Report on the Development of the Eternity Project

Iteration I

Submitted for COMP 354
June 6th 2019

By:
Mathieu Lajoie (40026331)
Olivier Hébert (40051654)
Jason Brennan (27793928)
Charles-Antoine Guité (40063098)
Manel Guay-Montserrat (29692304)

Table of Contents

Introduction

The goal of the project is to create a scientific calculator with transcendental functions built from scratch. This involves avoiding the use of math libraries that are available with essentially every single programming language. Writing these proved to be more complicated than it first appeared. Digging through mathematical theorems and lemmas was needed on more than one occasion. Most of the functions can be represented in the form of an infinite series, a rule which we made use of in our codebase.

How to Run the Application

Our calculator is a simple web application, accessible on any evergreen web browser. It is fully responsive for use on desktops, tablets and mobile devices. To run it, simply visit https://mattl75.github.io/eternity/ on your browser.

In order to run the application locally, follow the next few steps. A guide is also available in the readme markdown file of the source code.

1. Install Node.js and npm.
2. Install the Angular CLI by running *npm install -g @angular/cli*.
3. Navigate to the repository and enter the eternity directory.
4. Run *npm install*.
5. Run *ng serve* or *npm start*.
6. Visit *localhost:4200* in your browser.

Roles and Responsibilities

Team Leader: Mathieu was elected as team lead. His expertise in web development provided the rest of the team with a clear path to follow in developing the application.

Note Taker: Jason acted as note taker, recording the minutes of all team interactions.

Mathieu set up a GitHub repository for our project, and provided the initial framework and GUI for our application.

Each team member was assigned one function to implement in javascript as per the project requirements.

Each team member was to perform one interview, thereby creating a persona.

Mathieu and Olivier were assigned the task of coding the web application.

Jason, Charles-Antoine and Manel were assigned the task of writing the report.

The presentation slides were prepared by Mathieu.

Mathieu was assigned the role of lead presenter, with Manel as secondary presenter.

Overall, each team member share responsibility to contribute to the code, reviewing peer work and advancing work on the project.

Collaboration Patterns Followed

During iteration I of the development of the Eternity calculator, several collaboration patterns were followed. Here is a summarized view of the patterns we applied and how they helped us.

- **Clear up questions**: Some parts of the assignment description were vague and led several team members to have different interpretations of the requirements. During our initial meeting, we made sure we were on the same page. Afterwards, any remaining questions were asked to the professor in class.
- **Share expectations**: Because we are a group of students working together for the first time, we discussed our goals for the project and came to a group decision to prioritize project requirements (such as a detailed report) while having fun (making a GUI for the calculator).
- **Fill knowledge gaps**: When we were deciding on the technology for the project, it became obvious that each team member had a different skill set. This was acknowledged and once we chose to write our functions in TypeScript, experts on the team laid the foundation for the others to learn from. Now, the entire team can comfortably participate in the codebase.
- **Centralize work product management**: It was obvious to the entire team that we needed a centralized codebase, and we chose github for the project's version control system as we all had experience with it.
- **Manage the project**: To complete iteration I, we needed to constantly consider priorities and dependencies for ideal decision-making.
- **Start immediately**: As soon as we had the team formed, Mathieu was excited to make a calculator with Angular and quickly built up a prototype to get the project started.
- **Regularly check requirements fulfillment**: During iteration I, it was useful to have check-ins to make sure that work was progressing and that we were on track for the submission deadline.
- **Spread tasks appropriately**: This was made easier due to the fact that each team member must contribute his own function to the project, but we also gave specific roles to members to assure overall project progress.

Architecture and Technology

Although we were allowed by the professor to simply create a text-based application, we wanted something more practical. Something that would be easy to deploy, and would feature a very good user experience. Some of us also had extensive web knowledge, so we decided to go in that direction. We built our platform using Angular, so that meant making use of mainly HTML, CSS and TypeScript. We chose Angular because it is arguably the best frontend JavaScript framework available. Google is adopting it in its entire platform, plus the Angular team will soon release a new renderer (dubbed *Ivy)* which should significantly improve performance. Angular is also the best tool to develop Web Components, through its Angular Elements API. Web Components are on track to become the future of the web world and present a clear advantage to any framework that can exploit its strengths.

Initially, we wanted to have a simple frontend which would send equations to a Flask backend and then receive a response. However, after speaking with the professor, we were allowed to make use of TypeScript to do everything, even the mathematical processing. Having only a static frontend made it very easy to deploy our application. GitHub has a service called GitHub Pages where developers can host static content for free.

In terms of problems faced during the architecture process, we had quite a few. Normally, processing large numbers in TypeScript or JavaScript would be a significant problem. In fact, the largest integral value is 2^{53}-1, which was not sufficient for some of our algorithms which reach into truly big numbers. To resolve this issue, we made use of an open source library called bignumber.js. This allowed us to complete operations on much larger numbers, and thus reach a higher degree of precision. Furthermore, we made use of math.js to parse the mathematical expressions, in accordance to what the professor allowed during the lectures.

Our user interface loosely follows modern Material Design guidelines, which are developed at Google. We selected this design system for its maturity and ease of understanding the concepts. There is also a vast open source community surrounding this project, which makes finding answers to questions very simple and straightforward. We developed the web application with mobile in mind, and made it responsive for all devices. Users also do not need to use the mobile keyboard if they prefer, as our own built-in calculator offers buttons that are self-sufficient.

Accessibility is another large challenge that we decided to take on. Some of our use cases required high-contrast themes, or ease of use with keyboard, and even screen readers. While this journey is not yet complete as we have a lot of features left to implement, we do have a custom theming engine and the application has been tested with aXe, which is an automatic accessibility auditing tool.

