./

GENESIS - Learning Outcome & Mini-project Summary Report



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **To be Approved** | **Remarks/Revision Details** |
| 1. |  | Nitin N Shetty |  |  |  |
|  |  |  |  |  |  |

**Details**

Contents

[Contents 3](#_Toc69548708)

[1.0 Miniproject -1 SDLC [Team] 5](#_Toc69548709)

[1.1 Modules used 5](#_Toc69548710)

[1.2 Project Title 5](#_Toc69548711)

[1.3 Topic and Subtopics 5](#_Toc69548712)

[1.4 Objectives & Requirements 6](#_Toc69548713)

[1.4.1 High Level Requirement 6](#_Toc69548714)

[1.4.2 Low Level Requirement 8](#_Toc69548715)

[10](#_Toc69548716)

[1.5 Design 13](#_Toc69548717)

[1.6 Structural Diagram 13](#_Toc69548718)

[1.6.1 HLR of Calculator 13](#_Toc69548719)

[1.6.2 LLR Diagram(Exponential) 13](#_Toc69548720)

[1.6.3 LLR Diagram (Calculator) 14](#_Toc69548721)

[1.6.4 HLR Diagram(Logarithm) 14](#_Toc69548722)

[1.6.6 LLR Diagram (Complex Number) 15](#_Toc69548723)

[1.6.7 LLR Diagram(Arithmetic Operations) 16](#_Toc69548724)

[1.7 Behavioral Diagram 17](#_Toc69548725)

[1.7.1 HLR Calculator Diagram (Flow control) 17](#_Toc69548726)

[1.7.2 HLR Calculator Diagram (Sequence Diagram) 18](#_Toc69548727)

[1.7.3 LLR Diagram (Factorial) 19](#_Toc69548728)

[1.7.4 LLR Diagram (Logarithm) 19](#_Toc69548729)

[1.7.5 LLR Diagram (Finding a Power) 20](#_Toc69548730)

[1.7.6 LLR Diagram (Finding Root of Number) 20](#_Toc69548731)

[1.7.7 LLR Diagram (Plotting Graph) 21](#_Toc69548732)

[1.7.8 HLR Diagram (Arithmetic Operation) 21](#_Toc69548733)

[1.7.9 HLR Diagram ( Power, Root, Graph) 22](#_Toc69548734)

[1.8 Test Plan 23](#_Toc69548735)

[1.8.1 High Level Test Plan 23](#_Toc69548736)

[1.8.2 Low Level Test Plan 23](#_Toc69548737)

[1.9 Implementation Summary 25](#_Toc69548738)

[1.10 Video Summary 25](#_Toc69548739)

[1.11 Git Link 25](#_Toc69548740)

[1.12 Git Dashboard 25](#_Toc69548741)

[1.12.1 Badges:- 25](#_Toc69548742)

[1.12.2 Summary 26](#_Toc69548743)

[1.12.3 GIT inspector 27](#_Toc69548744)

[1.12.4 Set up for build 28](#_Toc69548745)

[1.12.5 Set up for build 29](#_Toc69548746)

[1.12.6 Set up for code Quality 29](#_Toc69548747)

[1.12.7 Outcome of code quality 30](#_Toc69548748)

[1.12.8 Set up for unity testing 31](#_Toc69548749)

[1.12.9 Outcome for unity testing 31](#_Toc69548750)

[1.13 Individual Contribution & Highlights 32](#_Toc69548751)

[1.14 Summary 32](#_Toc69548752)

[1.15 Challenges faced and how were they overcome 32](#_Toc69548753)

[1.16 Future Scope 33](#_Toc69548754)

[ Product will also use in banking sectors and also in other sectors where they want low price, more features 33](#_Toc69548755)

[2.0 Miniproject- 3. Embedded C [Team] 34](#_Toc69548756)

[2.1 Module: 34](#_Toc69548757)

[2.2 Topic and Subtopics 34](#_Toc69548758)

[2.3 Objectives & Requirements 34](#_Toc69548759)

[Objective: 34](#_Toc69548760)

[Requirements: 34](#_Toc69548761)

[LOW LEVEL REQUIREMENTS 34](#_Toc69548762)

[2.4 Test Plan 34](#_Toc69548763)

[2.5 Design 35](#_Toc69548764)

[2.6 Implementation Summary 36](#_Toc69548765)

[2.7 Individual Contribution & Highlights 36](#_Toc69548766)

[2.8 Summary 36](#_Toc69548767)

[2.9 Challenges faced and how they were overcome 37](#_Toc69548768)

[3.0 Miniproject -3 Python [Individual] 38](#_Toc69548769)

[3.1 Module Used 38](#_Toc69548770)

[3.2 Project title 38](#_Toc69548771)

[3.3 Topic and Subtopics 38](#_Toc69548772)

[3.4 Objectives 38](#_Toc69548773)

[3.5 Requirements 38](#_Toc69548774)

[3.5.2 High Level Requirement 38](#_Toc69548775)

[3.5.3 Low Level Requirement 39](#_Toc69548776)

[3.6 Design 40](#_Toc69548777)

[3.6.1 High Level Design 40](#_Toc69548778)

[3.6.2 Low Level Design 41](#_Toc69548779)

[3.7 Implementation Summary 41](#_Toc69548780)

[3.8 Git Link 41](#_Toc69548781)

[3.9 Summary 42](#_Toc69548782)

[3.10 Challenges faced and how were they overcome 42](#_Toc69548783)

[3.11 Future Scope 42](#_Toc69548784)

[4.0 Miniproject -4 [Individual] – Kernel Programming and Device Drivers 43](#_Toc69548785)

[4.1 Module/s: 43](#_Toc69548786)

[4.2 Topic and Subtopics 43](#_Toc69548787)

[4.4 Implementation Summary: 44](#_Toc69548788)

[4.4.1 Hands-on Activity that are implemented are as follows: 45](#_Toc69548789)

[4.4.2 User space code: 45](#_Toc69548790)

[4.4.3 kthread examples: 45](#_Toc69548791)

[4.5 Git Link: 45](#_Toc69548792)

[4.6 Summary 46](#_Toc69548793)

[4.7 Challenges faced and how were they overcome 46](#_Toc69548794)

# 1.0 Miniproject -1 SDLC [Team]

## 1.1 Modules used

SDLC (Software Development Life Cycle) and C Programming Modules are used in the project.

## 1.2 Project Title

Mini Calculator

## 1.3 Topic and Subtopics

* Introduction about SDLC
* C Programming
* Git Hub
* Code Analysis

Valgrind

CPP check

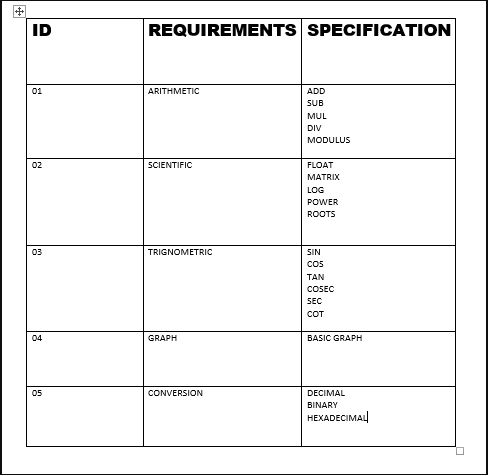
* Testing
* Unity Testing
* Features of Calculator
* Core Steps of SDLC
* Testing has been done for each function
* Makefile
* V Model
* Agile Model

