**A PROJECT REPORT**

**ON**

**INTERNSHIP PROJECT 1**

**CREATE A CALCULATOR USING HTML AND BOOTSTRAP**

**IN THE DOMAIN OF**

**FRONT-END WEB DEVELOPMENT**

**BY**

**MALLAVARAPU LAKSHMI PRAHARSHA**

**DATE:19-04-2024**

**TABLE OF CONTENTS**

**S. No Topic Page no.**

1. Abstract 3

2. Objectives 4-8

3. Introduction 9

4. Methodology 10-12

5. Code 13-20

6. Results and Outputs 21-23

7. Conclusion 24

**ABSTRACT**

This project report presents the design, development, and implementation process of creating a calculator application using HTML and Bootstrap. This project aimed to demonstrate how modern web technologies can be utilized to build a functional and user-friendly calculator interface that can perform basic arithmetic operations. The report begins with an introduction to the significance of calculator applications in daily life, emphasizing their role in facilitating quick and accurate mathematical calculations. It highlights the relevance of using HTML and Bootstrap for web development, citing their widespread adoption and versatility in creating responsive and visually appealing user interfaces. The implementation leverages HTML for structuring the calculator's layout and Bootstrap for styling, ensuring responsiveness across different screen sizes. Through the utilization of HTML forms and input elements, users can input numerical values and select operations with ease. Bootstrap's grid system facilitates the organization of elements, maintaining visual coherence and usability. The calculator supports addition, subtraction, multiplication, and division operations, enabling users to perform calculations effortlessly. JavaScript is employed to handle user inputs, perform computations, and display results dynamically within the browser interface. The project report provides a comprehensive overview of creating a calculator application using HTML and Bootstrap. It demonstrates the practical application of these technologies in building interactive and functional web-based tools while offering valuable insights for future web development projects.

**OBJECTIVE**

The following objectives are taken into account while creating a calculator using HTML and Bootstrap:

1. Functionality
2. User Interface Design
3. Responsive Design
4. Cross-Browser Compatibility
5. Keyboard Support
6. Memory Functionality
7. Performance Optimization
8. Documentation and User Guide

**Functionality:**

In ensuring the functionality of the calculator, it is imperative to meticulously address each of the basic arithmetic operations to guarantee accuracy and reliability. In addition, the fundamental operation must allow for the addition of multiple numbers, both positive and negative, while maintaining precision across varying magnitudes. Subtraction entails the subtraction of one number from another, with attention to the proper handling of negative results and borrowing in multi-digit calculations. Multiplication necessitates the ability to multiply numbers of any magnitude, considering factors such as decimal points and scientific notation for comprehensive functionality. Division, often the most complex operation, requires handling various scenarios such as division by zero and the display of both the quotient and remainder when applicable.

**User Interface Design:**

Designing an intuitive and user-friendly interface for the calculator involves careful consideration of layout, visual hierarchy, and interactive elements to ensure ease of navigation and comprehension for users of varying skill levels. Leveraging Bootstrap's components and grid system provides a solid foundation for creating a visually appealing layout that is responsive across different devices and screen sizes. By utilizing Bootstrap's predefined CSS classes and components such as buttons, forms, and typography, designers can streamline the development process while maintaining consistency and coherence in the interface design. The grid system offered by Bootstrap enables the creation of a structured layout with rows and columns, allowing for precise placement and alignment of interface elements. This facilitates the organization of the calculator's components, such as buttons for digits and operators, display panels for input and output, and auxiliary controls for memory functions or advanced operations. The responsive nature of Bootstrap ensures that the interface adapts seamlessly to various screen sizes, optimizing the user experience on both desktop and mobile devices. In addition to layout considerations, attention should be paid to visual elements such as color scheme, typography, and iconography to enhance usability and aesthetic appeal. Consistent use of colors for different types of buttons or interactive elements helps users quickly identify their purpose, while clear typography ensures the readability of numerical inputs and output. Furthermore, the use of intuitive icons or visual cues can aid in conveying functionality and guiding user interactions.

**Responsive Design:**

Implementing responsive design principles is essential to ensure that the calculator application provides an optimal user experience across a diverse range of devices and screen sizes, including desktops, laptops, tablets, and smartphones. Responsive design involves creating a flexible and adaptable layout that adjusts seamlessly to different viewport sizes and orientations, thereby accommodating users accessing the application from various devices and environments. By incorporating Bootstrap's responsive grid system and CSS media queries, developers can design a layout that dynamically reconfigures its structure and appearance based on the available screen real estate. This allows the calculator interface to scale proportionally and rearrange its elements to maintain usability and readability, regardless of the device being used. For example, on smaller screens such as smartphones, the layout may stack elements vertically or collapse navigation menus to conserve space and prioritize essential content. Moreover, responsive design encompasses more than just layout adjustments—it also entails optimizing user interactions and input methods for touch-based devices. By implementing touch-friendly controls and ensuring adequate spacing between interactive elements, designers can enhance usability and prevent accidental inputs on touchscreen devices like tablets and smartphones. Additionally, considering factors such as font size, button dimensions, and input field visibility helps ensure that users can interact with the calculator comfortably and accurately across different devices.