Application performance is very important to us, especially if users are going to be looking for high degrees of precision which will require many rounds of the infinite series. To that end, we decided to implement web workers, eventually. This will allow users to keep interacting with the page even while a calculation takes place. It is not currently production ready, so it has been omitted from iteration one, but will be a part of iteration two. We also regularly test our application with Google Lighthouse, which is an automatic performance auditor tool. It helps us keep our load times as low as possible.

We centralized our code in a private git repository hosted on Github. It is currently private to prevent plagiarism from other teams. Version control allowed us to work concurrently on many features and separate these features into branches, which need to be audited by 2 people in order to be pulled into the main branch, enforcing code review. We also make use of the “Issues” section with a kanban board of sorts, as each issue can be assigned custom tags and people to work on them. We can communicate directly on github for specific code review comments and questions as well as information about issues, however for general project discussion we use a free group communication software called Discord. It was initially developed for gaming communities, but it can serve team projects and other communities as well. We can share links and upload small files, as well have group voice calls if we cannot have in-person meetings.

Interview Questions

We have chosen an initial list of 15 questions to ask during our 5 interviews.

1) What is your name?
2) What is your age?
3) What is your profession (concentration if a student)?
4) Do you own or regularly use a desktop or laptop computer? What is the OS?
5) Do you own or regularly use a smartphone? What is the OS?
6) On a scale of 1-10 with 10 being the highest, how would you rate your knowledge of computer software?
7) On a scale of 1-10 with 10 being the highest, how easy do you find it to use desktop computer software? Mobile software?
8) Could you describe one of your favorite applications to use? What features if any contribute to this application being your favorite?
9) Are you aware of the following terms: localization, accessibility, and themes as they pertain to software? Could you describe what these terms mean to you?
10) When you want to perform a calculation, where is the first place that you look (ex: handheld calculator, phone, desktop app, google)
11) Do you find calculator apps easy to use? Do you expect them to be easy to use?
12) What is a feature that you'd expect to find in a calculator app?
13) What is a feature that you don't normally find in a calculator app, but you feel would be nice to have?
14) What are some expected behaviors when using a calculator app?
15) Can you describe an experience using a calculator app where you could not get your desired result (either from missing functionality, or lack of understanding about how to use the app)?

These questions numbers will be used as reference for the answer sheets to those interviews in Appendix A.

The answers collected during the interviews helped us create our personas used for the project. The main points that stood out during the interviews were:

- Basic functions are really what people need
- Memory and clear features are essential for an enjoyable experience
- Accuracy of results is crucial
- Order of operations must be respected, chaining operations functional
- Intuitive design and use

Personas

Initial Personas

Personas	**Use cases**	**Implementation**
Student	Use in dark environment, tooltips, Use on multiple screen sizes	Theming engine, responsive design, tooltips
French Person	Use in french (tooltips, FAQ)	Localization
Mechanical Disabilities	Cannot use mouse and/or keyboard	Keyboard support, screenreaders (accessible application)
Visual Disabilities	Color blindness, blindness	Theming engine, screenreaders (accessible application)
Researchers	Use on multiple screen sizes, complex functions guide, more than just complex, parsing math	Responsive design, information system, parse the formulas

Detailed Personas

Persona name: Julie Trang

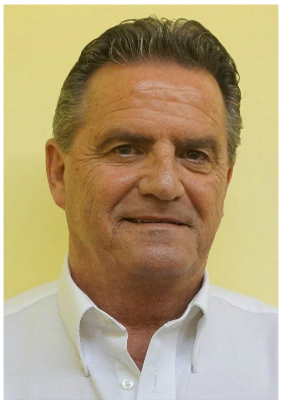
Specifics: Julie is a 21 years old mechanical engineering student. She uses her laptop and her cellphone everyday and is in general very comfortable with any software presented to her. Her degree makes it necessary for her to use a calculator very often, so she knows exactly what she wants as feature for this type of software. For her, at the most basic level, a calculator has to be precise and the software supporting it has to be responsive and be able to be used rapidly. One of her preferences is a dark mode theme that is easy on the eyes during the night when she studies.

[1]

Persona name: Nathalie Tremblay

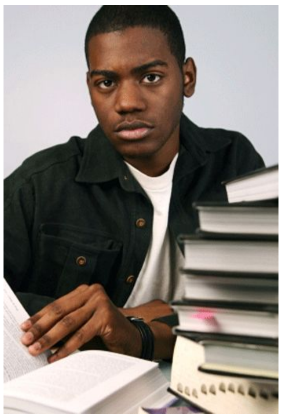
Specifics: Nathalie is a 37 year old high-school history teacher. She is decently comfortable with the software of her cellphone and her desktop computer, but is still not as good with it as the younger generations. Nathalie is a French speaker and does not speak great English. Surfing the web is sometimes frustrating for her, since a lot of websites are solely in English. When deciding to use a particular software, French support is the primary aspect she looks into, even if in this case the translation for a calculator are limited. After that, other basic features deemed necessary are like the other personas, such as precision and ease of use.
[2]

Persona name: Greg Walker



Specifics: Greg is a 55 year old factory worker. He is not very familiar with the various facets of software but he owns both a laptop and a mobile phone, which he uses to perform limited tasks, like sending emails and using Google maps. Greg prefers to use the mouse, but his severe arthritis sometimes limits him to only use the keyboard. In the context of his work, Greg rarely needs to use a calculator but he does use one on the computer often at home. For him, the most important things that any software needs, including a calculator, is to be easy to use and to be able to support both keyboard and mouse if possible.
[3]

Persona name: Tyrice James



Specifics: Tyrice is a 23 year old musician. Like most of his age, he is comfortable with technology and owns a laptop and an Android cellphone. Tyrice has a form of tritanopia, which is a colour blindness that affects the way he sees shades of blue and yellow. He is an avid fan of video games, and this deficiency sometimes affects him in understanding menus and color codes. He appreciates when developers of various apps add different colour themes that he can choose from to limit the effect of his colour blindness.