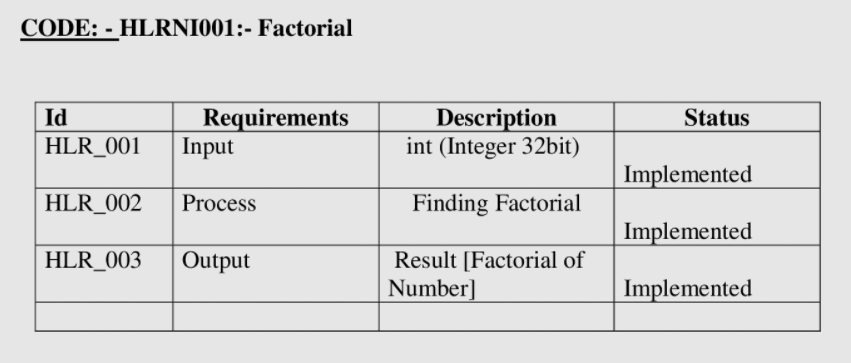
## 1.4 Objectives & Requirements

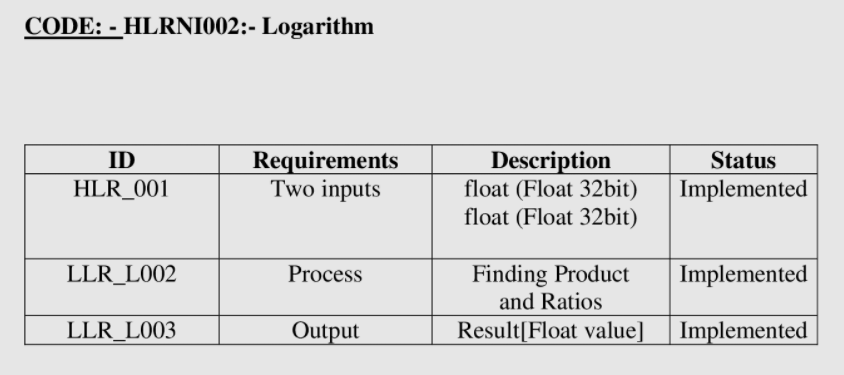
Objective: -  The simplest calculators can do only addition, subtraction, multiplication, and division. More sophisticated calculators can handle exponential operations, roots, logarithms, trigonometric functions, and hyperbolic functions So we are designing a calculator which contains all the functionalities

Of simplest calculator and sophisticated calculator.