**Cross-Browser Compatibility:**

Ensuring cross-browser compatibility is crucial to provide a consistent user experience for all users accessing the calculator application, regardless of the web browser they choose to use. Each web browser interprets HTML, CSS, and JavaScript slightly differently, which can lead to variations in the rendering and functionality of web applications. Therefore, thorough testing and implementation of compatibility measures are essential to address these differences and ensure that the calculator functions reliably across major web browsers such as Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge. One approach to achieving cross-browser compatibility is to adhere to web standards and best practices recommended by the World Wide Web Consortium (W3C). By following standardized HTML, CSS, and JavaScript coding practices, developers can minimize the likelihood of browser-specific issues and ensure a more consistent rendering of the application across different browsers. Additionally, leveraging modern web technologies and features supported by most modern browsers helps maximize compatibility while providing enhanced functionality and performance. Furthermore, utilizing CSS vendor prefixes and feature detection techniques can help address inconsistencies in CSS rendering across different browsers. Vendor prefixes such as -webkit- for Chrome and Safari, -moz- for Firefox, and -ms- for Edge allow developers to specify browser-specific CSS properties and ensure proper styling across different platforms. Additionally, employing feature detection libraries like Modernizr enables developers to detect and handle browser-specific feature support dynamically, ensuring graceful degradation or fallback solutions for unsupported features.

**Keyboard Support:**

Keyboard support is a critical aspect of accessibility and usability, enhancing the overall user experience by allowing users to interact with the calculator application using both mouse and keyboard inputs. Providing keyboard support ensures that users with mobility impairments or those who prefer keyboard shortcuts can efficiently input numbers and perform operations without relying solely on mouse or touch-based interactions. To implement keyboard support effectively, developers need to assign meaningful keyboard shortcuts or key bindings for common calculator functions such as numerical input, arithmetic operations, and clearing the input. This allows users to navigate the calculator interface and perform calculations using intuitive keyboard combinations, improving efficiency and accessibility. In addition to supporting basic numerical input, the calculator application should also accommodate keyboard input for performing arithmetic operations such as addition, subtraction, multiplication, and division. By mapping common mathematical operators to specific keys or key combinations, users can seamlessly enter expressions and execute calculations using the keyboard alone. Furthermore, providing keyboard shortcuts for auxiliary functions such as memory storage, memory recall, and clearing the input field enhances the versatility and usability of the calculator application. This enables users to perform advanced operations and manage calculations more efficiently, particularly when working with complex expressions or repetitive calculations. It's essential to ensure that the keyboard navigation and input mechanisms are implemented by established accessibility standards and guidelines, such as those outlined by the Web Content Accessibility Guidelines (WCAG). This includes providing keyboard focus indicators, ensuring logical tab order, and supporting keyboard navigation through interactive elements.

**Memory Functionality:**

Memory functionality enhances the versatility and convenience of a calculator application, allowing users to store intermediate results and perform repetitive calculations more efficiently. Implementing memory functionality involves providing intuitive controls for memory storage, recall, and clearing, as well as ensuring seamless integration with the calculator's interface and workflow. Memory storage enables users to store the current result or a specific value in memory for later use. This feature is particularly useful for storing intermediate results during complex calculations or for saving constants or frequently used values. Implementing memory storage typically involves providing a dedicated button or control that allows users to store the current result or input value into memory with a single click or key press. Memory recall functionality allows users to retrieve previously stored values from memory and incorporate them into their calculations. This feature enables users to reuse stored values without having to re-enter them manually, saving time and effort, especially when performing repetitive calculations or multi-step operations. Memory recall functionality can be implemented through a dedicated button or keyboard shortcut that retrieves the stored value and displays it in the calculator's input field.

Memory clearing functionality enables users to clear the memory storage and reset it to its default state. This feature is essential for maintaining the integrity of memory operations and ensuring that users can start fresh when needed. Implementing memory clearing typically involves providing a dedicated button or control that clears the memory storage with a single click or key press, restoring the calculator to its initial state. To ensure a seamless user experience, memory functionality should be integrated cohesively into the calculator's interface and workflow. This includes providing clear visual feedback when memory operations are performed, such as displaying a confirmation message or updating the display to reflect the current memory state. Additionally, memory functionality should be accessible via both mouse and keyboard inputs to accommodate different user preferences and accessibility needs.

**Performance Optimization:**

Performance optimization is crucial for ensuring that the calculator application delivers a responsive and smooth user experience, with fast loading times and minimal latency. This involves optimizing various aspects of the application, including code efficiency, resource utilization, and rendering performance. Techniques such as minimizing JavaScript file sizes, reducing the number of network requests, and implementing efficient algorithms for arithmetic operations can significantly improve loading times and responsiveness. Additionally, optimizing CSS and image assets, leveraging browser caching, and employing techniques like lazy loading can further enhance performance and ensure a seamless user experience, even on devices with limited processing power or network bandwidth. Thorough testing and profiling are essential to identify performance bottlenecks and validate the effectiveness of optimization efforts, ultimately resulting in a calculator application that provides fast and fluid operation across a wide range of devices and usage scenarios.

**Documentation and User Guide:**

Comprehensive documentation and a user guide are essential components of the calculator application, providing users with clear instructions on how to use the application effectively and maximize its features. The documentation should include detailed explanations of the calculator's interface elements, such as buttons, input fields, and memory functions, along with step-by-step instructions for performing common arithmetic operations. Additionally, it should cover any advanced features or customization options available, including keyboard shortcuts, memory management, and error handling. Visual aids such as screenshots or diagrams can further enhance understanding and usability. By providing comprehensive documentation and a user guide, developers empower users to make the most of the calculator application and troubleshoot any issues they may encounter, ultimately enhancing user satisfaction and adoption.

