “row” to create the row of the buttons. Now, add element with class “col-sm-3 col-md-3 col-lg-3 col-xl-3” to actually add the layout or partition for the buttons. Now inside it add <button> element and stylize it as wanted. Now, the number of elements with class “col-sm-3 col-md-3 col-lg-3 col-xl-3” will decide the number of buttons in a button row. Inside the <button> element, add the text you want to appear on the button itself. For example: “+” or “9” etc. Also assign “data-event_key=(key_symbol)”, this will bind particular keystroke to a particular button.

- **Step 3**: Add some custom CSS styles to the calculator elements to make them look more appealing. For example, change the background color, font size, border radius, margin, and padding of the buttons and the display. Here, I have made the button of different colors, circular shaped, with different colored texts etc. For display, I have used different font size and colors for different expression, made the display layout itself smooth, circular and gray. Made the calculator body transparent and with shadowed border. Made the WebPage background attractive with a gradient. Other modifications can also be done.

- **Step 4**: Write the JavaScript and jQuery code to handle the logic of the calculator. Use functions to perform the basic arithmetic operations, store expressions and display expressions. Some functions can be exclusively used to deal with exceptions/errors. Use event listeners to detect when a button is clicked and update the display accordingly. Use conditional statements to handle edge cases, such as division by zero, invalid input, and overflow and other such exceptions. Here we have used the following functions:

1. $(document).on('keypress',function(e){
   $('button[data-event_key="'+e.key+'"]').addClass('active')
   if((e.keyCode >= 48 && e.keyCode <= 57)||e.key=='.'){
   appendNumber(e.key)
   }
   else{
   if(e.key=='+'||e.key=='-'||e.key=='/'||e.key=='*'){
   generateExpression(e.key)
   }
   else if(e.key==';'){
   percent()
   }
   else if(e.key == '='){
   evaluateExpression()
   }
   else if(e.key == 'a'){
   clearCalc()
   }
   else if(e.key == 'c'){
   reduceExp()
   }
   }
   })
   $(document).on('keyup',function(e){

```
$('button[data-event_key="'+e.key+'"]').removeClass('active')
})
```

This function can be used to detect a keypress and eventually can be used to detect which key is actually pressed and accordingly we can code to perform certain action when certain key is pressed. For example: we can code that if the key pressed is a number or '.', append it to the input expression. If '+', '-', '*', '/' is pressed then generate the expression with the numbers in the input expression. If '=' is pressed then generate the result of the evaluation of the expression. If 'a' is pressed then clear the calculator, if 'c' is pressed then delete the input and if ';' is pressed then calculate the percentage etc.

2.
```
$('.calc-btn').on('click',function(e){
var key = $(this).data('event_key')
if(key!='+' && key!= '-' && key!= '*' && key!='/' && key!='c' && key!=';' &&
key!='=' && key!='a'){
appendNumber(key)
}
else{
if(key=='+'||key=='-'||key=='/'||key== '*'){
generateExpression(key)
}
else if(key == '='){
evaluateExpression()
}
else if(key == 'a'){
clearCalc()
}
else if(key == 'c'){
reduceExp()
}
```

```
else if(key == ';'){
percent()
}
}
})
```

This function can be used to detect a click in the calculator GUI and eventually can be used to detect which button is actually clicked and accordingly we can code to perform certain action when certain button is clicked. For example: we can code that if the button clicked is a number or '.', append it to the input expression. If '+', '-', 'x', '÷' is clicked then generate the expression with the numbers in the input expression. If '=' is clicked then generate the result of the evaluation of the expression. If 'AC' is clicked then clear the calculator, if 'Del' is clicked then delete the input and if '%' is clicked then calculate the percentage etc.

3.
```
function appendNumber(number){
var existingNumber = $("#number_div").html();
var currentString = number;
var outputString = '';
if(existingNumber != '' && existingNumber != undefined && existingNumber != null){
if(existingNumber == '0'){
outputString = number
}
else{
outputString = existingNumber+=currentString
}
}
else{
outputString = currentString;
}
```

```
$("#number_div").html(outputString);
}
```

This function will append digits and decimal from the input to the input expression.