### 1.4.1 High Level Requirement

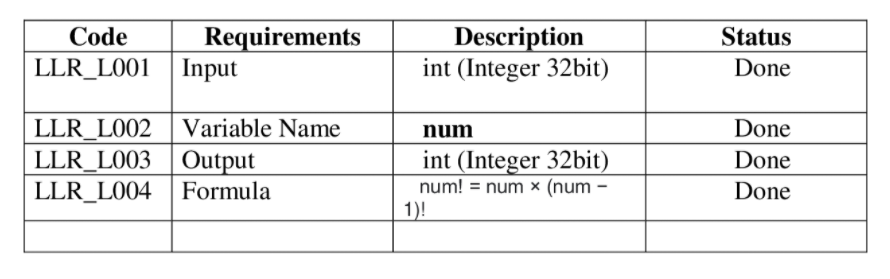




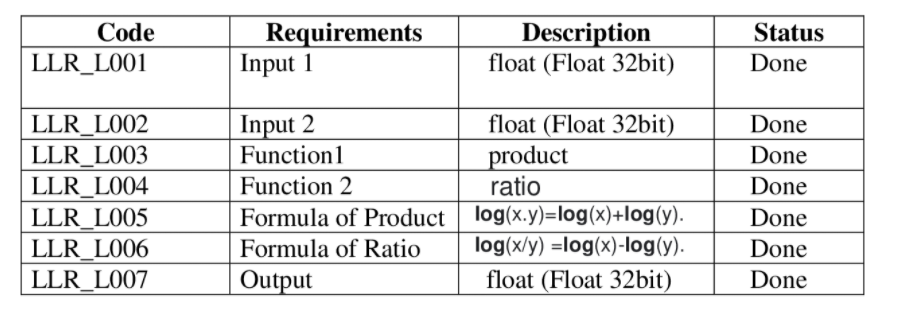


### 1.4.2 Low Level Requirement

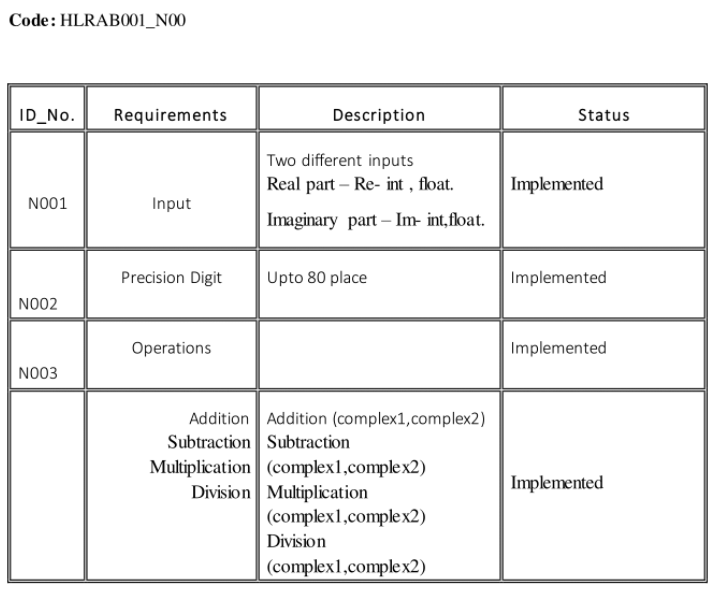
Factorial



Logarithm

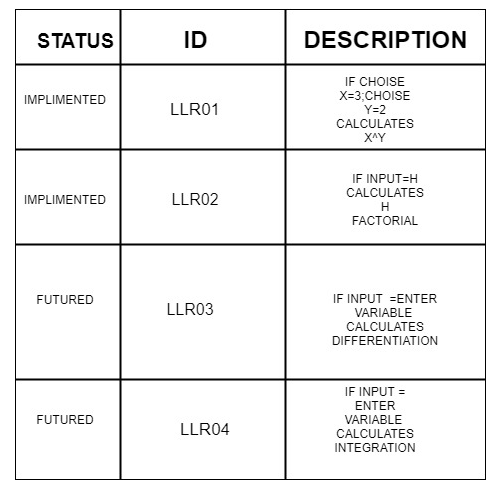


Complex Number

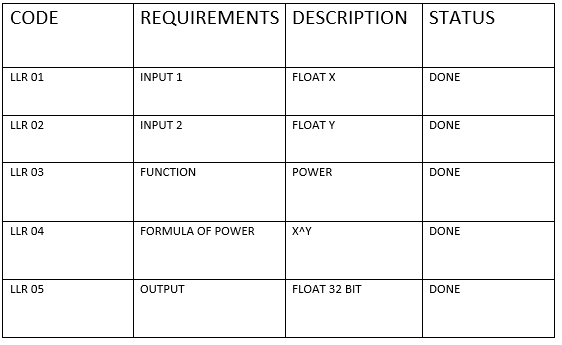


Arithmetic Operation

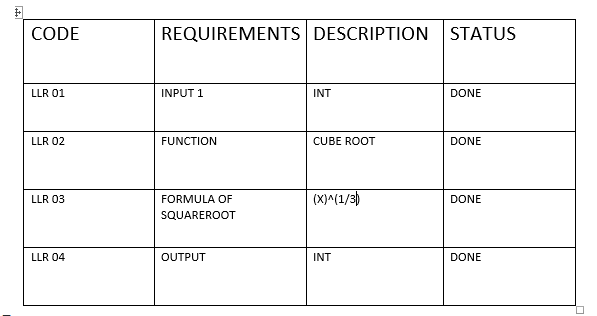
## 



Square Root



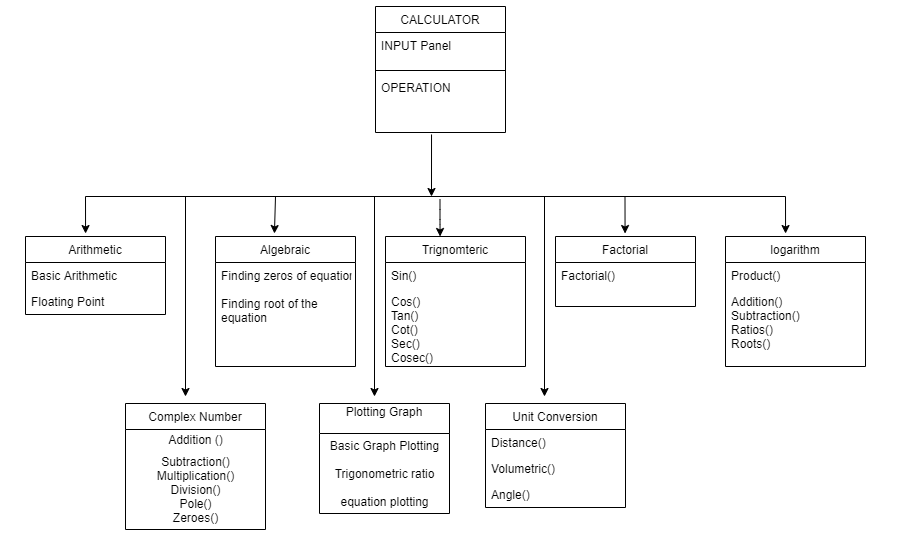
Cube Root



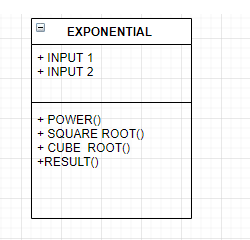
## 1.5 Design

## 1.6 Structural Diagram

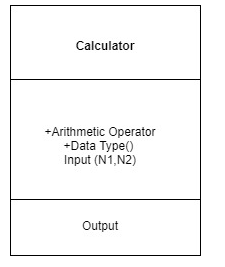
### 1.6.1 HLR of Calculator



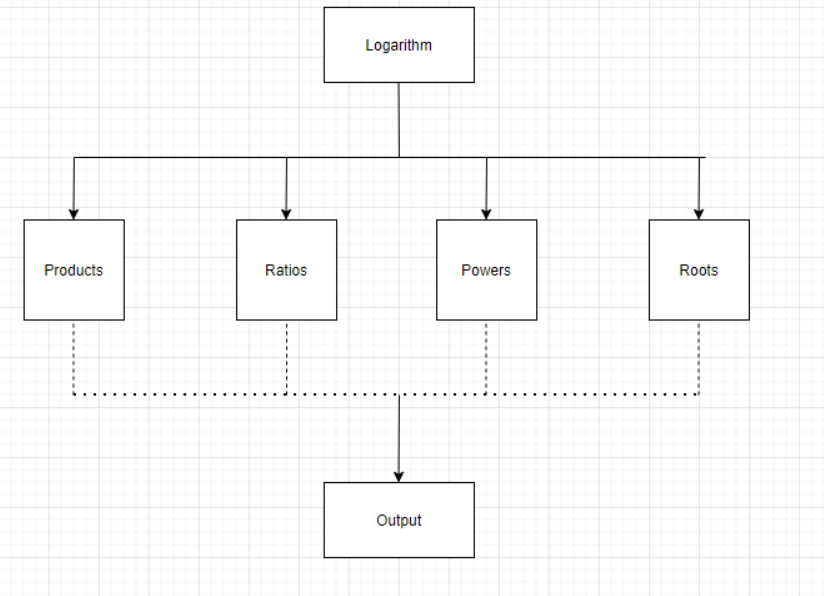
### 1.6.2 LLR Diagram(Exponential)



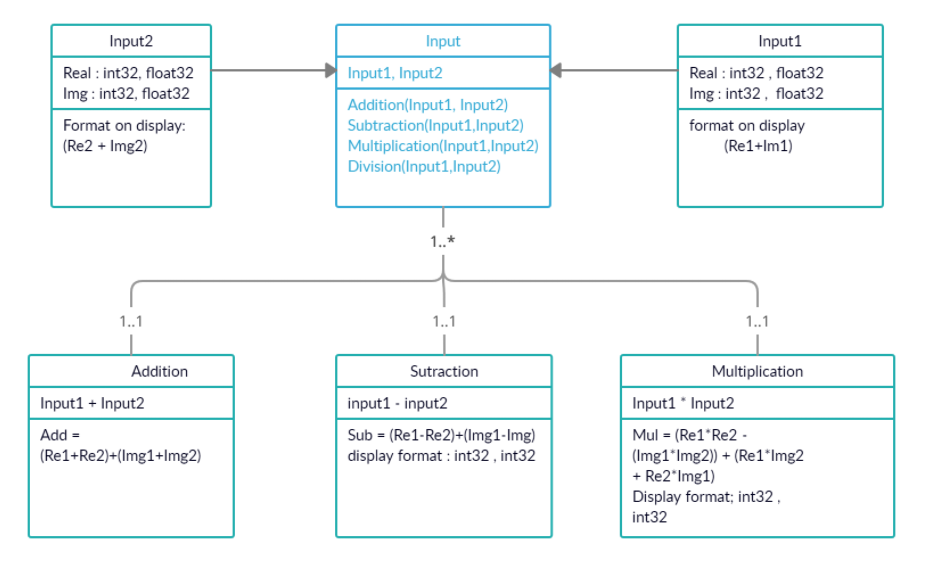
### 1.6.3 LLR Diagram (Calculator)



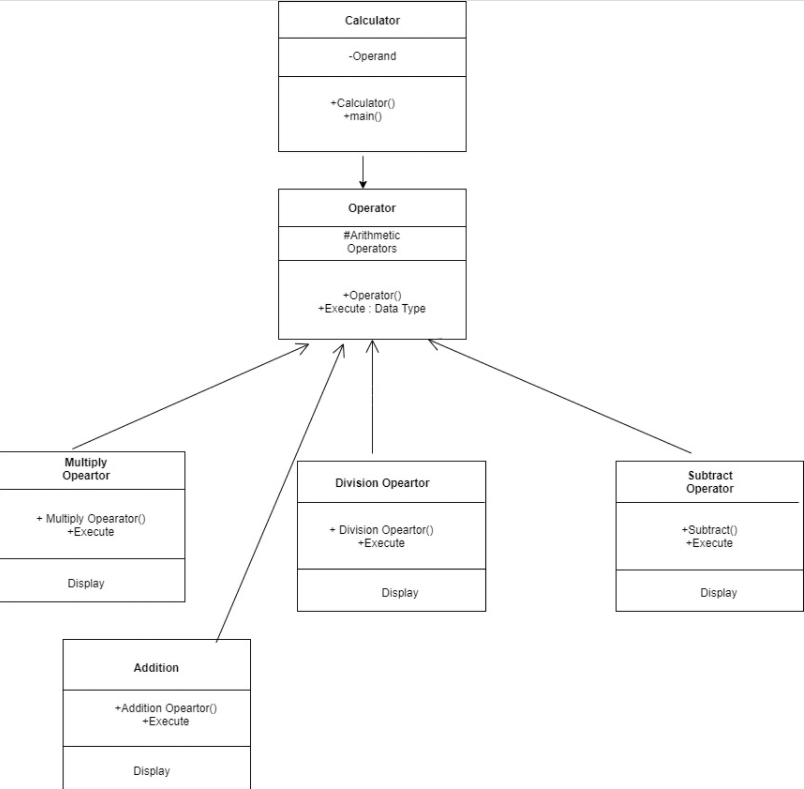
### 1.6.4 HLR Diagram(Logarithm)



