./

GENESIS – Mini-Project Report and Summary



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **To be Approved** | **Remarks/Revision Details** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Details**

Contents

[Contents 3](#_Toc58237596)

[MINIPROJECT-1 [TEAM] 4](#_Toc58237597)

[TITLE: SCIENTIFIC CALCULATOR 4](#_Toc58237598)

[1.INTRODUCTION: 4](#_Toc58237599)

[2.TOPICS & SUB-TOPICS: 4](#_Toc58237600)

[3.OBJECTIVES & REQUIREMENTS: 5](#_Toc58237601)

[3.1 OBJECTIVES: 5](#_Toc58237602)

[3.2 REQUIREMENTS: 5](#_Toc58237603)

[3.2.1 FUNCTIONAL REQUIREMENTS: 6](#_Toc58237604)

[3.2.2 FUNCTIONAL REQUIREMENTS: 7](#_Toc58237605)

[3.2.3 AGEING: 7](#_Toc58237606)

[3.2.4 COSTING: 7](#_Toc58237607)

[3.2.5 SWOT ANALYSIS: 7](#_Toc58237608)

[3.2.6 4W and 1H: 8](#_Toc58237609)

[4. DESIGN: 9](#_Toc58237610)

[4.1 HIGH LEVEL DIAGRAM (STRUCTURAL DIAGRAM): 9](#_Toc58237611)

[5. TEST PLAN: 11](#_Toc58237612)

[6. GITHUB 13](#_Toc58237613)

[6.1 GIT LINK-https://github.com/Genesis99003167/genesis\_miniproject 13](#_Toc58237614)

[6.2 GIT DASHBOARD: 13](#_Toc58237615)

[7. SUMMARY: 14](#_Toc58237616)

[8. INDIVIDUAL CONTRIBUTIONS & HIGHLIGHTS: 14](#_Toc58237617)

[9. CHALLENGS FACED AND HOW WERE THEY OVERCOME: 15](#_Toc58237618)

**List of Table**

Table 1 High Level Requirements**……………………………………………………………………………………………………..6**

[Table 2 Low Level Requirements 6](#_Toc58238294)

[Table 3 Costing Table 7](#_Toc58238295)

[Table 4 SWOT Analysis 8](#_Toc58238296)

[Table 5 Test Plan Table 12](#_Toc58238297)

[Table 6 Individual Contribution and Highlights 15](#_Toc58238298)

[Table 7 Challenges Faced and How were they overcome 16](#_Toc58238299)

**List of Figures**

[Figure 1 Structural Diagram of Scientific Calculator 10](file:///C:\Users\Training\Downloads\GENESIS%20-%20Mini-project_ScientificCalculator%20Report%20.docx#_Toc58239142)

[Figure 2 Class Diagram of Arithmetic and Trigonometric Function 11](#_Toc58239143)

[Figure 3 Structural Deployment Diagram 11](#_Toc58239144)

[Figure 4 Behavioral Diagram of Factorial function 12](#_Toc58239145)

[Figure 5 Behavioral Sequence Diagram 12](#_Toc58239146)

[Figure 6 Behavioral Sequence Diagram 13](#_Toc58239147)

MINIPROJECT-1 [TEAM]

# TITLE: SCIENTIFIC CALCULATOR

1.INTRODUCTION:

Ever since, dull mathematical issues in the organizations has certainly become a difficult task, tons of systems or conventions, , it can be said that the method includes a variety of procedures ranging from inserting figures for dreary measurement, to preparing of numbers and recovery of mistakes. Nevertheless, because of this, there is the need to achieve a broad examination to find the complexities engaged with the whole cycle. Hence it embarked on the design and execution of a basic scientific calculator for this same purpose.

A scientific calculator is a mathematical calculation system that distinguishes from a PC by a restricted issue area and interface advanced for intuitive estimation instead of programming. Adding machines can be equipment or programming, and mechanical or electronic, and are frequently incorporated into gadgets, for example, PDAs or mobile handsets.

