**Title Page**

SIMPLE CALCULATOR

Members:

Pulgan, Myraza Riann C.

Domingo, Jhana Christina

Malones, Mary Chel

Maguit, Ron Michael

Date : MARCH 3 2024

**Executive Summary (optional)**

**Briefly summarize the problem or inefficiency you're addressing.**

The problem being addressed is the lack of a straightforward and easy-to-use calculator for performing basic arithmetic operations, without unnecessary complexity or cluttered features that might confuse users.

**Highlight the proposed solution and its expected benefits.**

The proposed solution is to create a simple calculator interface with basic arithmetic functionality, including addition, subtraction, multiplication, and division. The interface will have a clean design with large buttons for each operation, making it easy for users to input calculations quickly and accurately. Additionally, the calculator will display results clearly, providing an intuitive user experience.

**Introduction**

**Set the context by explaining the current situation and the problem you're aiming to solve.**

The current situation involves users frequently needing to perform basic arithmetic calculations while coding in the VS Code environment. However, they often resort to external calculator applications or switch to separate calculator tools, disrupting workflow and productivity. This interruption leads to distractions, loss of focus, and unnecessary friction in the coding process.

**Top of Form**

**Provide evidence of the problem's significance and its impact on the organization (e.g., data, metrics, customer feedback).**

Workflow Disruption-Coding tasks often require frequent arithmetic calculations, such as converting units, determining numerical values, or testing mathematical expressions. Having to leave the coding environment to perform these calculations disrupts the natural workflow and slows down development.

**Current Process Analysis**

**Describe the current process or workflow in detail**

**Project Setup-** Create a new MVC project using your preferred development environment or IDE (e.g., Visual Studio).Choose the MVC project template to scaffold the necessary files and folders for your application.

**Model Creation-**Define the data model for the calculator application. In this case, the model could include properties such as Operand1, Operand2, Operator, and Result.

**Create a class file (e.g., CalculatorModel.cs)-** to represent the model. Define the properties and any necessary methods for performing calculations.

**View Creation-**Design the user interface for the calculator using HTML, CSS, and Razor syntax (if using ASP.NET MVC).

**Create a view file (e.g., Calculator.cshtml)-** to display the calculator interface to the user. Include input fields for operands, buttons for operators, and a display area for the result.

**Controller Implementation-**Create a controller class (e.g., CalculatorController.cs) to handle user interactions and business logic.

**Define action methods to respond to user input, such as entering operands, selecting operators, and calculating results.Within the action methods, interact with the model to perform calculations based on user input and update the view accordingly.**

**Routing Setup-**Configure routing in the MVC application to map incoming requests to the appropriate controller actions.

**Define routes in the RouteConfig.cs file**- to specify the controller and action method to be invoked for each URL pattern.

**Testing and Debugging**-Test the functionality of the calculator application by running it locally in your development environment. Use debugging tools provided by your IDE to identify and fix any errors or issues in the code.

**Deployment**-Once the application is complete and thoroughly tested, deploy it to a web server or hosting platform to make it accessible to users.

**Version Control with Git (Optional)-**If desired, initialize a Git repository for your MVC project to track changes and collaborate with other developers.Use Git commands or a Git client within your IDE to commit changes, create branches for new features, and push updates to a remote repository (e.g., GitHub).

**Proposed Improvement**

**Clearly outline your proposed solution for addressing the identified issues.**

**User Interface Design**- Designing a user-friendly and intuitive interface that provides clear feedback to the user. Issues such as cluttered layouts, confusing button labels, or inconsistent styling can impact usability.

**SOLUTION:**

**Simplify the Layout-** Keep the layout clean and uncluttered by arranging elements in a logical and organized manner. Group related buttons and controls together to make it easier for users to locate and interact with them.

**Testing and Quality Assurance-** Ensuring comprehensive testing coverage for all components of the MVC application, including unit tests for the model, integration tests for controller actions, and UI tests for the view layer. Inadequate testing can lead to undetected bugs or regressions.

**SOLUTION:**

**Code Reviews and Peer Testing-** Conduct code reviews to ensure code quality and adherence to best practices. Involve team members in peer testing to identify potential issues and provide feedback on the application's functionality and usability.

