

# **SRS**

## **(SOFTWARE REQUIREMENT SPECIFICATIONS)**

### **~~~~~::TEAM MATES::~~~~~~**

**1.Rahul Agarwal -140020017(Leader)**

**2.Satyam Agnihotri -140020062**

**3.Rudrajit Das -140020012**

### **~~~~~::FUNCTIONAL REQUIREMENTS::~~~~~~**

**1) Linux OS**

**2) Code Blocks**

**3) GNU compiler**

**4) EzWindows package(for graph plotting)**

**5) Fparser 4.3 (header file for parsing a string )**

### **~~~~~::LIBRARIES USED::~~~~~~**

**We will be using the following libraries for our project.**

- **stdio.h**
- **iostream**
- **string**
- **assert.h**
- **cmath**
- **cstring**
- **cstdio**

## ~~~~~::: **PURPOSE** :::~~~~~

It is a mathematical application designed by us for mathematicians ,physicists and chemists who quite frequently encounter cumbersome and complicated integrals , solutions to complicated equations ,derivatives etc. Using our project one can do these tasks very easily . Our project is inspired by Wolfram Alpha .

## ~~~~~::~: **DESCRIPTION** :::~~~~~

1.

**DIFFERENTIATOR**-This serves two purposes :

- >Calculates the derivative of a given function at a given point (if it is differentiable at that point)
- >Finds the maxima ,minima and point of inflection of a given function in a given interval

**void differentiator()**-The user is required to select one of the above options in this function.This function will then call the required function(described below) as per the user's choice and display the desired output.

**void derivative()**-If the user selects the first option in **void differentiator()**,then this function will be called by **void differentiator()**.The user is then required to provide a function as an input and also the value of the point at which the derivative is to be evaluated.Output of this function will be the derivative of the function at that point provided the function is differentiable at that point .If the function is not differentiable at that point ,then appropriate message will be displayed .

**void maximaMinimalInflection()**-If the user selects the second option in **void differentiator()**,then this function will be called by **void differentiator()**.The user is then required to provide a function as an input and also he/she is required to provide the interval in which he/she desires to locate the maxima /minima /inflection points.This function will display the positions of the maxima /minima/inflection points and if there are no maxima /minima/inflection points then appropriate message will be displayed.

**2. INTEGRATOR** -This also serves two purposes :

- >Calculates the definite integral of the given function in a given interval .

->Calculates the area bound by the given function in a given interval .

**void integrator()**-The user is required to select one of the above options in this function.This function will then call the required function(described below) as per the user's choice and display the desired output.

**void definiteIntegral()**-If the user selects the first option in **void integrator()**,then this function will be called.The user is then required to provide a function as an input and also the interval in which the definite integral is to be calculated.This function will display the value of the required definite integral .

**void area()**-If the user selects the second option in **void integrator()**, then this function will be called. The user is then required to provide a function as an input and also the interval in which the area is to be calculated.This will function will display the value of the required area.

The above two modules will output different values only if the function attains negative values in the given interval.

### **3. 2D-GRAPHER :**

This will take an input function and the desired scale in which the graph is to be plotted from the user.The user must type the function concerned and also the scaling factor as per the format described in the user draft manual.It will then plot the appropriate graph of the function.

### **4. QUADRATIC EQUATION SOLVER:**

The user is supposed to enter the quadratic equation in the specified format as per the user manual. The function evaluates the roots of the equation using the quadratic formula. It also tells about the nature of roots, whether real or imaginary using the theory of roots.

### **5. CUBIC EQUATION SOLVER:**

This feature finds the real roots of the cubic equation entered by the user in the specified format. It also tells about the number of real and imaginary roots out of the total three roots. The concept uses the functionality of evaluating the function at the points of extrema and then analyzing the various possibilities that arises due to the sign of coefficients.

The functions used are:

**double eval(double x,double a,double b,double c,double d)**-Evaluates the value of the cubic equation with coefficients a ,b, c and d at the point x.

**double findRoot(double a,double b,double c,double d,double startPoint)**-Finds a real root of the cubic equation with coefficients a, b, c and d in the range starting from startPoint to plus infinity, if  $a > 0$  and if  $a < 0$ , then it locates the root in the range starting from minus infinity to startPoint.