2.TOPICS & SUB-TOPICS:

Our mini project is a simple scientific calculator implemented in C using agile model of software development. The various mathematical calculations like arithmetic operations, trigonometric operations, conversions and some special operations are implemented in C language using Agile model of Software Development Life Cycle(SDLC).

* 1. **C Language**

C is a powerful programming language for general use. Software including operating systems, databases, compilers, and so on can be built using it. For beginners, C programming is an excellent language to learn to program. This language was designed to be easy to learn and use. The language not only allows programmers to create simple many applications but can also develop fairly  
complex applications as well. Our project was done using C as it proved to be easily managing. Modifying the functions during troubles was simple later. The C language is easily readable and understandable. Low-level memory access, easy keyword collection, and clean style are the main characteristics of C language, making C language suitable for system programming such as operating system or compiler development. Low-level memory access, easy keyword collection, and clean style are the main characteristics of C language, making C language suitable for system programming such as operating system or compiler development.

* 1. **Agile Model**

The agile model is used in the entirety of the software development group for manufacturing. The related rising expenses with the production of software and unsatisfactory reliability, quality and the resulting app functionality has inspired software engineers to build comparing these new ones which is difficult. Software development models because their disciples also use multiple model’s terminology, and except for their origins, models frequently have little in common.

The use of such a model is-

I. Encourages us to specify (i.e., to define) what the device is supposed to do Requirements before the construction of the device(i.e., designing).

II. Encourages us to schedule how components can communicate (i.e., Designing) until the components are assembled (i.e., coding).

III. Allows project managers to more reliably monitor and uncover progress possible early slippage.

IV. Demands that a series of records be produced by the development process which can be used to validate and maintain the device later.

V. Lowers the cost of production and maintenance for all of the above purposes.

VI. Enables the company that produces the framework to be more coordinated and also manageable.

3.OBJECTIVES & REQUIREMENTS:

3.1 OBJECTIVES:

The objectives of our project are to:

* Be familiar with Visual studio, it’s source control and other Git services.
* Get conceptually clear with C and its unit test frameworks.
* Implement various mathematical and special operations in C language.
* Be able to clone repository, commit changes, push and pull the code within a team.

3.2 REQUIREMENTS:

The calculator has the following keys: 0 to 9, ., +, -, \*, /, ±, =, C, CE and 1-operand keys: 1/x, x2 , sqrt. There are also trigonometric functions and some special functions like factorial, even-odd, prime numbers detection. In any situation the calculator must produce a correct result defined by the well-known arithmetic rules. If the calculations are impossible the calculator has to display information helping the user to resolve the situations.

3.2.1 FUNCTIONAL REQUIREMENTS:

All the arithmetic and science operations whose buttons can be seen are practical specifications, as well as memory store, recall and clear. It will also define turning on and off as practical.

Table 1 High Level Requirements

|  |  |
| --- | --- |
| ID | DESCRIPTION |
| HL1 | Arithmetic functions |
| HL2 | Trigonometric Function |
| HL3 | Logical operators |
| HL4 | Conversion Functions |
| HL5 | Special Functions |

Table 2 Low Level Requirements

|  |  |
| --- | --- |
| ID | DESCRIPTION |
| HL1\_L1 | Add Function |
| HL1\_L2 | Subtract Function |
| HL1\_L3 | Multiplication Function |
| HL1\_L4 | Division Function |
| HL2\_L1 | Sin Function |
| HL2\_L2 | Cosine Function |
| HL2\_L3 | Tangent Function |
| HL2\_L4 | Cosecant Function |
| HL2\_L5 | Secant Function |
| HL2\_L6 | Cotangent Function |
| HL3\_L1 | AND Function |
| HL3\_L2 | OR Function |
| HL3\_L3 | NOT Function |
| HL3\_L4 | NAND Function |
| HL3\_L5 | NOR Function |
| HL3\_L6 | XOR Function |
| HL3\_L7 | XNOR Function |
| HL4\_L1 | Kilometer to Meter |
| HL4\_L2 | Meter to Kilometer |
| HL4\_L3 | KMPH to MPH |
| HL4\_L4 | MPH to KMPH |
| HL5\_L1 | Modulo Function |
| HL5\_L2 | Power Function |
| HL5\_L3 | Even Odd Function |
| HL5\_L4 | Prime Function |
| HL5\_L5 | Factorial Function |
| HL5\_L6 | Square Function |
| HL5\_L7 | Exponential Function |
| HL5\_L8 | Log Function |