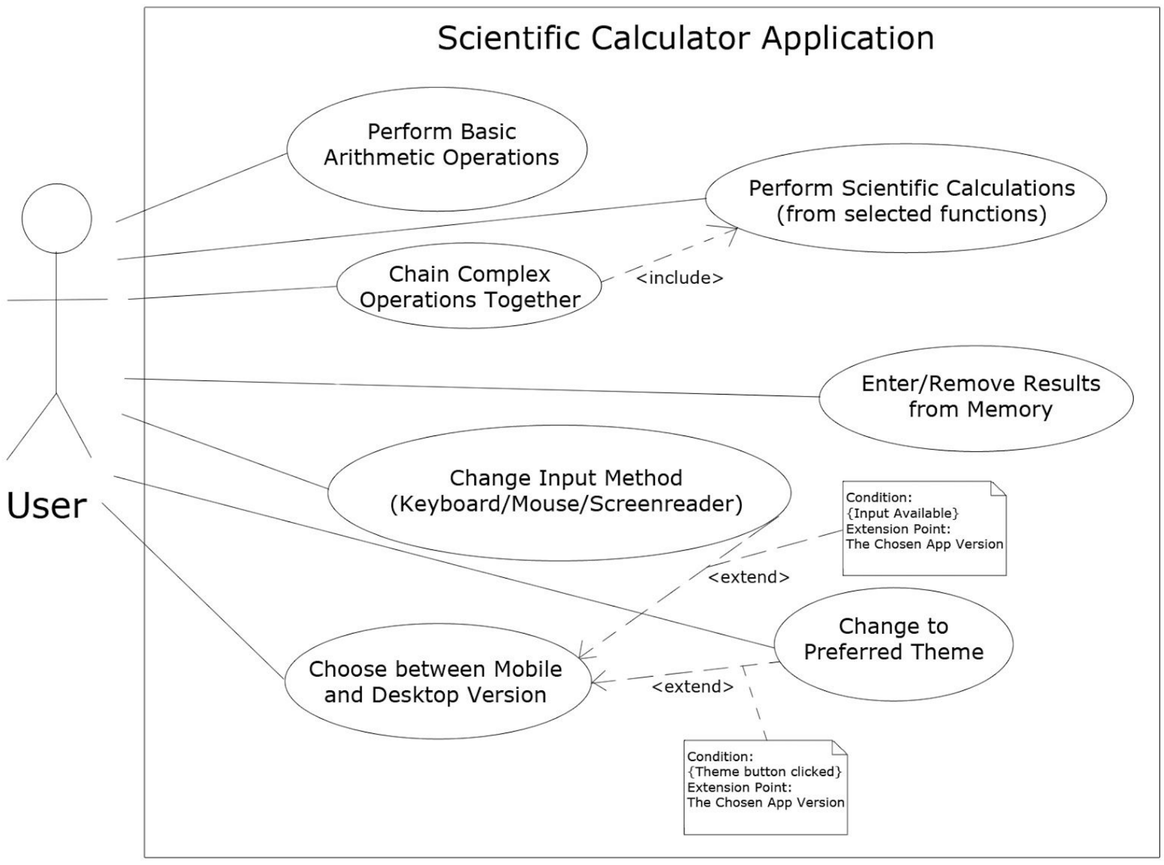
[4]

Persona name: Rishu Raj

Specifics: Rishu is a 44 year old PhD researcher in his university's department of physics. In the context of his work, he has access and has used complex technology and software. Any calculator that Rishu uses seriously, needs to have at its disposition a scientific mode that let him parse extensive formulas and functions. Multiple quality of life features are also necessary for him, since he will only use the best software that is available to him. Things like support of multiple screen size, history of equations, graphs of functions, etc.

[5]

Use Cases


Scientific Calculator Application
User
Perform Basic Arithmetic Operations
Perform Scientific Calculations (from selected functions)
Chain Complex Operations Together
<include>
Enter/Remove Results from Memory
Change Input Method (Keyboard/Mouse/Screenreader)
Condition: {Input Available} Extension Point: The Chosen App Version
<extend>
Change to Preferred Theme
Choose between Mobile and Desktop Version
<extend>
Condition: {Theme button clicked} Extension Point: The Chosen App Version

Implementation of Use Cases

1. Perform Basic Arithmetic
   The basic operations present on any standard calculator are available as buttons in the application.
2. Perform Scientific Calculations (from Selected Functions)
   The transcendental functions selected to be the main purpose of the calculator are present as buttons in the application.
3. Chain Complex Operations Together
   This was not in the scope of iteration I.
4. Enter/Remove Results from Memory
   This was not in the scope of iteration I.
5. Change Input Method (Keyboard/Mouse/Screenreader)
   On desktop, both keyboard and mouse work, as well as accessible screen reader Options. On mobile, the buttons all work with touch.
6. Choose between Mobile and Desktop Version
   The application is functional and fully responsive on desktop and mobile.
7. Change to Preferred Theme
   A theme button allows the user to select from many different themes.

Selected Functions

e^x

To implement this transcendental function, we have chosen to describe it as a Taylor Series.

$e^x = 1 + x + x^2/2! + x^3/3! + x^4/4! \ldots$

To implement this series, two sub-functions are necessary: a power function that only uses integers as powers and a recursive factorial function

```
**FUNCTION** ePowerX(val, rounds)
finalResult ← 1 + val
**for** i =0 **to** i < rounds **do**
        finalResult = powerWithInt(val, i + 2) / factorial(i + 2)

**return** Final_Result
**END**
```

Limitations:
After testing the function, we realized that the Max_Iteration value was limited by the memory that a BigNumber could store . A limited Max_Iterations means that above a certain threshold, the accuracy of the final result starts to drop, which limits the domain of the x value. It is possible we could find an improvement to that problem in Iteration 2.

