Term Individual Project

Report

Web Calculator and Finite State Machine

Vinay Kumar

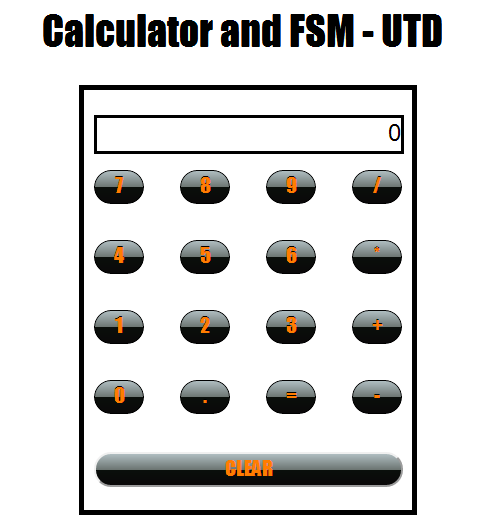
[Vxk123530@utdallas.edu](mailto:Vxk123530@utdallas.edu)

2021147431

INTRODUCTION

I have designed the calculator using the following technologies

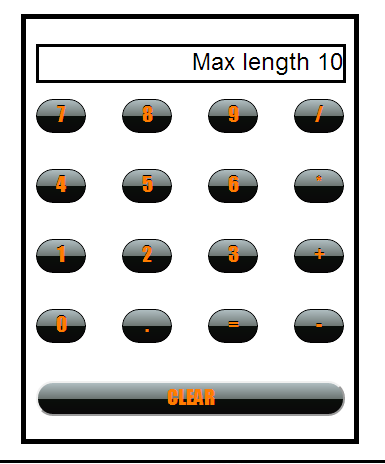
1. CSS 3.0
2. Javascript
3. HTML
4. ASP.NET 4.0

I have also designed a finite state machine on the basis of which the calculator was developed. I also took the liberty to host the calculator and the finite state machine on a free web hosting service. [http://www.fsmcalculator.somee.com](http://www.fsmcalculator.somee.com#)

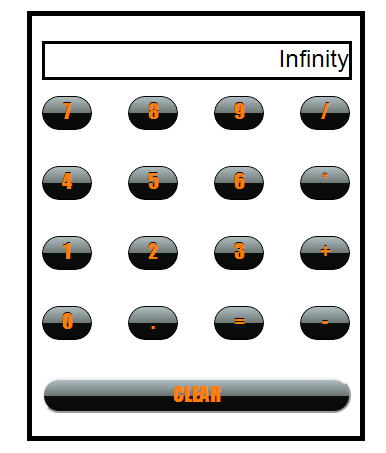
**INPUT AND OUTPUT AND ERRORS**

I have attempted to replicate a real calculator. Certain deviations will however be visible, owing to the project guidelines.

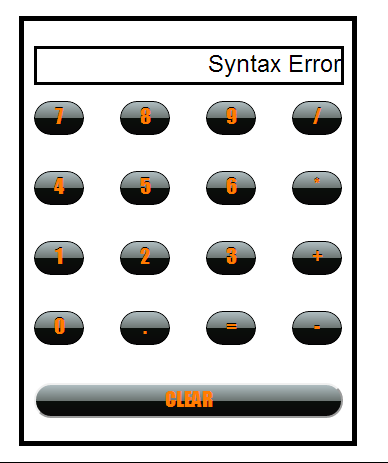
* + - 1. Errors – When the length of the input exceeds 10 Digits, then the user gets a messages – “MAX LENGTH : 10” . This number can be easily modified by changing the value of a global variable.



