

Agile Multi-File C Language Scientific Computation System

A Final Project Report

Submitted in the partial fulfillment of the requirements Adv Topics in
Software Engineering (CSE 6324)

By

Name	ID Number
Mr. Satyadev Kalakonda	1002059676
Mr. Sai Akhilesh Veldi	1002079415

Under the
supervision of
Dr. Farnaz Farahanipad



UNIVERSITY OF
TEXAS
ARLINGTON

INDEX & CONTRIBUTIONS

S.NO	TITLE	PAGE NO
1	Introduction	3
2	4w's & 1H	4
3	Project Goals	5-6
4	Functional & Non-Functional Requirements.	6-7
5	High-level & Low-level Requirements	7-8
6	Choosing the Appropriate SDLC Model for the Project.	9
7	Cost & Time Estimation	10
8	Planning & Scheduling Tools	12-15
9	Analysis of the times you were behind or ahead of your ideal schedule	15
10	High-level & Low – level Design	16-21
11	Implementation & Sample Output	21 -29
12	High-level & Low-level Test plan	29-34
13	Challenges Faced & How it was overcome	34
14	Each Team Member's role and Contributions	35
15	Conclusion	36

Introduction

Brief Overview of the project:

A Scientific calculator is a device that can perform some mathematical operations like arithmetic operations & scientific operations i.e., here in this project we are going to implement some calculator operations using C Language with a Command Line Interface (CLI) where it can perform functions like

- Arithmetic Operations
 - MUL
 - SUB
 - DIV
 - ADD
- Trigonometric Operations
 - SIN
 - COS
 - TAN
- Scientific Operations
 - LOG
 - POWER
 - SQUARE ROOT

The Scientific calculator doesn't involve any complex operations or operations. It is easy to apply the values in the system to find the required result rather than remembering and applying it in the formula. Thus, by studying all these facts, the necessary functions have been implemented to get the required results in a better way.

4W's and 1'H

Who:

- Everyone in day-to-day life uses the project. All the students and employees who require the calculations will use the program to find their answers. The main objective of this project is the user should be satisfied with a friendly interface and fast calculation.

What:

- We can accept the results that the user can perform mathematical operations like arithmetic, Trigonometric & Scientific operations.
- All the basic functionalities are included in the project. Users can use the program efficiently and get the required results.

When:

- The project can be used when the calculations are required and get the results for basic, and Math related problems and the result will be obtained fastly.

Where:

- In all the Domains it can be used, and the project is portable and user-friendly, It should overcome all the drawbacks of the Old existing system and most important of all meet the user requirements.

How:

- The Project is going to be implemented in C language for the Both Windows & Linux OS
- The constraints of the project are to develop using industry standards with a multi-file approach.

Project Goals

For this project, we have utilized the SMART goals technique to ensure that our objectives are specific, measurable, achievable, relevant, and time-bound, allowing us to develop a user-friendly and accurate scientific calculator in C language.

Specific:

- Develop a command line calculator in C language that supports basic arithmetic operations and scientific calculations such as logarithm, power, square root, sine, cosine, and tangent.
- The calculator should be able to accept mathematical expressions as input from the user and output the result of the expression after performing the appropriate calculations using the functions provided.
- The calculator should have a simple and intuitive interface that allows the user to input the desired mathematical expression and receive the result quickly and accurately.

Measurable:

- The calculator should be able to perform basic arithmetic operations and scientific calculations accurately and efficiently.
- The calculator should have a user-friendly interface that can be easily understood and used by anyone.
- The calculator should be tested and debugged thoroughly to ensure that it is error-free.

Achievable:

- The calculator can be developed using C language and the appropriate functions can be used for arithmetic and scientific calculations.
- The calculator's interface can be designed using simple and intuitive input/output methods.
- The calculator can be tested and debugged using test cases to ensure accuracy and reliability.

Relevant:

- The calculator will be useful for students, professionals, and anyone who needs to perform quick and simple calculations in their daily work.
- The calculator will help improve the user's knowledge of programming concepts and mathematical calculations.
- The calculator will provide a quick and easy solution to perform scientific calculations without the need for a physical calculator.

Time-bound:

- The calculator will be developed and tested within a specified time frame of 7 weeks.
- The calculator will be ready for user testing and feedback within 8 weeks.
- The calculator will be finalized and ready for release within 11 weeks.

Desired Requirements & Constraints

Functional requirements:

- Perform arithmetic operations like addition, subtraction, multiplication, and division.
- Perform scientific operations like square root, exponential, logarithm, and power.
- Perform trigonometric operations like sine, cosine, and tangent.
- Provide a Command Line Interface (CLI) for the user to input their calculations and receive output.
- Handle edge cases like division by zero and invalid input.

Non-functional requirements:

- The calculator should be fast and accurate in its calculations.
- The calculator should be easy to use and intuitive for the user.
- The calculator should be reliable and have a low error rate.
- The calculator should be secure and prevent unauthorized access to the system.

- The calculator should be compatible with different operating systems Like windows & Linux.

Constraints:

- The calculator should be implemented using the C language.
- The calculator should have a small memory footprint and be efficient in its resource usage.
- The calculator should be delivered within a reasonable time frame and budget.
- The calculator should work on windows, Linux & MacOS Platforms.

Detail requirements

High-level Requirements:

ID	Description	Category	status
H001	A user should be able to perform arithmetic operations	Technical	Completed
H002	A user should be able to perform scientific operations	Technical	Completed
H003	A user should be able to perform Trigonometric operations	Technical	Completed

Low-Level Requirements:

ID	Description	HLR ID	Status
L001	When a user performs the ' + ' operation the result should be the sum of two numbers	H001	Completed
L002	When a user performs the ' - ' operation the result should be the subtraction of two numbers	H002	Completed
L003	When a user performs the ' * ' operation the result should be the multiplication of two numbers	H003	Completed
L004	When a user performs the ' / ' operation the result should be a division of two numbers	H004	Completed
L005	When a user performs the ' LOG ' operation the user should get the desired result of N number	H005	Completed
L006	When a user performs the ' POWER ' operation the user should get the desired result of (n^n) numbers	H006	Completed
L007	When a user performs the ' SQUARE ROOT operation the user should get the desired result of N number	H007	Completed
L008	When a user performs the ' SIN ' operation the user should get the desired result of N number	H008	Completed
L009	When a user performs the ' COS ' operation the user should get the desired result of N number	H009	Completed
L010	When a user performs the ' TAN ' operation the user should get the desired result of N number	H010	Completed.

Choosing the Appropriate SDLC Model for the Project

The Agile methodology Scrum framework is a great fit for this project, as it emphasizes flexibility, collaboration, and incremental development. The project can be divided into sprints, which are time-boxed iterations lasting 1-4 weeks each. During each sprint, the development team works on a specific set of features and requirements, using daily stand-up meetings to stay on track and address any issues.

The Scrum framework encourages frequent testing and feedback, allowing for continuous delivery of a working product. The product backlog is prioritized by the product owner, and the team works to complete the highest priority items during each sprint. Regular sprint review meetings are held with stakeholders to demonstrate progress and gather feedback.

Scrum also provides regular retrospective meetings, where the team reflects on the previous sprint and identifies opportunities for improvement. This helps to continuously optimize the development process and ensure that the team is working efficiently.

Overall, the Scrum SDLC framework provides a flexible and adaptable approach to software development that allows the team to respond quickly to changes and issues. It encourages collaboration and communication, helping to ensure that the final product meets the needs of stakeholders and users alike.

Cost & Time Estimation

Time Estimation:

Task	Optimistic	Most Likely	Pessimistic
Conduct initial research and analysis	1 week	1.5 weeks	2 weeks
Define project requirements and scope	1 week	1.5 weeks	2 weeks
Create a High Level & Low-Level Design	1 week	1 week	1.5 weeks
Command Line interface design	1 week	1 week	1.5 weeks
Setup development environment and tools	1 week	1.5 weeks	2 weeks
Develop software components	2 weeks	2.5 weeks	3 weeks
Develop test cases and test plans	1 week	1 week	1.5 weeks
Perform testing and bug fixing	1 week	1 week	1.5 weeks
Conduct final testing and bug fixing.	1 week	1 week	1.5 weeks
Documentation	0.5 week	0.5 week	1 week

- Optimistic: 10 weeks
- Most Likely: 12.5 weeks
- Pessimistic: 17 weeks

PERT Estimate = (Optimistic + 4 x Most Likely + Pessimistic) / 6

Plugging in the values we have:

PERT Estimate = (10 + 4 x 12.5 + 17) / 6

PERT Estimate = 13.08 weeks

Therefore, the PERT estimate for this project is 13.08 weeks.

Cost Estimation:

This is a sample cost estimation for this project.

Size of the project: 1000 lines of code (LOC) that needs to be written.

The complexity of the project: The complexity of the software being developed is medium.

