EECS 448 – Software Engineering I Project 04 System Documentation

Group 08:

Huzaifa Zahid

Mir Shazil Faisal

Junyi 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-Project4	🟨 Homework Assignment 5
	🟨 Homework Assignment 5	🟨 Homework Assignment 6	⬜ Lab 7-Software Testing	⬜ Lab 8-Web Language Intro	⬜ EECS 448-Project3

Legend:

🟩	**EECS 168**
🟦	**EECS 268**
🟨	**EECS 368**
🟧	**EECS 388**
⬜	**EECS 448**

Estimate of Hours/Story Point:

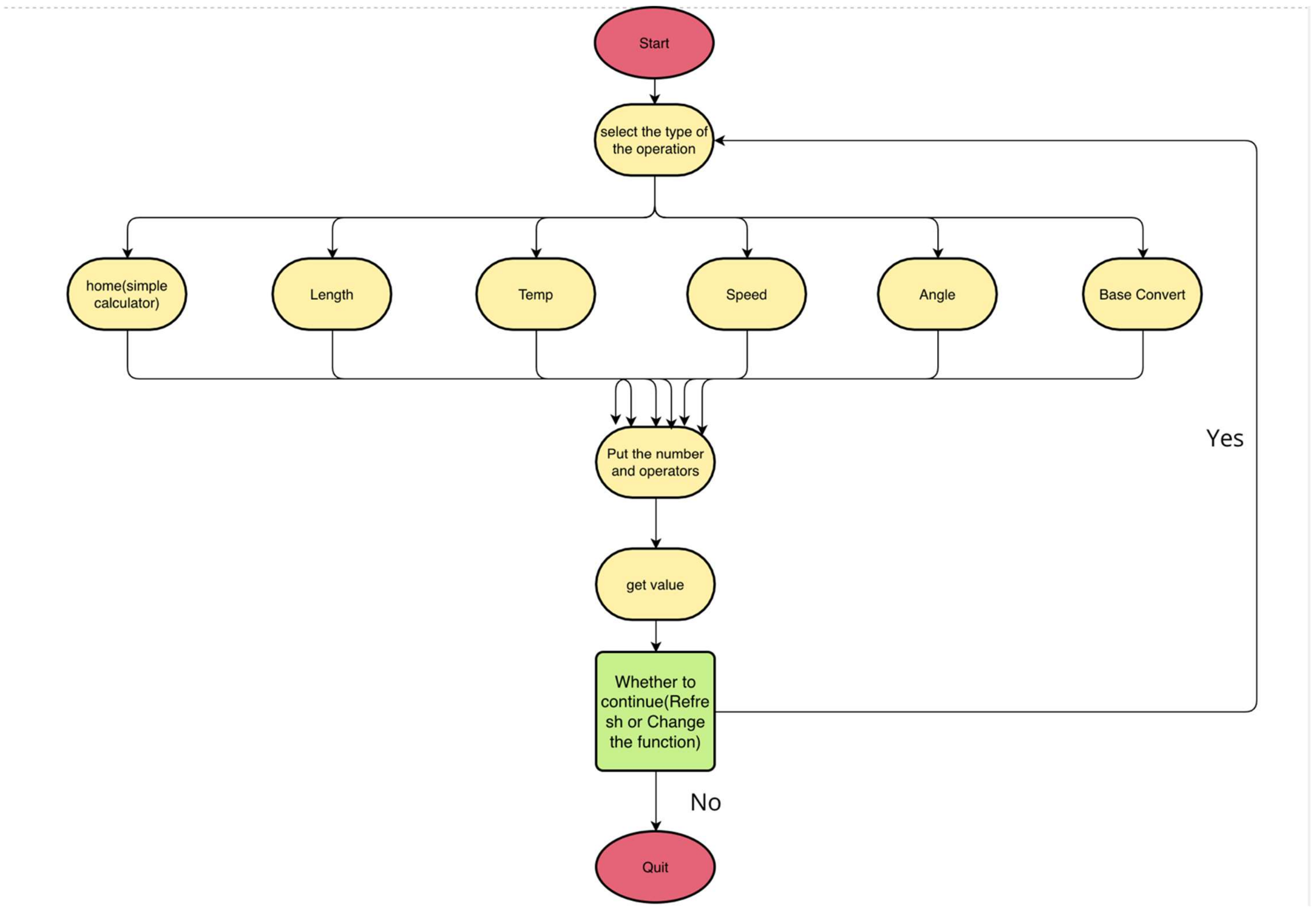
2 hours/story point.

We placed project4 under 8 story points because we thought although project 4 is a final project, it is just an upgraded version of project 3 and adds some functions. However, keeping in mind the complexity of some functions and making sure the correct functionality of the project (after debugging), we decided it would be best to place it in a 8 story point pool with 2 hours/story point.

