EECS 448 – Software Engineering I Project 04
System Documentation

Group 08: Huzaifa

Zahid Mir Shazil

FaisalJunyi Zhao

Zhenzhou Wang

Story Point Analysis of Previous EECS Projects

1	2	3	5	8	13
Lab 1 - Simple Printing Exercise	Lab 6-Array reversal, file I/O	Lab 3- Lidear Sensor	Lab 10-Intro to OOP	Lab 11- DMV Class	Lab 7 - Recursion& Backtracking
Lab 4 - Fibonacci Numbers, ASCII Conversion	Lab 1- Command-line argument file manipulation	Lab6 – Recursion exercises	Lab4 – Elevator(Queues&Stacks)	Lab9,Lab10- Binary SearchTree	EECS 448- project1
Homework Assignment 1	Lab8 – Time Complexities	Homework Assignment 2	Homework Assignment 3	Lab5- Browser Tracking (List)	EECS 448- Project2
	Homework Assignment 4	Lab 6- Unified Modeling Language		EECS 448- Project3	Homework Assignment 5
	Homework Assignment 5	Homework Assignment 6	Lab 7-Software Testing	Lab 8-Web Language Intro	EECS 448- Project4

Legend:

	EECS 168
	EECS 268
	EECS 368
	EECS 388
	EECS 448

Estimate of Hours/Story Point:

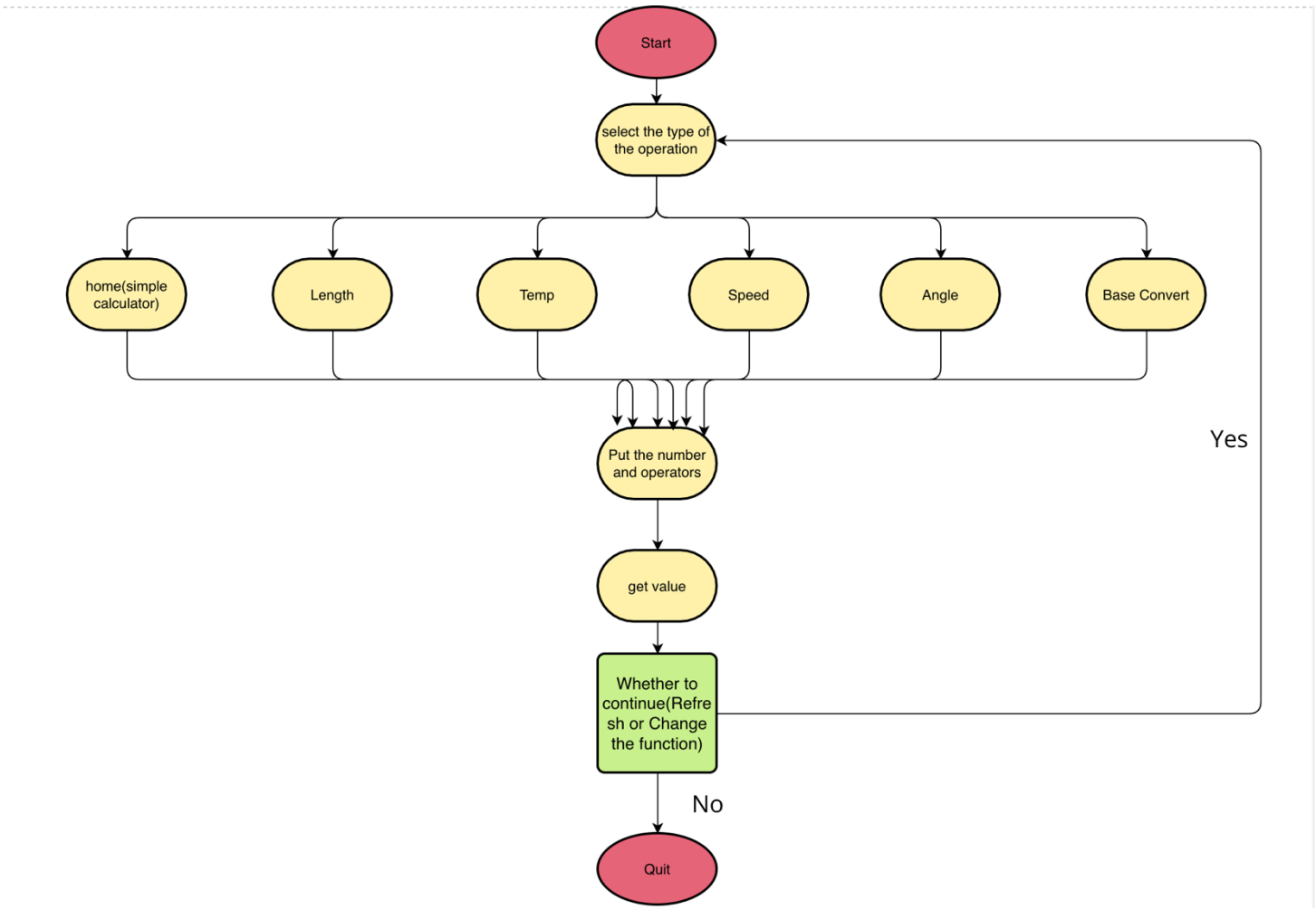
1 hour/story point.