--

$\sqrt{x}$

We chose to use the Babylonian method for approximating the square root [6]. The method uses an initial guess for $\sqrt{x}$ and then recursively iterates through the algorithm, using the obtained result as the next guess until the difference between iteration n and n-1 is within some tolerance.

The algorithm is described as follows:

```
epsilon = 0.00001 (or some other suitably small number)
FUNCTION sqrt(val, guess)
        x1 ← (guess  + (val / guess )) / 2
        if (x1 – guess ) < epsilon then
                return x1
        else
                return sqrt(val, x1)
```

END

Despite it’s recursive nature the method is quite fast; able to calculate the square root of small numbers in 5-10 iterations, while larger numbers (in the hundreds of thousands) require around 10-20 iterations.

Limitations: As the size of the input increases, so do the number of required iterations. Since this is a recursive function, care must be taken to avoid stack overflow. Although this is unlikely with today’s hardware and the expected function input, it is nonetheless something to be considered. We will explore less memory intensive methods of calculating $\sqrt{x}$ for iteration 2.

--

Sin(x) and Cos(x)

Considering their similarity, we decided to implement both sin(x) and cos(x) functions. Both can be derived from relatively simple series based on the Taylor Theorem [Guichard].

$$sin(x) = \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k+1)!} x^{2k+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - ...$$

The above series can be approximated using the following pseudocode.

```
FUNCTION sin(val, rounds)
    val ← val * (PI / 180)
    retained ← 0
    denominator← null
    numerator ← null
    multiplier ← null

    for i = 0 to i < rounds do
        numerator ← power(-1, i)
        denominator ← factorial(2 * i + 1)
        multiplier ← power(val, 2 * i + 1)
        retained ← retained + (numerator / denominator) * multiplier

    return retained
END
```

$$cos(x) = \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k)!} x^{2k} = x - \frac{x^2}{2!} + \frac{x^4}{4!} - ...$$

The above series can be approximated using the following pseudocode.

```
FUNCTION cos(val, rounds)
    val ← val * (PI / 180)
    retained ← 0
    denominator← null
    numerator ← null
    multiplier ← null

    for i = 0 to i < rounds do
        numerator ← power(-1, i)
        denominator ← factorial(2 * i + 1)
        multiplier ← power(val, 2 * i + 1)
        retained ← retained + (numerator / denominator) * multiplier

    return retained
END
```

---

x^y

As both numbers can be real, we cannot simply multiply value x for y number of times. An equivalent expression to x^y, using logarithms, is $e^{y*ln(x)}$ **.** We chose this due to e^x already being implemented. Since exponents follow $x^y = x^1 * x^{y-1}$, we can extract the fractional part of the exponent by subtracting it and multiplying both the results together, simplifying the operation. We define $ln(x)$ in the Subordinate Functions.

```
FUNCTION power(x, y):
    intPart ← floor(abs(y))
    floatPart = abs(y) - intPart
    initialResult ←  ePowerX(floatPart * ln(x)) * power(x, intPart)
    If y > 0 then
        return initialResult
    Else
        return 1 / initialResult
END
```

10^x

Similarly to x^y , 10^x uses the formula $e^{x*ln(10)}$ to find its result.[7] Since the base is always 10, it was possible to extract a constant value instead of always making extra calculations by calling *ln(x)*. For optimal precision, the formula was separated into 2 parts, one for the integer part of the exponent and one for the remainder.

```
FUNCTION tenPowerX(val)
    intVal ← floor(val)
    remain ← val - intVal
    retained ← 0

    if (intVal < 0) then
        retained ← 1 / power(10, intVal) * ePowerX(remain * LN10)
    else
        retained ← power(10, intVal) * ePowerX(remain * LN10)

    return retained
END
```

Subordinate Functions

Power with integer function

```
FUNCTION powerWithInt(val, power)
    finalResult ← 1
    for i =0 to i < power do
        finalResult = finalResult * val ;

    return finalResult ;
END
```

Factorial function

```
FUNCTION factorial(val)
    If val = 0 then
        return 1
    else
        return val * factorial(val - 1)
END
```

----

ln(x) Function

This function relies on a logarithmic series [8] described as follows:

$$ln(x) \; = \; 2[\; \frac{x-1}{x+1} \; + \; \frac{1}{3}(\frac{x-1}{x+1})^{\,3} + \; \frac{1}{5}(\frac{x-1}{x+1})^{\,5} + \; ...] \; : \; (x > 0)$$

As a default, we calculate this series until the fraction reaches $1/400$, although this can be changed to increase either precision or efficiency. In pseudocode, this was implemented as

```
FUNCTION ln(x, rounds = 400):
    Sum ← 0.0
    for i = 1 to rounds do
        If i is odd then:
            Sum ← sum + (1 / i) * fraction(x, i)
    Return sum
END
FUNCTION fraction(x, i):
    Accumulator ←  ((x - 1.0) / (x + 1.0))
    For j = 1 to i do
        Accumulator ← accumulator *  ((x - 1.0) / (x + 1.0))
    Return Accumulator
END
```

This algorithm has a few shortcomings. Since it is based on approximations, the result cannot be guaranteed a precision. It also does not account for negative base numbers, as the natural log of negative numbers are undefined. By extension, it doesn't address imaginary parts of numbers either, which can occur when dealing with exponentiation of negative real numbers.