### 1.6.6 LLR Diagram (Complex Number)

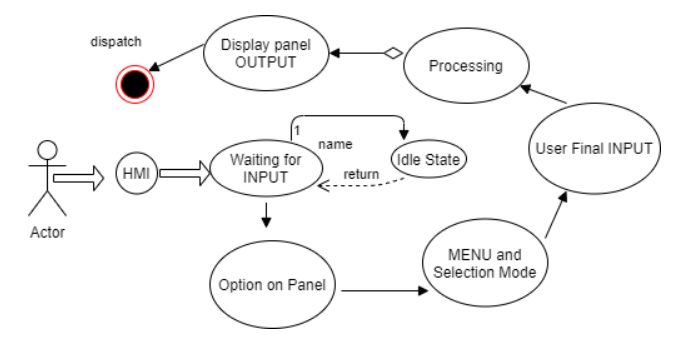


### 1.6.7 LLR Diagram(Arithmetic Operations)

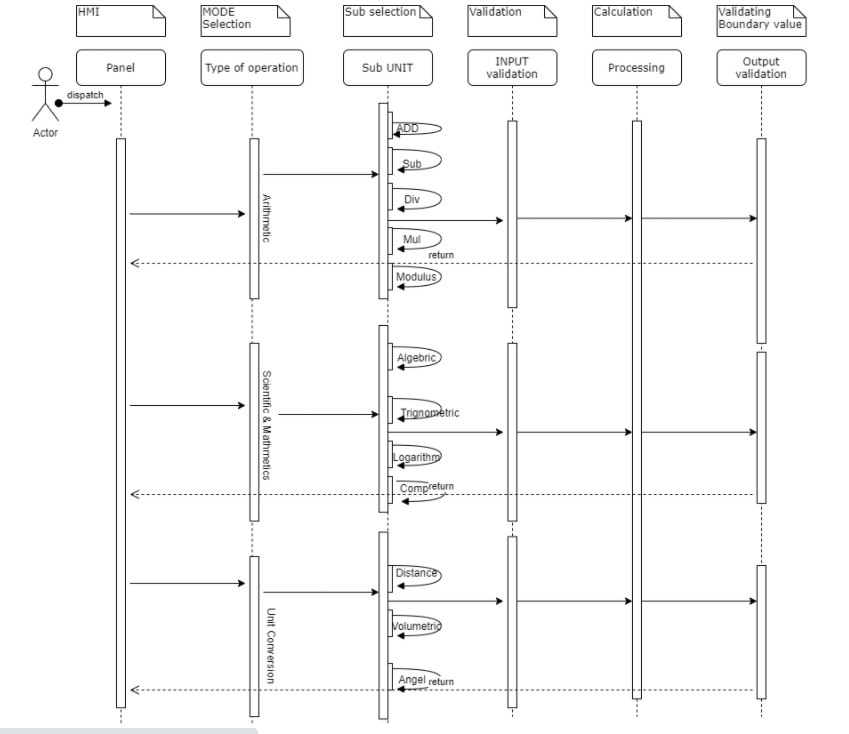


## 1.7 Behavioral Diagram

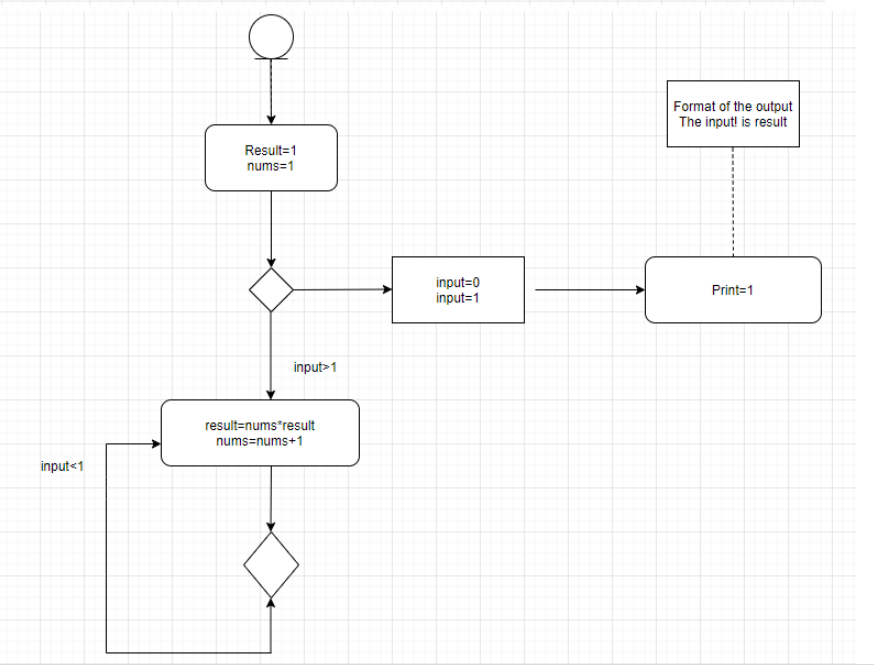
### 1.7.1 HLR Calculator Diagram (Flow control)



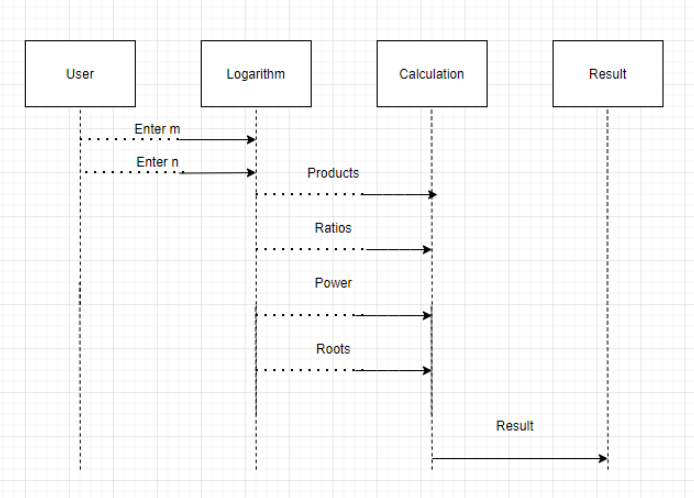
### 1.7.2 HLR Calculator Diagram (Sequence Diagram)



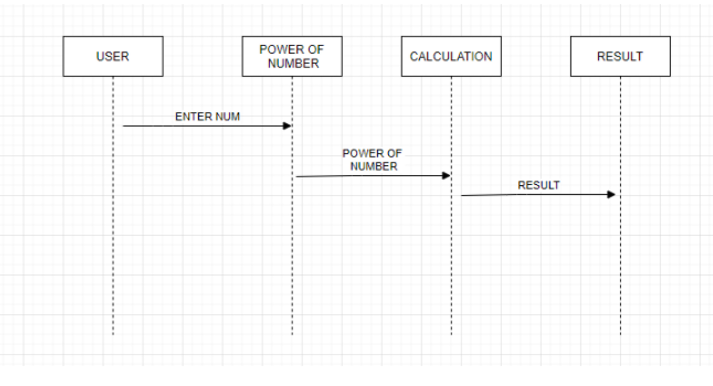
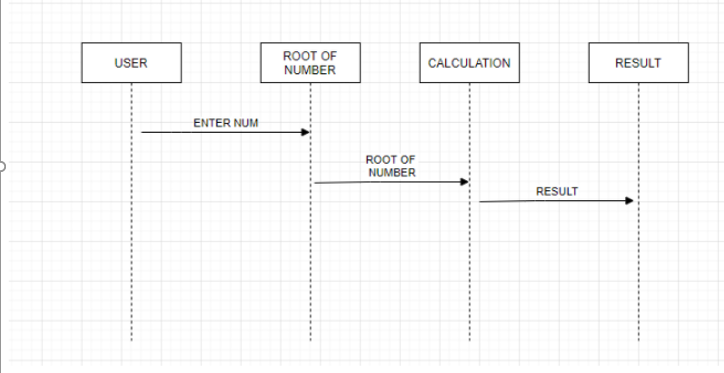
### 1.7.3 LLR Diagram (Factorial)



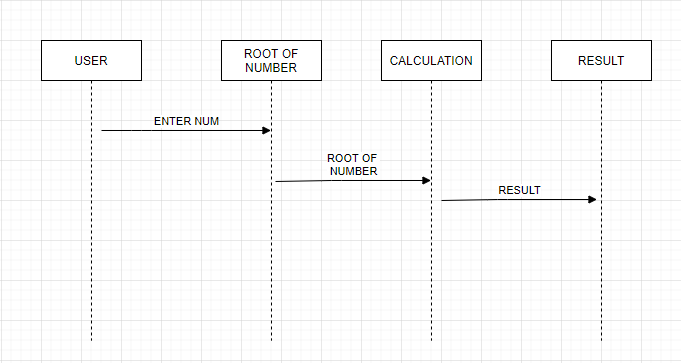
### 1.7.4 LLR Diagram (Logarithm)



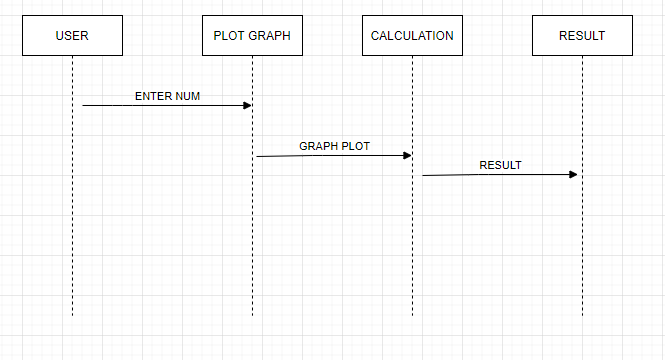
### 1.7.5 LLR Diagram (Finding a Power)



