

# GENESIS - Learning Outcome & Mini-project Summary Report



LTTS  
GLOBAL  
ENGINEERING  
ACADEMY



*L&T Technology Services*



**Details**

Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	To be Approved	Remarks/Revision Details
1.		Nitin N Shetty (Ps.no 99003746)			

## Contents

<b>CONTENTS.....</b>	<b>3</b>
<b>1.0 MINIPROJECT -1 SDLC [TEAM].....</b>	<b>5</b>
1.1 MODULES USED.....	5
1.2 PROJECT TITLE .....	5
1.3 TOPIC AND SUBTOPICS.....	5
1.4 OBJECTIVES & REQUIREMENTS.....	6
1.4.1 High Level Requirement.....	6
1.4.2 Low Level Requirement.....	8
.....	10
1.5 DESIGN.....	13
1.6 STRUCTURAL DIAGRAM .....	13
1.6.1 HLR of Calculator.....	13
1.6.2 LLR Diagram(Exponential).....	13
1.6.3 LLR Diagram (Calculator).....	14
1.6.4 HLR Diagram(Logarithm).....	14
1.6.6 LLR Diagram (Complex Number).....	15
1.6.7 LLR Diagram(Arithmetic Operations).....	16
1.7 BEHAVIORAL DIAGRAM .....	17
1.7.1 HLR Calculator Diagram (Flow control) .....	17
1.7.2 HLR Calculator Diagram (Sequence Diagram).....	18
1.7.3 LLR Diagram (Factorial).....	19
1.7.4 LLR Diagram (Logarithm).....	19
1.7.5 LLR Diagram (Finding a Power).....	20
1.7.6 LLR Diagram (Finding Root of Number) .....	20
1.7.7 LLR Diagram (Plotting Graph) .....	21
1.7.8 HLR Diagram (Arithmetic Operation) .....	21
1.7.9 HLR Diagram ( Power, Root, Graph).....	22
1.8 TEST PLAN.....	23
1.8.1 High Level Test Plan.....	23
1.8.2 Low Level Test Plan.....	24
1.9 IMPLEMENTATION SUMMARY .....	26
1.11 GIT LINK .....	26
HTTPS://GITHUB.COM/99003738/APPLIEDSDLC_N3.GIT.....	26
1.12 GIT DASHBOARD.....	27
1.12.1 Badges:-.....	27
1.12.2 Summary.....	28
1.12.3 GIT inspector.....	29
1.12.4 Set up for build.....	30
1.12.5 Set up for build.....	31
1.12.6 Set up for code Quality.....	31
1.12.7 Outcome of code quality .....	32
1.12.8 Set up for unity testing.....	32
1.12.9 Outcome for unity testing.....	33
1.13 INDIVIDUAL CONTRIBUTION & HIGHLIGHTS .....	33
1.14 SUMMARY .....	33
1.15 CHALLENGES FACED AND HOW WERE THEY OVERCOME.....	34
1.16 FUTURE SCOPE .....	34

<b>2.0 MINIPROJECT- 2. EMBEDDED C [TEAM].....</b>	<b>35</b>
2.1 MODULE: .....	35
2.2 TOPIC AND SUBTOPICS.....	35
2.3 OBJECTIVES & REQUIREMENTS.....	35
OBJECTIVE: .....	35
REQUIREMENTS: .....	35
2.4 TEST PLAN.....	35
2.6 GIT LINK.....	36
2.7 DESIGN.....	36
2.8 IMPLEMENTATION SUMMARY .....	38
2.9 INDIVIDUAL CONTRIBUTION & HIGHLIGHTS .....	38
2.10 SUMMARY .....	39
2.11 CHALLENGES FACED AND HOW THEY WERE OVERCOME.....	39
<b>3.0 MINIPROJECT -3 PYTHON [INDIVIDUAL] .....</b>	<b>40</b>
3.1 MODULE USED.....	40
3.2 PROJECT TITLE.....	40
3.3 TOPIC AND SUBTOPICS .....	40
3.4 OBJECTIVES.....	40
3.5 REQUIREMENTS.....	40
3.5.2 High Level Requirement.....	41
3.5.3 Low Level Requirement.....	41
3.6 DESIGN.....	42
3.6.1 High Level Design .....	42
3.6.2 Low Level Design .....	43
3.7 IMPLEMENTATION SUMMARY .....	43
3.8 GIT LINK.....	43
3.9 SUMMARY .....	44
3.10 CHALLENGES FACED AND HOW WERE THEY OVERCOME.....	44
3.11 FUTURE SCOPE .....	44
<b>4.0 MINIPROJECT -4 [INDIVIDUAL] – KERNEL PROGRAMMING AND DEVICE DRIVERS .....</b>	<b>45</b>
4.1 MODULE/S: .....	45
4.2 TOPIC AND SUBTOPICS .....	45
4.3 OBJECTIVES & REQUIREMENTS:.....	46
4.3.1 Requirements:.....	46
4.4 IMPLEMENTATION SUMMARY: .....	46
4.4.1 HANDS-ON ACTIVITY THAT ARE IMPLEMENTED ARE AS FOLLOWS: .....	47
4.4.2 USER SPACE CODE:.....	47
4.4.3 KTHREAD EXAMPLES: .....	47
4.5 GIT LINK:.....	47
4.6 SUMMARY .....	48
4.7 CHALLENGES FACED AND HOW WERE THEY OVERCOME .....	48