Floor Function

```
FUNCTION floor(val)
    i ← 1
    prev ← 1
    result ← 0
    prevResult ← 0
    if (val > 0) then
        while (val > result) do
            if (val >= prev) then
                result ← prev
                prev ← prev + i
                i ← i * 2
            else
                if (prevResult = result) then
                    return result
                prevResult ← result
                prev ← result
                i ← 1
    else if (val < 0) then
        prev ← -1
        while (val < result) do
            if (val <= prev) then
                result ← prev
                prev ← prev - i
                i ← i * 2
            else
                if (prevResult = result) then
                    return result
                prevResult ← result
                prev ← result
                i ← 1
    return result
END
```

--

Abs Function

```
FUNCTION abs(val)
    if val < 0 then
        return -val
    else
        return val
END
```

Process

To complete this project, we decided as a team to take an agile approach. Considering that the project requires two iterations of the product, this seemed natural. We had regular meetings and check-ins and we always kept each other up to date to have a good idea of the progress we were doing. Using GitHub's features, we set up a kanban board, tracked issues for the work to be done, and implemented a rigorous review system. Thanks to the process, we were able to produce an early prototype and continuously improve our product.

Inclusions

- 6 selected functions from the project description
- Basic operators
- Degree/Radian modes
- Customizable Graphical User Interface
- Mobile responsiveness
- Keyboard and mouse functionality
- Screen reader support
- 4 different themes
- Localization

Exclusions

- Accessibility Testing (aXe)
- Automated Testing
- Performance Testing (Lighthouse)
- Function chaining
- Result saving in memory
- Web workers (basically web multithreading, page doesn't freeze during long calculations, ability to display a loading spinner, etc)
- Web storage (Store which theme the user has set)
- Better keyboard support & layout
- Quality of life fixes

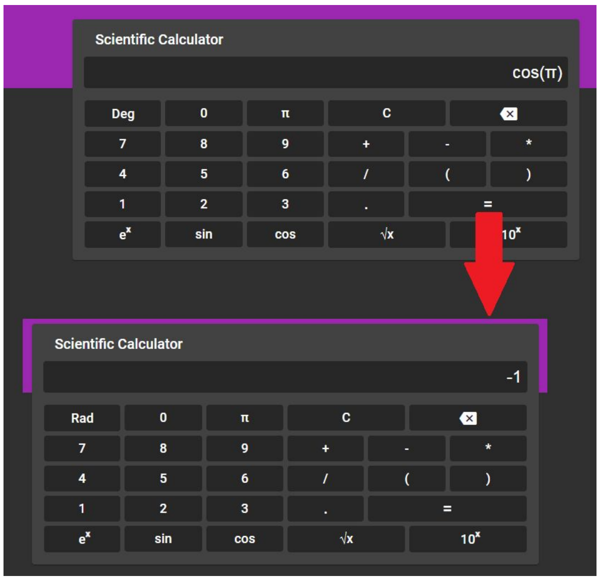
Testing Implementations and Results

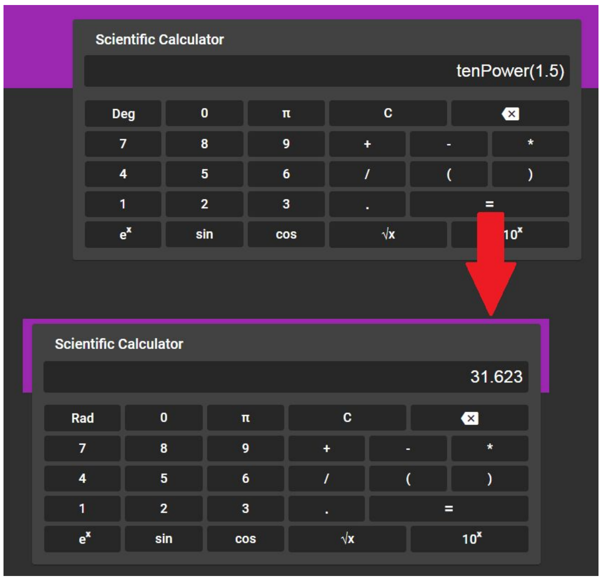
In iteration I, testing was performed manually in two ways: through debugging and within the calculator application itself. Furthermore, each function provides an output for the following two cases:

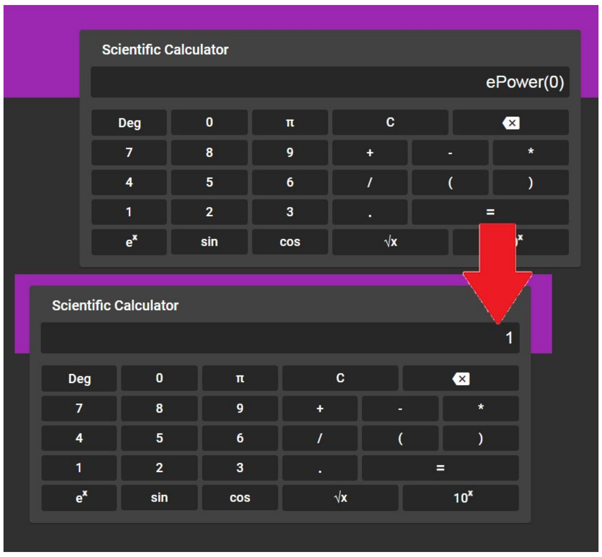
1. Input is 0
2. Input is transcendental

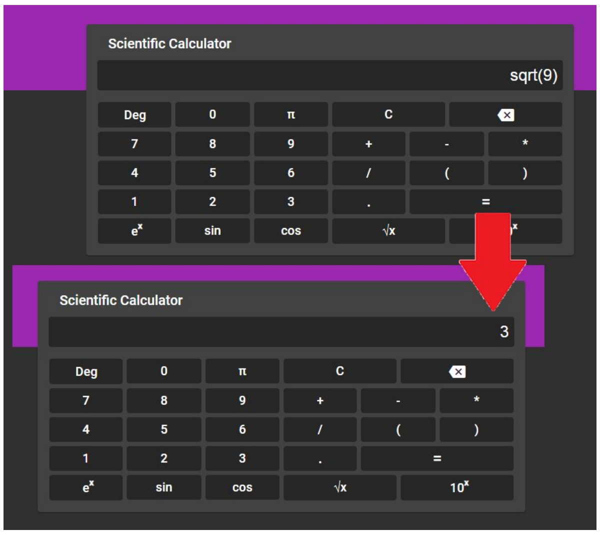
Before making new changes and optimizing in iteration II, it will be important to setup some automated tests to ensure functionality does not break.