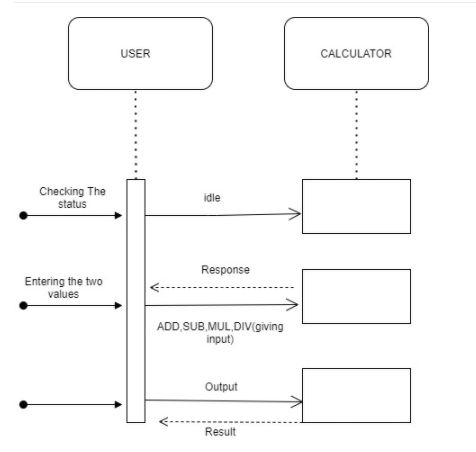
### 1.7.6 LLR Diagram (Finding Root of Number)



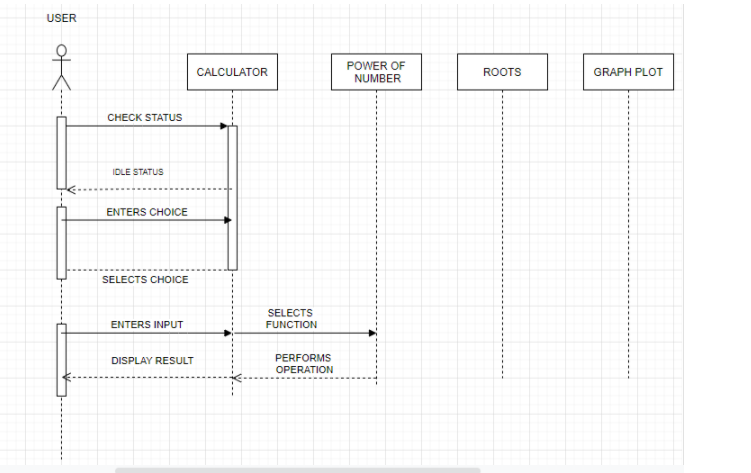
### 1.7.7 LLR Diagram (Plotting Graph)



### 1.7.8 HLR Diagram (Arithmetic Operation)



### 1.7.9 HLR Diagram ( Power, Root, Graph)



## 1.8 Test Plan

### 1.8.1 High Level Test Plan

### 1.8.2 Low Level Test Plan

Complex Number

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test\_ID** | **Description** | **Expected Input** | **Expected Output** | **Actual Output** | **Type of Test** |
| L001 | Giving right value to check the function | (1,1) (2,1) | Add-(3,2) | Add-(3,2) | Requirement Based |
| L002 | Checking the function subtraction | (3,6) (2,4) | Sub-(1,2) | Sub-(1,2) | Requirement Based |
| L003 | Subtraction  Input1<Input2 | (4,5) (6,7) | (-2,-7) | (-2,-7) | Scenario Based |

Factorial

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test\_ID** | **Description** | **Expected Input** | **Expected Output** | **Actual Output** | **Type of Test** |
| TCN001 | Standard Input | 5! | 120 | 120 | Requirement Based |
| TCN002 | Standard Input | 1! | 1 | 1 | Scenario Based |
| TCN003 | Rational Number | 1/2 | Error | Error | Scenario Based |

Logarithm

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test\_ID** | **Description** | **Expected Input** | **Expected Output** | **Actual Output** | **Type of Test** |
| TCN001 | Standard Input | m=5  n=6 | Product=116  Ratio=300 | Product=116  Ratio=300 | Requirement Based |
| TCN002 | Standard Input | m=9  n=10 | Product=456  Ratio=554 | Product=456  Ratio=554 | Requirement Based |
| TCN003 | Particular Input | m=0  n=5 | Error | Error | Scenario Based |

Basic Arithmetic Operations

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test\_ID** | **Description** | **Expected Input** | **Expected Output** | **Actual Output** | **Type of Test** |
| AO\_001 | Standard Input  (ADD) | n1=2  n2=5 | 7 | 7 | Requirement Based |
| AO\_002 | Standard Input  (SUB) | n1=8  n2=3 | 5 | 5 | Requirement Based |
| AO\_003 | Standard Input  (DIV) | n1=1  n2=0 | Error | Error | Scenario Based |

Power

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test\_ID | Description | Expected Input | Expected Output | Actual Output | Type of Test |
| TC\_001 | Standard Input | Input1: 2  Input2: 3 | 8 | 8 | Requirement Based |
| TC\_002 | Standard Input | Input1: 5  Input2: 2 | 25 | 25 | Requirement Based |
| TC\_003 | Standard Input | Input1: 10  Input2: 2 | 100 | 100 | Requirement Based |

Roots

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test\_ID** | **Description** | **Expected Input** | **Expected Output** | **Actual Output** | **Type of Test** |
| TC\_004 | Standard Input | 4 | 2 | 8 | Requirement Based |
| TC\_005 | Standard Input | 0.25 | 0.5 | 0.5 | Scenario Based |
| TC\_006 | Standard Input | 16 | 4 | 100 | Requirement Based |

## 1.9 Implementation Summary

Now days calculator is contributing in each one of us life, some of us using for very basic arithmetic calculation and some of using for calculating such a complex problem which take so many hours to calculate by manually. Hence in market there are various categories of calculator available based on your requirements. Some of them are made to be very specific in term of their using and some of them are used by different-different class of people who are using it. Like students, graduate students, business man, local shops and etc. If define this device in very short then we can say it is a device that performs arithmetic operations on numbers. Implementation folder has all the source files, header files, test files or different features of the calculator.

Here inc folder contains all header files with ‘.h’ extension. It contains prototype of all functions.

* The src folder holds all the source file with ‘.c’ extension. It has definition of all function whose prototype is define in inc folder.
* The test folder contain test.c file for testing of source code based on requirement, scenario, and boundary.
* The unity folder contains file which holds prototype and definition of the standard unity test case functions.
* And then there is Makefile

## 1.10 Video Summary

“Please upload a short video on the repo for the walkthrough of the project (Team/Individual) less than 7min and less than 30MB File Size. Start is the Standard opening slide with title of miniproject + Team members followed by the walkthrough ”

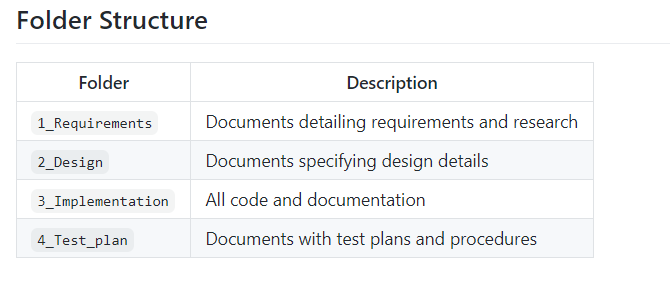
## 1.11 Git Link

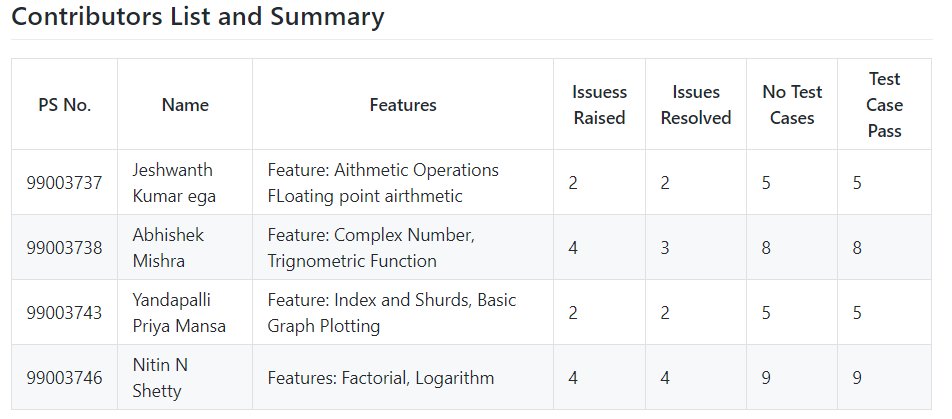
<https://github.com/99003738/AppliedSDLC_N3.git>

## 1.12 Git Dashboard

### 1.12.1 Badges:-







### 1.12.2 Summary

It is a simple electronic hardware/software device that is capable of performing the simple calculations such as addition, subtraction, multiplication, division, calculating power of number, exponential function, logarithmic function, permutation and combination, trigonometry, inverse-trigonometric functions, factorial of a number, binary to decimal conversion etc.

