MyCalculator project

-Testing Process Report -

Author: Filip Neagoe

Contents

TESTING STRATEGY

Due to the fact that the application proposed to be tested **is not accompanied by any formal documentation** such as Functional Requirements Document (FRS) or Project Plan document, the testing process will not be performed in a “classic” manner.

The testing phase associated with the application development process will not fully follow the steps of Software Testing Life Cycle (STDL) due to the situation we face.

Considering the fact that this application is not a newer version of an old product then I have not access to the old specifications to refer. The client is not available for clearing queries and doubts because such an approach (testing without requirements) requires frequent meetings and doubt clarification sessions to make it run smoothly.

Also I have not the knowledge of all the documents that the developers are referring to for coding purposes, so that I can refer to the same to get an idea about how the product will work.

The development team is not available to consult.

Given the testing scenario I am in, I will further present the strategy I will follow to successfully complete this testing process and achieve a satisfactory result.

The process follows the following steps:

1. Proper Planning

Most lacking area in such projects is planning. Testers must do proper planning at the very early stage. Planning is required to ensure which deliverable are expected from testing team and types of testing we are going to perform.

Even if the testing process does not follow the classic recipe and some of the testing methods that will be used do not focus on this aspect, a good organization is essential for performing a quality test.

I believe that the tester should have control of the process even if it gives him more "freedom" (as in this case). Efficiency is an important feature to consider.

The tester should know which feature will be executed first depending on their hierarchy.

2. Let's get to work

Given the ideas mentioned above, my approach strategy looks like this:

First of all, I will make sure that I received the build from the development team and it can be accessed. Next I will deal with the Smoke Testing part which is part of the Tester Acceptance Testing category. I will make sure that the application can be successfully installed and run in the development environment.

Assuming that both the Unit Testing and Integration Testing (here it depends on the case, but in this situation I will consider the fact that this set of tests were performed) processes have been successfully performed by the developers, I will refer to the application as a fully functional system.

Since I was not constantly involved in the testing process of the application, I will consider that the application was developed using the V model strategy because in a real scenario, following an Agile development strategy such as the Scrum method, the delivered build, even the initial one would have shown fewer defects (but this is a demo application and I understand its purposes). So I will deal strictly with the Validation part of the process. It is just a personal remark.

2.1 Testing strategies and methods

The test methods performed belong exclusively to the dynamic testing regime. Static testing cannot be performed under the given conditions due to the lack of documentation corresponding to the project.

For this application, all tests will be performed manually. I don't know anything about the application development process and I don't have access to any such document. So the testing process falls into the **Black Box Testing category (known as Behavioural Testing).**

Both functional and non-functional aspects will be tested, as well as aspects related to the interaction with the user and his experience in using the applications **(GUI aspects).**

In principle, I will test the system as a whole, but without having access to the requirements. I will use test methods and techniques that are ultimately summarized and fall into the **System Testing category.**

So after completing the **Smoke Testing** stage, I will focus on the **Exploratory Testing method.**

Initially I did a **freestyle testing** of the application because I had no interaction with the application until that moment. In freestyle testing, I did not follow any rules, there is no maximum coverage, and I explored the application just to roughly verify its functionality and to get used to it. I intended to get familiar with the software.

Then, for an optimal coverage of testing process, I migrated to **Strategy based and Scenario based types of Exploratory Testing.**

Strategy based exploratory testing can be performed with the help of multiple testing techniques such as risk-based, boundary value analysis, and equivalence partitioning. For this particular case, I used **risk based** and **error guessing techniques** along the process. In this approach, I derived the values or inputs based on their understanding or assumption of the requirements, and I do not follow any kind of rules to perform error guessing technique. It helped me to save lots of time.

Performing Risk Based analysis allowed me to make choices on what tests needs to be run first in a time crunch situation. Before I started testing, I identified all areas in the user most likely to find issue. In such a testing scenario, decisions should be discussed with other team members, both development and testing, to know that the chosen direction is correct. In this case it is not possible, but that is part of the game.

In this sense, **I divided the testing process into 3 areas:** first I aimed to check the detailed functionality of the application. Then I turned my attention to the experience that the application offers the user. In this category I tested various aspects related to GUI. In the third category I included the verification of the aspects related to the interaction of the application with the test environment (operating system), as well as the interaction with other softwares in various scenarios. This division allowed me to perform a more elaborate test, focused on individual areas and explore the application in more detail.

Scenario-based exploratory testing is performed with the help of multiple scenarios such as end-to-end, test scenarios, and real user scenarios.

The test engineer can find defects and also checks various set of possibilities for the multiple scenarios with their application knowledge while they were exploring the application. I also relied on these aspects after I got used to using the application and gained experience with it.

I used Exploratory testing method to understand the flow of the application. This is the main purpose of this approach.

I tried to avoid random testing because certain aspects could have been omitted.

It can be said that I also performed a **kind of Ad hoc testing**, more precisely Monkey Testing. Although the main purpose of the testing was not to break the system, I think that this type of testing is somewhat found in the approached strategy. But I am reserved about this statement.

"Think from the user perspective" it is also a procedure I have used. It can help you to make the software more useful. Give suggestions and solutions. There are various users who are going to use the software in different ways. If I am able to think from their perspective then the software becomes more compatible and versatile.

I also tried to test the application based on ideal user experience. Even if I don't know exactly what a customer intended for a given feature, I can assess what actions would make the most sense to the end user.

Another "procedure" I used in testing the application is the following: self analyze the system and understand the functionality by my own but this may get more time as this is based on trial and error process. This step can be associated with freestyle exploratory testing phase. Even if it's something a user wouldn't likely do, it's good to include in a test case. Brainstorm any possible action you can do with any particular feature is the main idea here.

What I did next was to research similar features on other apps. Just because a feature is new to the app you're testing doesn't mean it's revolutionary in the tech space. So I analyzed the calculator default software provided by my PC system. It is a stable, well-developed, tested and optimized application, so it is a very good benchmark to test the initial version of the MyComputer application.

Both positive and negative testing were used in this category of functional and behavioral testing.