Examples:


Scientific Calculator
cos(π)
Scientific Calculator
-1

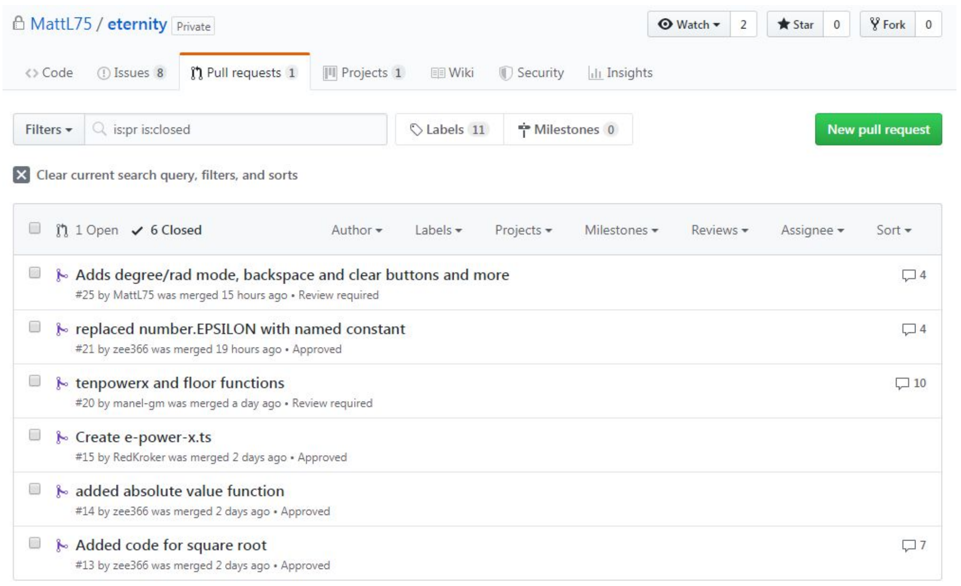

Scientific Calculator
tenPower(1.5)
Scientific Calculator
31.623


Scientific Calculator
ePower(0)
Scientific Calculator
1


Scientific Calculator
sqrt(9)
Scientific Calculator
3

Code Review

All code integrated in the application must follow an adequate workflow. Work is done on separate branches before being merged into the master branch through pull requests.



Adds degree/rad mode, backspace and clear buttons and more
#25 by MattL75 was merged 15 hours ago • Review required — 4

replaced number.EPSILON with named constant
#21 by zee366 was merged 19 hours ago • Approved — 4

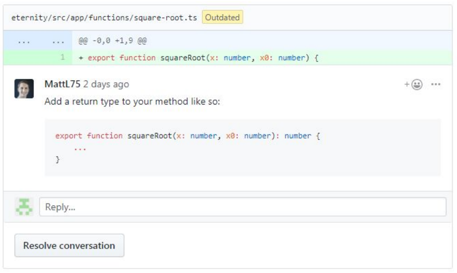
tenpowerx and floor functions
#20 by manel-gm was merged a day ago • Review required — 10

Create e-power-x.ts
#15 by RedKroker was merged 2 days ago • Approved

added absolute value function
#14 by zee366 was merged 2 days ago • Approved

Added code for square root
#13 by zee366 was merged 2 days ago • Approved — 7

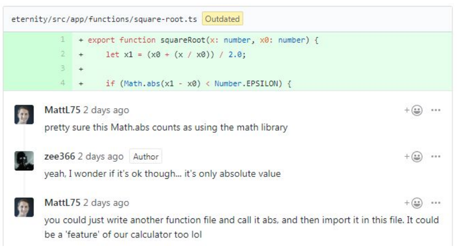
Furthermore, there is an asymmetric code review process in place where at least 2 team members must approve the new code before it is merged. Here are examples of this process in action:


eternity/src/app/functions/square-root.ts Outdated

```
@@ -0,0 +1,9 @@
+ export function squareRoot(x: number, x0: number) {
```

MattL75 2 days ago
Add a return type to your method like so:

```
export function squareRoot(x: number, x0: number): number {
    ...
}
```

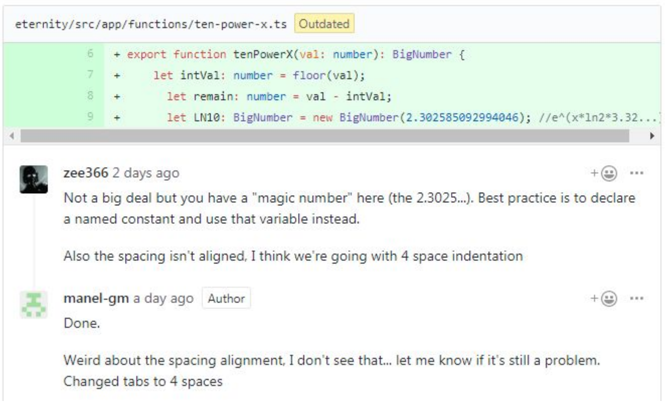

eternity/src/app/functions/square-root.ts Outdated

```
+ export function squareRoot(x: number, x0: number) {
+     let x1 = (x0 + (x / x0)) / 2.0;
+
+     if (Math.abs(x1 - x0) < Number.EPSILON) {
```

MattL75 2 days ago
pretty sure this Math.abs counts as using the math library

zee366 2 days ago Author
yeah, I wonder if it's ok though... it's only absolute value

MattL75 2 days ago
you could just write another function file and call it abs, and then import it in this file. It could be a 'feature' of our calculator too lol


eternity/src/app/functions/ten-power-x.ts Outdated

```
+ export function tenPowerX(val: number): BigNumber {
+     let intVal: number = floor(val);
+       let remain: number = val - intVal;
+       let LN10: BigNumber = new BigNumber(2.302585092994046); //e^(x*ln2*3.32...
```

zee366 2 days ago
Not a big deal but you have a "magic number" here (the 2.3025...). Best practice is to declare a named constant and use that variable instead.