#### Git inspector summary

### 1.12.3 GIT inspector

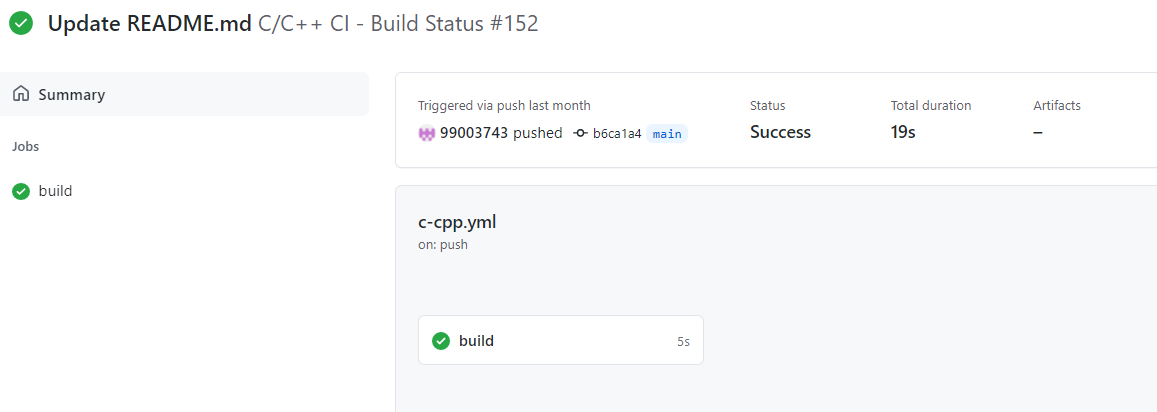
#### 

#### Build

### 1.12.4 Set up for build

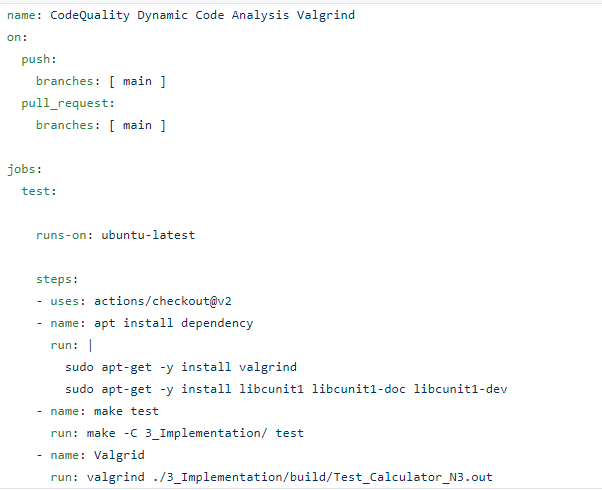


### 1.12.5 Set up for build

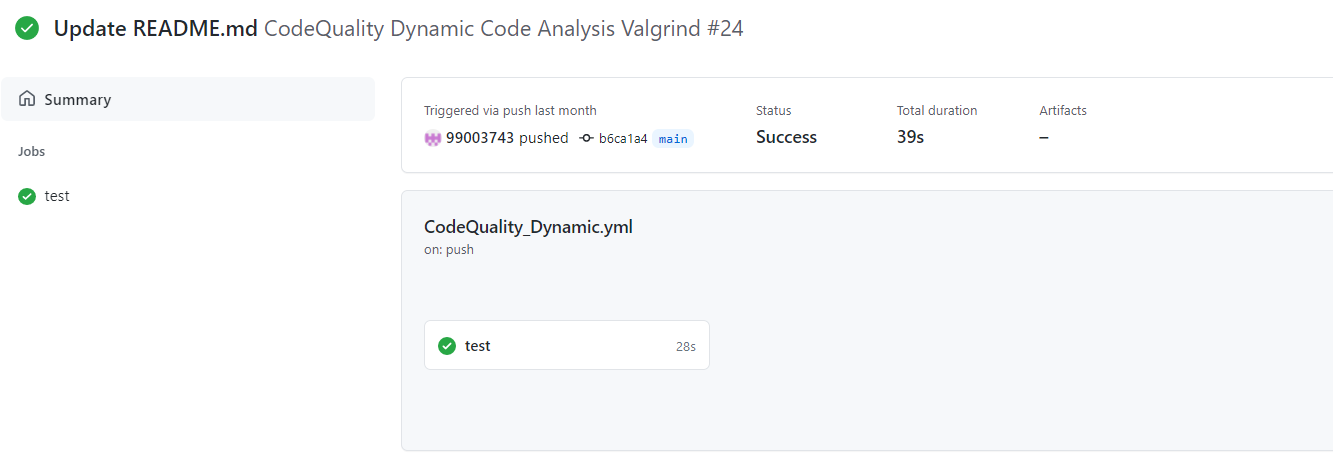


#### Code quality and Issues or Bug Tracking

### 1.12.6 Set up for code Quality

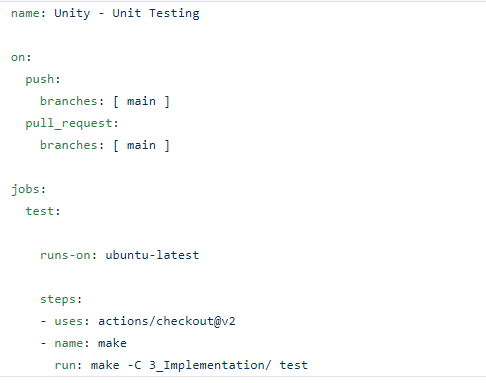


### 1.12.7 Outcome of code quality

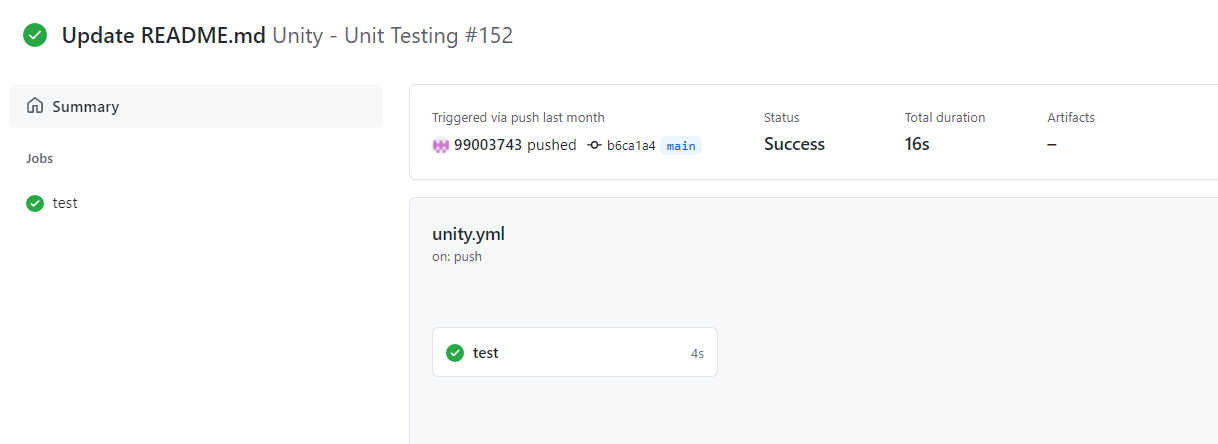


#### Unit Testing

### 1.12.8 Set up for unity testing



### 1.12.9 Outcome for unity testing



## 1.13 Individual Contribution & Highlights

* In Calculator Project, I have done Fraction and Logarithm Feature
* Implemented both the feature in C language
* Implemented all possible test cases for the same
* High level and Low-Level Requirements are listed
* Plotted UML (Structural and Behavioral) diagrams for High level and Low-Level Requirements
* Listed 4W1H for the Project
* Issues are raised and respective issues are solved.
* Contributed in overall workflow and in project implementation.

## 1.14 Summary

The main motto is to design a calculator with certain features according to the specific requirements. The target customers for the designed calculator are students, shopkeepers, banking executives and engineers.

Technical:

Improved implementation of C Concepts

Practical Implementation of SDLC life Concepts

Source code management(Github)

Soft skills:

* Project management
* Conflict management

## 1.15 Challenges faced and how were they overcome

* Running the make file as its resolved by defining its correct path(.out for linux and -lm for math functions)
* Synchronizing the VS code to github, colleague help to resolve the issue
* Making the function call in correct path
* Open git log while committing, thus went to github desktop and pulled origin and then pushed origin.
* Test case code for the boundary problem. Added code with the help of internet
* Integration problem since lots of header file was there. Changes made in header file to remove the multiple occurrence error.

