**Intro**

For the project, we’ll be doing a graphing calculator primarily in Javascript, some CSS, and some HTML Our group has majors in computer science, math, and computer engineering. Javascript experience includes everything covered in Programming Language Paradigms.

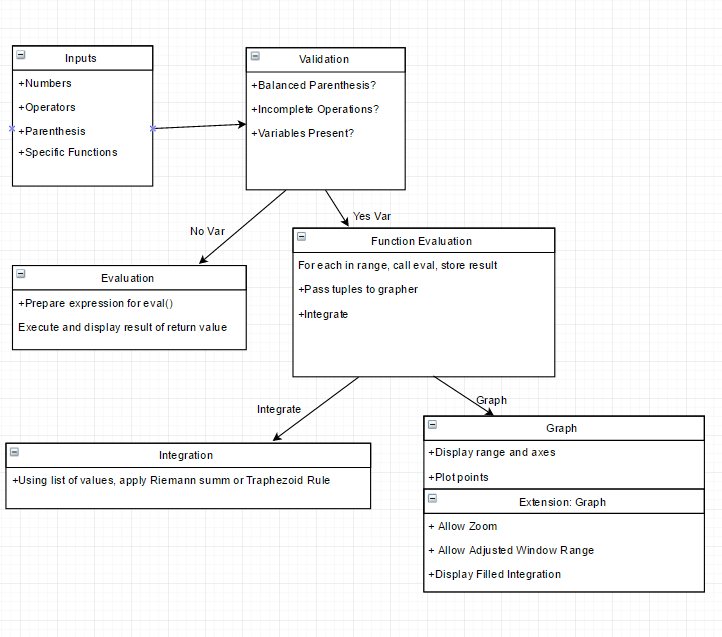
**Details**

We are trying to create a javascript-based web application that takes in user input and performs graphing calculator operations and integrations. Currently, graphing calculators of today are physical and cost hundreds of dollars to obtain. Our application seeks to change those limitations by hosting it in a web-based application as opposed to an embedded system and allowing it to be free to use as often as necessary, provided the user has an internet connection.

Our calculator should perform graphing of common functions, which is more than the basic calculator installed in most devices have, and will be available freely on the web. These basic ideas and functions will make it stand out for people who can’t afford graphing calculators and those who forget their graphing calculators and would enjoy a web-based application available for use anywhere. Like so many other web applications, any limitations are offset by convenience and cost. However, there are risks which may undermine the utility of our app. If the graphing calculator does not perform as planned, this could cause great problems for the user and could also lead to poor reputation for the developers. Furthermore, if it is difficult or counterintuitive in its design, no one will use it and it will not be worth the cost to host it. That said, if it performs as planned, the application could reduce costs and burden among its demographic in college students. If necessary, ads can also be placed on the page which will allow the calculator to pay for the cost of hosting it.

The calculator is planned to only have website hosting costs, which would be approximated to be around $10 per month, if free servers are not used. For this project, the domain and servers are expected to be hosted on a prepaid domain and server.

The project is estimated to take around 4 weeks. The midterm check will occur at 2 weeks in and will check if individual components work with basic functions. The final check will test if the entire project works together. The development process will roughly follow the tentative schedule included at the bottom.



**Division of Labor**

Jackson J Schilmoeller -

I’ll be writing the integral calculator function for our program, which will be implemented using Riemann sums - specifically the trapezoid rule. Advanced and additional functionalities could include: implementation of other methods of approximation of integrals for comparison - left, middle, right sums; piece-wise integration; and, perhaps creation of objects to display on the graphing canvas that illustrate the Riemann sums.

Michael Neises -

I will be in charge of the mapping function, which takes in as parameters a function string as inputted by the user, a lower domain bound, and an upper domain bound. It will then output an array of point objects, each of which holds X and Y coordinates as properties. I will also be in charge of parsing the input string and handling arguments that cannot be directly evaluated by the eval() function, such as powers, exponentials, and sinusoidal functions. The advanced functionality will be to create points so that the max distance between points is controlled, which will increase visual fidelity.

Stephen Longofono -

I will be writing the code to manage, update, and adjust the display of the graphed function. We are planning to use a canvas object to display the tuples passed in, with the x and y coordinates mapped to the appropriate spot on the canvas. The basic functionality will also include axes marked at regular intervals. Improvement ideas include adding the ability to adjust the displayed range, zooming, or three predefined views at low, medium, and high resolution.

Cammy Vo -

I’ll be doing the display which shows the calculator, the input, and the graph, and takes in input to pass into the mapping function for parsing and evaluation. The input should only be in the form of buttons and allow numbers, variables, operators, and parentheses. The input should include basic arithmetic operators, numbers 0-9, an integral button, evaluation button, parenthesis, and exponential function.

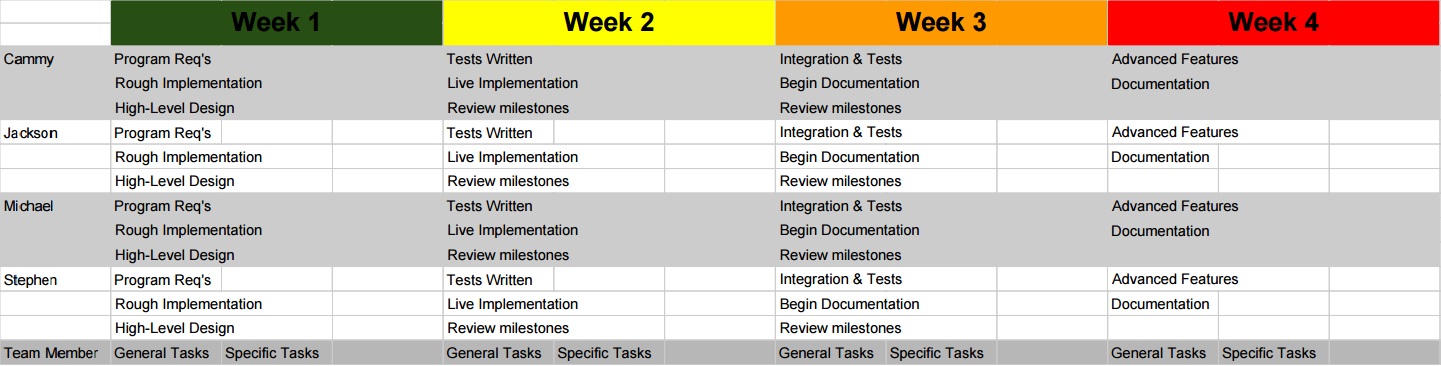
**Timeline**

Week 1: Design program, decide on requirements, begin to distribute workloads, define milestones.

Week 2: Individual functions meet baseline requirements. Midterm Checkup - revisit milestones, verify minimum requirements met.

Week 3: Integration & Testing: Program works altogether, adjustments and helpers as needed. Revisit milestones if necessary, begin documentation.

Week 4: Advanced individual functionality added, additional refinements. Complete documentation.



**Github https://github.com/SLongofono/EECS-368-Javascript**