Regarding the non-functional testing of the software, I performed tests related to accessibility and data volume.

Defects identified during the testing process were highlighted and reported in the test report. Tracking their evolution will be done using specific tools (such as Bugzilla; not used in this case).

Next, the development team will decide on the measures to be taken to eliminate these defects.

For the next build/version of the application the older/current version of the application can be used as a reference to test the future release of a software product. Now, I admit this is in negation to the rule, "Never write test cases using the application as a reference". However, when we are

working in a less than perfect situation, we have to mold the rules to fit our needs. Or, preferably, at the next update we will have the related documentation.

In the next version or when the reported defects are fixed, regression tests will be performed.

The sanity tests, a subset of the regression tests, will be run first to make sure that the features that have suffered errors work correctly. Performing these tests manually is time and resource consuming (especially human resources). Therefore, some of them can be automated using specialized software for writing the necessary scripts (such as Selenium or QTP; this step is not approached in this work report, but is on the “TO LEARN AND TO DO” list).

Also, in the subsequent development and testing session, approaching an Agile strategy method would be an optimal option for the process.

One of the disadvantages of exploratory testing is that the test engineer can misunderstand the feature as a bug and bugs can be misunderstood as a feature. Although I think exploratory testing is a helpful method, my main concern is how to know if I'm seeing a bug or a "feature." Be prepared to spend significant time in research or reporting defects that are not really issues.

From this point of view, the reporting process of defects becomes quite complex and risky. As developers and customers cannot be consulted in making such decisions (my case), the task lies exclusively with the tester. Using his knowledge and experience, he is forced to make the most appropriate choices.

3. Conclusions

As you don’t have any formal requirement listed, hence exploratory testing is the best option. It is a hands-on approach in which tester is involved in exploring the software, what it does and doesn’t do. Beauty of Exploratory testing is, testers do minimum planning and maximum execution. Test design and test execution activities run in parallel, hence testers documents key aspects of the application during execution.

Still, it is good to have a standard list of expectations for any feature. Even if the list is short, there are requirements you can expect for any feature.

Being asked to test without requirements can be nerve-wracking.

Provide a summary of what you'll be testing. When you know what to test and how to test, you can make your own documents.

Honestly, there is really no substitute to a well documented functional / system requirement document with elaborate use cases and mock up screens. Although we have to admit that this is becoming a rarity in the industry due to rapid development cycles and a paradigm shift towards minimum or no documentation.

No requirements? No problem!

Even though having no documents creates a challenge, it can also occasionally work as an advantage. I find this kind of testing to be fun and exciting when I get a chance to perform more exploratory testing than being restricted by a defined test plan. This challenge can help think beyond and produce better, more considered output.

TESTING PROCESS WALK-THROUGH

This section provides a closer look at the testing process. Screenshots made during the entire process will complete the entire puzzle.

Step 1

Make sure that I received the build and I can accesed it. (**PASSED**)

Step 2

Setup the environment. (**DONE**)

Step 3

Execution of installation tests and smoke tests. (**PASSED**)

I received the .exe executable file from the team. It is accessible and can be run. Test data can be entered.

The application is stable, can be opened and closed. The interface is displayed, it is visible, the characters are legible. The window basic control commands work. (minimize, restore, maximize, close).

Step 4

Test data preparing. Determining the types of input data for a comprehensive test of the application. In this case, integers and decimals, both signed and unsigned, as well as other characters prohibited in the use of a calculation application will be entered.

Step 5

The actual testing process of the application.

First the functional aspects of the application will be tested, then those related to the interaction with the user. Finally, we will test the interaction of the software with the operating system and other applications.

Testing functional aspects of the initial build of MyComputer application

Test no.1 (PARTIALLY PASSED)

Title: Opening the application.

I noticed that when accessing, the application always opens in the taskbar which is pretty annoying for me as a user. Each time I access the application, I need to open the application window from the taskbar. It is good to keep this in mind.

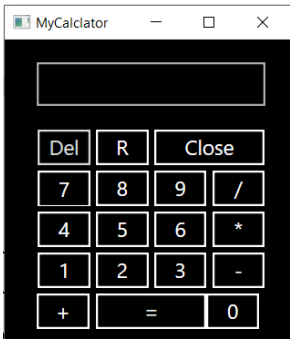
Test no.2 (PASSED)

Title: Test the functionality of “Del” button

It seems that pressing the ‘Del’ button deletes the entire display content.

This function is equivalent to the Clear all (C) function.

It works properly, but it is not very intuitive.

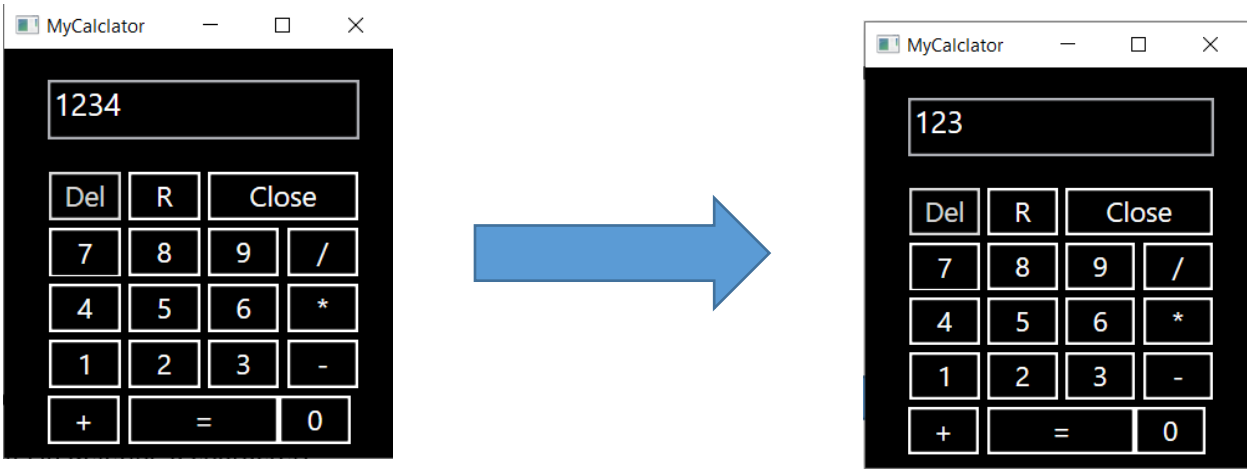


Test no.3 (POSSIBLY PASSED)

Title: Functionality of “R” button

Pressing the ‘R’ button results in deleting a single character at once. Not intuitive at all.