**INTRODUCTION**

In today's fast-paced world, the significance of calculator applications in daily life cannot be overstated. From simple arithmetic tasks to complex calculations, calculators play a vital role in facilitating quick and accurate mathematical computations across various domains, including education, finance, engineering, and more. Whether it's calculating monthly budgets, solving equations, or analyzing data, the convenience and efficiency offered by calculator applications have become indispensable tools for individuals and professionals alike.

HTML and Bootstrap have emerged as powerful frameworks in web development, offering developers a robust toolkit for creating responsive and visually appealing user interfaces for calculator applications. HTML, the backbone of the web, provides the structural foundation for designing and organizing the elements of a calculator interface, including input fields, buttons, and display panels. Its simplicity and versatility make it an ideal choice for building interactive web applications that can be accessed across different devices and platforms.

Bootstrap, on the other hand, further enhances the development process by offering a comprehensive set of pre-designed components, CSS styles, and JavaScript plugins that streamline the creation of responsive and visually engaging user interfaces. With its grid system, typography, and reusable components, Bootstrap enables developers to build calculator applications that not only function seamlessly but also adapt gracefully to various screen sizes and devices, ensuring a consistent user experience.

The widespread adoption of HTML and Bootstrap in web development underscores their relevance and effectiveness in meeting the demands of modern calculator applications. Their intuitive syntax, extensive documentation, and active community support make them accessible to developers of all skill levels, empowering them to create polished and professional-grade calculator interfaces with ease.

In this report, we explore the design, development, and implementation process of creating a calculator application using HTML and Bootstrap, showcasing their combined capabilities in delivering an intuitive, responsive, and visually appealing user experience. By leveraging these technologies, developers can meet the evolving needs of users and provide them with powerful tools for performing quick and accurate mathematical calculations in their daily lives.

**METHODOLOGY**

The "methodology" in the domain of web development refers to a systematic approach or framework used to plan, design, build, deploy, and maintain web applications or websites. It encompasses the processes, practices, and techniques employed by developers and teams to ensure the successful completion of web development projects. Methodologies in web development help streamline the development process, improve efficiency, maintain consistency, and ultimately deliver high-quality products.

**Requirement Analysis:**

During the initial phase of the project, a comprehensive analysis of the project requirements is conducted to lay the foundation for the development of the calculator application. This analysis involves a detailed exploration of the desired functionality, encompassing fundamental arithmetic operations such as addition, subtraction, multiplication, and division, as well as additional features such as memory functions for storing and recalling values. Furthermore, careful consideration is given to user interaction features, including input methods such as mouse clicks and keyboard input, as well as preferences for the user interface design such as color schemes, layout preferences, and accessibility requirements. Additionally, identifying the target audience and understanding their needs and preferences play a crucial role in shaping the design and functionality of the calculator application, ensuring that it meets the expectations and usability requirements of its intended users.

**Planning and Design:**

In the planning phase, meticulous attention is devoted to conceptualizing the design and layout of the calculator application based on the insights gained from the requirement analysis. This involves defining the structural elements of the interface, including the arrangement of input fields, buttons, and display panels to ensure intuitive navigation and ease of use. Organizing the placement of these elements is crucial for facilitating efficient user interaction and enhancing the overall usability of the application. Furthermore, determining the visual styling using Bootstrap components enables the creation of a cohesive and visually appealing interface that aligns with modern design trends while leveraging the responsive capabilities of Bootstrap to ensure compatibility across various devices and screen sizes. By prioritizing user experience and aesthetics in the planning phase, the foundation is laid for developing a calculator application that meets the needs and expectations of its intended users.

**HTML Structure:**

Once the design is finalized, the focus shifts to implementing the HTML structure of the calculator application. This phase involves translating the design specifications into code by coding the markup for the various interface elements. Input fields are created for displaying numbers and results, while buttons are added to facilitate arithmetic operations and memory functions. Throughout the implementation process, semantic HTML elements are prioritized to ensure accessibility and optimize search engine visibility. By adhering to semantic HTML practices, developers ensure that the calculator application is accessible to users with disabilities and is more easily indexed by search engines, enhancing its usability and discoverability. Additionally, semantic HTML promotes clean and maintainable code, facilitating future updates and modifications to the calculator application.

**Styling with Bootstrap:**

In the styling phase, Bootstrap is leveraged to enhance the visual appeal and responsiveness of the calculator application. The Bootstrap grid system provides a flexible and responsive layout framework, allowing developers to create a consistent user experience across various devices and screen sizes. By utilizing the grid system, the interface elements of the calculator application are arranged and aligned systematically, ensuring optimal readability and usability. Additionally, Bootstrap's extensive library of CSS classes is applied to customize the appearance of buttons, input fields, and other interface components according to the design specifications. This enables developers to achieve a cohesive and visually appealing design while maintaining consistency and compatibility across different browsers and devices. Through the use of Bootstrap, the calculator application not only looks polished and professional but also offers a seamless user experience across diverse platforms.