Development mode: The development mode can be organic.

$$\text{Effort} = a * (\text{KLOC})^b * (c1 + c2 * (\text{Cplx}))^d * (e1 + e2 * (\text{Size}))^f$$

where:

- $a = 2.4$ (constant for organic development mode)
- $b = 1.05$ (constant for organic development mode)
- $c1 = 0.75$ (constant for organic development mode)
- $c2 = 0.75$ (constant for organic development mode)
- $d = 0.26$ (constant for organic development mode)
- $e1 = 1.0$ (constant for organic development mode)
- $e2 = 0.0$ (constant for organic development mode)
- $f = 0.38 + 0.2 * (d - 0.91)$ (constant for organic development mode)

Given:

- $\text{KLOC} = 1$ (project size in thousands of lines of code)
- $\text{Cplx} = 1.17$ (complexity factor for a medium complexity project)

$$\text{Effort} = 2.4 * (1)^{1.05} * (0.75 + 0.75 * (1.17))^{0.26} * (1.0 + 0.0 * (1000))^f$$

$$\text{Effort} = 2.4 * 1.116 * 1.0525 * (1.38)^{0.26} * 1.0^f$$

$$\text{Effort} = 3.63 \text{ person-months}$$

The cost of the project can be estimated using the average hourly rate of \$100 for software development services and the number of person-months required:

Cost = Effort * PM Rate

Assuming a PM rate of \$8,000, the cost estimation using COCOMO II is:

Cost = 3.63 * \$8,000 = \$29,040

Planning & Scheduling Tools

Planning:

Jira Board is a Project Management tool, which is helpful to know the status of the project and track it. The whole board is divided into three sections mainly, they are:

- 1.To do
- 2.InProgress
- 3. Done

1. To-do tasks refer to the tasks that are planned to be started in the upcoming sprint or iteration. These tasks are usually identified during the planning phase and are assigned to the relevant team members for execution. In the Jira board provided, there are two tasks that are currently in the To-do column, indicating that they are yet to be started.

2. In-progress tasks refer to the tasks that are currently being worked on by the team members. These tasks are usually in progress and are being actively worked upon to achieve the desired outcome. In the Jira board provided, there are five tasks that are currently in the InProgress column, indicating that the team is actively working on these tasks.

3. Done tasks refer to the tasks that have been completed successfully by the team. These tasks have achieved the desired outcome and have been approved by the relevant stakeholders in this project we are the stakeholders, so we assume that is being achieved. In the Jira board provided, there are five tasks that are currently in the Done or Completed column, indicating that the team has successfully completed these tasks.

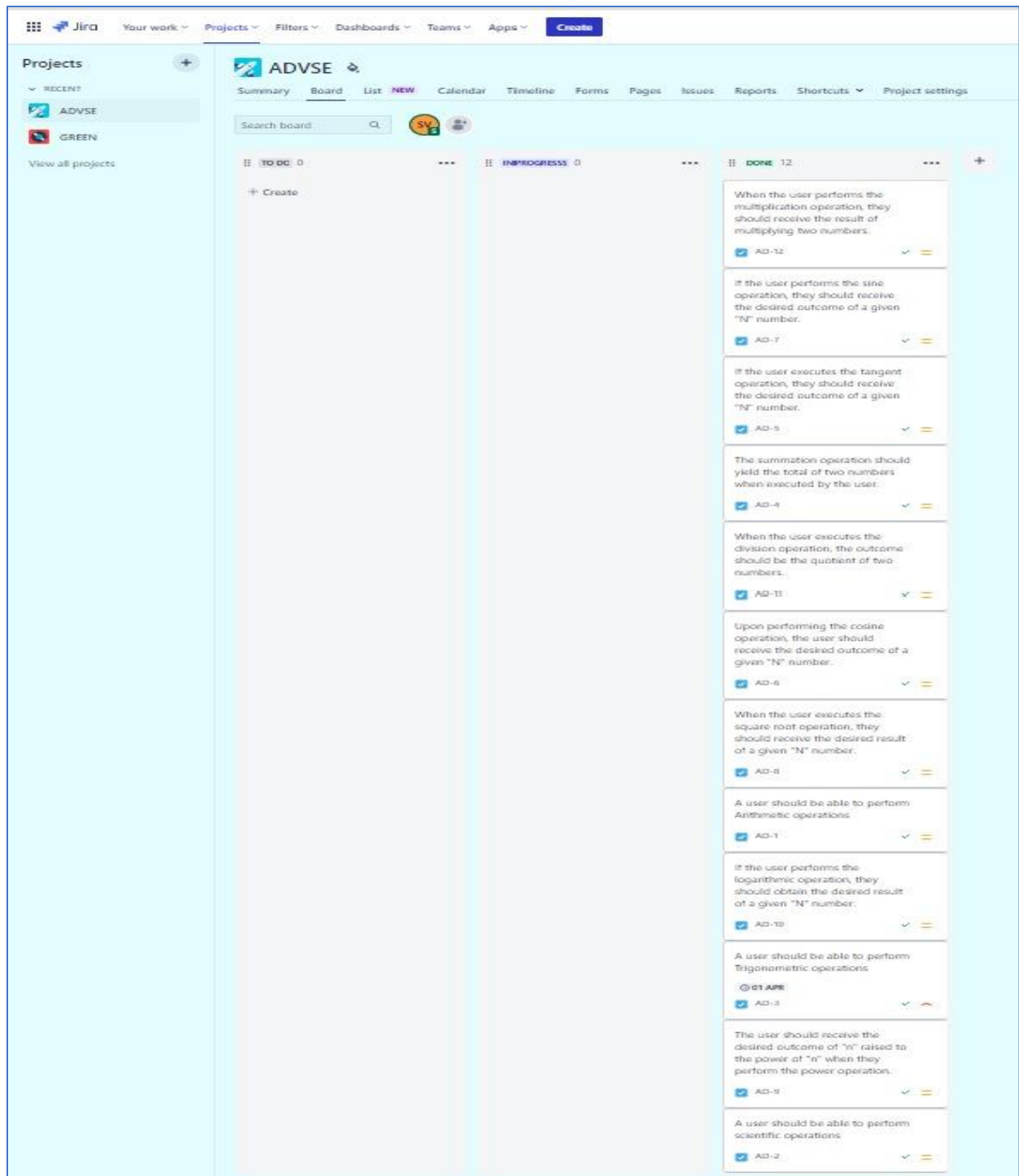


Fig Jira Board displaying the task details.

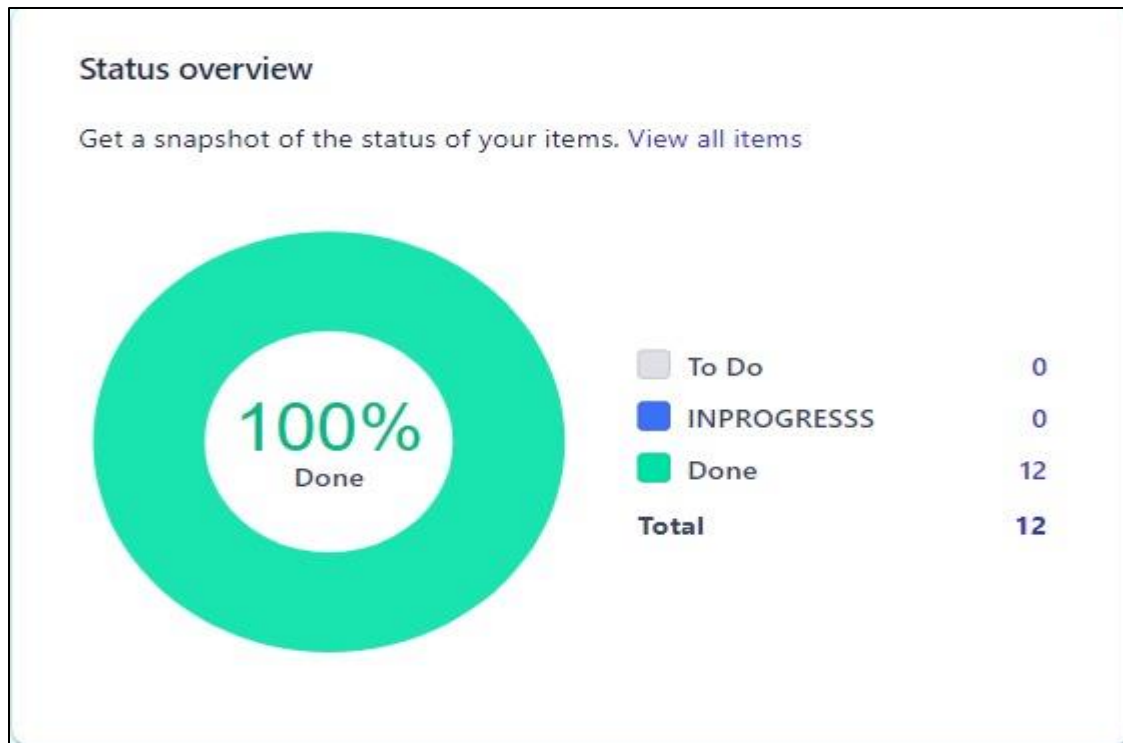


Fig Displaying Status overview.

Scheduling:

In our project, we have utilized Teams for conducting daily Scrum meetings, as well as for Sprint planning and review sessions, to ensure effective communication and collaboration among team members.