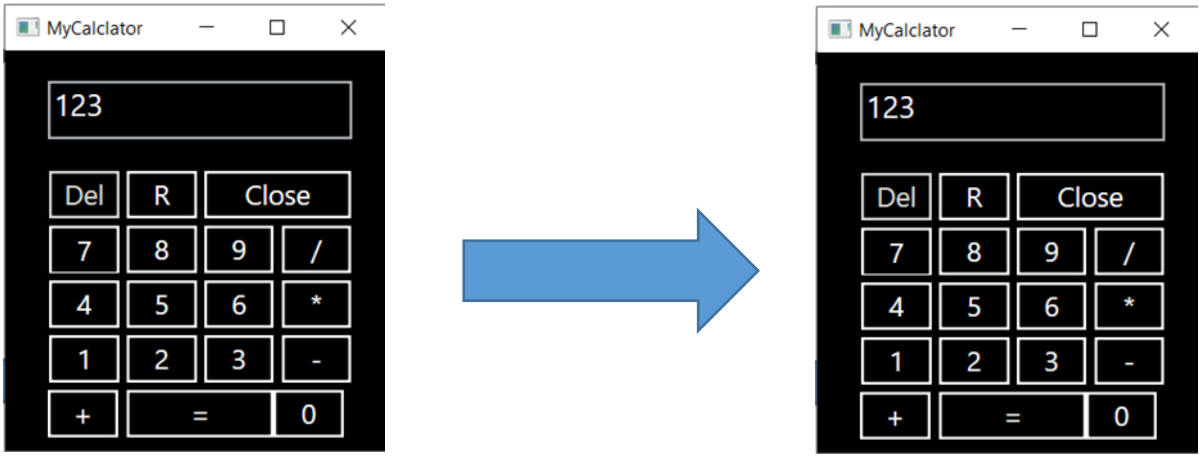
After one press of the button, the situation will be something like this.



Test no.4 (FAILED)

Title: Functionality of “Clear” button

Pressing the ‘Clear’ button does not produce any results. I think its functions was not implemented yet considering that it is the initial build. However, it is quite confusing for the user.



Test no.5 (PASSED)

Title: Functionality of “Minimize” button

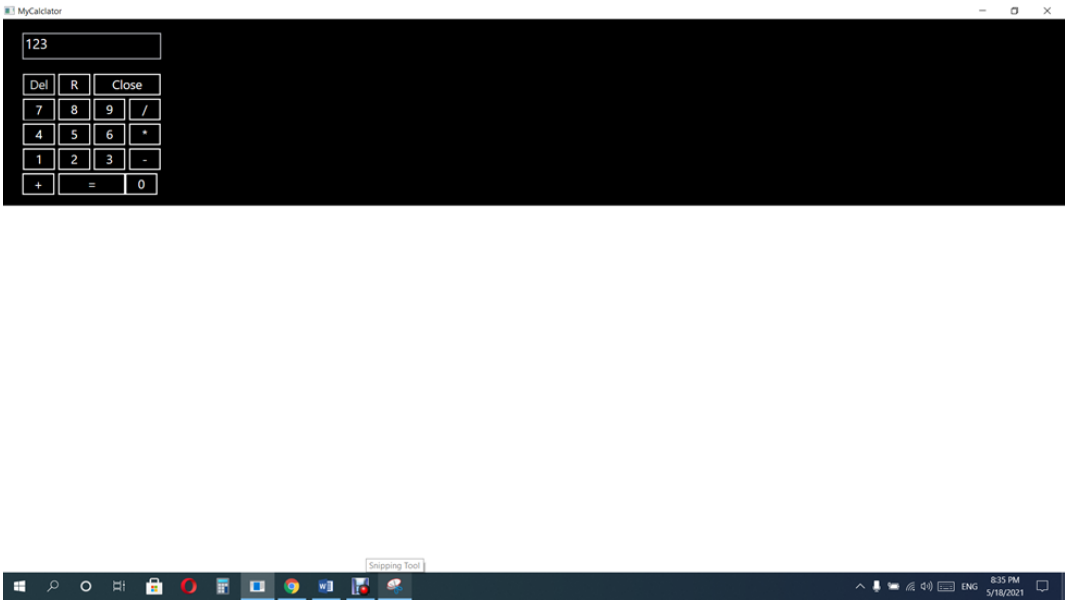
Pressing the ‘Minimize’ button, the application window is removed from the screen in the taskbar.

Test no.6 (FAILED)

Title: Functionality of “Maximize” button

Pressing the ‘Maximize’ button the application window should be resized to occupy the entire screen. Also, the graphic elements of the application must be

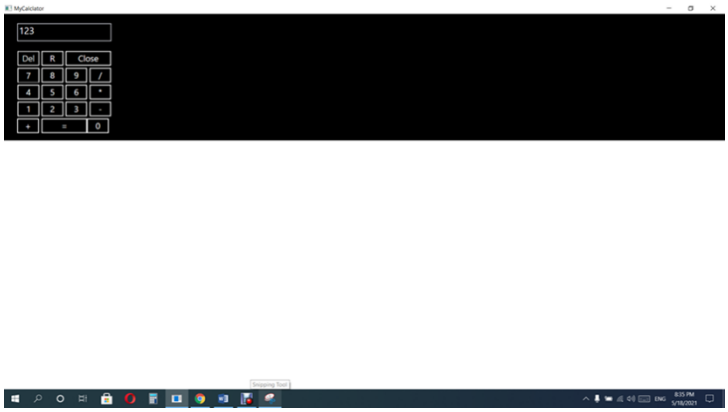
resized accordingly. This is not happening. Not even the background color is evenly distributed.