**void cubic(double a,double b,double c,double d)**-Displays all the real roots of the cubic equation with coefficients a, b, c and d. It uses the functions eval and findRoot, implementing their functionalities to find the real roots of the cubic equation. It basically uses all the mathematical tests to find the number of real roots and to locate them.

## **6. PARSER:**

This parser is made by us. It is used for accepting functional input and then interpreting it so as to find the value of the function at given point which is then used to plot graphs, find integrals and derivatives.

**bool sinechk(char fn[],int i)**- This function checks if the user has entered sine function.

**double sineval(char fn[],int i,double x)**-Calculates sine(x).

**bool cosinechk(char fn[],int i)**- This function checks if the user has entered cosine function.

**double cosineval(char fn[],int i,double x)**-Calculates cosine(x).

**bool tangentchk(char fn[],int i)**- This function checks if the user has entered tangent function.

**double tangentval(char fn[],int i,double x)**-Calculates tangent(x).

**bool cosecantchk(char fn[],int i)**- This function checks if the user has entered cosecant function.

**double cosecantval(char fn[],int i,double x)**-Calculates cosecant(x).

**bool secantchk(char fn[],int i)**- This function checks if the user has entered secant function.

**double secantval(char fn[],int i,double x)**-Calculates secant(x).

**bool cotangentchk(char fn[],int i)- This function checks if the user has entered cotangent function.**

**double sineval(char fn[],int i,double x)-Calculates cotangent(x).**

**bool exponentialchk(char fn[],int i)- This function checks if the user has entered exponential function.**

**double exponential(char fn[],int i,double x)-Calculates exp(x).**

**bool powerchk(char fn[],int i)- This function checks if the user has entered power function.**

**double powerval(char fn[],int i,double x)-Calculates  $x^i$ .**

**bool logchk(char fn[],int i)- This function checks if the user has entered log function.**

**double logval(char fn[],int i,double x)-Calculates log(x).**

**bool modulochk(char fn[],int i)- This function checks if the user has entered modulus function.**

**double moduloval(char fn[],int i,double x)-Calculates modulus(x).**

**bool sqrtchk(char fn[],int i)- This function checks if the user has entered square root function.**

**double sqrtval(char fn[],int i,double x)-Calculates square root x.**

**bool constantchk(char fn[],int i)- This function checks if the user has entered a constant value.**

**double constantval(char fn[],int i,double x)-Finds the constant value.**

**bool signchk(char fn[],int i)- This function checks if there is any operator{+,-,\*,%}.**

**void composite(char fn[],int end,double x)-Computes the value of composite functions entered by the user, by using the above functions**

**double composite2(char fn[],double x)-This is used to compute the value of functions which are linear combinations/product of the basic or composite function. It uses the composite function.**

**bool validate(char fn[],int end,double x)-Checks if a point is in the domain of a basic function,(i.e doesn't make the function discontinuous).**

**bool validate1(char fn[],int end,double x)-Checks if a point is in the domain of a composite function.**

**bool validate2(char fn[],double x)-Checks if a point is in the domain of functions which are are linear combinations/product of the basic or composite function.**

**~~~~~::LIMITATIONS::~~~~~~**

**Our scope is restricted only to continuous(not piece wise continuous) and bounded functions (i.e trigonometric,exponential,logarithmic,polynomials,inverse and some composite functions).**

**Also the coefficients of the functions and the quadratic/cubic/bi-quadratic must be real .**

**Our integrator becomes ineffecient as the size of the intervals increases.**

**Our differentiator may not work perfectly for non-differentiable points in a function .**

**~~~~~::ACKNOWLEDGEMENTS::~~~~~~**

**We would like to thank our CLTA Binay Gupta for assisting us by giving us ideas for our codes and the libraries that we may use.We would like to acknowledge our respected professors, Dr.Deepak Phatak and Supratik Chakraborty who have taught us C++ and have given us an opportunity to make this project.We would also like to thank Juha Nieminen and Joel Yliluoma who have developed the function parser library.We would also like to express our gratitude towards Firuza Ma'am who helped us with the graphics part and also by providing us the projects made by previous year students,from which we gained inspiration and ideas.**