## 1.16 Future Scope

### Product will also use in banking sectors and also in other sectors where they want low price, more features

* Price of the products is less than other calculator which are the same features available in the market.
* Students can use this product because to solve complex problems.

# 2.0 Miniproject- 3. Embedded C [Team]

## 2.1 Module:

The modules used in this are SDLC, Embedded C and was implemented on the hardware STM32.

## 2.2 Topic and Subtopics

* + The Car feature requirements for sub system was found.
  + The window, seat and lighting system of car was developed.
  + The code was dumped on the STM32 board.
  + SPI, UART,I2C
  + External Interrupt

## 2.3 Objectives & Requirements

## Objective:

To implement body control module functionalities using STM32 development board.

## Requirements:

• STM32 development board.

• LDR sensor

• Push buttons

• Bread board

• Buzzer

• Jumper wires

• LEDs

## LOW LEVEL REQUIREMENTS

## 2.4 Test Plan

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Description** | **Expected Input** | **Expected Output** |
| ID1 | Door Control | Push Button | Green LED glows & Buzzer beeps. |
| ID2 | Seat Control | Push Button | Green LED glows. |
| ID3 | Power Window | Push Button | 2 LEDs glow in RGB module |
| ID4 | Headlight Control | LDR sensor | Green LED glows |
| ID5 | AC Control | Push Button | Blue LED glows |
| ID6 | Wiper Control | Moisture Sensor | Orange LED glows |

## 2.5 Design

## 2.6 Implementation Summary

* The main objective was to create a Body Control Module using STM32 board. The features selected to be implemented are Door Control, Seat Control, Power Window, Headlight Control, AC Control, Wiper Control. Push buttons & sensors were used to create a prototype of the BCM. These modules were tested individually and later integrated into one single code and dumped on the STM32 board.
* Door Control: A push button is used to indicate the state of door. When button is pressed it indicates that the door is open and Green LED glows and Buzzer beeps. When button is released, LED & Buzzer is turned off.
* Seat Control: A push button is used to indicate the seat status. When button is pressed it indicates that the seat is in a particular position and Green LED glows. When button is released LED is turned off.
* Power Window: A push button is used to indicate the status of Window. When button is pressed it indicates that the window is open and 2 LEDs from RGB module glow. When button is released RGB is turned off.
* Headlight Control: A LDR sensor is used as input in this module. When LDR sensor does not get any light, it indicates that the headlight should be turned on, i.e. Green LED glows. When LDR senses Light, the Green LED is turned off.
* AC control: A push button is used to indicate the on/off status of AC. When button is pressed Blue LED is turned on. When button is released Blue LED is turned off.
* Wiper Control: A moisture sensor is used as input in this module. When water is detected by moisture sensor, Orange LED glows. When there is no water the LED is turned off.

## 2.7 Individual Contribution & Highlights

1. Headlight Control

2.Wiper Control

I have Implemented these two features for Headlight control I used LDR Sensor so that sensor detects light and during night time sensor does not detects any light and it automatically turns on the light. And also for wiper control used a moisture sensor when water is detected by sensor it turns on the wipers

So that I have used green led for headlight control and orange led for wiper control.

## 2.8 Summary

Each module was written and tested individually. Once all module was working as intended, all modules were integrated into one.

## 2.9 Challenges faced and how they were overcome

1. It was challenging to showcase all the functionality with minimal components.
2. Integration of the modules.
3. It was also the challenging to show all the features in the single board.

# 3.0 Miniproject -3 Python [Individual]

## 3.1 Module Used

In this Module we used Advanced python for project.

3.2 Project title : Retrieving particular data from different sheets and adding retrieved data in Master Sheet

## 3.3 Topic and Subtopics

* Basic Python

Subtopics

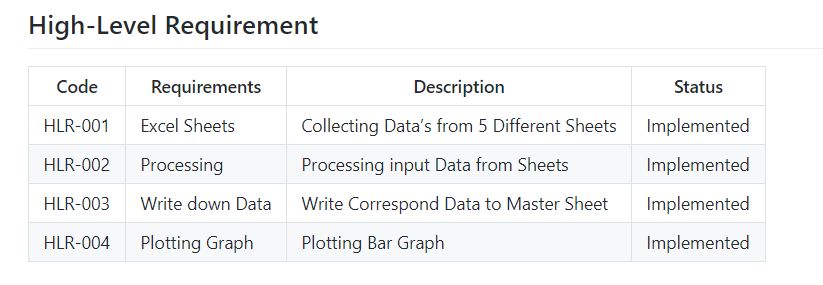
* Data Types
* Arithmetic Operations
* String Operations
* If-else statements
* While loops
* For loops
* Functions
* List
* Tuple
* Set
* Dictionary
* Open multiple excel file
* Excel sheet creations
* Reading excel sheet
* Writing excel sheet

## 3.4 Objectives

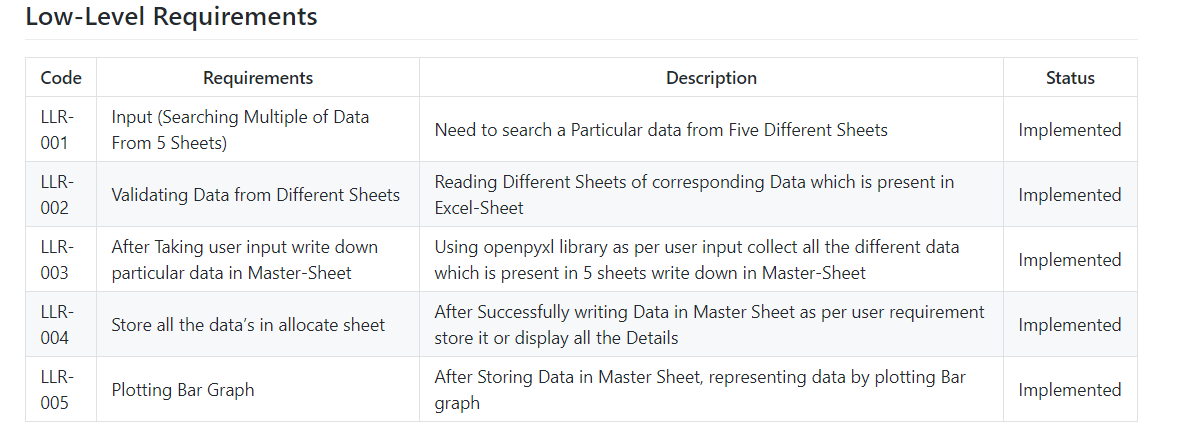
I am having collection data in Different Sheets. This program takes a user input as a keyword and search the occurrence of the word in the Excel file and assembles all corresponding data from 5 sheets, and store these details in a Master Sheet.