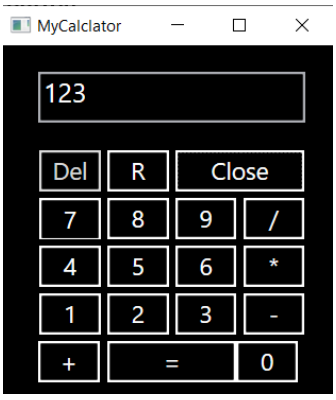


Test no.7 (PASSED)

Title: Functionality of “Restore” button

Pressing the ‘Restore’ button the application window should be resized to the default size. Also, the graphic elements of the application must be resized accordingly.





Test no.8 (PASSED)

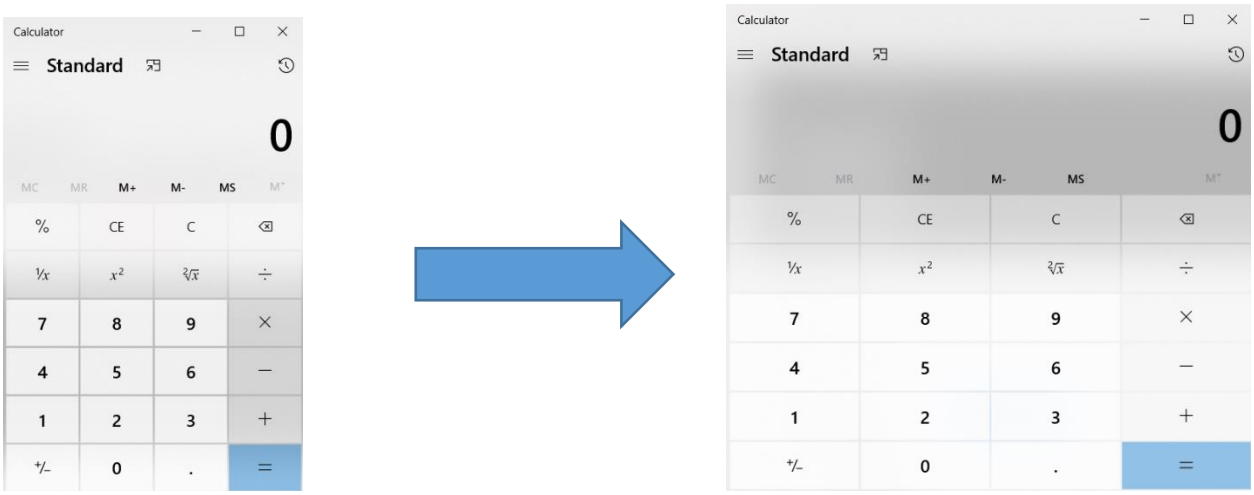
Title: Functionality of “Close” button

Pressing the ‘Close’ button the application should end its activity. Application main window is closed as well. The application icon disappear even from the taskbar.

Test no.9 (FAILED)

Title: Application window expanding

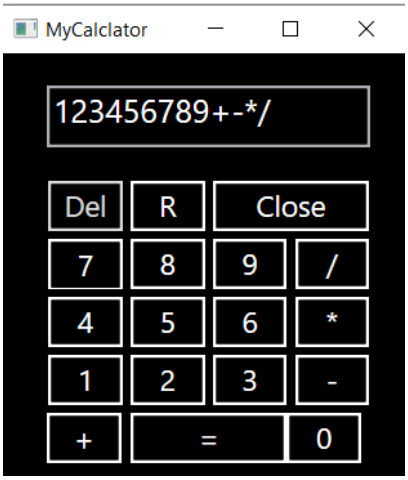
If we try to resize the application window with the cursor, we will notice that the buttons and the computer display are not resized as they should be. In direct comparison with the default PC calculator, MyComputer software seriously fails at this function. The interface is not adaptable.



In the image above we can see how the default computer fulfills this function.

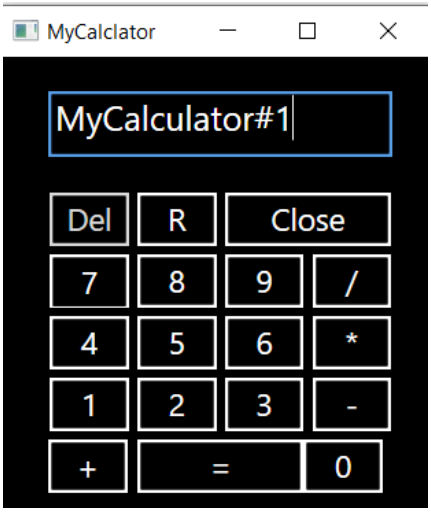
Test no.10 (PASSED)

Title: Input digits, numbers and basic mathematical characters (+, -, *, /) is allowed.



Test no.11 (FAILED)

Title: Enter letters and other characters that not belong to a calculator input system (~, ! , @, #, $ etc.) is not allowed.

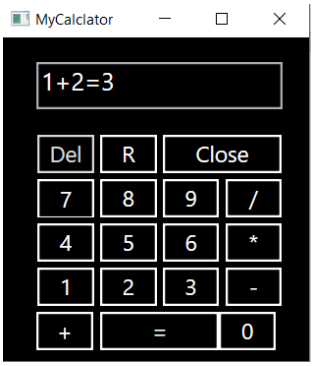


Following this test, it is observed that any kind of characters are allowed.

Test no.12 (PASSED)

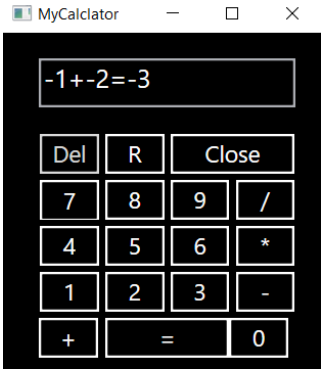
Title: Testing the sum function.

Case 1: Both numbers are positive.