3.2.2 FUNCTIONAL REQUIREMENTS:

It would be a split second to provide the response, any slower non-functional criteria and you would reject the calculator. To be able to deal with big numbers, it should also be scalable. Another significant non-functional requirement is that the key layout should suit a calculator's standard layout, and for usability, the keys should be the correct size and shape.

3.2.3 AGEING:

* The first scientific calculator that included all of the basic ideas above was the programmable Hewlett-Packard HP-9100A released in 1968.
* The HP-35, introduced on February 1, 1972, was Hewlett-Packard's first pocket calculator and the world's first handheld scientific calculator .
* Handheld scientific calculator on January 15, 1974, in the form of the SR-50.
* First graphics calculator developed in 1990 Tl-81 for math education that adds a new visual dimension to mathematics instruction.

3.2.4 COSTING:

Table 3 Costing Table

|  |  |  |
| --- | --- | --- |
| **YEAR** | **MODEL** | **COST** |
| 1972 | Tl-2500 | $149.95 |
| 1985 | fx-7000G | $75 |
| 1988 | Tl-68 | $55 |
| 1992 | Tl-85 | $130 |
| 2020 | Tl-30XS | $18.13 |

3.2.5 SWOT ANALYSIS:

Table 4 SWOT Analysis

|  |  |
| --- | --- |
| **STRENGTH**   * Innovation-driven goods * Strong technology experience that drives new companies * Board product range across market segments * Effective team of advertisers and big sponsorships | **WEAKNESS**   * Stereotyped Brand Image * Slow to launch goods as per the rapidly evolving needs of the market |
| **OPPORTUNITIES**   * Youth-centered plan for the introduction of new goods * Advertisement and promotion of brands * Adapting to emerging technologies and forming new companies * Expand development in the company of timepieces and educational science calculators | **THREATS**   * Huge rivals * Economic volatility can have a huge effect on revenue. * Requirements at a low level |

3.2.6 4W and 1H:

* **WHAT**- A scientific calculator is a calculator designed to help you quantify problems in science, technology, and mathematics. It has a lot more buttons than your regular calculator, which allows you to do your four-simple addition, subtraction, multiplication, and division arithmetic operations.
* **WHEN**- This calculator is use when  various kinds of numbers and problems such as these- Trigonometry problems, scientific numbers that have a multiplication by 10 to certain power, Pi problems, logarithm problems with base 10 and the natural base, probability problems that use the factorial function and many more complicated functions.
* **WHERE**- Earlier, whole classes were dedicated to studying how to calculate such equations, and it would not have been expected for students to really learn what those equations were for and how they would use them in future careers. But anyways, scientific calculator has made it easy to perform several functions to influence a career in science, mathematics or engineering.
* **WHY**- The evolution of scientific calculator has made it possible to avoid the problems of solving by hand for long and difficult expressions. It also does the computation in the following order f operations.
* **HOW**- The way calculator works has got actually the heart of it. The perfect technical explanation for the same can be is it includes describing and comparing the decimal numbers we use in a different format, called binary, with electrical circuits known as logic gates.