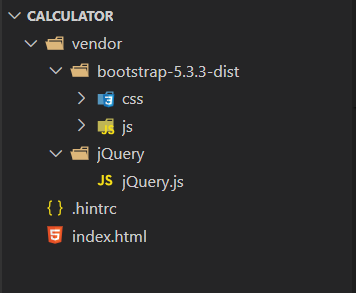
**JavaScript and jQuery Functionality:**

During the implementation phase, JavaScript and jQuery are integrated to imbue the calculator application with essential functionality. JavaScript functions are meticulously crafted to manage user interactions seamlessly, including arithmetic operations, and memory management functions such as storage, recall, and clearing, ensuring a comprehensive user experience. Event listeners are strategically employed to capture user inputs from both mouse clicks and keyboard interactions, facilitating real-time updates to the calculator display. Through the effective utilization of JavaScript and jQuery, the calculator application transforms into a dynamic and interactive tool, enabling users to perform calculations effortlessly while maintaining accuracy and reliability. This phase is crucial in bringing the design to life and providing users with a fully functional and responsive calculator application.

**Testing and Debugging:**

During the development process, thorough testing is essential to ensure the reliability and usability of the calculator application. Unit tests are conducted to validate the correctness of arithmetic operations and memory functions, ensuring accurate results under various scenarios. Usability tests assess the intuitiveness and efficiency of the user interface, ensuring that users can navigate the application effortlessly and perform calculations with ease. Additionally, compatibility testing is performed to verify consistent performance across different web browsers and devices, ensuring a seamless user experience regardless of the platform used. Through rigorous testing, any issues or inconsistencies are identified and addressed promptly, ensuring that the calculator application meets the highest standards of functionality, usability, and compatibility.

**File Structure:**



**CODE**

**index.html**

<!DOCTYPE html>

<html lang="en">

    <head>

        <meta charset="utf-8">

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

        <title>HTML Calculator | Minor Project 2</title>

        <link  rel="stylesheet" href="vendor/bootstrap-5.3.3-dist/css/bootstrap.css" >

        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-icons.min.css">

        <style>

            .calc-btn{

                width: 60px;

                height: 60px;

            }

            .calc-display{

                background-color: black;

                color: white;

                height: 100px;

                width: 310px;

                border-radius: 5px;

                margin-top: 5px;

            }

            .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: 20px;

            }

            .card{

                background-color: #ECE9EC;

            }

            .card-header{

                background-color: #ECE9EC;

                font-weight: bold;

            }

            #instructions {

                width: 40%;

                float: left;

                margin-right: 70px;

                margin-top: 5px;

                color: black ;

            }

            @media (max-width: 768px) {

                #instructions {

                width: 100%;

                margin-right: 0;

                }

            }

        </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" id="instructions">

                    <div class="card">

                        <div class="card-header">

                            INSTRUCTIONS

                        </div>

                        <div class="card-body">

                            <ul>

                                <li>Use the number buttons to enter a number.</li>

                                <li>Use the operator buttons (+, -, \\*, /) to perform an operation.</li>

                                <li>Use the 'Del' button to delete the number or operator entered.</li>

                                <li>Use the 'CE' button to clear the current number.</li>

                                <li>Use the '=' button to calculate the result of the operation.</li>

                            </ul>

                        </div>

                    </div>

                </div>

                <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>

        </div>

        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/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="vendor/bootstrap-5.3.3-dist/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.keyCode >= 96 && e.keyCode <= 105)){

                    appendNumber(e.key)

                }else{

                    if(e.key=='+' || e.key == '-' || e.key == '/' || e.key == '\*'){

                        generateExpression(e.key)

                    }else if(key=='='){

                        evaluateExpression()

                    }else if(key == "Delete"){

                        clearCalc()

                    }

                }

                console.log(e.key);

            })

            $(document).on('keyup', function(e){

                $('button[data-event\_key="'+e.key+'"]').removeClass('active')

                console.log(e.key);

            })

            $('.calc-btn').on('click', function(e){

                var key = $(this).data('event\_key')

                if(key != '+' && key != '-' && key != '=' && key != '.' && key != '\*' && key != '/' && key != 'Delete' && key != 'NumLock'){

                    appendNumber(key)

                }else{

                    if(key=='+' || key == '-' || key == '/' || key == '\*'){

                        generateExpression(key)

                    }else if(key=='='){

                        evaluateExpression()

                    }else if(key == "Delete"){

                        clearCalc()

                    }

                }

                console.log(key)

            })

            function appendNumber(number){

                var existingNumber = $("#number\_div").html();

                console.log(existingNumber);

                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 = parseInt(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 + parseInt(existingNumber);

                    $("#expression").html(expression)

                }

                result = eval(expression);

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

                $("#number\_div").html(result)

                $("#value\_div").html(result)

            }

            function clearCalc(){

                $("#number\_div").html("")

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

                $("#expression").html("")

                $("#value\_div").html("")