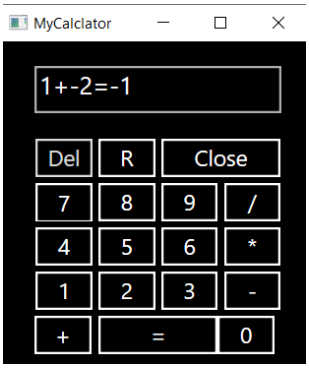
It works as it should.

Case 2: Both numbers are negative.



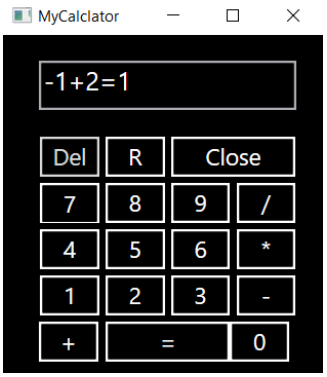
It works as it should.

Case 3: First number is positive & the second one is negative.



It works as it should.

Case 4: First number is negative & the second one is positive.

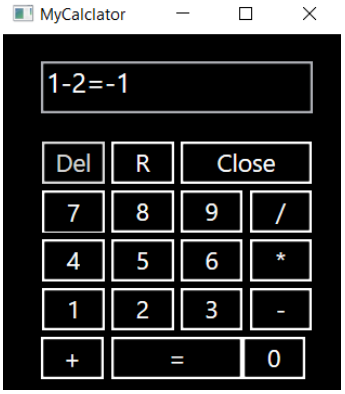


It works as it should.

Test no.13 (FAILED)

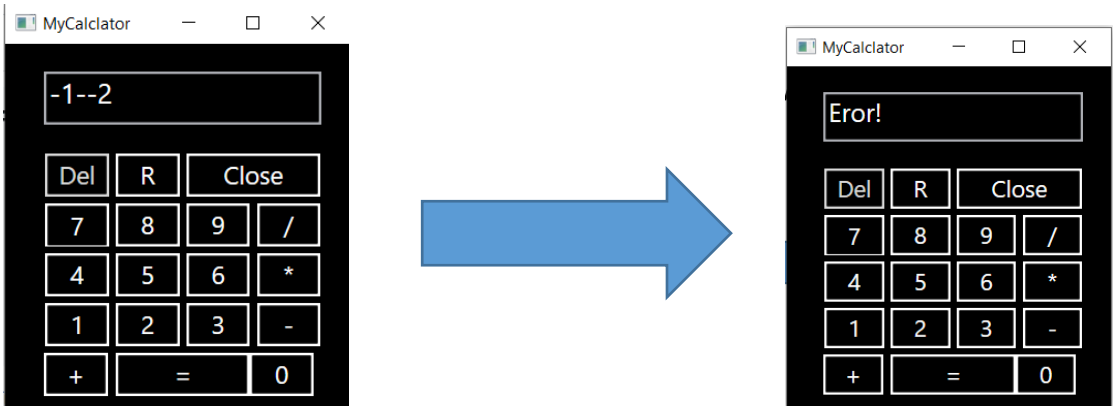
Title: Testing the subtraction function.

Case 1: Both numbers are positive. (PASSED)

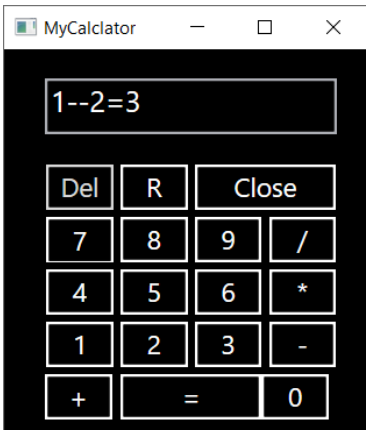


It works as it should.

Case 2: Both numbers are negative. (FAILED)

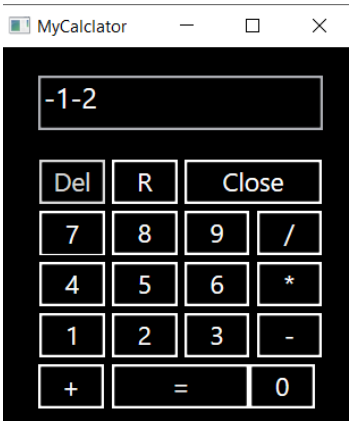


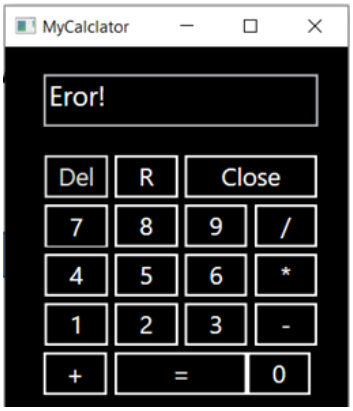
Case 3: First number is positive & the second one is negative. (PASSED)



It works as it should.

Case 4: First number is negative & the second one is positive. (FAILED)

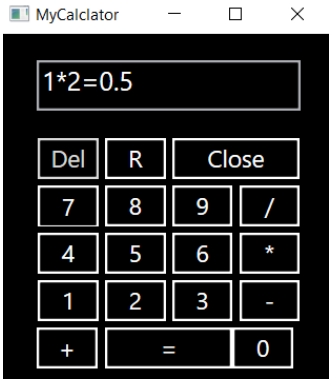




Test no.14 (FAILED)

Title: Testing the multiplication function.

Case 1: Both numbers are positive (FAILED).



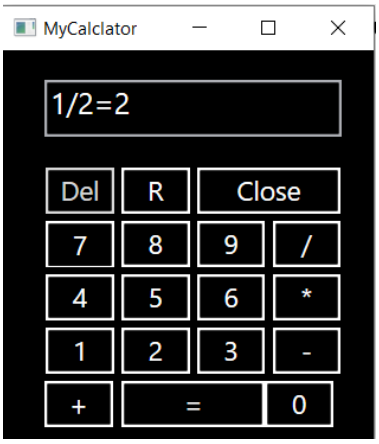
Mention:

For the other three test cases, an error message is returned (“Eror!”). To save time, their example has not been attached.

