Maanasi Bulusu

Project Description:

My term project named “SolveIT” is an easy math solver. This software takes in any equation and can solve for its derivatives (first and the second derivative), integral, domain, range, roots, and asymptotes as well as graph the function, its derivatives, and integrals.

Competitive Analysis:

There are several projects similar to my term project in both that they are actual products with user interfaces and others that are just libraries. Firstly, several products such as “Symblolab”, “Derivative Calculator”, “Desmos”, and “Wolfram Alpha” are widely popular software’s that are available online for anyone’s use. These softwares run similar calculations to what my project “SolveIT” does such as solving for derivatives, integrals, and roots. However, the key difference between my project and these existing softwares is in the writing of the code. All of these products use some sort of library, mostly Sympy, to solve for the values that my project, SolveIT, solves for without the use of any libraries. Thus where most of the existing products are aimed to organize the solved for values from the SymPy library into one user-friendly product, my project uses mathematical concepts and ideas on the topics of derivatives, integrals, domain, range, roots, and asymptotes and I create my own code solving for these values. However, similar to these products, I do use the libraries Mathplotlib and NumPy to create a graph of the function, derivatives, and integral.

This algorithmic creation of my own code solving for these values is comparable to that the library, Sympy, does. However, as Sympy is a very complex library, my project works on solving certain values with my own algorithmic code and creates an easy to use user interface to do so. Thus, my project is distinguishable from the library Sympy in that it is not a library and therefore a user will not need to write any code to solve for the values they want rather can simply type in the equation to do so. Furthermore, my project also provides a 3D graph of not only the inputted function but also that of its derivatives and integral which SymPy does not do without several lines of codes and other libraries imported as well. However, as my code solving for these values is written by myself, it has some limitations/ exceptions that it will not run. Another feature that is unique to my product, SolveIT, is that it not only solves for the derivatives, integral, domain, range, roots, and asymptotes but it also allows for there to be an individualized use for any user in that they can input their name and store previously inputted equations that are accessible to them with just a mouse click.

Structural Plan:

My project will be organized into several different files, each solving a different function from the listed calculated values. For example, I will have a folder “Calculator” with the files: “Derivative”, “Integral”, ‘Roots”, “Domain”, “Range”, “Asymptotes”, and “Init”. Additionally, I will have a folder

“Users” in the bigger folder “Calculator” that will store all the previously written equations of each new user.

Algorithmic Plan:

The most complex aspect of my project is writing the function for derivatives. This is simply because of the fact that there are so many rules and organizations that need to be followed to create the function. This can then be used to create an integral function as it is just the opposite of this function. After this, the domain, roots, range, and asymptotes functions are relatively easy to create. The organization of the derivative function is as follows. The operators: “+.-.*,/, and **” are defined as symbols initially with their properties. After this, each function is separated by the operators “+” and “-” into a series each with an “element” of the function. This is then classified by name (ex: string), type (ex: string), base (element), coefficient (float), constant (float), and basic element (element). The classification, the base will always be what is surrounding the x. This is then called for its derivative which is multiplied to the derivative. The basic element is the surrounding types outside of the x. These outside elements and their derivatives are distinguished by the various element functions (constant (integer), basic (constant, can be x), power (ax^n), e^x, lnx, sinx, cosx, tanx, secx) , are further classified into the same element class to see whether they are the same and the derivative of those elements are printed. This value is taken back to the overall element and if the operator is not “*” or “/” is there, the value is taken back to the series derivative. If it is, the derivative UV method is called. This method separates the element by the operator and calls the product rule on u and v values. For division, I still use the product rule except I call the v as 1/v. The separated u and v then call the element class from above and add them back together. This derivative is then added back to the series derivative. This series derivative is then printed at the end.

Timeline:

4/14 - Derivative function is finalized

4/17 - Integral function is finalized

4/19 - All the functions, Domain, Range, Roots, and Asymptote function finalized

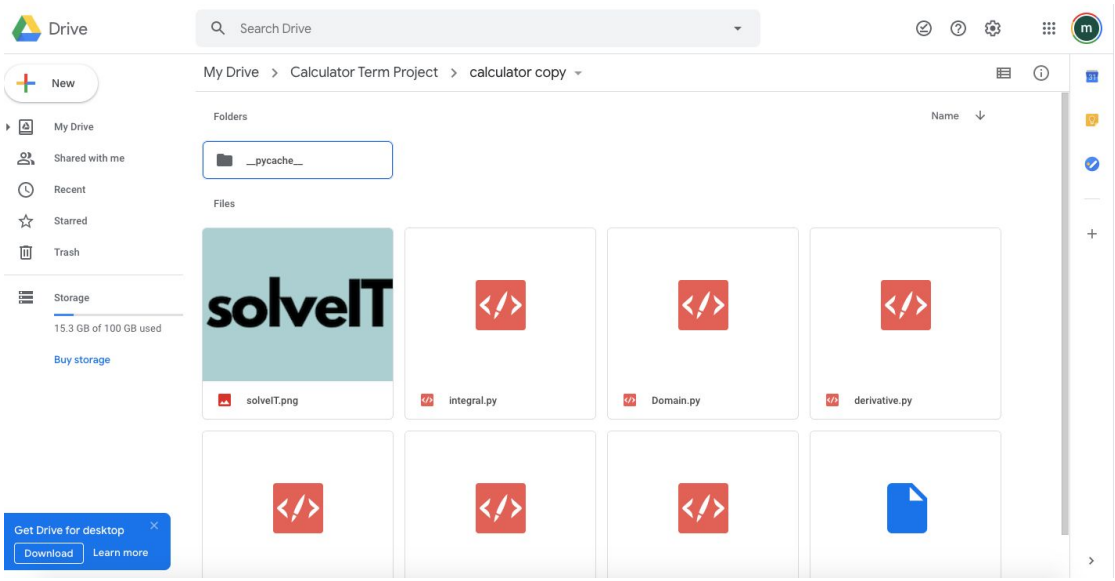
4/23 - Graphics finished for the most part

4/25 - All elements are put together and tested for function

4/28 - Video, testing, Read Me file, and all extras are created and tested

Version Control Plan:

I will upload all files every time I edit them onto google drive folder.



Module List:

- MathPlotLib
- NumPy

TP2 Update:

- I took out the graph IT option for the derivative and integral section of the display and instead elongated that box to show the derivative located in the solver screen
- I took out the calculate again button in the derivative section and instead elongated the Previous Derivates box, replacing the words “Previous Derivatives” with the number of the derivative
- I switched the order of the Enter Equation and Enter Name input boxes in the home screen
- I took out the clear Previous equations button on the previous equations display box to show instead 10 the previous equations and come down one when there are more than 10 equations

TP3 Update:

- The simplify function and keyboard scroll function has been added to every input box in the program