4. DESIGN:

4.1 HIGH LEVEL DIAGRAM (STRUCTURAL DIAGRAM):

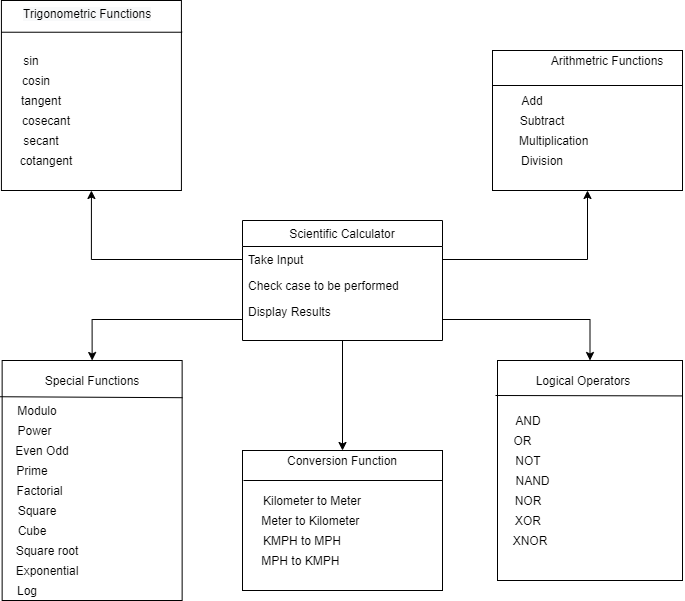


Figure 1 Structural Diagram of Scientific Calculator

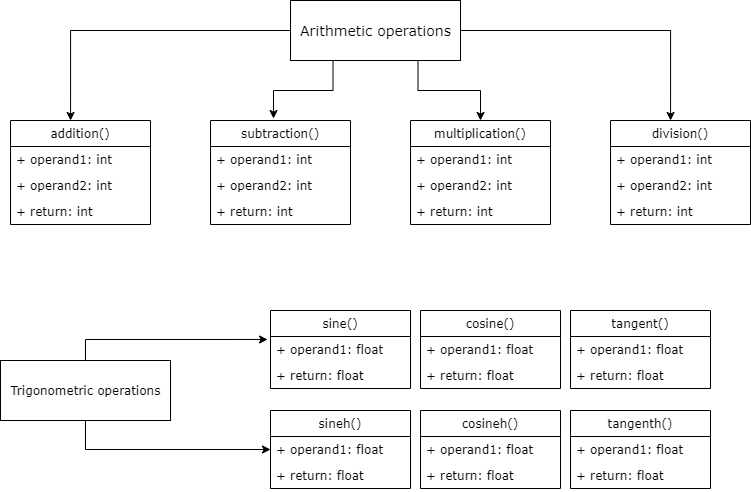


Figure 2 Class Diagram of Arithmetic and Trigonometric Function

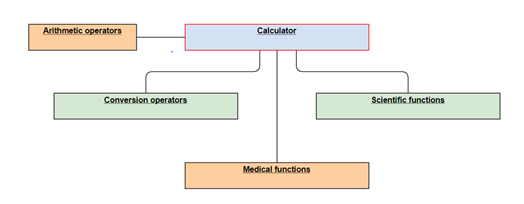


Figure 3 Structural Deployment Diagram

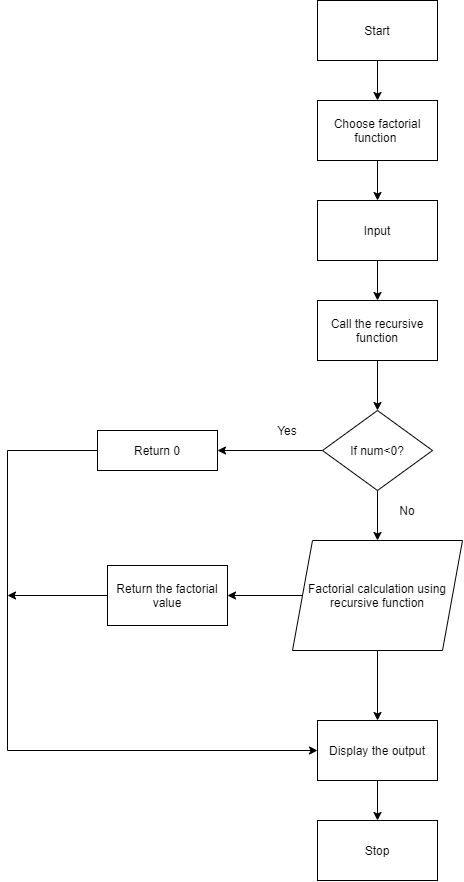
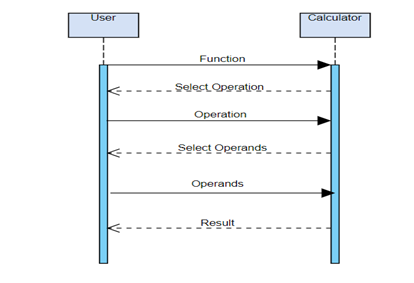