We placed the project4 under 13 story points because we thought although project 4 is an upgrade from project 3 and adds some functions. But the added functions are much more complicated than project3, including some algorithms.

Introduction:

Our project 4 is a Scientific calculator program implemented in JavaScript, HTML and CSS. In project3, we create a simple calculator performs addition, subtraction, multiplication, and division. In the project 4, we create a scientific calculator based on the project 3 that can calculate slightly more complex trigonometric and calculus equations, this includes the conversion of trigonometric functions, as well as the conversion of international units such as length and temperature and speed and angle.

Project 4 Scientific Calculator Flow Chart



defect tracking tool

Date reported	Apr 23rd	Apr 23rd
Who reported it	Junyi Zhao	Mir, Shazil Faisal
Brief description of defect	sin cos and tan function will throw out the error if no brackets are added	When trigonometric functions add something, it will throw out error.
Date fixed	Apr 23rd	Apr 24th
Who fixed it		Huzaifa
Description of how it was fixed	The defects do not need to be fixed, just add the brackets for trigonometric functions	Fix the Syntax error

Code Integrated:

In Project 4, we used JavaScript, HTML and CSS, where the HTML files is responsible for the layout, the CSS files for the interface design and the JavaScript files for the logic, and modularity is evident in our project, especially in the corresponding .JS files for the different functions. For example, Home, length, temperature, speed, angle, and base convert are seven modules. So, the team division of labor can become very clear. Before integration, you just need to design different modules and give them to different people, and then put the different code together for testing.

The Integration Strategy we used in the Project 4 should be the Sandwich Integration. The Sandwich Integration has several features. It is combines two strategies of topdown and bottom-up which maximizing their strengths and minimizing their weaknesses. The sandwich integration artition modules into two groups (basically layout and logical). Implement and integrate logic artifacts top down(in the JavaScript files, which is the decision making flow and generally situated close to root) and Implement and integrate operational artifacts bottom up (in the HTML files,which is Performing actual operations and Generally found in lower levels). Last but not the least, test interfaces between logic artifacts and operational artifacts (both HTML and JavaScript files).

Our Project 4 is designed to meet these features from design to distribution to testing. And the reason we do the Sandwich Integration because it has three strengths, which is more easier fault isolation, major design faults show up early and the operational artifacts are adequately tested.

Deployment Plan:

The steps we did are brainstorming about the use case of the potential project, design of each part of the project, coding of the project, and testing of the project. When brainstorming the project, we were thinking about how an amateur project like a class project can do beyond just a GPA reaper. We found that we can deal with something people typically google for. However, sadly, google does not work every time and internet is not a human right. After all, it is a privilege. Therefore, we decided that we can deal with what typically internet does when internet is not an option. This includes some essential work: calculation of units, trig calculations, and some basic deg/rad transforming job. This is extremely important when using internet is limited, say during an exam, in military bases, or using internet is not appropriate, like during educational sessions with kids – internet is a disturbing factor too early to them. Those limitations give our product a right place to use.

Knowing our use case for the project – exams, military bases, and kids. We should think about the aim of our project. Most exams are time constrained. It is even more important for military – a second could mark between life and death. For kids, to stop them from distracting is very important. Therefore, on designing our project, we aimed on its size, speed, and easiness to read the code. For likes of military, security is very important. To sell our project, one of the most important elements is the security of our code. Therefore, ease to be checked by officials is very important. It is also important to create one-size-fits-all project. Our use cases are very diverse. Making only a simple calculator will not work. We should design a calculator that sorts as much needs as possible. When brainstorming about what to do, my teammate comes up with the “excel sheets” plan: to integrate different calculators like sheets in an excel file. We simply click to the one in need, do the calculation and go back to the next step. This design has two benefits: continuation of calculation. We can finish one side of calculation and directly feed that calculation into the next step. This is convenient. The other benefit is finish conversion in one click. Multiplication by 180 and division by pi is too hard. Same to change between C and F. Therefore, one-click conversion is necessary.

Speaking about the cost of our project, coding is not the biggest cost source. There are many same or similar product on the market and our aim is to lobby or advertise the relevant organizations to use our product. We should develop personal business relations by business dinners and other means of socials. Another cost is to pass a background check. Our code needs to be thoroughly tested by the officials to be 100 per cent sure that no malicious code is detected. We also need to collaborate with education sectors to sell our product as a accessible and easy mean of education of kids.