Test no.15 (FAILED)

Title: Testing the division function.

Case 1: Both numbers are positive (FAILED).

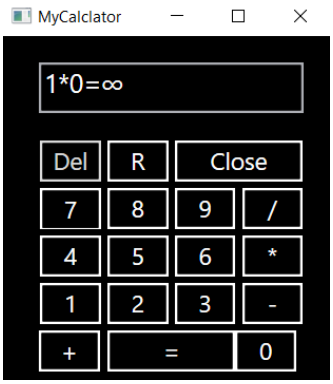


Mention:

For the other three test cases, an error message is returned (“Eror!”). To save time, their example has not been attached.

! After testing on several data sets, it can be seen that the multiplication and division functions are reversed.

Experiment: I tried to make a division at 0 using the inverted functions. The computer does not display any error to warn the division at 0, an operation that is neither possible nor allowed. Instead, it calculates the limit for a function with denominator equals to 0 and a positive numerator.

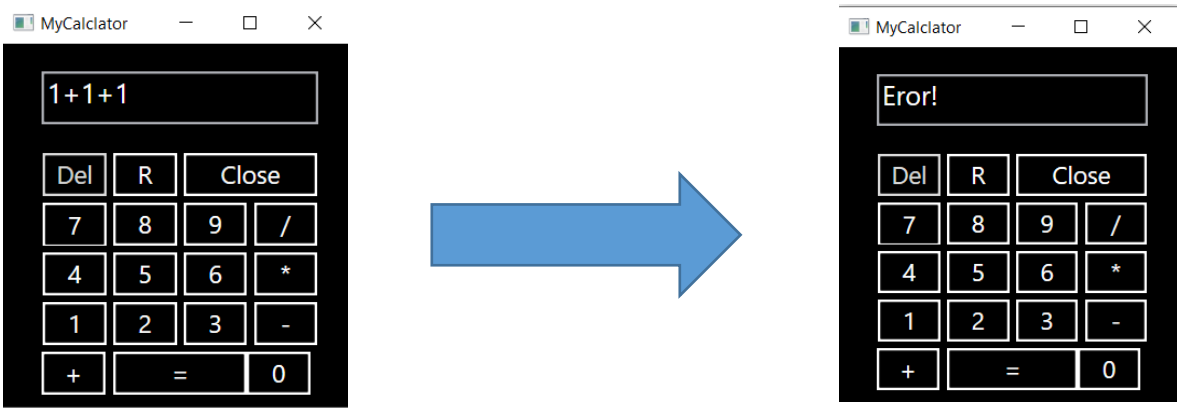


Mention: Despite the fact that the function of entering decimal numbers using the application interface is not available, decimal numbers can be entered into the application via the keyboard. The test results for arithmetic operations using integers are also valid for decimal numbers.

Test no.16 (FAILED)

Title: Input more than 2 parameters for executing an operation.

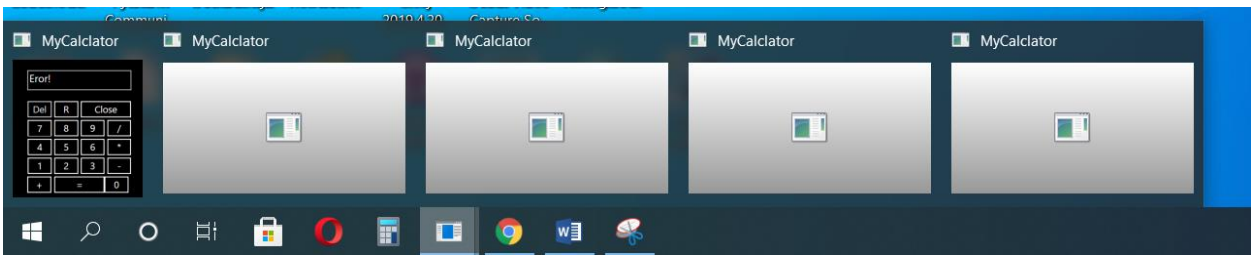
Case 1: Enter 3 parameters.



Test no.17 (PASSED)

Title: Creating multiple instances (open the application multiple times).

Mention: When multiple instances are created, the window of none of them is opened on the desktop.

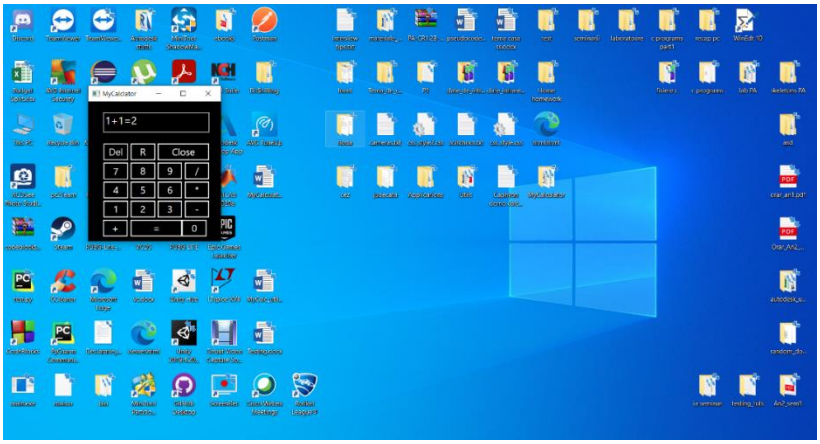


Test no.18 (FAILED)

Title: After close the last open instance of the application, if a new one is created afterwards, it will open in the same place where the last one was closed.

The default calculator of the PC system does this (if a new window is opened, it will be opened where the last one was closed).

All the new instances will be opened in the upper left corner of the desktop.



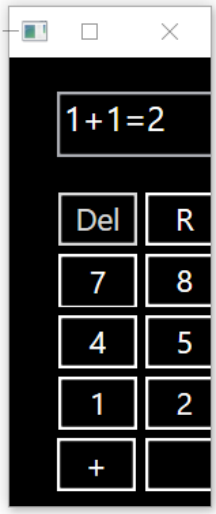
Test no.19 (FAILED)

Title: After close the last open instance of the application, if a new one is created afterwards, it will have the dimensions of the last closed one.