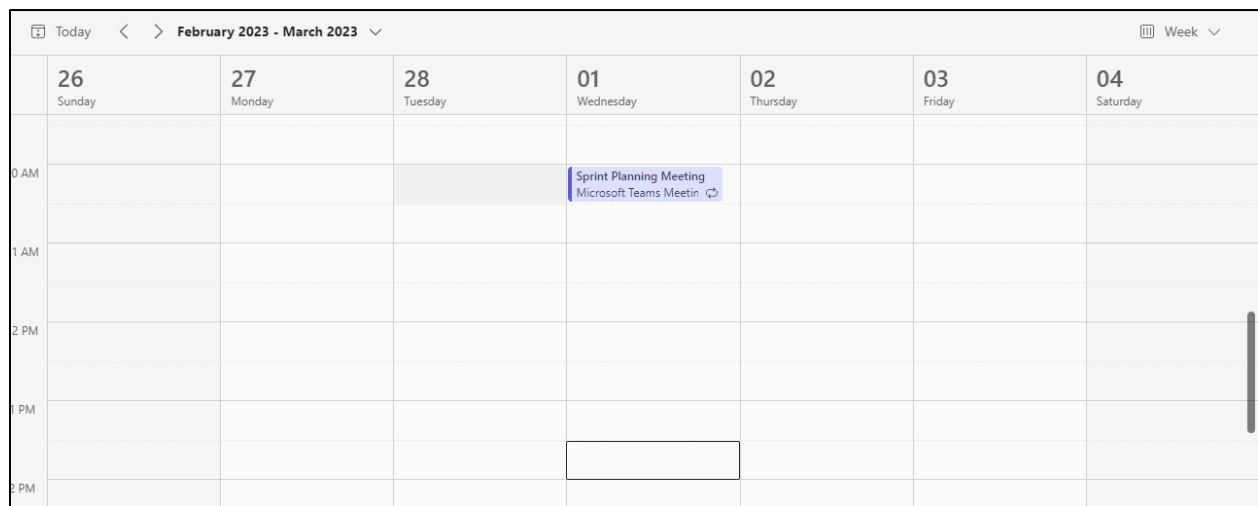


Fig Displays Meetings scheduled for the project.

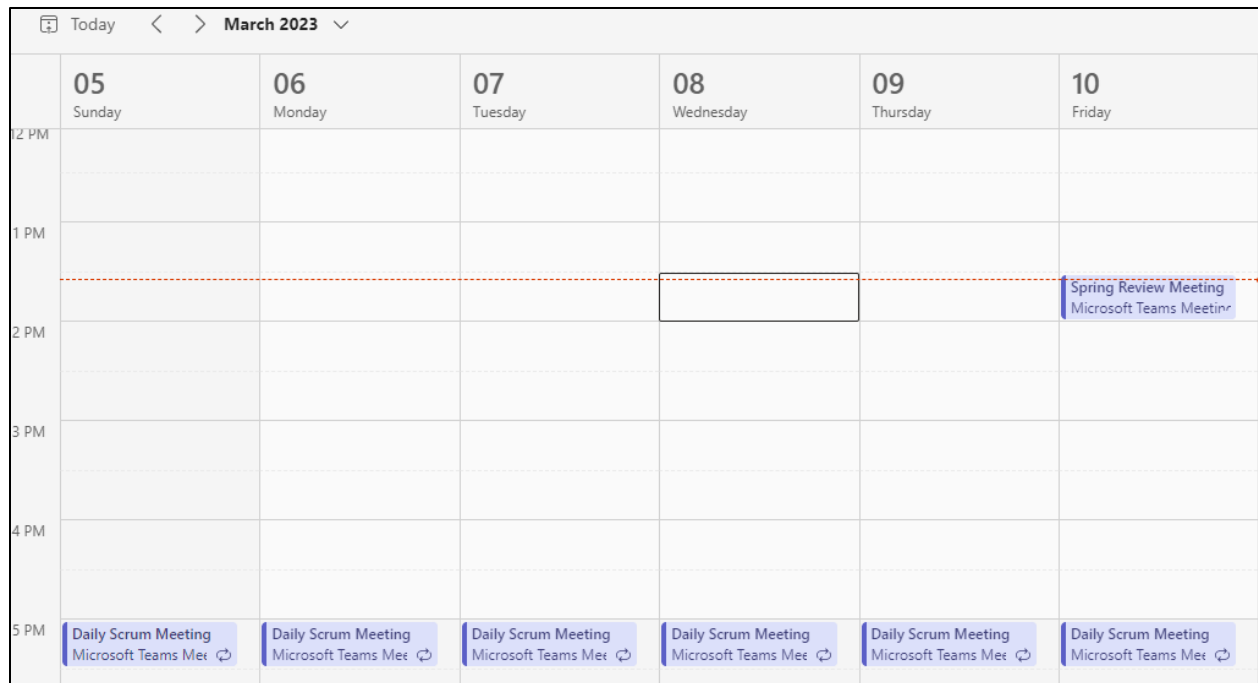


Fig Displays Meetings scheduled for the project.

Analysis of the times you were behind or ahead of your ideal schedule.

While working on the scientific calculator project, we found that we were ahead of schedule during the designing phase. This was due to the clear and concise requirements gathered during the initial stages of the project, which allowed our team to create high-level design specifications and low-level design specifications with ease. We were able to create accurate and comprehensive behavioral diagrams, making the implementation phase much smoother.

In addition, we were also ahead of schedule during the implementation phase for certain features of the calculator. Our team was able to work efficiently and effectively, translating the design specifications into working code with minimal errors or defects. This allowed us to complete the implementation of some features ahead of schedule, providing us with extra time to test and refine them. Our team was also able to perform more rigorous testing on these features, ensuring that they functioned correctly and met the requirements specified.

Overall, being ahead of schedule during the design and implementation phases was a positive indicator of our team's capabilities and efficient working practices. It

allowed us to complete the project ahead of schedule while still maintaining a high level of quality and meet all the requirements.

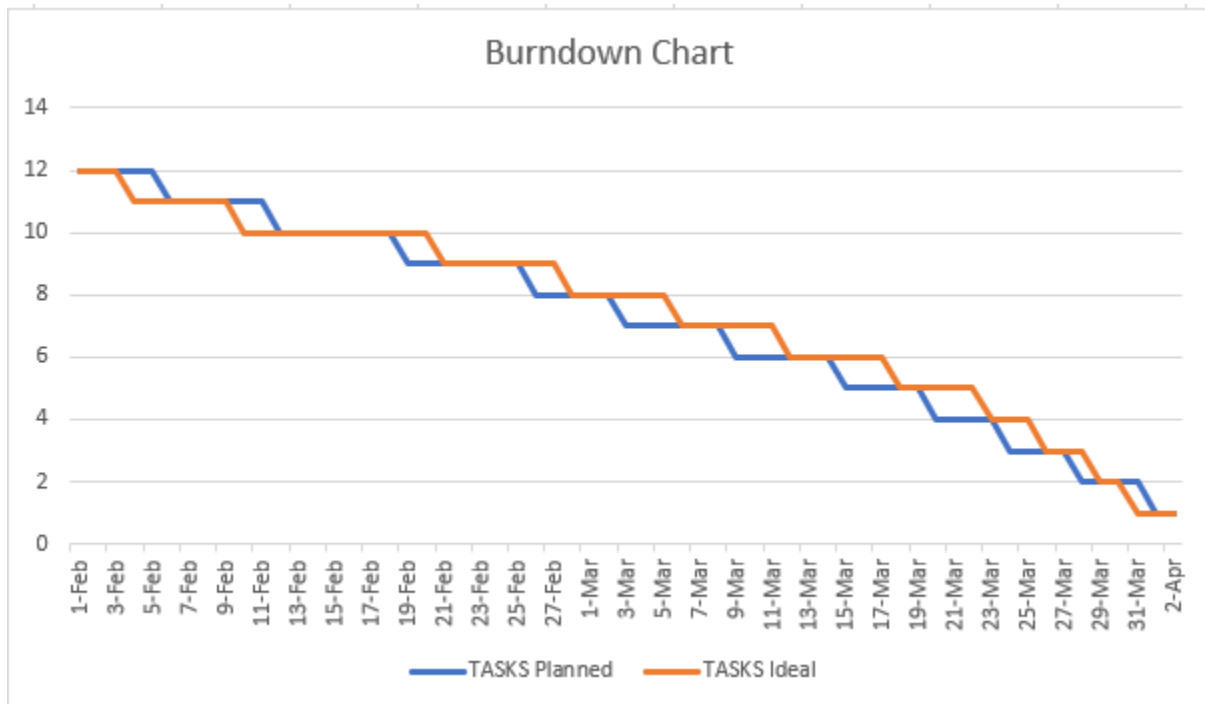


Fig Burndown chart

High-Level & Low-Design

High-Level Design:

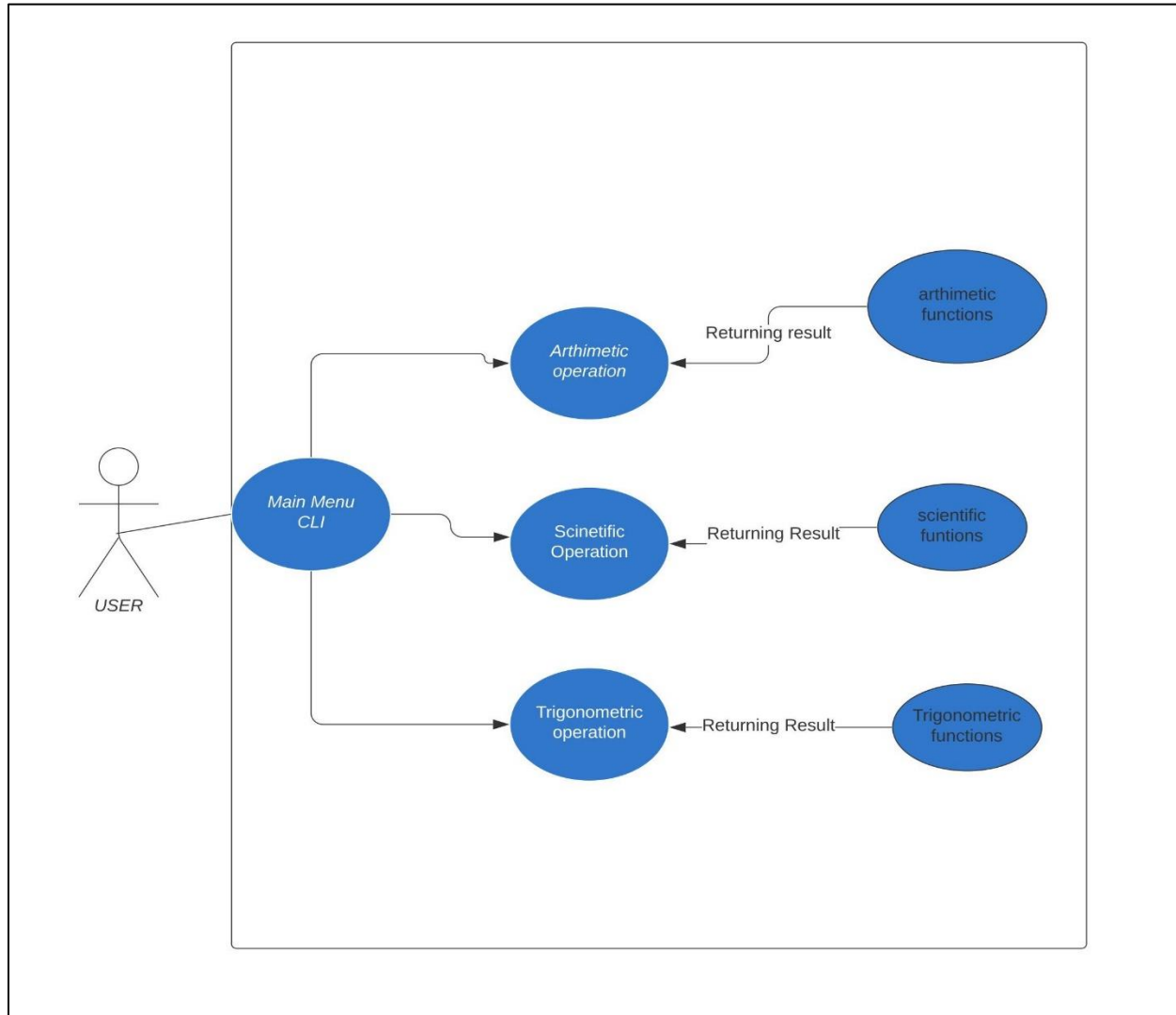


Fig High-Level Design

- The project includes a high-level design diagram, which illustrates the overall structure of the project. This diagram provides an overview of the project's architecture and how different components are connected.

- When the user accesses the Main Menu in the Command Line Interface (CLI), they are presented with options to select from arithmetic, scientific, and trigonometric functions. This allows the user to choose the type of mathematical operation they want to perform.
- If the user selects the arithmetic function, they are presented with options for addition, subtraction, and multiplication. This allows the user to perform basic arithmetic operations using the CLI.
- If the user selects the scientific function, they are presented with options for functions such as square root, logarithm, and exponential. This allows the user to perform more advanced mathematical operations using the CLI.
- If the user selects the trigonometric function, they are presented with options for functions such as sine, cosine, and tangent. This allows the user to perform trigonometric calculations using the CLI.
- The user can exit the program at any time by selecting the exit option, which will terminate the program.

Low-Level Design:

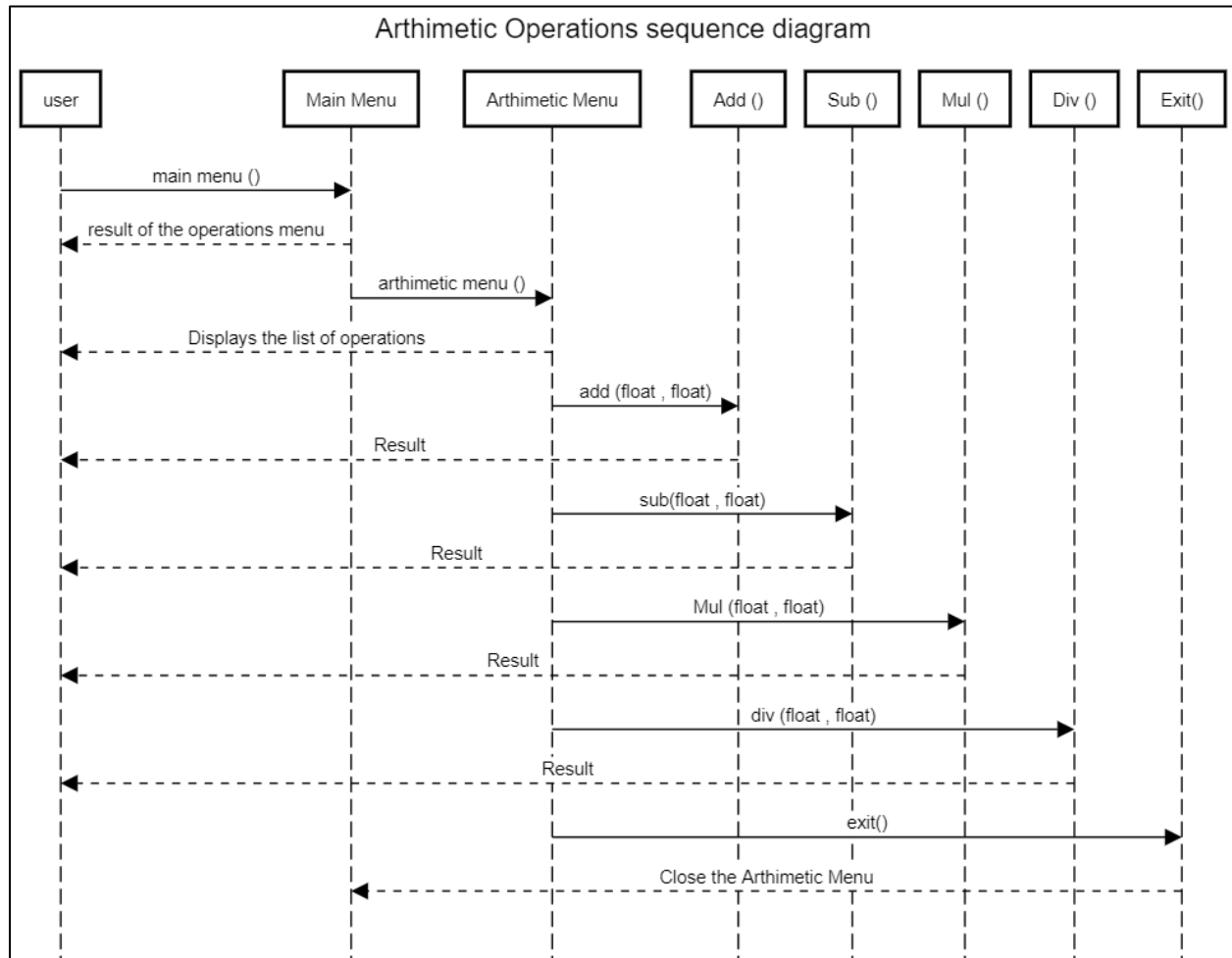


Fig Low-Level Design for Arithmetic Operation Module.

- The system presents a main menu to the user, which includes a comprehensive list of available operations. This allows the user to easily navigate the system and select the desired operation.
- If the user selects the arithmetic menu, the system presents a set of options for addition, subtraction, and multiplication. This makes it clear to the user what operations are available within the arithmetic menu.
- Once the user selects a specific operation, the system validates and processes the input parameters. This ensures that the inputs are correct and that the operation can be performed correctly.

- If the user selects the exit option, the system directs them back to the arithmetic menu for further operations. This allows the user to easily perform multiple operations without having to go back to the main menu each time.
- Finally, the system operates with a high level of professionalism and efficiency.