Also the spacing isn't aligned, I think we're going with 4 space indentation

manel-gm a day ago Author
Done.

Weird about the spacing alignment, I don't see that... let me know if it's still a problem.
Changed tabs to 4 spaces

Glossary

Agile - Agile software development is an approach to software development under which requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customer(s)/end user(s).

Angular - A platform for developing mobile and desktop web applications.

Automated Testing - Process to validate that software functions appropriately and meets requirements before it is released into production, using scripts that are executed by testing tools.

aXe - Automatic accessibility testing Chrome extension.

Evergreen - The last version of the browsers that automatically update themselves. This includes Safari >= 10, Chrome >= 55 including Opera, Edge >= 13.

CI - Continuous Integration. A practice where developers integrate their code into a shared repository at a high frequency. Code is verified by an automated build which allows for quick error detection.

Kanban Board - Agile project management tool designed to help visualize work, limit work-in-progress, and maximize efficiency.

Lighthouse - Automatic web application auditing tool made by Google.

Localization - Providing different translations of the application.

Material Design - Design system developed at Google.

Node.js - Open source JavaScript runtime.

Repository - A central place in which an aggregation of data is kept and maintained in an organized way.

Responsive Design - Making an application function as expected on multiple devices of different screen sizes.

TypeScript - An open-source programming language. Typescript is a superset of JavaScript.

Web Components - A set of guidelines and conventions which allow components to be read the same way by all browsers and all frameworks.

References

[1] Freepik, Portrait of smiling asian female student with book Photo | Free Download. Available: https://www.freepik.com/free-photo/portrait-smiling-asian-female-student-with-book_1148015.htm

[2] Harrow School, “Teacher Profiles”. Available: https://www.harrowschoolshortcourses.co.uk/online-tuition/teacher-profiles/

[3] H. Carson, “Totally Random”. Available: https://www.yelp.com/biz/totally-random-myrtle-beach

[4] Getty Images, “Student Studying”. Available: https://www.istockphoto.com/ca/photo/student-studying-gm182145723-1122633

[5] University of Michigan Health System, “Team finds new genetic anomalies in lung cancer”. Available: https://medicalxpress.com/news/2014-12-team-genetic-anomalies-lung-cancer.html

[6] D. Fowler and E. Robson, "Square Root Approximations in Old Babylonian Mathematics: YBC 7289 in Context", Historia Mathematica, vol. 25, no. 4, pp. 366-378, 1998. Available: https://www.sciencedirect.com/science/article/pii/S0315086098922091. [Accessed 3 June 2019].

[7] D. Guichard, “Single Variable Calculus”, p.85, https://www.whitman.edu/mathematics/calculus/calculus_04_Transcendental_Functions.pdf

[8] R. Nave, "Natural Logarithm Series", Hyperphysics.phy-astr.gsu.edu. [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/Math/lnseries.html. [Accessed: 05- Jun- 2019].

Appendix A: Interviews

1st Interview

1) Benoit Boisvert
2) 53 years old
3) I am working for a fiberglass oil tank company as an installer and repairman
4) Of course! I mostly use a laptop that runs Windows 7.
5) Yes that too, I own a samsung phone that runs Android.
6) Oh not that high, probably a 5 or a 6 (*laugh)*
7) I find mobile software to be generally easier to use, or at least the applications that I use on it. So I would probably give mobile an 8 while desktop would be more around 7
8) Mmm, I think I like mobile banking apps, like Acces D or the Tangerine one. The menus are usually very clear and easy to follow.
9) I know what localization means, but in terms of software, I can't say I know what the link is. Accessibility would be to me how easy to access and use the software is for different users. And themes are the colors of the software I think ? Yeah I think that's it.
10) If I'm on my laptop, I usually use the windows calculator of the computer and the one on my cellphone in other cases.
11) Oh very easy, they are pretty straightforward aren't they ? (*Laugh)*. Of course I would expect them to be easy to use, with how simple and widely used they are.
12) A scientific mode, or at least a part of it. I hate when a calculator only has like, the 4 basic operations and nothing else. I usually switch to a different one if that's the case.
13) Mmm, On top of my head I can't really think of anything… Maybe a way to remember multiple old answers ? You don't see that often
14) The result should be exact, I hope *(laugh).* Anything else is just a bonus.
15) Actually I do have one. I have pretty bad arthritis in my hands and wrists and it sometimes limits what I can do on a computer. I usually prefer to use the mouse to enter the number in my laptop calculator, but some days, using it is very painful to me. So I have to use the keyboard a lot more, which can be operated without much wrist movement. But the calculator I had at the time did not allow to use the keyboard, which was quite frustrating. Having both of these input methods sounds pretty important to me.