**Benefits & Impact**

One enhancement we're proposing for our simple calculator project is the integration of both History and Memory features. This entails incorporating functionalities to showcase a log of past calculations, enabling users to review or reuse prior results. Additionally, we aim to introduce memory storage capabilities, permitting users to save and access intermediate outcomes throughout their calculations.

**Quantify the benefits whenever possible (e.g., cost savings, increased efficiency, improved quality).**

For the improvement of our project I can say that it can improved the quality of our simple calculator and the features of our project.

**Project Plan & Timeline**

Present a step-by-step plan for implementing the project.

**Phase 1: Planning and Design**

* Define the requirements and features of the calculator.
* Design the user interface layout.
* Determine the programming language and platform for development.

**Phase 2: Implementation**

* Develop the basic functionalities of addition, subtraction, multiplication, and division.
* Implement the user interface according to the design.
* Include error handling for invalid inputs.

**Phase 3: Testing and Debugging**

* Test the calculator for accuracy and functionality.
* Debug any issues or errors found during testing.
* Ensure compatibility across different devices and screen sizes.

**Phase 4: Deployment**

* Prepare the application for deployment.
* Publish the calculator app to the desired platform.

**Break down the plan into phases with clear milestones and deadlines.**

Week 1: Planning and Design

* Define requirements: Days 1-2
* Design UI layout: Days 3-4
* Choose development platform: Day 5

Week 2: Implementation

* Develop addition functionality: Days 1-2
* Develop subtraction functionality: Days 3-4
* Develop multiplication functionality: Days 5-6
* Develop division functionality: Days 7-8

Week 3: Testing and Debugging

* Test basic functionalities: Days 1-2
* Debug issues: Days 3-4
* Compatibility testing: Days 5-6

Week 4: Deployment

* Prepare for deployment: Days 1-2
* Publish app: Days 3-4
* Final testing before release: Days 5-6

**Identify the resources needed (e.g., personnel, equipment, budget).**

**Personnel:**

* Software Developers: Responsible for writing the code for the calculator application. Depending on the scale of the project, you may need one or more developers proficient in the chosen programming language
* UI/UX Designer: Designs the user interface (UI) of the calculator application, ensuring it is intuitive and user-friendly.
* Project Manager: Oversees the development process, coordinates tasks, and ensures deadlines are met.
* Technical Support: Provides assistance to users and addresses any technical issues that may arise post-release.

**Equipment:**

* Computers: Each team member will need a computer with appropriate hardware specifications to develop and test the application.
* Development Software: IDEs (Integrated Development Environments) such as Visual Studio, Eclipse, or IntelliJ IDEA for writing and debugging code.
* Version Control System: Utilizing a version control system like Git for collaborative development and code management.

**Evaluation & Risk Assessment**

**Describe how you'll measure the success of the project.**

Our completed simple calculator project successfully performs basic arithmetic operations such as addition, subtraction, multiplication, and division accurately. The user interface is designed to be user-friendly and intuitive, ensuring ease of use. During testing, the application runs smoothly, providing a seamless experience. However, currently, the project is only compatible with one type of device, specifically laptops. Despite these achievements, we continue to work on expanding its compatibility and further enhancing its functionality.

**Top of Form**

**Identify potential risks associated with the project and propose mitigation strategies.**

Technical Complexity- Despite its simplicity, implementing the MVC architecture correctly might pose a challenge, especially for developers who are not familiar with this design pattern. Ensuring proper separation of concerns and maintaining code readability could be challenging. All my team members found coding using MVC challenging due to their limited familiarity with the architecture. Despite this, everyone is putting forth their best efforts to overcome this hurdle and ensure successful project completion.

Top of Form

**Conclusion**

**Summarize the key points of your proposal.**

* Objective- Our project's is to develop a simple calculator with a clean and intuitive user interface, allowing easy input and viewing of calculations. It will perform addition, subtraction, multiplication, and division accurately and efficiently. Currently limited to laptops, our aim is to enhance compatibility for use across various devices.
* Development Approach- Our team will follow an iterative development approach, starting with planning and design.
* Top of Form

**Reiterate the value proposition of your project and its potential impact.**

Our project is about making a simple calculator that works perfectly for adding, subtracting, multiplying, and dividing. Right now, it only works on laptops, but we want to make it work on other devices too. This will make it easier for people to do calculations quickly, no matter what device they're using. It's all about making math easier and more convenient for everyone.