Maintenance Plan:

Maintenance of software usually requires several different types and phases. Corrective maintenance, adaptive maintenance, perfective maintenance, and preventive maintenance. For each of these types of maintenance, a number of maintenance strategies can be adopted to control maintenance costs. Corrective maintenance is the process of correcting problems that happened during the system development phase but were not noticed during the system testing phase. This part of the maintenance burden accounts for 17–21% of the total maintenance workload. Adaptive maintenance is the process of modifying software to respond to changes in information technology and management requirements. This component of the maintenance burden accounts for 18% to 25% of the total maintenance workload. Perfective maintenance is the modification of an existing software system to expand its functionality and improve its performance, mainly by adding functional and performance features that were not specified during the system analysis and design phase. These features are necessary to improve the functionality of the system. This aspect of maintenance accounts for 50% to 60% of the overall maintenance work, a large proportion. It is also an important aspect in relation to the quality of the system development. Preventive maintenance is necessary to improve the reliability and maintainability of the application software and to adapt to future changes in the software and hardware environment. This aspect of the maintenance workload represents around 4% of the overall maintenance workload.

Ideally, our code would require minimum maintenance. Unfortunately, it is hard to have God smile on us every single time. therefore, having a proper maintenance plan is important.

Our program is a html-based program. It can be both on local or online location. If on an online location, even on a local server location, maintenance of such server may apply. Say a most basic azure server. A two-core CPU web server costs around $15 per month on a yearly basis. Therefore, the maintenance of the server may cost $180 a year.

We should also consider employment of maintenance staff. This is a small project, so we can assign the maintenance tasks along with other projects. I would say these will cost around $3000 for salary per year.

To best brace for impact, we should also prepare for the deprecation of some features of html that may affect our project. This happens once in a while. For example, Adobe no longer supports Flash Player after December 31, 2020 and blocked Flash content from running in Flash Player beginning January 12, 2021. Therefore, all Flash based web projects need to be rewritten. to maintain the longevity of our project, we should sign a bet agreement with an insurance company that our code could be needed to rewrite within 15 years. I would suggest a $300 per year for the contract given web based language's historical length of support.

Therefore, I would say a yearly maintenance cost of $3500 to $4000 is probable. Depending on the proportion of the maintenance costs and the budget, different maintenance components will require

different amounts of money; Corrective maintenance will cost approximately $700-$800, Adaptive maintenance will cost approximately $700-$900 and Perfective maintenance will take up half of the cost, approximately $1750-$2000. Finally, Preventive maintenance is the least expensive, costing $200.

Code Review:

Code review is an effective way of detecting a fault and the faults are detected early time during the software process. As a mid sized code, the way to find bugs are limited. Our testing methods include extreme testing (say tan 180 degrees, division by 0), illegal input testing (missing ")"), etc .We used the inspection to review the code, Inspection is a five-step process: The first is Overview, second is Preparation, third is inspection, fourth is Rework and fifth is Follow-up. After following the five steps, here are the faults we have found:

1. Fault Description: Trig functions do not work without paratheses. It should. If we input sin 1 instead of sin (1), it will end in error. (Junyi Zhao Reported)

   Junyi ran the codes and reported the bug, which is indeed a fault, but it is not a fault that has to be fixed. It makes sense to force the addition of parentheses, as it better separates the following operators from the values, but the only thing that doesn't make sense is that adding parentheses is a pain. However, the rest of our team felt that our calculator had to make a rigorous judgement that parentheses had to be forced when using trigonometric functions. We intentionally let the format be with parenthesis and couldn't let pi in or the other thing be added because for that we'd have to change the entire structure. So after some discussion, it was decided that the glitch here could be avoided by adding parentheses.
2. Fault Description: rad and degree issue. Reversing the angular frequency and degree caused some problems with the display of trigonometric functions. (Zhenzhou Wang Reported)

   Zhenzhou Wang reported the bug and Huzaifa was assigned to fix the fault. The reason for the error was that he had mistakenly written rad and degree backwards when applying the method. He changed the structure slightly and adjusted the parameters to align the wrong object to the correct place and the error was fixed.
3. Fault Description: When using the temperature conversion function, when converting from Fahrenheit to Celsius, the system will report an error if the degree of Fahrenheit is below -10000 degrees. (Shazil reported)

   Shazil reported the bug and Huzaifa fixed it, the reason for the error was that he had incorrectly set the wrong range of degrees when applying the method, which caused the error to be reported when the range was exceeded.
4. Fault Description:

   In the function for speed conversion, the operator for trigonometric functions was introduced by mistake. This resulted in a system throw out an error.

   The reason for the error was that the speed conversion function is a one-click conversion of numbers and the reference to trigonometric functions would lead to an error in the system's operations. The trigonometric functions are not used in speed conversions in real life applications, so the solution was to remove the trigonometric functions from the interface after discussion with the team and considering the user experience and user requirements.