All the new created instances will have the same default size.

Test no.20 (FAILED)

Title: When the application default window is minimized to the maximum the buttons in the top bar will not exceed the size of the window.



Test no.21 (FAILED)

Title: When the application default window is minimized the size of the buttons and the computer display adapts so that the interface and functionality are not affected.

Test no.22 (PASSED)

Title: When we double-click on the top bar of the application window, it is maximized and minimized.

Test no.23 ()

Title: The error message can be removed to perform another operation.

The error message can ONLY be removed to perform another operation if the "Del" key is pressed.

In the case of the default calculator, any key can be pressed to proceed to another operation.

Test no.24 ()

Title: If error message is displayed, it can be erased letter by letter by pressing "R" button (weird functionality).

I think it should simply disappear all at once by pressing any key in order to proceed further.

Test no.25 ()

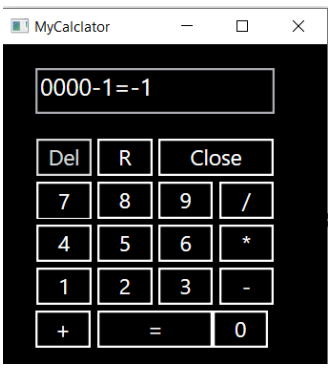
Title: The application allows the user to input more than 2 characters such as +, - , *, / after a number.

This is not possible in the default calculator. But in this case, this possibility makes up for the lack of parentheses.

Test no.26 ()

Title: Zero can be added more than once when executing operations.

Exemple:



This thing is useless in calculation and make no sense.

Test no.27 ()

Title: CE (Clear Entry) function isn't available.

This functions permit to the user to delete just the last entrance in an operation.

Example: 1 + 2. Press CE and 2 will be removed (reseted to 0).

Test no.28 (PASSED)

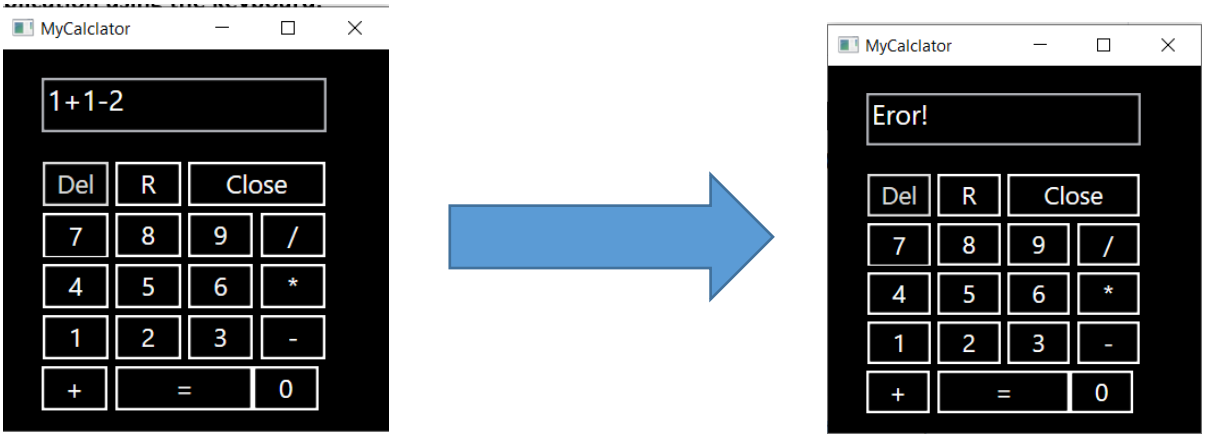
Title: Navigate the application using the keyboard.

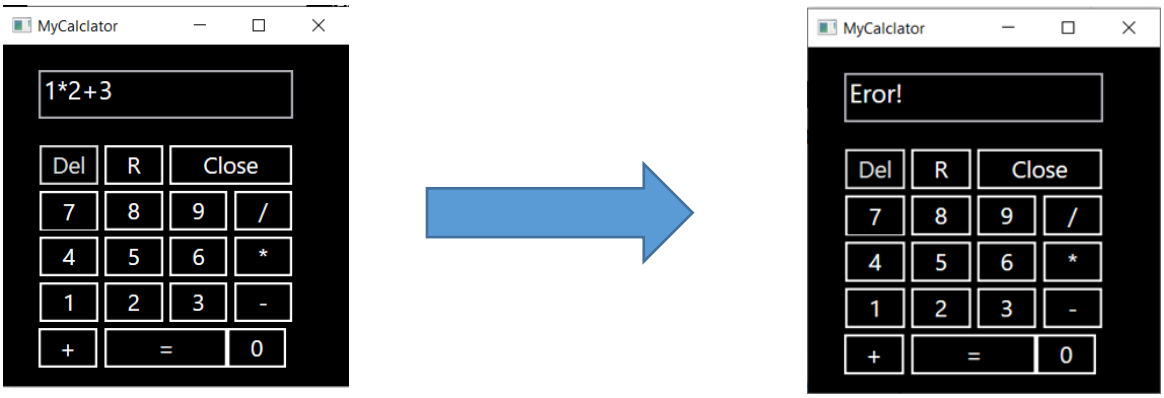
Navigation in the application using the keyboard is possible. Unfortunately, the current button we are on is highlighted only by pressing. Thus, navigation from the keyboard is difficult.

The selection role has both the Space key and the Enter key, which can be confusing.

Test no.29 (FAILED)

Title: Multiple operations are possible.





Test no.30 ()

Title: The cursor from the calculator display can be moved using the mouse or pressing the Space key.

I think this should be not possible in such an application where the user work with numbers and basic symbols.

Test no.31 (FAILED)

Title: The result of an operation can be copied.

The default calculator offer this possibility.

Test no.32 (FAILED)

Title: Input values are limited.