4.
```
function generateExpression(operator){
var existingNumber = $("#number_div").html();
var currentOperator = operator;
var expression = '';
var savedExpression = $("#savedExpression").val();
if(savedExpression == '' || savedExpression == null || savedExpression == undefined){
expression = parseFloat(existingNumber) + operator
}
else{
expression = savedExpression + existingNumber + operator;
}

$("#number_div").html("")
$("#savedExpression").val(expression)
$("#expression").html(expression)
}
```

This function generates the expression, that is, it combines the numbers and operators to generate the expression to be evaluated. Once the operator is encountered in the input, it takes the operator and apply it to the already existing expression and the input number, and assigns it back to the stored expression. It also clears the input expression for another input.

5.
```
function evaluateExpression(){
var result = '';
var expression = $("#savedExpression").val();
```

```
var existingNumber = $("#number_div").html();
if(existingNumber != " || existingNumber != null || existingNumber != undefined){
expression = expression + parseFloat(existingNumber);
$("#expression").html(expression)
}

try{
result = eval(expression);

$("#savedExpression").val("")
$("#number_div").html(result)
if((result+").length>17){
result = (result+"").substring(0,17)
$("#value_div").html(result)
}
else{
$("#value_div").html(result)
}
if($("#number_div").html()==""){
$("#number_div").html("")
$("#expression").html("")
$("#value_div").html("Error")
}
}

catch(error){
if(error instanceof EvalError){
$("#number_div").html("")
$("#expression").html("")
$("#value_div").html("Error")
}
else{
$("#number_div").html("")
$("#expression").html("")
```

```
$("#value_div").html("Error")
}
}
}
```

This function evaluates the expression stored as the saved background expression or the current expression and then returns the result to the output portion of the calculator display. Also, this function handles some exceptions such as Screen display overflow or evaluation of a meaningless or wrong expression. The exception handlers snippets can be modified as needed. I have coded it to display “Can’t Display” for a screen overflow type error. For illegal expression, I have coded it to display “Error” in the output screen and clear up the existing and input expressions.

6.
```
function clearCalc(){
$("#number_div").html("0")
$("#savedExpression").val("")
$("#expression").html("")
$("#value_div").html("")
}
```

This function is for All Clear (AC) button in the calculator, and as the name suggests, this clears up all the expressions, be it the input expression, existing expression, saved background expression, and the output expression.

7.
```
function reduceExp(){
var existingNumber = $("#number_div").html();
if(existingNumber.length>0){
existingNumber = existingNumber.substring(0,existingNumber.length-1);
$("#number_div").html(existingNumber);
}
```

```
else{
var expression = $("#savedExpression").val();
expression = expression.substring(0,expression.length-1);
$("#savedExpression").val(expression);
$("#expression").html(expression);
}
}
```

This function is the reflection of the delete button of the calculator and basically deletes expressions, numbers and operators. This is achieved by taking the substring of the expression from 0^{th} index to the 2^{nd} last index, and assigning it back to the expression variable.

8.
```
function percent(){
var key = $("#number_div").html();
var res = parseFloat(key)/100;
key = res+";
$("#number_div").html(key);
}
```

This function is for calculating the percentage of a number inside the input expression. This is done simply just by dividing the input expression number by 100 and assigning it back to the input expression.

- **Step 5**: Test the calculator and debug any errors or bugs. Use the browser's console or debugger tools to inspect the code and find any syntax or logical errors. Use different test cases to check if the calculator works as expected.