Figure 5 Behavioral Sequence Diagram

Figure 4 Behavioral Diagram of Factorial function

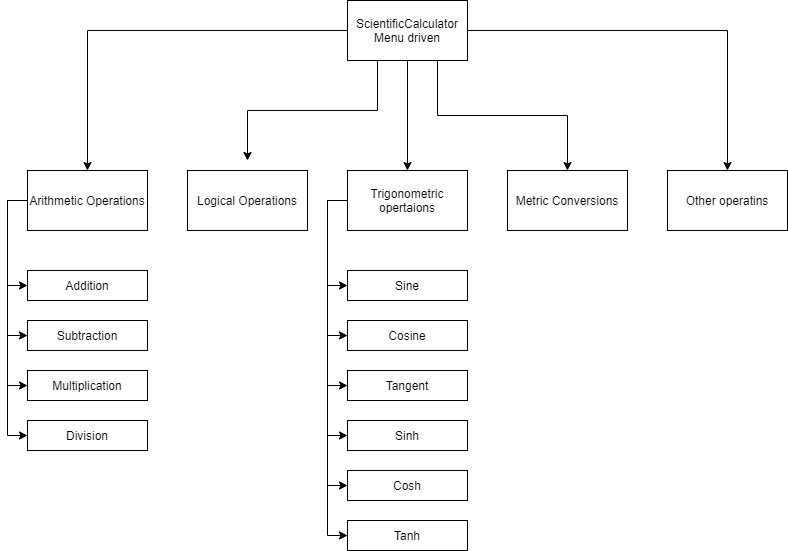


Figure 6 Behavioral Sequence Diagram

5. TEST PLAN:

Table 5 Test Plan Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Description** | **Expected Input** | **Expected Output** | **Actual Output** |
| HL1\_L1 | Add Function | Enter one/two value greater than 65535 | Error |  |
| HL1\_L1 | Add Function | Enter two values | Sum of inputs |  |
| HL1\_L2 | Subtract Function | Enter one/two value greater than 65535 | Error |  |
| HL1\_L2 | Subtract Function | Enter two values | Difference between inputs |  |
| HL1\_L3 | Multiplication Function | Enter two values | Multiplication of inputs |  |
| HL1\_L4 | Division Function | Enter divisor value as 0 | Error |  |
| HL1\_L4 | Division Function | Enter two non-zero values | Division of inputs |  |
| HL2\_L1 | Sin Function | Enter one int/float value | Sin of the given input in float |  |
| HL2\_L2 | Cosine Function | Enter one int/float value | Cosine of the given input in float |  |
| HL2\_L3 | Tangent Function | Enter one int/float value | Tangent of the given input in float |  |
| HL2\_L4 | Cosecant Function | Enter one int/float value | Cosecant of the given input in float |  |
| HL2\_L5 | Secant Function | Enter one int/float value | Secant of the given input in float |  |
| HL2\_L6 | Cotangent Function | Enter one int/float value | Cotangent of the given input in float |  |
| HL3\_L1 | AND Function | Enter two int values | AND of the inputs |  |
| HL3\_L2 | OR Function | Enter two int values | OR of the inputs |  |
| HL3\_L3 | NOT Function | Enter single input | Not of the input |  |
| HL3\_L4 | NAND Function | Enter two int values | NAND of the inputs |  |
| HL3\_L5 | NOR Function | Enter two int values | NOR of the inputs |  |
| HL3\_L6 | XOR Function | Enter two int values | XOR of the inputs |  |
| HL3\_L7 | XNOR Function | Enter two int values | XNOR of the inputs |  |
| HL4\_L1 | Kilometer to Meter | Enter two int/float values greater than 0 | Conversion from KM to Meter |  |
| HL4\_L2 | Meter to Kilometer | Enter two int/float values greater than 0 | Conversion from Meter to KM. |  |
| HL4\_L3 | KMPH to MPH | Enter two int/float values greater than 0 | Conversion from KMPH to MPH |  |
| HL4\_L4 | MPH to KMPH | Enter two int/float values greater than 0 | Conversion from MPH to KMPH |  |
| HL5\_L1 | Modulo Function | Enter two non-zero values | Modulo of the inputs |  |
| HL5\_L1 | Modulo Function | Enter divisor value as 0 | Error |  |