## 3.5 Requirements

### 3.5.2 High Level Requirement

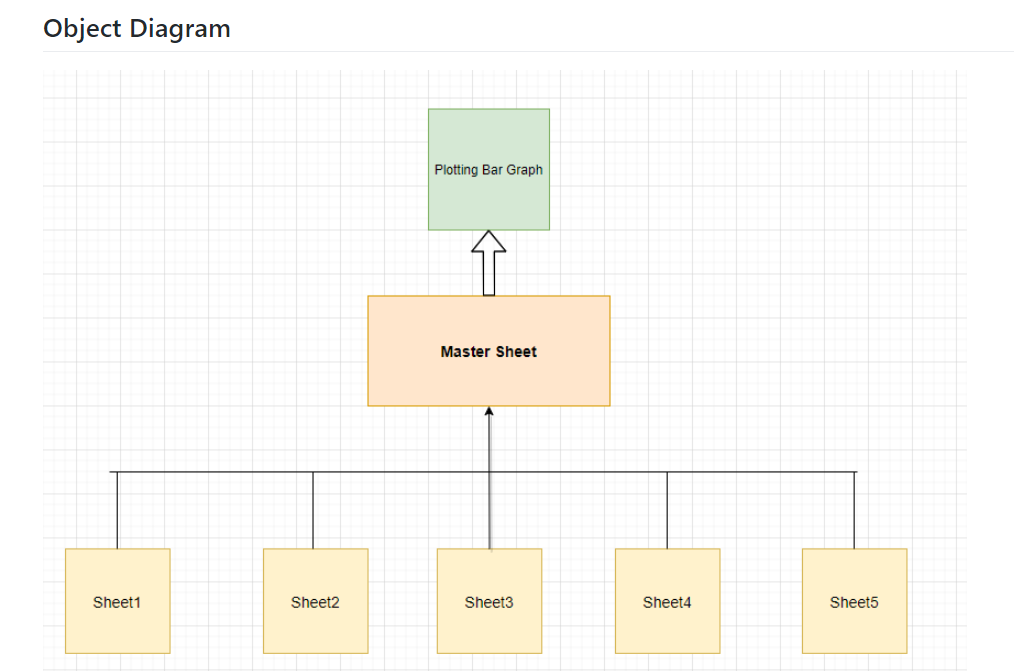


### 3.5.3 Low Level Requirement

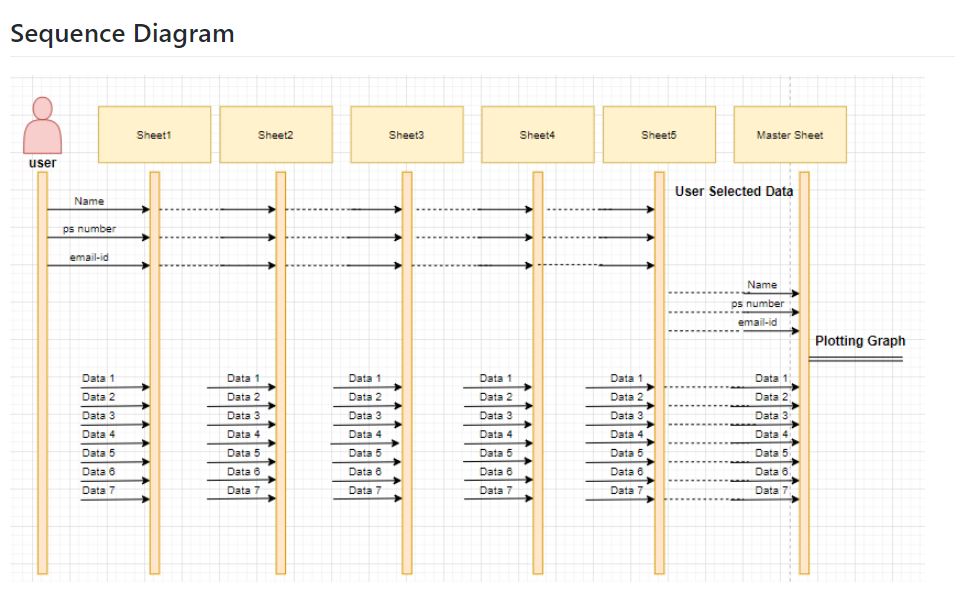


## 3.6 Design

### 3.6.1 High Level Design



### 3.6.2 Low Level Design



## 3.7 Implementation Summary

I am having collection of students marks data from Sem 1 to Sem 5 in 5 Different Sheets. This program takes a user input as a keyword and search the occurrence of the word in the Excel file and assembles all corresponding data from 5 sheets, and store these details in a Master Sheet.

1. There are 10 columns in each sheet, All Sheets have 3 common data i.e. Name, PS Number & Email. The remaining columns are unique to each sheet.
2. The user provides these 3 common values to search for a Data. Using these 3 values the record is searched in all available sheets.
3. User can also search repeated Name, PS-no and also email-id
4. This record from all the sheets is appended in a list and is Written in the Master Sheet in a single row, against the name.
5. The recent data from the Master Sheet is Represented by Plotting Graph.

## 3.8 Git Link

<https://github.com/99003746/Mini_Project_Python.git>

## 3.9 Summary

Technical:

Improved implementation of Python Concepts

* Practical Implementation of Python Concepts
* Source code management
* Used predefined modules ( openpyxl, pandas)

Soft skills:

* Project management
* Conflict management

## 3.10 Challenges faced and how were they overcome

* Differentiation of high level and low level.
* Committing to GitHub, pull and push in GitHub.
* Converting pictures and tables into readme file.
* System issues (crashing and Interfacing)
* Understanding the modules of openpyxl and pandas.

## 3.11 Future Scope

* Searching or Analyzing huge amount of data can be made easily
* Easily we can create new master sheet and we can store required data in master sheet.

# 4.0 Miniproject -4 [Individual] – Kernel Programming and Device Drivers

## 4.1 Module/s:

The modules used in this are Linux and Kernel Device drivers.

## 4.2 Topic and Subtopics

* Basic Linux commands.
* Qemu Based Emulation.
* Creation of SD card.
* Building custom Kernel.
* Cross Compilation.
* Static and dynamic libraries.
* System calls.
* Adding system calls in kernel space.
* Invoking system calls from user space.
* Kernel modules.
* In-Tree modules: Dynamic.
* In-tree modules: static.
* Basics of Kernel Device Drivers.
* Registering Char Driver.
* Kernel Data Structure.
* Kfifo API.
* List API.
* IPC Kernel
  + Concurrency.
* Kernel Threads.
* Locking and Synchronization.
* Mutex.
* Semaphore.
* Spinlocks.
* Wait queues.
* IOCTL.
  + Driver model.

4.3 Objectives & Requirements:

The main objective of this module is to apply the concepts of Linux kernel, kernel device drivers to develop:

* + Custom kernel.
  + Create char drivers.
  + Developing cross compiled code for target qemu.
  + Creating own system calls.

4.3.1 Requirements:

* Basic Linux commands.
* Programming in Linux Environment.
* Custom kernel.
* zImage
* vexpress-v2p-ca9.dtb
* rootfs.img
* Operating system Basics.
* IPC concepts.
* Concurrency.
* File handling using system calls.
* Virtual Memory concept.

## 4.4 Implementation Summary:

For System Calls: -

* Generate new zImage after adding system calls definition and prototype. Adding its definition file name in Makefile.

-> make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabi- zImage

* Then mount the SD card and copy the output file of user space code in it and then unmount it by following commands: -

-> sudo mount –o loop, rw, sync rootfs.img /mnt/rootfs

-> sudo cp a.out rootfs.img /mnt/rootfs/home/root

->sudo umount rootfs.img /mnt/rootfs

* Then run Qemu using following command

-> qemu-system-arm -M vexpress-a9 -m 1024 -serial stdio \

-kernel zImage -dtb vexpress-v2p-ca9.dtb \

-sd rootfs.img -append "console=ttyAMA0 root=/dev/mmcblk0 rw"

Then run the ./a.out file on Qemu and output is display either on VGA console or in serial console based on the system call. (give command line input only for first

## 4.4.1 Hands-on Activity that are implemented are as follows:

* Register char driver
* Register file operations
* Device Create, Class Create
* Read, write operations using global buffer
* Read, write operations using kfifo.
* ioctl operations, returning length/remaining space, reset operation
* ioctl operations - filling length/remaining space in structure
* synchronization in char driver - using wait queue

## 4.4.2 User space code:

* simple read, write
* multiple read, multiple write
* User space code for IOCTL operations

## 4.4.3 kthread examples:

* + simple two threads
  + Race condition scenarios
  + Mutual exclusion using semaphore, mutex, spinlock
  + Synchronization using semaphores, wait queues
  + Device Tree based platform driver code -- dummy UART
  + Activity that are implemented are as follow:
  + System calls -- echo back the given string.
  + System calls—traverse process list print pid and ppid.
  + System calls—length of string.
  + System calls—taking simple parameter.
  + IOCTL operation traverse the list.

## 4.5 Git Link:

<https://github.com/99003746/Embedded-Linux_Kernel-Programming.git>

## 

## 4.6 Summary

In this project, custom system calls for a particular kernel is made by modifying internal syscalls.h, syscall.tbl, kernel /Makefile and its definition in c file in kernel folder of kernel source.

In user-space code of the system call a special system call number is mentioned to use the custom system call which is defined system call table (syscall.tbl). Finally, it’s test on serial console and VGA console according to expected input and output.

## 4.7 Challenges faced and how were they overcome

* Unable to directly access string in kernel space from userspace and vice-versa – Using copy\_from\_user () and copy\_to\_user () solved this issue.
* Traversing through system process list was an issue- It was solved by using for\_each\_process () and task\_struct.
* Traversing through node list was issue that was resolved using list\_for\_each () method.