- CODE:

  (Note: Indents have been removed from the code below so as to fit it in the page)

```
<!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="vendor/bootstrap/css/bootstrap.css" rel="stylesheet">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-
icons/1.11.1/font/bootstrap-icons.min.css" integrity="sha512-
oAvZuuYVzkcTc2dH5z1ZJup5OmSQ000qlfRvuoTTiyTBjwX1faoyearj8KdMq0LgsBTHMrRuMek7s+CxF8yE+w=="
crossorigin="anonymous" referrerpolicy="no-referrer" />

<style>
.calc-btn{
width: 60px;
height: 60px;
background-color: rgba(255, 255, 255, 0.722);
border-radius: 30px;
font-weight: medium;
padding-top:3%;
border-color: #eb92be;
border-width: 1px;
}

.calc-display{
background-color: white;
color: black;
height: 100px;
width: 310px;
border-radius: 15px;
}

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

.expressionString{
margin-top: 5px;
height: 20px;
display: block;
```

```
font-size: 20px;
}

.valueString{
margin-top: 5px;
height: 20px;
display: block;
font-size: 30px;
color: blue;
text-align: end;
}

</style>
</head>
<body style="background: linear-gradient(320deg, #eb92be, #ffef78, #63c9b4);">
<h1 style="text-align: center; color: goldenrod;">HTML Calculator</h1>

<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" style="background-color: gray; opacity: 0.7; border-radius: 30px; box-
shadow: 0 4px 30px;">
<div class="d-flex justify-content-center align-items-center p-4">
<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="a" style="color: red;
font-size: 25px; font-weight: bold;">
AC
</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="c" style="color:
orange; font-size: 20px; font-weight: bold;">
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=";" style="font-size:
30px; color: rgb(4, 145, 4);">
&#37
</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="/" style="font-size:
30px; color: rgb(4, 145, 4);">
&#247
</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" style="font-size:
30px;">
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" style="font-size:
30px;">
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" style="font-size:
30px;">
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="*" style="font-size:
30px; color: rgb(4, 145, 4);">
&#215
</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" style="font-size:
30px;">
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" style="font-size:
30px;">
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" style="font-size:
30px;">
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="-" style="font-size:
30px; color: rgb(4, 145, 4);">
&#8722
</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" style="font-size:
30px;">
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" style="font-size:
30px;">
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" style="font-size:
30px;">
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="+" style="font-size:
30px; color: rgb(4, 145, 4);">
+
</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; font-size: 30px;"
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="." style="font-size:
30px;">
.
</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="=" style="font-size:
30px; color: white; background-color: rgb(4, 145, 4);">
=
</button>
</div>
</div>
</div>
</div>
</div>
</div>
</div>

<script src="https://code.jquery.com/jquery-3.7.1.js" integrity="sha256-
eKhayi8LEQwp4NKxN+CfCh+3qOVUtJn3QNZ0TciWLP4=" crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/2.9.2/umd/popper.min.js"
integrity="sha512-
2rNj2KJ+D8s1ceNasTIex6z4HWyOnEYLVC3FigGOmyQCZc2eBXKgOxQmo3oKLHyfcj53uz4QMsRCWNbLd32Q1g=="