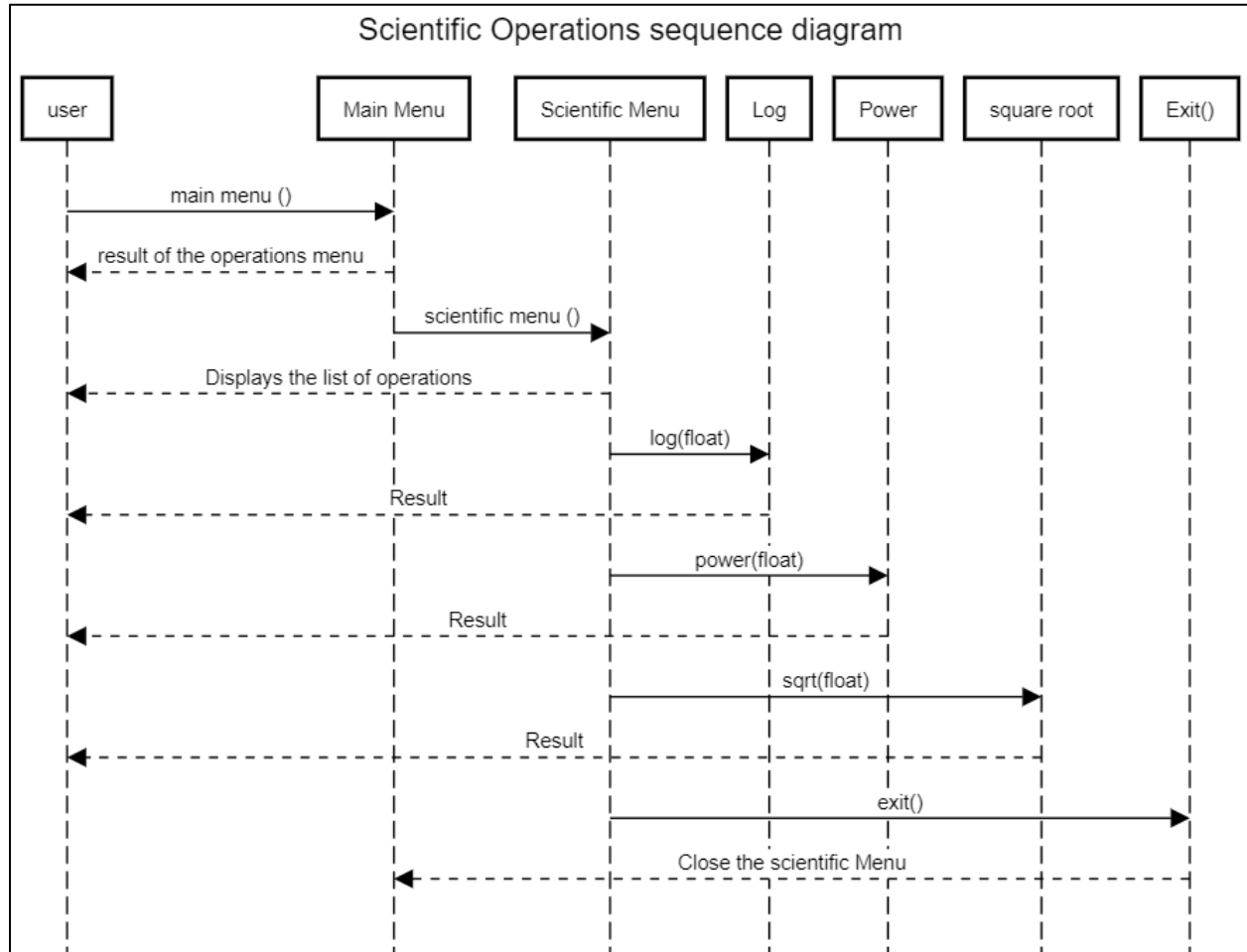


Fig Low-Level Design for Scientific operations

- The main menu provides an extensive list of available operations to the user. This allows the user to choose the operation they need, without having to navigate through multiple sub-menus.
- Selecting the scientific menu displays options for logarithmic, power, and square root operations. This allows the user to perform more complex mathematical operations that are commonly used in scientific calculations.

- Upon selecting a specific operation, the system validates the input parameters to ensure that they are correct and appropriate for the operation. This ensures that the operation is performed correctly and that the output is accurate.
- After validating the input parameters, the system processes them accordingly. This involves applying the appropriate mathematical formula or algorithm to calculate the desired result.
- Selecting the exit option directs the user back to the scientific menu for further operations. This allows the user to perform multiple operations within the scientific menu without having to go back to the main menu.

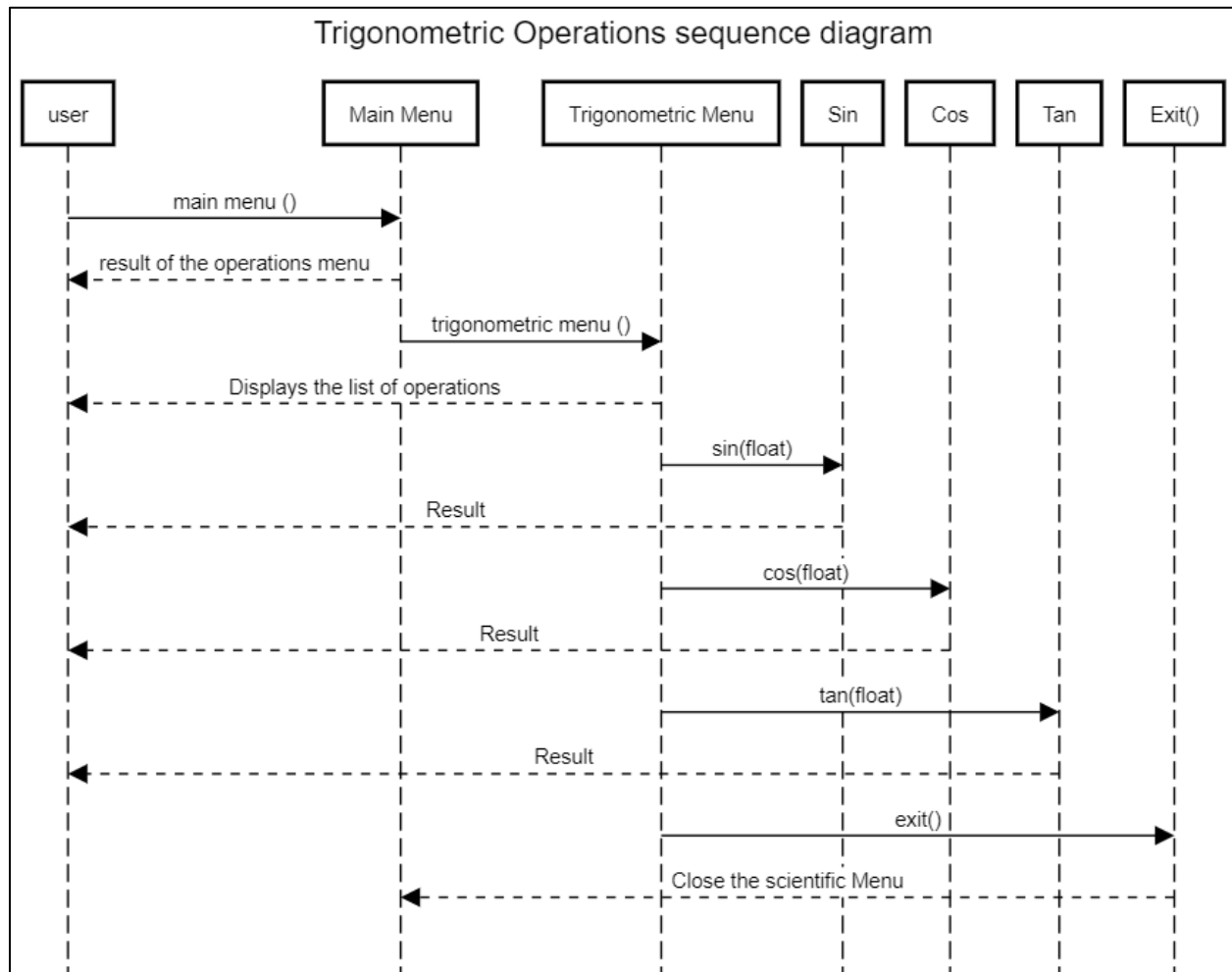


Fig Low-Level Design for Trigonometric operations

- Upon visiting the main menu, the user is presented with a comprehensive list of available operations. This allows the user to choose the operation they need without having to navigate through multiple sub-menus.
- Selecting the trigonometric menu displays options for sine, cosine, and tangent operations. These trigonometric functions are commonly used in mathematics, physics, and engineering.
- Upon selecting a specific operation, the system validates and processes the input parameters. The system checks to make sure that the input parameters are appropriate for the selected operation. For example, the input parameters for a sine function must be in radians.
- After validating the input parameters, the system processes them to perform the desired operation. For example, if the user selects the cosine function, the system applies the mathematical formula for cosine to the input parameters to calculate the result.
- Selecting the exit option directs the user back to the trigonometric menu for further operations. This allows the user to perform multiple operations within the trigonometric menu without having to go back to the main menu.