2nd Interview

1) Andréa
2) 21
3) Student, multimedia integration
4) Yes, with Windows 10
5) Yes, with Android
6) 7
7) 9
8) Adobe Photoshop. It has many unique features and good UI/UX
9) Localisation is when you translate and adapt software for other languages and cultures. Accessibility is adapting software to work with screenreaders, add options for high contrast colors, make sure the buttons have good visibility, etcé Themes are color schemes, maybe with different fonts or UI.
10) On android, the calculator app. On desktop, either the calculator app or chrome, whichever is closest.
11) Yes for both
12) Basic operations, parentheses support, square root and power functions. Able to support multiple operations in the same equation, and a history of equations
13) Parentheses in the basic operation mode (non-scientific)
14) Accuracy, speed, offline capability, efficient, and that it shows the result
15) Once, parentheses were in the scientific menu and I had already entered many values, however it was all erased when I opened the scientific menu because windows treats the calculators as different.

3rd Interview

1) Catherine Bouchard
2) 39
3) Category Manager
4) Laptop with Windows 7
5) 1 iPhone7 (personal), 1 iPhone8 (business)
6) 5
7) 7 for computer, 8 for mobile
8) I like using Instagram. It’s easy to upload pictures and create stories.
9) I assume that I use the localization and accessibility every day with Google Maps.
10) At my job, I use a regular calculator every day. At home I would use the calculator on my smartphone.
11) I only use the one that came with my iPhone. I’ve never looked for other calculator apps, but I would expect them to be easy to use.
12) I would expect to find the basic features, but also quick margin calculator buttons.
13) It would be nice to have the memory feature.
14) I only expect the calculator to give me the right answer!
15) I don’t often use calculator apps, but I would say that any app that is too complicated or not intuitive enough would quickly cause me to lose interest and I would probably go back to a regular calculator.

4th Interview

1) Sergei Makotov
2) 31
3) SOEN
4) Yes, Windows
5) Yes, Android
6) 8
7) 7, 8
8) Steam (gaming platform) The interface is intuitive, the application itself is stable and responsive and new features and improvements are consistently added.
9) Yes, localization is translating language/currency/timezone according to the location of the user, accessibility is how easy is the program to use for novice and disabled users, themes are different versions of the same interface (in terms of color, shape of icons, etc…)
10) On the go: phone. At home: computer.
11) Yes, yes.
12) The ability to perform the 4 basic arithmetic functions.
13) Calculator apps should have a unit converter handy.
14) Output correct results… When a number is stored in memory, it can be retrieved later. When reset is pressed, it should clear all functions and memory.
15) Tried to do a simple multiplication with bracket on the Windows calculator program, it didn’t work in the basic version (it would multiply then add). I believe even the most basic calculator apps should have that functionality.

5th Interview

1) Travis Miller
2) 28
3) I am a PhD student working as aid research in biochemistry
4) Yes, I own a Macbook laptop that I use very regularly
5) Yes, I have an android cellphone
6) A solid 8 probably
7) A 9, I don’t have difficulty with a lot of computer software. For mobile, an 8 I would say
8) I like Discord a lot. It has a clean UI, lots of customization options and features. Controls are also quite responsive
9) Yes. Localization would be adapting the software to the language of a certain country or zone. Accessibility would be making sure that a maximum number of users can use your software even with things like disabilities. Finally, I believe theming is simply have the option of different color themes.
10) Definitely the one on my cellphone, it’s the one that is always at my disposition
11) Yes, most calculators are very easy to use. Some of the ones I have used, such as advanced scientific calculator are a bit more complex to use, but that is to be expected.
12) Basic operations, parenthesis support is essential too.
13) Functions graphing is not that much found in computers and cellphones calculators but is very practical to have.
14) Being accurate of course. After that, calculations should not take too long to execute or it becomes frustrating.
15) I don’t really recall an experience like that… For a while, I only have been using scientific calculators that do exactly what I want to for my job.

Appendix B: Meeting Minutes

Following is an example of the documents created after a team meeting.

Meeting Minutes
Notes by: Jason Brennan
May 31, 2019

In no particular order…

D1:
- We will edit the same document on google drive, then we put it all in Latex
- We will be using python for now, may switch to another language for D3/D4
- Jason, Manel, Charles are to work on the report, due June 5. Olivier and Mathieu will work on the backend but may be available to help finish up the report.
- We need 5 personas / interviews. Each with names, picture and possible transcript of the interview. Because of the need for 5, Olivier and Mathieu will probably need to conduct interviews as well.
- Some possible features of the app: Dark Mode (color theme), localization, accessibility, color blindness support (goes with color theme), screen reader.

Personas:
- Student - French speaking person - Person with visual disability - Person with mechanical disability - Researcher

Use Cases:
- Choosing a theme - Using the keyboard - Using a screen reader - Basic operators (+, -, etc) - Concatenate operations - Responsive design (mobile use) - Tooltips

D2:
- Mathieu will be major presenter, Manel minor presenter - We need a functional prototype for June 7

Some questions raised during the meeting:
- Do we need an abstract for the report?
- x^y: Can we use the ** (exponent) operator from python? We decided to use it for now, can change later if it's not allowed.
- Server choices: AWS, Google Cloud, or Digital Ocean?
- Sending input to the server, POST or GET? We decided POST - Can we do the processing in javascript? This would eliminate the need for python and a backend.
- Is the app expected to handle negative real numbers? Obvious issues with log(x) and sqrt(x)
- Should the app gracefully handle erroneous input? Division by 0, sqrt(-1), using x^y button before inputing LHS (is LHS considered 0 by default?)

-Some personal thoughts:
. In the essence of fairness, and with the due date not far away, I feel that we should all contribute 1 persona for D1. I'm working on a sheet with interview questions that we can all use. This way we're all asking the same things and it would standardize things a bit.
Mathieu has done a lot of initial work in setting up the app as that is his area of expertise. However, our decision to make the app web-based has some potential issues. I think Manel, Charles and myself have little experience in that department, meaning it would be hard for us to contribute code aside from our assigned function. I don't know how much work is required to connect the UI and backend but I'm worried that this becomes a bigger issue later as we try to divide the tasks we need to perform. Just something to be aware of.