crossorigin="anonymous" referrerpolicy="no-referrer"></script>
<script src="vendor/bootstrap/js/bootstrap.js"></script>

<script>
$(document).on('keypress',function(e){
$('button[data-event_key="'+e.key+'"]').addClass('active')
if((e.keyCode >= 48 && e.keyCode <= 57)||e.key=='.'){
appendNumber(e.key)
}
else{
if(e.key=='+'||e.key=='-'||e.key=='/'||e.key=='*'){
generateExpression(e.key)
}
else if(e.key==';'){
```

```
percent()
}
else if(e.key == '='){
evaluateExpression()
}
else if(e.key == 'a'){
clearCalc()
}
else if(e.key == 'c'){
reduceExp()
}
}

})

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

})

$('.calc-btn').on('click',function(e){
var key = $(this).data('event_key')
if(key!='+' && key!= '-' && key!= '*' && key!='/' && key!='c' && key!=';' && key!='=' &&
key!='a'){
appendNumber(key)
}
else{
if(key=='+'||key=='-'||key=='/'||key== '*'){
generateExpression(key)
}
else if(key == '='){
evaluateExpression()
}
else if(key == 'a'){
clearCalc()
}
else if(key == 'c'){
reduceExp()
}
else if(key == ';'){
percent()
}
}

})

function appendNumber(number){
var existingNumber = $("#number_div").html();
```

```
var currentString = number;
var outputString = '';
if(existingNumber != '' && existingNumber != undefined && existingNumber != null){
if(existingNumber == '0'){
outputString = number
}
else{
outputString = existingNumber+=currentString
}
}
else{
outputString = currentString;
}
$("#number_div").html(outputString);
}

function generateExpression(operator){
var existingNumber = $("#number_div").html();
var currentOperator = operator;
var expression = '';
var savedExpression = $("#savedExpression").val();

if(savedExpression == '' || savedExpression == null || savedExpression == undefined){
expression = parseFloat(existingNumber) + operator
}
else{
expression = savedExpression + existingNumber + operator;
}

$("#number_div").html("")
$("#savedExpression").val(expression)
$("#expression").html(expression)
}

function evaluateExpression(){
var result = '';
var expression = $("#savedExpression").val();
var existingNumber = $("#number_div").html();

if(existingNumber != '' || existingNumber != null || existingNumber != undefined){
expression = expression + parseFloat(existingNumber);
$("#expression").html(expression)
}

try{
result = eval(expression);

$("#savedExpression").val("")
```

```
$("#number_div").html(result)
if((result+'').length>17){
result = (result+"").substring(0,17)
$("#value_div").html(result)
}
else{
$("#value_div").html(result)
}
if($("#number_div").html()==''){
$("#number_div").html("")
$("#expression").html("")
$("#value_div").html("Error")
}
}

catch(error){
if(error instanceof EvalError){
$("#number_div").html("")
$("#expression").html("")
$("#value_div").html("Error")
}
else{
$("#number_div").html("")
$("#expression").html("")
$("#value_div").html("Error")
}
}
}

function clearCalc(){
$("#number_div").html("0")
$("#savedExpression").val("")
$("#expression").html("")
$("#value_div").html("")
}

function reduceExp(){
var existingNumber = $("#number_div").html();
if(existingNumber.length>0){
existingNumber = existingNumber.substring(0,existingNumber.length-1);
$("#number_div").html(existingNumber);
}

else{
var expression = $("#savedExpression").val();
expression = expression.substring(0,expression.length-1);
$("#savedExpression").val(expression);
$("#expression").html(expression);
```

```
}

}

function percent(){
var key = $("#number_div").html();
var res = parseFloat(key)/100;
key = res+'';
$("#number_div").html(key);
}

</script>
</body>
</html>
```