Implementation & Sample Output:

inc	changed header
src	Update func_trigo_menu.c
test	added total testcases
unity	Created Folder Structure
LICENSE	Created Folder Structure
Makefile	commenting the make file
Readme.md	Created Folder Structure
project_main.c	added testing

Fig Project folder structure for implementation

Folder Structure of the project:

Folder	description
inc	All header files
src	Main source code for calculator
test	All source code and data for testing purposes
build	Build output (included in git)

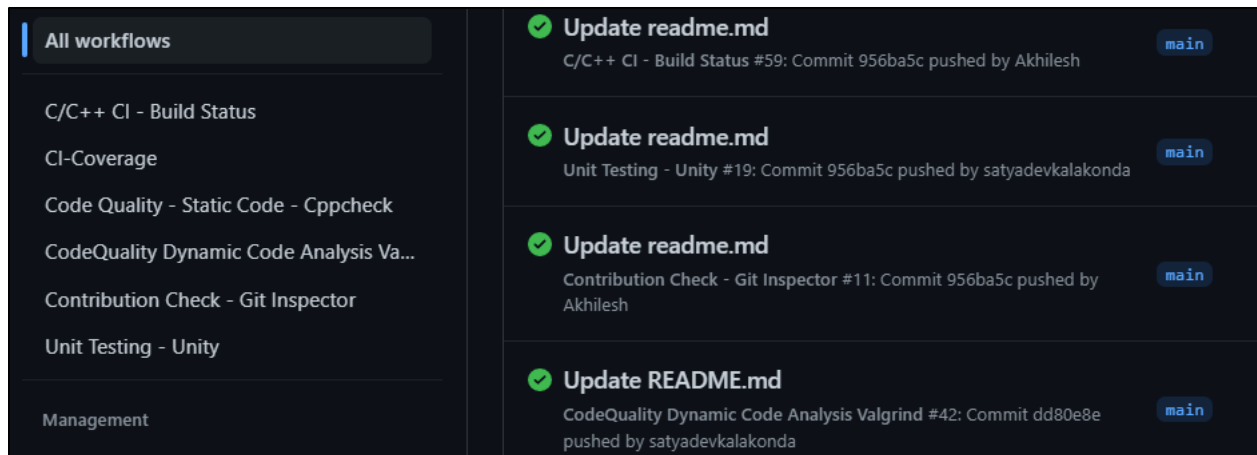


Fig CI/CD Build Pipeline using GitHub Actions.

- In this project we have used GitHub actions Pipeline for CI/CD, which allows for continuous integration and delivery of the project.
- By using GitHub actions Pipeline, the project benefits from automated testing and deployment, which helps to ensure that code changes are integrated smoothly and delivered efficiently.

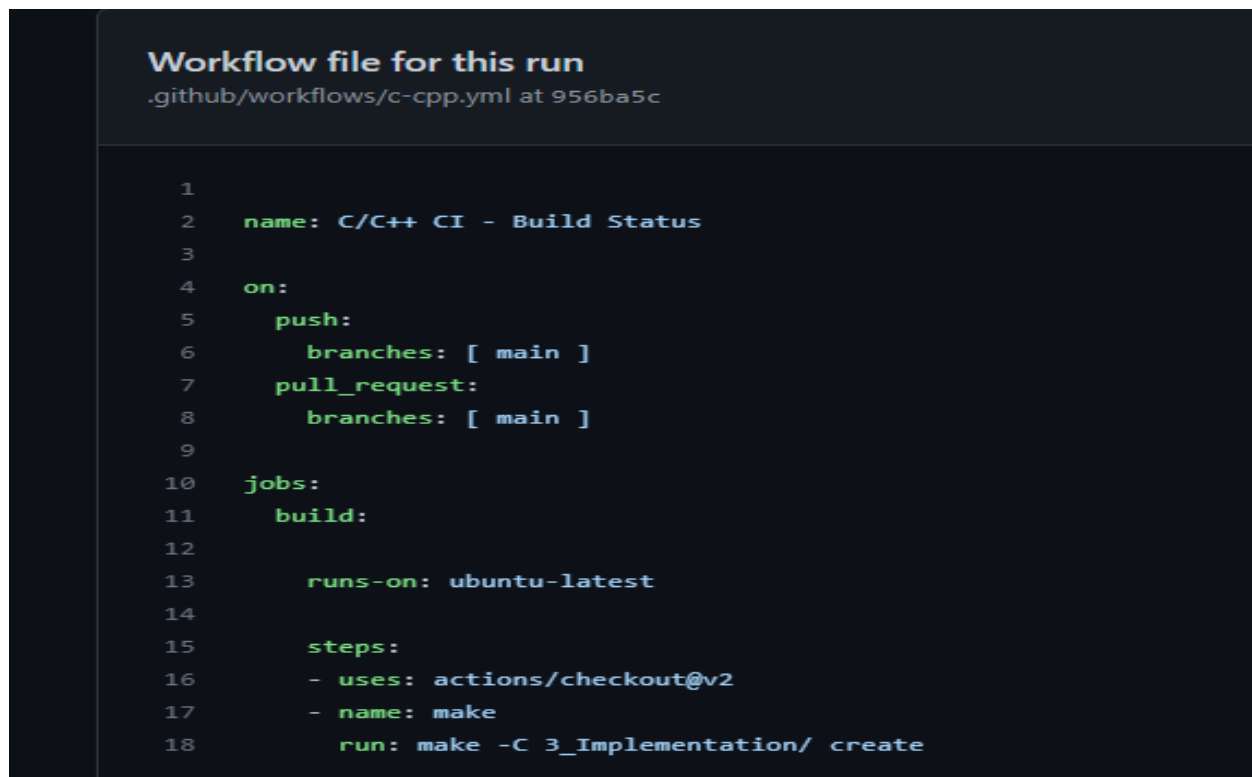
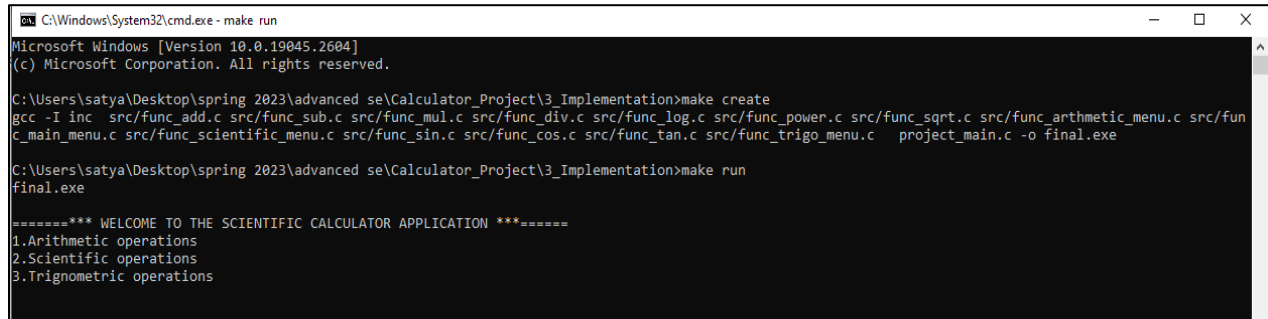


Fig Sample Workflow code used for C/C++ Build

- Above fig shows the code used in the workflow for the C Build.
- The workflow is designed to automate the build process for C code changes, ensuring that the code is compiled correctly and efficiently.



```

C:\Windows\System32\cmd.exe - make run
Microsoft Windows [Version 10.0.19045.2604]
(c) Microsoft Corporation. All rights reserved.

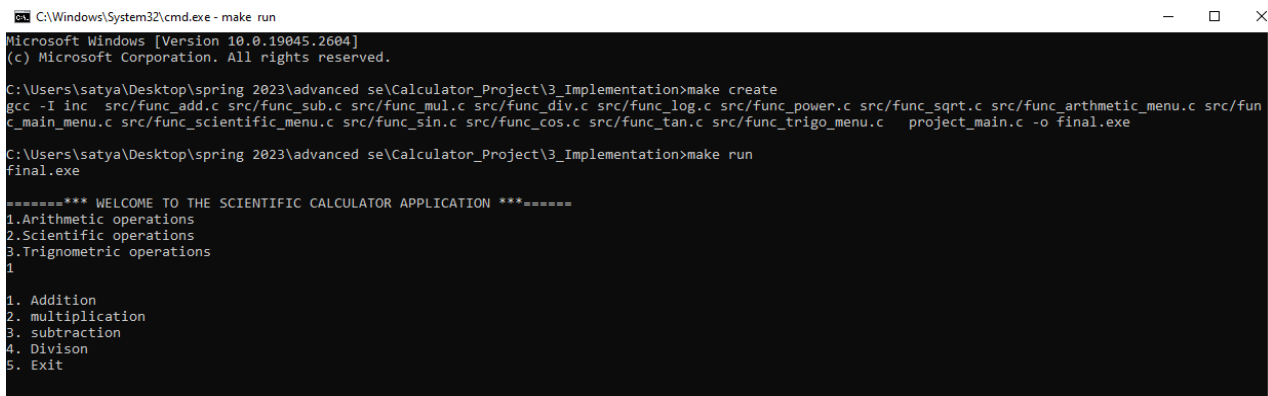
C:\Users\satya\Desktop\spring 2023\advanced se\Calculator_Project\3_Implementation>make create
gcc -I inc src/func_add.c src/func_sub.c src/func_mul.c src/func_div.c src/func_log.c src/func_power.c src/func_sqrt.c src/func_arithmetic_menu.c src/func_main_menu.c src/func_scientific_menu.c src/func_sin.c src/func_cos.c src/func_tan.c src/func_trigo_menu.c project_main.c -o final.exe

C:\Users\satya\Desktop\spring 2023\advanced se\Calculator_Project\3_Implementation>make run
final.exe

***** WELCOME TO THE SCIENTIFIC CALCULATOR APPLICATION *****
1.Arithmetic operations
2.Scientific operations
3.Trigonometric operations