            }

        </script>

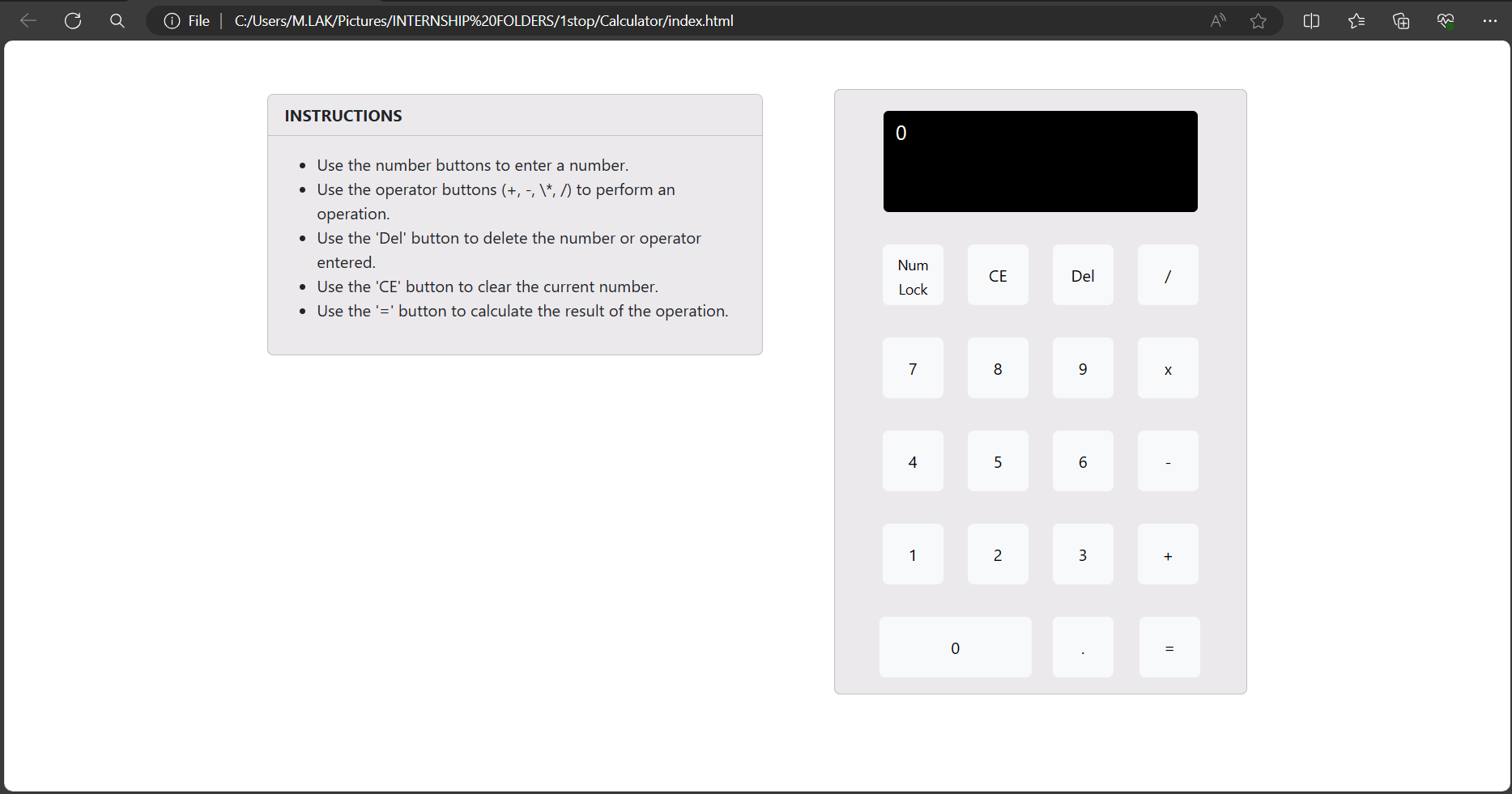
    </body>

</html>

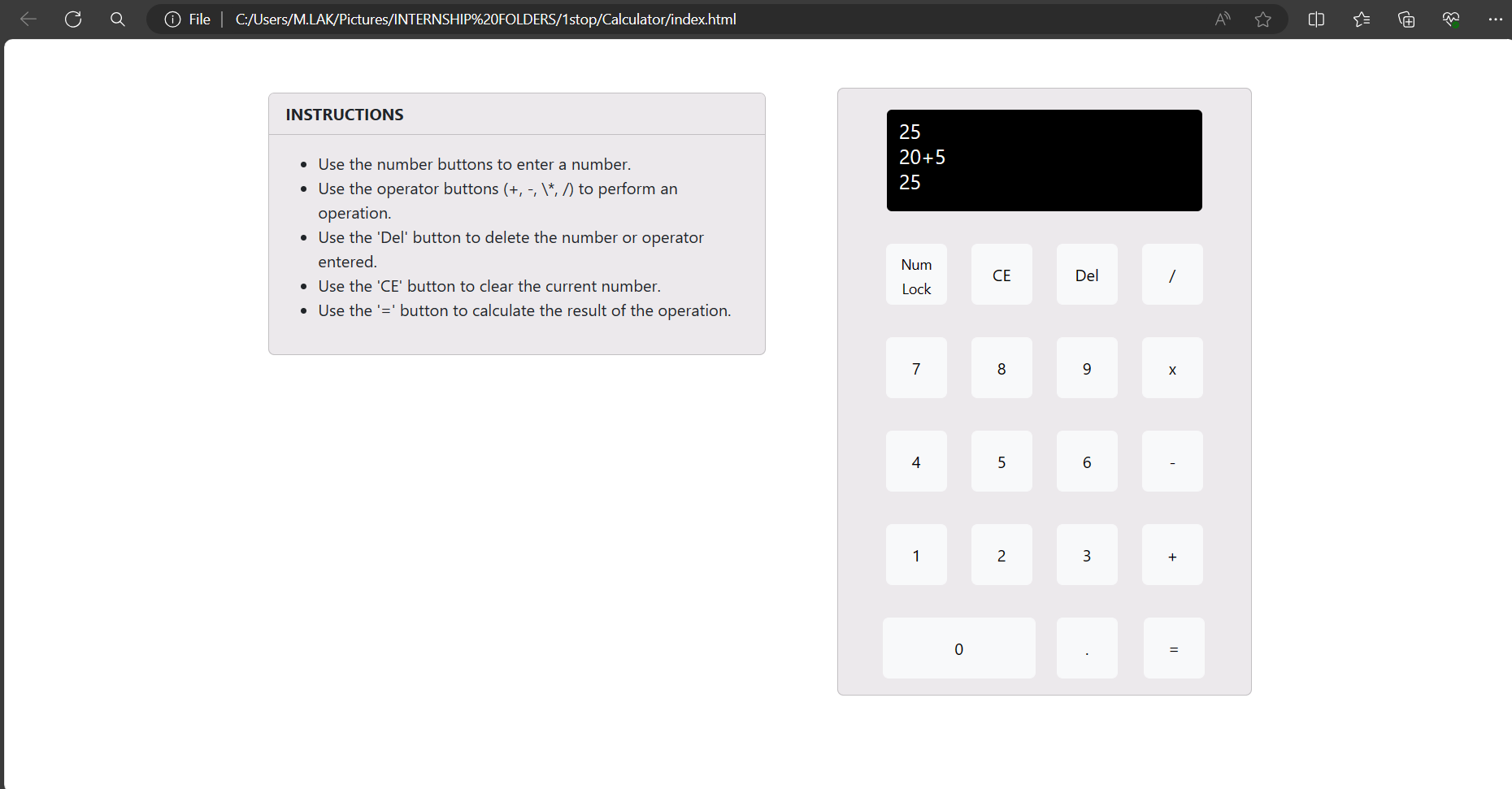
The code provided is an HTML document that displays a simple calculator interface on a webpage. The calculator interface is composed of a display area and a set of buttons for entering numbers and operators. The display area is divided into three sections: the input string, the expression string, and the value string. The input string displays the current number being entered, the expression string displays the current expression being calculated, and the value string displays the result of the calculation. The buttons for entering numbers and operators are arranged in a grid, with each button corresponding to a specific number or operator. The calculator interface is styled using CSS, with the display area having a black background and white text, and the buttons having a light gray background and black text. The calculator interface is also made responsive using Bootstrap's grid system, with the instructions div taking up 25% of the width on medium and large screens and 100% of the width on small screens. The calculator interface is interactive, with the buttons triggering JavaScript functions that update the display area and perform calculations. The JavaScript code uses jQuery to handle user input and update the display area. The calculator interface is also designed to be accessible, with the instructions div providing clear instructions on how to use the calculator.