- INSTRUCTIONS TO USE THE CALCULATOR:
    1. Press the numbers in your keyboard or click the number buttons in the calculator.
    2. Press the + or click the + for addition.

       Press the – or click the – for subtraction.

       Press the * or click the x for multiplication.

       Press the / or click the ÷ for division.

       Press the . or click the . for decimal.

       Press the ; or click the % for percentage.

       Press a or click AC to all clear.

       Press c or click Del to delete expressions.
    3. In the display, the top portion is the input expression, the middle portion is the saved or current expression, and the bottom portion is the result.

- RESULTS/OUTPUT:

Normal Expression Evaluation

INPUT OUTPUT

Evaluation of number divided by zero

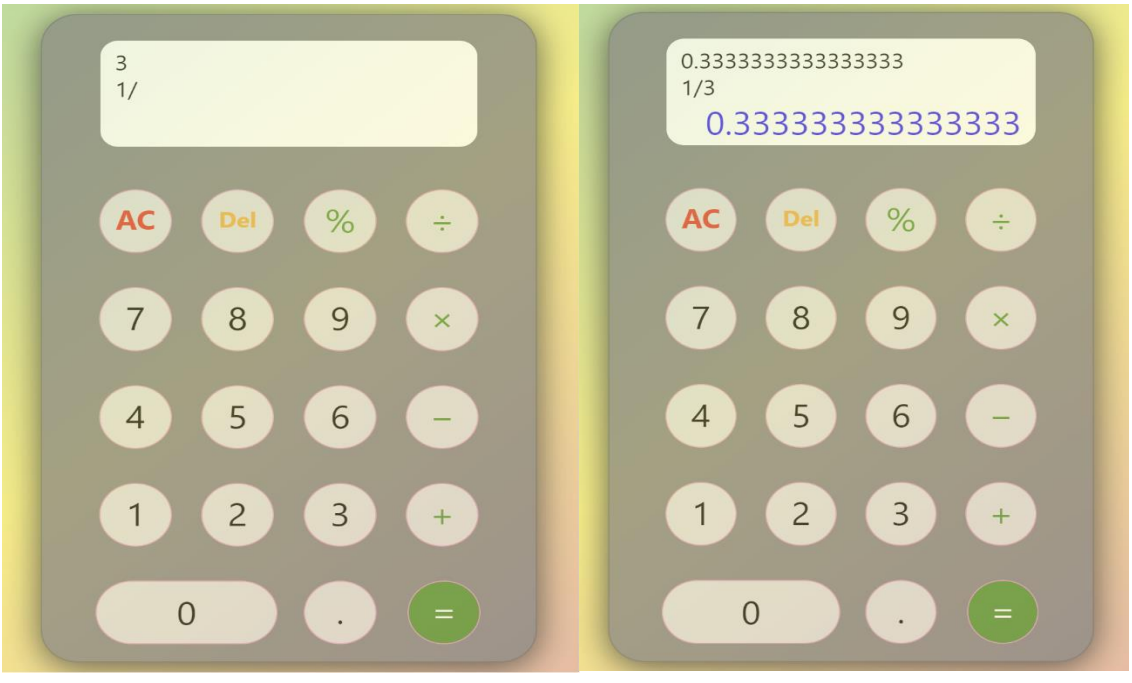
INPUT OUTPUT

Display of error whilst evaluating illegal expression

INPUT OUTPUT

Display of long sized results in shortened form in the output display

INPUT OUTPUT



- CONCLUSION:

- In this project, I have created a simple and functional calculator using HTML, Bootstrap, CSS, JavaScript, and jQuery. I have used the Bootstrap grid system and CSS classes to design the layout of the calculator buttons and display. I have used JavaScript and jQuery to handle the logic of the calculator operations and events. I have tested the calculator with different test cases and debugged any errors or bugs. The calculator can perform basic arithmetic operations, such as addition, subtraction, multiplication, and division. It can also handle edge cases, such as division by zero, invalid input, and overflow.

- The main objectives of this project were to demonstrate the use of Bootstrap and HTML to create a responsive and user-friendly web interface, and to practice the skills of web development using various technologies and tools. I have learned how to use Bootstrap components and classes to create a grid layout, how to use CSS styles to customize the appearance of the elements, how to use Javascript and jQuery to manipulate the DOM and add interactivity, and how to use the browser's console and debugger tools to inspect and debug the code.

- The main challenges of this project were to implement the logic of the calculator operations correctly, to handle the edge cases gracefully, and to ensure the compatibility and performance of the code across different browsers and devices. I have overcome these challenges by using functions, variables, conditional statements, event listeners, error handling, and testing techniques.

- The main limitations of this project were that the calculator can only perform basic arithmetic operations, and that it does not have any advanced features, such as memory, scientific functions, or history. These limitations can be addressed by adding more buttons, functions, variables, and events to the code.
- The main outcomes of this project were that I have successfully created a Bootstrap and HTML Calculator that works as expected, that I have improved my web development skills and knowledge, and that I have gained more confidence and interest in creating web applications.