```

Fig Sample Output of the code displaying Main Menu.



```

C:\Windows\System32\cmd.exe - make run
Microsoft Windows [Version 10.0.19045.2604]
(c) Microsoft Corporation. All rights reserved.

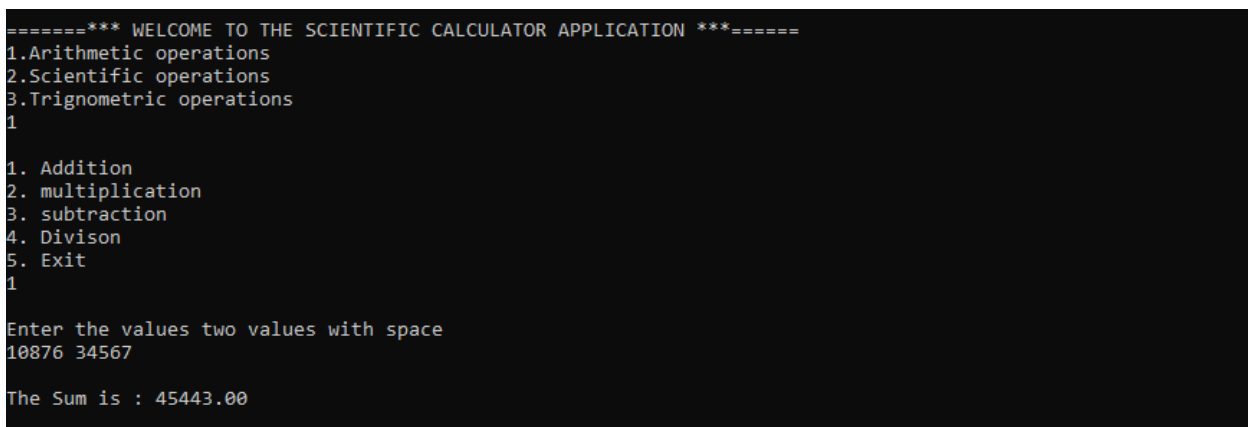
C:\Users\satya\Desktop\spring 2023\advanced se\Calculator_Project\3_Implementation>make create
gcc -I inc src/func_add.c src/func_sub.c src/func_mul.c src/func_div.c src/func_log.c src/func_power.c src/func_sqrt.c src/func_arithmetic_menu.c src/func_main_menu.c src/func_scientific_menu.c src/func_sin.c src/func_cos.c src/func_tan.c src/func_trigo_menu.c project_main.c -o final.exe

C:\Users\satya\Desktop\spring 2023\advanced se\Calculator_Project\3_Implementation>make run
final.exe

***** WELCOME TO THE SCIENTIFIC CALCULATOR APPLICATION *****
1.Arithmetic operations
2.Scientific operations
3.Trigonometric operations
1
1. Addition
2. multiplication
3. subtraction
4. Divison
5. Exit

```

Fig Sample output of the code Displaying List of Operations available in Arithmetic Menu



```

***** WELCOME TO THE SCIENTIFIC CALCULATOR APPLICATION *****
1.Arithmetic operations
2.Scientific operations
3.Trigonometric operations
1
1. Addition
2. multiplication
3. subtraction
4. Divison
5. Exit
1
Enter the values two values with space
10876 34567
The Sum is : 45443.00

```

Fig Sample Output of the Addition function with two values input

```

=====*** WELCOME TO THE SCIENTIFIC CALCULATOR APPLICATION ***=====
1.Arithmetic operations
2.Scientific operations
3.Trignometric operations
1
1. Addition
2. multiplication
3. subtraction
4. Divison
5. Exit
2
Enter the values two values with space
17896 23456
The Multiplication is: 419768576.00

```

Fig Sample output of Multiplication function with 2 values as input

```

=====*** WELCOME TO THE SCIENTIFIC CALCULATOR APPLICATION ***=====
1.Arithmetic operations
2.Scientific operations
3.Trignometric operations
1
1. Addition
2. multiplication
3. subtraction
4. Divison
5. Exit
3
Enter the values two values with space
50 100
The subtraction is: -50.00

```

Fig Sample output of Subtraction function with 2 values as input

```

=====*** WELCOME TO THE SCIENTIFIC CALCULATOR APPLICATION ***=====
1.Arithmetic operations
2.Scientific operations
3.Trignometric operations
1
1. Addition
2. multiplication
3. subtraction
4. Divison
5. Exit
4
Enter the values two values with space
405678 2
The Divison is: 202839.00

```

Fig Sample output of Division function with 2 values as input

```
C:\Windows\System32\cmd.exe - make run
C:\Users\satya\Desktop\spring_2023\advanced se\Calculator_Project\3_Implementation>make run
final.exe

=====*** WELCOME TO THE SCIENTIFIC CALCULATOR APPLICATION ***=====
1.Arithmetic operations
2.Scientific operations
3.Trignometric operations
2

1. Log10
2. Power
3. square-root
4. Exit
1
```

Fig Sample output of the code Displaying List of operations available in Scientific Menu.

```
=====*** WELCOME TO THE SCIENTIFIC CALCULATOR APPLICATION ***=====
1.Arithmetic operations
2.Scientific operations
3.Trignometric operations
2

1. Log10
2. Power
3. square-root
4. Exit
1

Enter the value to calculate :
1067
3.03
```

Fig Sample output of Log10 function with 1 value as input

```
=====*** WELCOME TO THE SCIENTIFIC CALCULATOR APPLICATION ***=====
1.Arithmetic operations
2.Scientific operations
3.Trignometric operations
2

1. Log10
2. Power
3. square-root
4. Exit
2

Enter the two values with space
12 34
492223530701160540000000000000000000000000.00
```

Fig Sample output of power function with 2 values as input

```

=====*** WELCOME TO THE SCIENTIFIC CALCULATOR APPLICATION ***=====
1.Arithmetic operations
2.Scientific operations
3.Trignometric operations
2
1. Log10
2. Power
3. square-root
4. Exit
3
Enter the value to calculate :
234
15.30

```

Fig Sample output of square root function with 1 value as input

```

C:\Windows\System32\cmd.exe - make run
C:\Users\satya\Desktop\spring 2023\advanced se\Calculator_Project\3_Implementation>make run
final.exe
=====*** WELCOME TO THE SCIENTIFIC CALCULATOR APPLICATION ***=====
1.Arithmetic operations
2.Scientific operations
3.Trignometric operations
3
1. Sin
2. Cos
3. Tan
4. Exit

```

Fig Sample output of the code Displaying a list of operations available in the Trigonometric Menu

```

=====*** WELCOME TO THE SCIENTIFIC CALCULATOR APPLICATION ***=====
1.Arithmetic operations
2.Scientific operations
3.Trignometric operations
3
1. Sin
2. Cos
3. Tan
4. Exit
1
Enter the value
12
The Sin value is : 0.21

```

Fig Sample output of sin function with 1 value as input.

```

=====*** WELCOME TO THE SCIENTIFIC CALCULATOR APPLICATION ***=====
1.Arithmetic operations
2.Scientific operations
3.Trignometric operations
3
1. Sin
2. Cos
3. Tan
4. Exit
2
Enter the value
34
The cos value is: 0.83
=====*** WELCOME TO THE SCIENTIFIC CALCULATOR APPLICATION ***=====

```

Fig Sample output of cos function with 1 value as input

```

=====*** WELCOME TO THE SCIENTIFIC CALCULATOR APPLICATION ***=====
1.Arithmetic operations
2.Scientific operations
3.Trignometric operations
3
1. Sin
2. Cos
3. Tan
4. Exit
3
Enter the value
56
The tan value is: 1.48
=====*** WELCOME TO THE SCIENTIFIC CALCULATOR APPLICATION ***=====

```

Fig sample output of Tan function with 1 value as input.

Hardware Requirements:

1. **Processor:** Any processor with a clock speed of 1 GHz or higher should suffice.
2. **RAM:** At least 512 MB of RAM is required to run the program efficiently.
3. **Display:** A monitor or screen capable of displaying at least 800x600 resolution is necessary.
4. **Keyboard:** A standard keyboard for inputting data is required.

5. **Storage:** At least 100 MB of free storage space to store the program files and the output data.
6. **Operating System:** The calculator program can run on any operating system, but a Windows, macOS, or Linux-based system should work fine

Software & Library Requirements:

1. **C Compiler:** To write and compile the C code for the calculator, you need a C compiler such as GCC, Clang, or mingw.
2. **Command-Line Interface:** A command-line interface (CLI) is required to run the calculator program. The CLI allows the user to interact with the program by typing commands into the terminal.
3. **Make:** Make is a build automation tool that automates the compilation and linking of the calculator program. It is helpful for managing large projects, but it is not necessary for small programs.
4. **Version Control System:** A version control system (VCS) is useful for managing and tracking changes to the calculator program's source code. Git and GitHub are used in this project.
5. **Testing Framework:** A Unity testing framework is helpful for testing the calculator program's functionality and ensuring that it produces accurate results.