**RESULTS AND OUTPUTS**

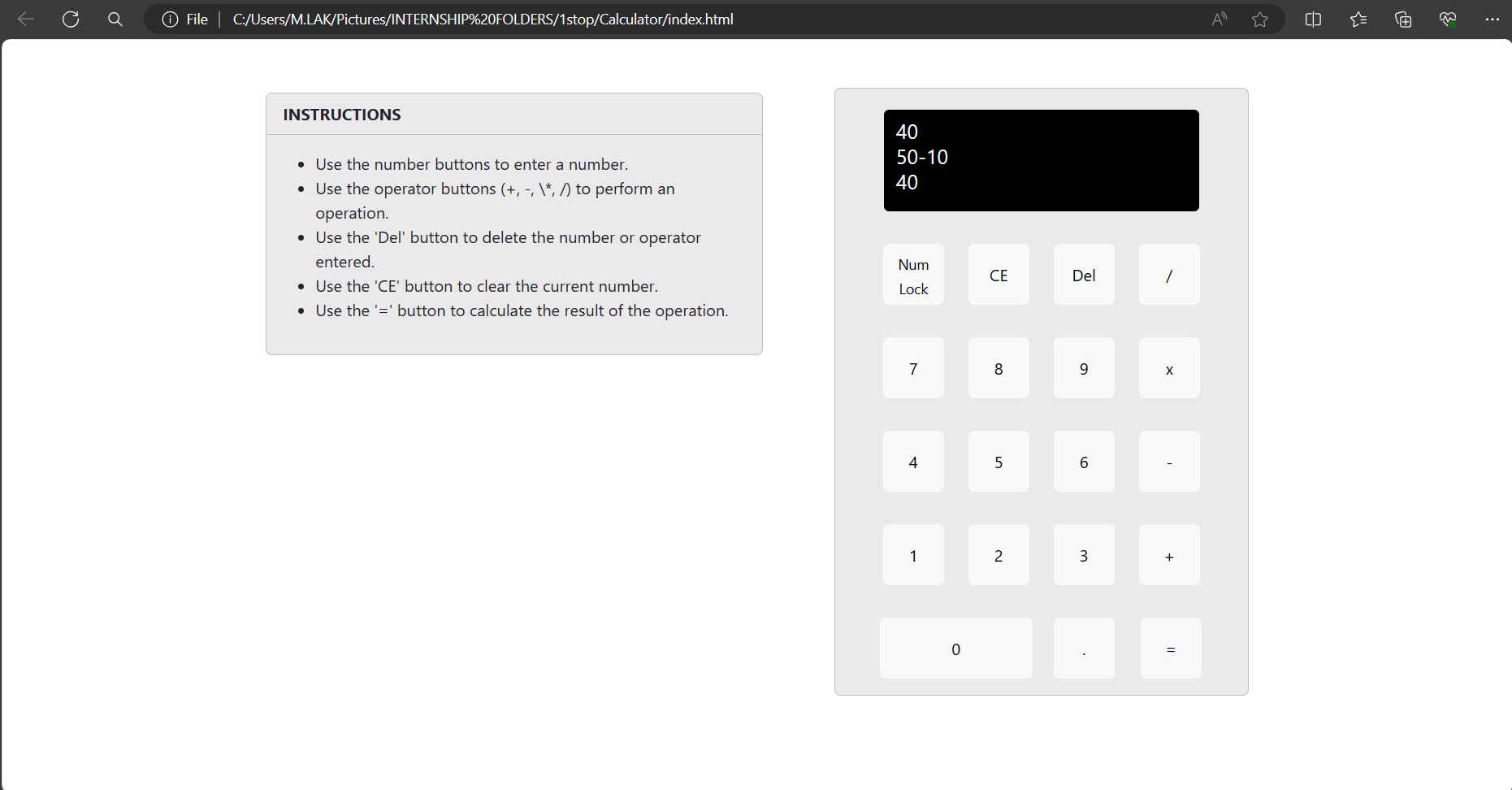
**Simple Calculator webpage:**

****

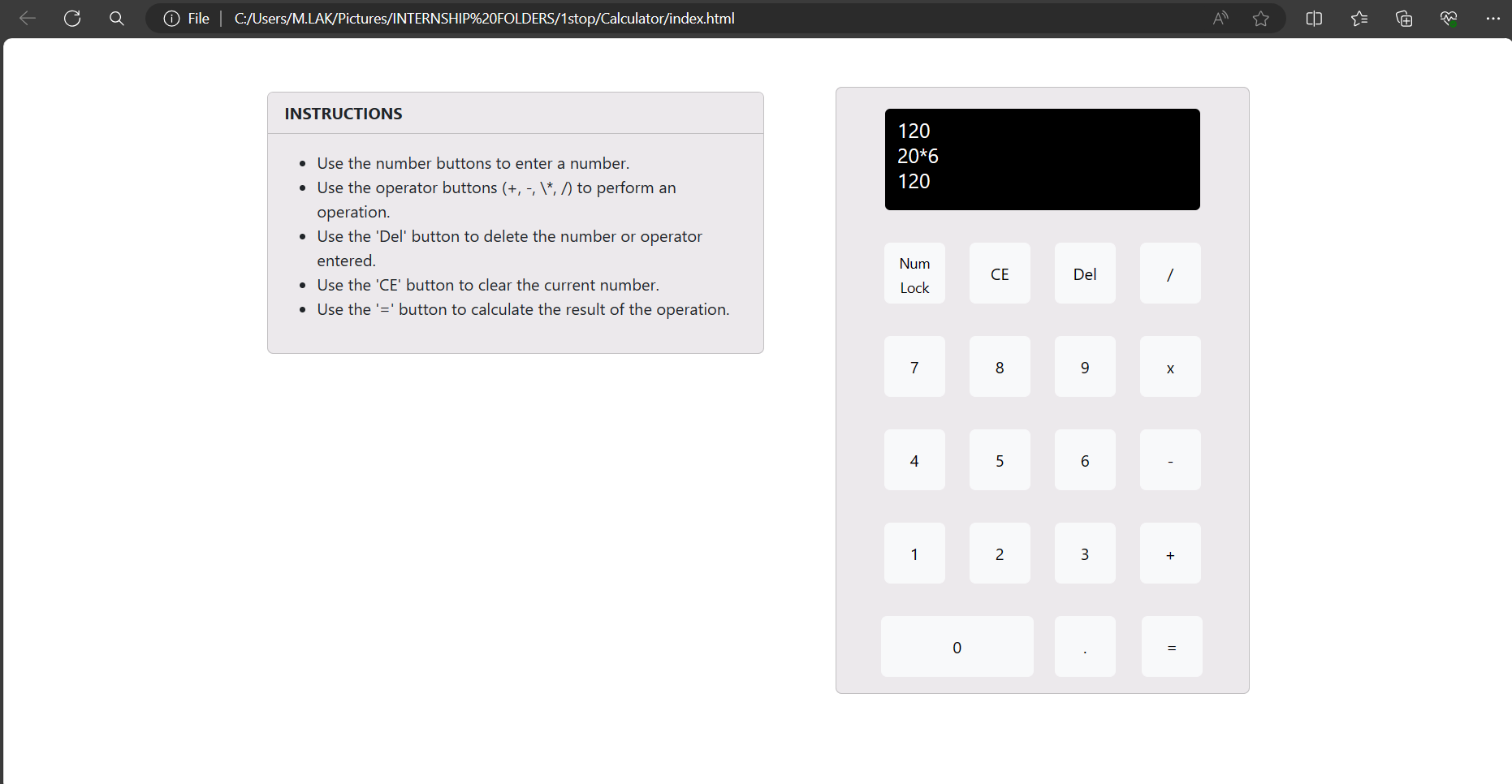
**Addition Operation:**

****

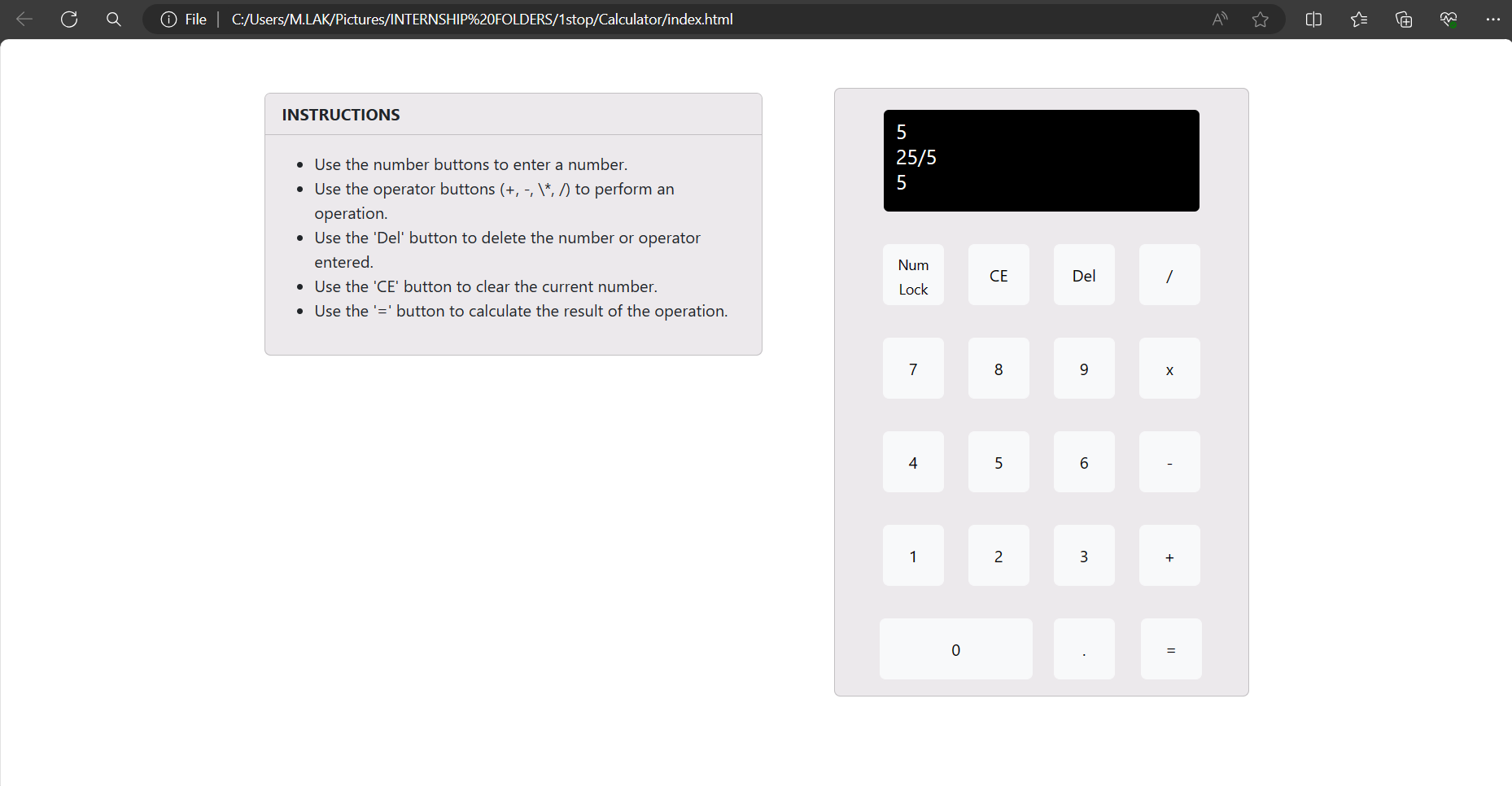
**Subtraction Operation:**

****

**Multiplication Operation:**

****

**Division Operation:**

****

**CONCLUSION**

In conclusion, the development of a calculator application using HTML and Bootstrap has been a rewarding journey, culminating in the creation of a versatile and user-friendly tool for performing mathematical calculations. Throughout the project, we have successfully leveraged HTML and Bootstrap to design and implement a responsive and visually appealing user interface, accommodating users across various devices and screen sizes. The integration of JavaScript has enriched the calculator application with essential functionality, enabling users to perform arithmetic operations, manage memory functions, and interact with the interface seamlessly. Through meticulous planning, implementation, and testing, we have ensured that the calculator application meets the needs and expectations of its intended users, offering a reliable and efficient solution for everyday mathematical tasks. Moving forward, the experience gained from this project serves as a valuable foundation for future endeavours in web development, empowering us to create innovative and impactful applications that enhance the digital experience for users worldwide. Overall, the development of the calculator application using HTML and Bootstrap has been a testament to the power of modern web technologies in creating accessible, functional, and aesthetically pleasing solutions for a diverse range of users.