目录

[Executive Summary 1](#_Toc117257803)

[Requirements 2](#_Toc117257804)

[Design 2](#_Toc117257805)

[View 2](#_Toc117257806)

[Model 2](#_Toc117257807)

[View Model 2](#_Toc117257808)

[Meeting Notes 3](#_Toc117257809)

[Sync-up meeting 10.19 3](#_Toc117257810)

[Takeaways: 3](#_Toc117257811)

[To-Learn lists: 3](#_Toc117257812)

[To-Do lists: 3](#_Toc117257813)

[Sync-up meeting 10.26 3](#_Toc117257814)

[Questions: 3](#_Toc117257815)

# Executive Summary

Goal: create a WPF desktop application that mimics a simple calculator with the following operations, +, -, \*, / and clear, and a standard keypad layout

Deliverables:

1. software requirements, break tasks into smaller tasks underneath each requirement if necessary
2. a git repository on GitHub
3. written in C# using Window Presentation Foundation (WPF) and Model-View-View Model (MVVM) design pattern
4. unit tests
5. a 25-min presentation on MVVM, using the example(s) from the code
6. a working application

Target demo date: 10/27

# Requirements

The software application shall allow the user to input integer and decimal numbers.

The software application shall allow the user to perform basic calculations including addition, subtraction, multiplication, and division.

The software application shall have a layout like Microsoft Calculator. All the digit button cluster in the left bottom. Calculator sign buttons on the right in cyan. Clear, Delete, Equal Button in the top in gold, just below the Text Box.

The software application shall

* display the input digits when the user pushes the button from 0~9 and dot.
* Display calculator signs too, but just like the Microsoft calculator, when you type a calculator sign in the same place multiple times, it only shows the last typed calculator sign.
* Display the calculated result when pressing the = button, in the form “A+B=C”。
* After showing the calculated result, the next input should clean the screen.

The software application shall be able to detect valid input and inform the user.

* Detect number with two dots
* Detect two calculator signs in different position.

Mark: The scope is the same as the requirements.

# Design

## View

The layout mimics the Microsoft calculator, With buttons: 0123456789, +,-,×, ÷, =, delete, and clear.

Display the num buttons in the left bottom and in order as the Microsoft calculator. Text Box in the very top, and the clear, delete, = button below it in gold. The calculator sign in the right and in cyan.

The Text in the text box should starts at right.

日历

描述已自动生成

## Model

The only data is a string which shows in the text box.

## View Model

An attribute bound to the view’s text box.

A Command bound to the view’s button. The business logic can be injected into the command using interface.

Interface: ICalculator

Takes the current text box string and the input string as input. Return the new text box string.

The business logic: Calculator

Calculator is inherited from ICalculator, thus can be injected into the viewmodel. The

The calculator calculates on the form “num1 = function (num1, num2)”, and shows up as “A+B=C”

|  |  |  |  |
| --- | --- | --- | --- |
| **Cases** | **Main actions** | **Edge cases1** | **Edge case2** |
| Calculator sign | Change the function | -A\*-B🡪minus input; | -+, 1+2+🡪multiple input |
| Number | Cast value to the number | Decimal input |  |
| Equal | Fire the function | Fire clear after equal |  |
| Clear | Clear everything. |  |  |

It is better if we put input validation and input casting part in “Tools.cs”, since it will be more convenient to add new features without touching the internal logic of the calculator.

The logic shows in pipeline form like this:

图示

描述已自动生成

# Meeting Notes

## Sync-up meeting 10.19

### Takeaways:

* Development loophole: Requirements => Scope => Design Implement Test => Deploy => Gather feedback => Requirements.
* Make more documentation and notes inside of the code.
* Word documentation for planning, Readme for code deployment; Notes inside of the code for maintenance.

### To-Learn lists:

* ~~The package: CommunityToolkit.Mvvm~~
* ~~Using Command Parameters to merge all the functions into one big function.~~

### To-Do lists:

* ~~Omit the name and click mode in the button, the press and release modes both are ideal for the job.~~
* ~~Using proper signs for multiply and divide.~~
* ~~Characters inside the text box should start from the right.~~

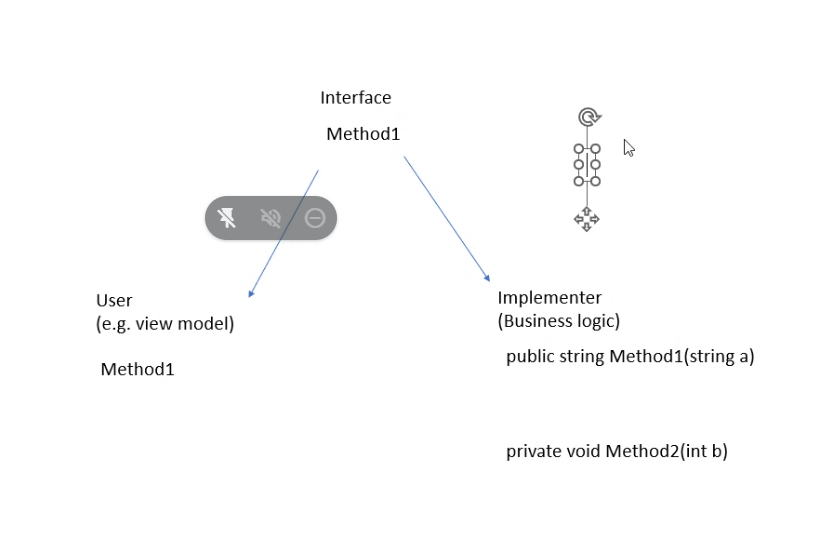
## Sync-up meeting 10.26

### Takeaways:

* Inject the business logic into the View Model, the view model should just be responsible for maintaining the view.
* Using more precise language and graphs to pin-point requirements like the input validation parts.
* Using diagrams more, like UML, sequence diagrams.

### To-Do lists:

* Using interface and “app.cs” to inject the business logic into the view model, and thus decouple the view, view model and model.



Github reposity: <https://github.com/screw-44/Csharp-Projects>

The Calculator Final is the software itself, and the Test does the unit test.