High-Level & Low-Level Testing

To test the developed software, I have utilized the unit testing method, which is a widely used testing approach that focuses on testing individual components or units of a software system in isolation. For this purpose, I have used the Unity framework, which is a popular open-source framework that provides an extensive set of tools for writing and executing unit tests in a variety of programming languages.

With Unity, I have been able to create test cases for each function in the software system, specifying expected input and output values and verifying the results against the actual output of the function. This approach has allowed me to identify

any errors or bugs in the software code and make necessary corrections to ensure that the software meets the specified requirements and performs as intended.

Moreover, by utilizing the unit testing method with Unity, I have been able to automate the testing process, reducing the likelihood of human error and increasing the efficiency and effectiveness of the testing process. Overall, the combination of unit testing and the Unity framework has provided a robust and reliable means of testing the developed software and ensuring its quality and functionality.

High-Level Test Plan:

Test ID	Description	Exp I/P	Exp O/P	Actual Out
H_01	Calling Arithmetic Menu	arithmetic_menu()	TRUE	TRUE
H_02	Calling Scientific Menu	scientific_menu()	TRUE	TRUE
H_03	Calling Trigonometric Menu	trigonometric_menu()	TRUE	TRUE

Low-level Test plan:

Test ID	Description	HL ID	Exp IN	Exp OUT	Actual Out	Type Of Test
L_01	Performing Arithmetic operation '+'	H_01	(200,100)	300	300	Requirement based
L_02	Performing Arithmetic operation '+'	H_01	(20*2,-9+5)	36	36	Scenario-based

Test ID	Description	HL ID	Exp IN	Exp OUT	Actual Out	Type Of Test
L_03	Performing Arithmetic operation '+'	H_01	(-99999,9999)	0	0	Boundary based
L_04	Performing Arithmetic operation '-'	H_01	(15,5)	20	20	Requirement based
L_05	Performing Arithmetic operation '-'	H_01	(15*82,-5+28)	1207	1207	Scenario-based
L_06	Performing Arithmetic operation '-'	H_01	(-888889,77778)	11111	11111	Boundary based
L_07	Performing Arithmetic operation '**'	H_01	(159,286)	45474	45474	Requirement based
L_08	Performing Arithmetic operation '**'	H_01	(-15,-26)	390	390	Scenario-based
L_09	Performing Arithmetic operation '**'	H_01	(159999,286666)	-126667	-126667	Boundary based
L_10	Performing Arithmetic operation '/'	H_01	(3888,24)	162	In-progress	Requirement based

Test ID	Description	HL ID	Exp IN	Exp OUT	Actual Out	Type Of Test
L_11	Performing Arithmetic operation '/'	H_01	(1,0)	ERROR_DIV_ZERO	In-Progress	Scenario-based
L_12	Performing Arithmetic operation '/'	H_01	(9999,3)	3333	3333	Boundary based
L_13	Performing Scientific operation 'Log'	H_02	(8869)	3.95	To-Do	Requirement based
L_14	Performing Scientific operation 'Log'	H_02	(124)	3.95	To-Do	Scenario-based
L_15	Performing Scientific operation 'Log'	H_02	(124)	3.95	To-Do	Scenario-based
L_16	Performing Scientific operation 'sqrt'	H_02	(24)	4.09	In-progress	Requirement based
L_17	Performing Scientific operation 'sqrt'	H_02	(45*32)	37.95	In-progress	Scenario-based
L_18	Performing Scientific operation 'sqrt'	H_02	(9999)	99	In-Progress	Boundary based

Test ID	Description	HL ID	Exp IN	Exp OUT	Actual Out	Type Of Test
L_19	Performing Scientific operation ' p ower '	H_02	(5,20)	953674335 51872	953674335 51872	Requirement based
L_20	Performing Scientific operation ' p ower '	H_02	(2*2,4)	256	256	Boundary based
L_21	Performing Trigonometric operation ' S in '	H_03	(10)	0.17	0.17	Requirement based
L_22	Performing Trigonometric operation ' S in '	H_03	(1000)	-0.98	-0.98	Boundary based
L_23	Performing Trigonometric operation ' C os '	H_03	(9)	0.99	To-Do	Requirement based
L_24	Performing Trigonometric operation ' C os '	H_03	(8888)	-0.37	To-Do	Boundary based
L_25	Performing Trigonometric	H_03	(6)	0.11	0.11	Requirement based

Test ID	Description	HL ID	Exp IN	Exp OUT	Actual Out	Type Of Test
	operation 'Tan'					
L_26	Performing Trigonometric operation 'Tan'.	H_03	(5555)	-0.47	-0.47	Boundary based

```

C:\Users\satya\Desktop\spring_2023\advanced se\Calculator_Project\3_Implementation>make create_test
gcc -I inc -I unity src/func_add.c src/func_sub.c src/func_mul.c src/func_div.c src/func_log.c src/func_power.c src/func_sqrt.c src/func_arithmetic_menu.c src/func_main_menu.c src/func_scientific_menu.c src/func_sin.c src/func_cos.c src/func_tan.c src/func_trigo_menu.c unity/unity.c test/test_project.c -o final_test.exe

C:\Users\satya\Desktop\spring_2023\advanced se\Calculator_Project\3_Implementation>make run_test
final_test.exe
test/test_project.c:263:test_add:PASS
test/test_project.c:264:test_sub:PASS
test/test_project.c:265:test_mul:PASS
test/test_project.c:266:test_div:PASS
test/test_project.c:267:test_log:PASS
test/test_project.c:268:test_sqrt:PASS
test/test_project.c:269:test_sin:PASS
test/test_project.c:270:test_cos:PASS
test/test_project.c:271:test_tan:PASS

-----
9 Tests 0 Failures 0 Ignored
OK

C:\Users\satya\Desktop\spring_2023\advanced se\Calculator_Project\3_Implementation>

```

Fig shows all tests passed for high-level & low-level test plans.

Challenges Faced and How Was It Overcome

No	Challenge	Solution
01	Resource unavailable while using system commands	Uninstalled MacAfee anti-virus
02	Make the file not working even after following all steps	Renamed to Mingw/bin/mingw32-make.exe to Mingw/bin/make.exe

Each team member's role

Satyadev Kalakonda:

- Conducted a SWOT analysis.
- Defined low-level requirements.
- Developed a 4W's and 1H analysis.
- Used the PERT tool to analyze project dependencies and potential delays.
- Implemented Scientific operations and was involved in writing various low-level & high-level test plans and tested using the unity framework.
- Also involved in checking code coverage using the gcov tool, and part of analyzing the code developed Val grind tool.
- Combinedly worked on developing arithmetic operations and was involved in writing various low-level & high-level test plans.
- Developed CI/CD workflows yaml configuration files for GitHub actions.
- Assisted in developing and commenting for Make files.
- Written formal comments in the code for easier understanding using the Doxygen tool for Scientific & Arithmetic operations.
- Scheduled Meetings for daily scrum calls using teams' software.
- Involved in developing behavioral diagrams for Scientific & Arithmetic operations.
- Involved in developing project documents.

Sai Akhilesh:

- Defined high-level requirements.
- Assisted with SWOT analysis.
- Developed a Gantt chart to track the project timeline.
- Created a JIRA board for tracking project tasks.
- Combinedly worked on developing arithmetic operations and was involved in writing various low-level & high-level test plans.
- Developed Make file for easier access and building the code through CI/CD pipeline.
- Implemented Trigonometric operations and was involved in writing various low-level & high-level test plans and tested using the unity framework.
- Assisted in developing yaml configuration files.
- Written formal comments in the code for easier understanding of the Doxygen tool for Trigonometric operations.
- Scheduled meeting for sprint reviews and spring planning.

- Developed Behavioral diagrams for trigonometric operations & assisted in developing Arithmetic operations.
- Involved in developing Project documentation.

Pair Programming: Pair Programming is the concept of guiding a programmer while writing code. We implemented this practice of Extreme Programming, as it helps to take on bugs that were ignored by the programmer.

Conclusion

This project has been a fantastic opportunity for us to learn and apply various software development principles, tools, and methodologies. We were able to gain practical experience in utilizing different planning and scheduling tools, such as Gantt charts, Trello boards, and Kanban boards. Additionally, we learned how to use Git and GitHub effectively for version control and collaborative work, which helped us develop a solid understanding of CI/CD workflows - essential for modern software development practices.

Moreover, we gained a thorough understanding of software development cycles, including Agile and Waterfall, and experienced the benefits and limitations of each approach. This enabled us to identify and mitigate potential risks and challenges while effectively managing our resources and timelines.

Furthermore, we also developed a project with multiple files that could be reused across different projects, thus gaining an understanding of modularization and reusability concepts. We implemented unit and integration testing to ensure the quality and functionality of our code and gained knowledge of code coverage and Doxygen, enabling us to document and test our code more effectively.

Overall, this project has provided us with a comprehensive understanding of software development principles, methodologies, and tools. It has equipped us with essential skills for future software development projects, enabling us to confidently approach and manage them.