Introduction:

Our project 4 is a Scientific calculator program implemented in JavaScript, HTML and CSS. In project3, we create a simple arithmetic calculator that performed addition, subtraction, multiplication, and division. In this project (4), we created a scientific calculator based on the previous project that can calculate slightly more complex trigonometric operations (Sin, Cos, and Tan), and conversions of units of measurement such as length, temperature, speed, and angle, and number system (binary and denary) conversion.

Project 4 Scientific Calculator Flow Chart


Start
select the type of the operation
home(simple calculator)
Length
Temp
Speed
Angle
Base Convert
Put the number and operators
get value
Whether to continue(Refresh or Change the function)
Yes
No
Quit

Code Integrated:

In Project 4, we used JavaScript, HTML and CSS, where the HTML file is responsible for the layout, the CSS file 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 would be the Sandwich Integration. The Sandwich Integration has several features. It combines two strategies of topdown and bottom-up approaches which maximizes their strengths and minimizes their weaknesses. The sandwich integration describes two groups (basically layout and logic). 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 is because of its strengths; easier fault isolation, major design faults show up early, and the operational artifacts are adequately tested.

Deployment Plan:

We started with brainstorming the use case of the potential project, design of each part of the project, coding of the project, and testing of the project. At this point, our project is not refined enough to be distributed in the market. Even though our functions that we intended to have in our calculator work, they could use more thorough testing and need to be able to integrate better with each other, for example, we need to make trigonometric functions work with basic arithmetic too, which our calculator does not do yet. Apart from this, we also need to advertise our calculator to our target audience – school-going kids, people in academia, and laymen who need everyday unit conversions at ease – before we deploy the project into the market. Right now, the calculator contains basic unit conversions, but before deployment, it should be updated to include a thorough suite of units that anyone can use.

On designing our project, we aimed on its size, speed, and easiness to read the code. It is also important to create one-size-fits-all project. Our use cases are very diverse. Making only a simple calculator would not work, which is why we designed 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 an 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, several 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, but to be safe, having a proper maintenance plan is important. Our program is a html-based program. It can be both on local and 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 occasionally. 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.

Defect Tracking:

Date reported	Apr 22nd	Apr 22nd	Apr 23rd
Who reported it	Junyi Zhao	Huzaifa Zahid	Huzaifa Zahid
Brief description of defect	Radian to Degree Conversion didn't have pi	The code doesn't handle the error when an operation is made to Sin, Cos, Tan functions	Incorrect conversion of Fahrenheit to Celsius
Date fixed	Apr 22nd	Apr 23rd	Apr 23rd
Who fixed it	Mir Shazil Faisal	Mir Shazil Faisal	Huzaifa Zahid
Description of how it was fixed	Pi button added to that mode of the calculator	An Error thrown	IF-statement that led into the code was incorrect.

Date reported	Apr 24th	Apr 24th	Apr 24th
Who reported it	Huzaifa Zahid	Zhenzhou Wang	Mir Shazil Faisal
Brief description of defect	Temperature Calculator doesn't have negative values	Screen doesn't reset after the mode is changed on the calculator	'=' button not working
Date fixed	Apr 24th	Apr 24th	Apr 24th
Who fixed it	Mir Shazil Faisal	Zhenzhou Wang	Mir Shazil Faisal
Description of how it was fixed	A button for '-' added	An Error thrown	There was a syntactical mistake in the declaration

Code Review:

Code review is an effective way of detecting a fault and the faults are detected early 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 some of the faults, apart from the defect tracking above, that we found:

1. **Fault Description**: Trig functions did not work with paratheses. If we input sin(90) instead of sin90, it will end in error. (Junyi Zhao Reported)
   Junyi ran the codes and reported the bug. It wasn’t a major bug but still Huzaifa Zahid corrected it. It made sense to force the addition of parentheses, as it better separates the operators from the values. So, not according to our format, sin(90) will be accepted but not sin90.

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 calculator produced bogus values (Mir Shazil Faisal Reported)
   Shazil reported the bug and fixed it himself, the reason for the error was that he had entered the if statement incorrectly that led to the part of the code that converted the temperature value. Instead of writing *if(initial == "C° to F°" || initial == "F° to C°"),* he had written *if(initial == "C° to F°" || initial == "F to C").*

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. This was reported and rectified by Zhenzhou Wang.

*Note: coverage directory, node_modules directory, package.json, and package-lock.json files are not written by us, but were included as part of jest testing. We tried using Jest testing to test the correct functioning of our methods, but for some reason the test wasn't identifying our methods and we couldn't use our test suite. So, instead we did manual testing of our methods. However, we have left the test files as part of our submission.