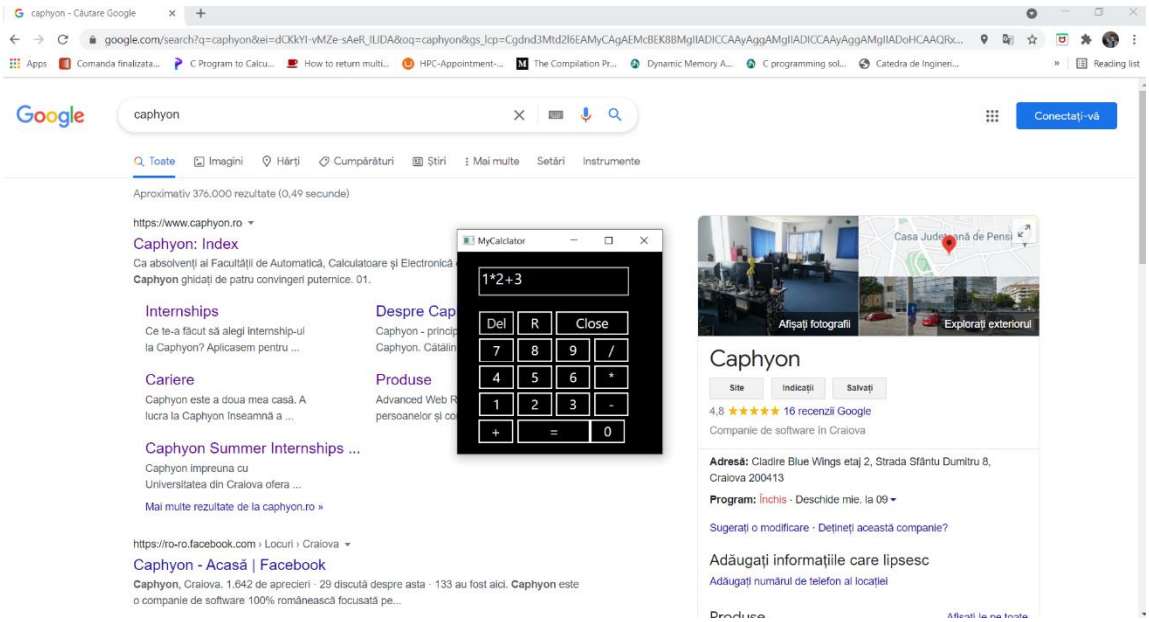
It seems that numbers with a huge number of digits can be entered in this application. All other applications of this kind are limited from this point of view. Anomalies may result. Also, the computer does not display any errors in case of overflow.

Testing the interaction of the software with other applications

Test no.32 (FAILED)

Title: When MyCalculator app is open on desktop (even in default size or expanded size) and I try to open another application or software, the opened application should be in the foreground.

The other opened application remains in background until I close or put in taskbar the MyComputer application.Even when the other open application window is explicitly selected, the MyComputer window does not run in background.



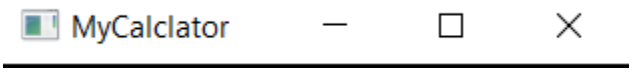
Test no.33 (PASSED)

Title: The MyCalculator application icon can the added to taskbar.

Testing the user interaction with the application interface

Test no.33 (FAILED)

Title: The name of the application is displayed correctly.



Test no.34 (FAILED)

Title: The graphic interface alignment is accurate.

When the application is opened in the default form and size, the left and right margins are not equal, “+” button box is move slightly to the left compared to the other boxes on the same column, there is no space between “=” and “0” button boxes.

The bottom line of the 7-digit box is inaccurate (thinner than the other ones).

“+”, “-“, “*”, “/” and “=” symbols are not properly centered in the boxes.

Between the third and fourth columns a bigger space is left compared to the others columns.

Test no.35 ()

Title: When enter in the application for the first time, “1+1=2” message is displayed.

I think it should display ‘0’.

Test no.36 (PARTIALLY PASSED)

Title: When an error occurred in the calculation, the thrown message is displayed.

“Eror!” is displayed instead of “Error!” .

Test no.37 (FAILED)

Title: The chosen colors for the GUI are optimal.

The numbers are hard to see when selected.

I think “Del” button font is coloured in another color then the others. Also, when it is selected, its color is slightly different. Anyway, they are pretty hard to notice.

Test no.38 (FAILED)

Title: Buttons are intuitive.

R does not reset the result and Close does nothing so far.

Also I would used all capitals for these primary functions of the calculator.

I would also reverse the positions of the "+" and "0" keys.

Test no.38 (FAILED)

Title: When introduce numbers using the keyboard, they are highlighted.

This feature would significantly improve the user experience of the application.

Test no.39 (FAILED)

Title: Both input and output data are displayed efficiently on the display.

The computer display is rather a box in which text of any length can be entered. The information is not displayed properly. When entering long numbers, the user

no longer has visual access to characters that exceed the display format. He has to scroll to be able to access all the information, which is not practical at all.

Test no.40 ()

Title: If the user clicks the display edges, these become highlighted.

The display behave like a text box. This behavior is not found in such applications.

Regarding the **non-functional testing** of the application, I focused on the **acceptance and data volume tests.**

From the point of view of the experience offered to the user, the application does not excel by far.

For a regular user the application offers a relatively decent interface and functionality that meets certain criteria. However, many of the basic functions are unstable and do not provide satisfactory results. In any case, the application does not meet the conditions so that a user with special needs to use it.

Regarding the behavior of the application in working with a high volume of data, it behaves quite well. Implemented functionalities that work correctly provide satisfactory results for various volumes of data, even for large values of numbers involved in computational operations in this case.

Those are the majority of tests I have implemented for this project. There could be many for sure. Exhaustive testing is not possible and bug free software is a myth.

Some of the performed tests have no status (passed / failed) because their subject can be a feature wanted by customer. This is one of the “risks” of testing without proper documentation.

Next, I would have created test scenarios and test cases in a tabelar form using Excel for a better management of the performed test and to analyze the coverage of the test performed as well.

Defects encountered during the process will be reported and brought to the attention of the development team.

Documents such as RTM and Test Metrics can be made, although they are not very conclusive in the case of such a test scenario.