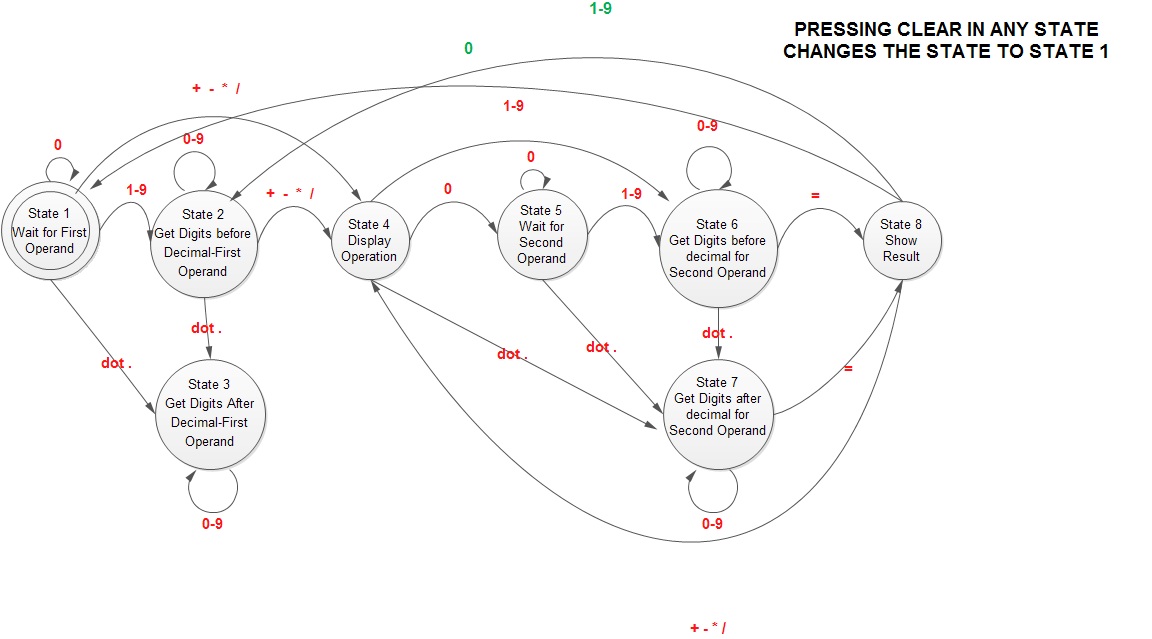
* + - 1. A user is allowed to press a + , - \* , / symbol directly because in a real calculator this is allowed. The first operand is taken as ZERO, which is visible by default.
      2. When the user divided a number by 0 , then a message “Infinity” appears on the screen.



* + - 1. When a user exceeds the digit length of 10 while entering the second operand, then he again gets a chance to enter the second operand, instead of having to enter the both the first and second operands all over again.
      2. The calculator displays a message “Syntax Error” when the user enters the following sequence of characters 3+= . This is because a real calculator also displays this error. The second operator has not been entered yet.



**FINITE STATE MACHINE DIAGRAM EXPLANATION**

****

I had drawn the exhaustive FSM before starting the coding. This ensured that the coding process was smooth.

I have broken down the process into 8 states for greater clarity of thought process and implementation.

I have also included a clear button, pressing which leads to the setting of the calculator to state 1. This button can be pressed in any state and then the user can start all over again.

Now I have considered the first state to the state in which the calculator is waiting for input from the user. The default first operand is 0. The user is not allowed to enter multiple zeros in this state. The user will have to enter digits 1 to 9 before he is allowed to enter any more zeros. Also the user cannot enter multiple decimal points in this calculator. Multiple “dots” when pressed are ignored.

The user can directly press an operator sign. In that case the first operand will be taken as zero. This is a slight deviation from the original specification, however I included this because this is how a real calculator works.

The calculator displays a message “Syntax Error” when the user enters the following sequence of characters 3+= . This is because a real calculator also displays this error. The second operator has not been entered yet. Thus it cannot be assumed to be zero in this case as there is no default zero on screen in this case.

When the user presses an operator sign then the first operand disappears from the screen and the operand sign appears. Now the user enters the second number. When the user press the “=” sign, the second number displayed in the calculator screen disappears, the correct arithmetic result gets displayed on the calculator screen.

Also if the user presses an operator sign after displaying the result then the result is considered as the first operand and the user is can to enter a second operand and perform the following action. This feature has been included because this is feature is present in a real calculator as well.

**IMPLEMENTATION STEPS**

The project was implemented in two steps. Initially all the functions were implemented for the + function only. Once everything turned out to be perfect, all the other functions were included into the operator.

At present the calculator only offers the four basic operations of PLUS , MINUS , MULTIPLY , DIVIDE.

Inclusion of other functions like *sin* will require additional states in the FSM. This can be implemented in the future versions.