| HL5\_L2 | Power Function | Enter two int values | Power operation for the given inputs |  |
| HL5\_L3 | Even Odd Function | Enter one int value | Tells whether input is even or odd |  |
| HL5\_L4 | Prime Function | Enter one int value | Tells whether input is prime or not prime number |  |
| HL5\_L5 | Factorial Function | Enter value less than 0 | Error |  |
| HL5\_L5 | Factorial Function | Enter value greater than 0 | Factorial of the input |  |
| HL5\_L6 | Square Function | Enter input | Square of the given input |  |
| HL5\_L7 | Exponential Function | Enter valid input | Exponential of the given input |  |
| HL5\_L8 | Log Function | Enter input as 0 | Error |  |
| HL5\_L8 | Log Function | Enter input less than 0 | Error |  |
| HL5\_L8 | Log Function | Enter input greater than 0 | Natural log of the given input |  |

# 6. GITHUB

6.1 GIT LINK-<https://github.com/Genesis99003167/genesis_miniproject>

6.2 GIT DASHBOARD:

Graphical user interface, text, application

Description automatically generated

7. SUMMARY:

Our project started with the installation and setup of Visual studio code (VSC). Then, a separate GitHub id is made by each member of the group. A member of the group adds her team members as contributors so that they can clone the repository and start making changes to their code. After adding their codes, these were committed and pushed to the repository through the source control action of VSC. Git was also installed in our systems. The GitHub account is then linked through enabling authentication because any Git action requires GitHub authentication such as pushing to a repository or cloning a repository. Using GitHub actions, robust and dynamic automations are built.

8. INDIVIDUAL CONTRIBUTIONS & HIGHLIGHTS:

Table 6 Individual Contribution and Highlights

|  |  |  |  |
| --- | --- | --- | --- |
| **PS NO.** | **No of Lines of code** | **NO of functions** | **status** |
| 99003165 | 64 | 10 | Passed |
| 99003166 | 68 | 9 | passed |
| 99003167 | 70 | 8 | passed |

9. CHALLENGS FACED AND HOW WERE THEY OVERCOME:

Table 7 Challenges Faced and How were they overcome

|  |  |
| --- | --- |
| **CHALLENGES** | **SOLUTION** |
| * Disabled  source control action | * Proper authorization of GitHub |
| * Error for user name and user mail id | * In git, executed the command line configurations for the same |
| * Got error’s while doing unit testing | * Some test cases for float value failed and rest for int it got passed. |
| * Setup of GitHub in visual studio code | * Checked and watched the videos of the setup and help from friends |