## **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

ID	REQUIREMENTS	SPECIFICATION
01	ARITHMETIC	ADD SUB MUL DIV MODULUS
02	SCIENTIFIC	FLOAT MATRIX LOG POWER ROOTS
03	TRIGNOMETRIC	SIN COS TAN COSEC SEC COT
04	GRAPH	BASIC GRAPH
05	CONVERSION	DECIMAL BINARY HEXADECIMAL

**CODE: - HLRNI001:- Factorial**

<b>Id</b>	<b>Requirements</b>	<b>Description</b>	<b>Status</b>
HLR_001	Input	int (Integer 32bit)	Implemented
HLR_002	Process	Finding Factorial	Implemented
HLR_003	Output	Result [Factorial of Number]	Implemented

**CODE: - HLRNI002:- Logarithm**

<b>ID</b>	<b>Requirements</b>	<b>Description</b>	<b>Status</b>
HLR_001	Two inputs	float (Float 32bit) float (Float 32bit)	Implemented
LLR_L002	Process	Finding Product and Ratios	Implemented
LLR_L003	Output	Result[Float value]	Implemented

### 1.4.2 Low Level Requirement

Factorial

Code	Requirements	Description	Status
LLR_L001	Input	int (Integer 32bit)	Done
LLR_L002	Variable Name	<b>num</b>	Done
LLR_L003	Output	int (Integer 32bit)	Done
LLR_L004	Formula	$\text{num!} = \text{num} \times (\text{num} - 1)!$	Done

Logarithm

Code	Requirements	Description	Status
LLR_L001	Input 1	float (Float 32bit)	Done
LLR_L002	Input 2	float (Float 32bit)	Done
LLR_L003	Function1	product	Done
LLR_L004	Function 2	ratio	Done
LLR_L005	Formula of Product	$\log(x.y) = \log(x) + \log(y).$	Done
LLR_L006	Formula of Ratio	$\log(x/y) = \log(x) - \log(y).$	Done
LLR_L007	Output	float (Float 32bit)	Done



Complex Number

**Code:** HLRAB001\_N00

ID_No.	Requirements	Description	Status
N001	Input	Two different inputs Real part – Re- int , float. Imaginary part – Im- int,float.	Implemented
N002	Precision Digit	Upto 80 place	Implemented
N003	Operations		Implemented
	Addition Subtraction Multiplication Division	Addition (complex1,complex2) Subtraction (complex1,complex2) Multiplication (complex1,complex2) Division (complex1,complex2)	Implemented

## Arithmetic Operation

ID_NO.	REQUIRMENTS	Description
LLR_N001	INPUT 1	DATATYPE INT (INTEGER 32 BIT)
LLR_N002	INPUT 2	DATATYPE INT (INTEGER 32 BIT)
LLR_N003	VARIABLE NAME	OPERAND1, OPERAND 2
LLR_N004	FUNCTION 1	SUBTRACTION
LLR_N005	FUNCTION 2	MULTIPLICATION
LLR_N006	FUNCTION 3	DIVISION
LLR_N007	FUNCTION 4	ADDITION
LLR_N008	FORMULA ADDITION FORMULA SUBTRACTION FORMULA MULTIPLICATION FORMULA DIVISION	OPERAND1 + OPERAND2 OPERAND1 - OPERAND2 OPERAND1 * OPERAND2 OPERAND1 / OPERAND2
LLR_N009	OUTPUT	INT INTEGER(32BIT)

STATUS	ID	DESCRIPTION
IMPLIMENTED	LLR01	IF CHOISE X=3;CHOISE Y=2 CALCULATES $X^Y$
IMPLIMENTED	LLR02	IF INPUT=H CALCULATES H FACTORIAL
FUTURED	LLR03	IF INPUT =ENTER VARIABLE CALCULATES DIFFERENTIATION
FUTURED	LLR04	IF INPUT = ENTER VARIABLE CALCULATES INTEGRATION

## Square Root

CODE	REQUIREMENTS	DESCRIPTION	STATUS
LLR 01	INPUT 1	FLOAT X	DONE
LLR 02	INPUT 2	FLOAT Y	DONE
LLR 03	FUNCTION	POWER	DONE
LLR 04	FORMULA OF POWER	$X^Y$	DONE
LLR 05	OUTPUT	FLOAT 32 BIT	DONE

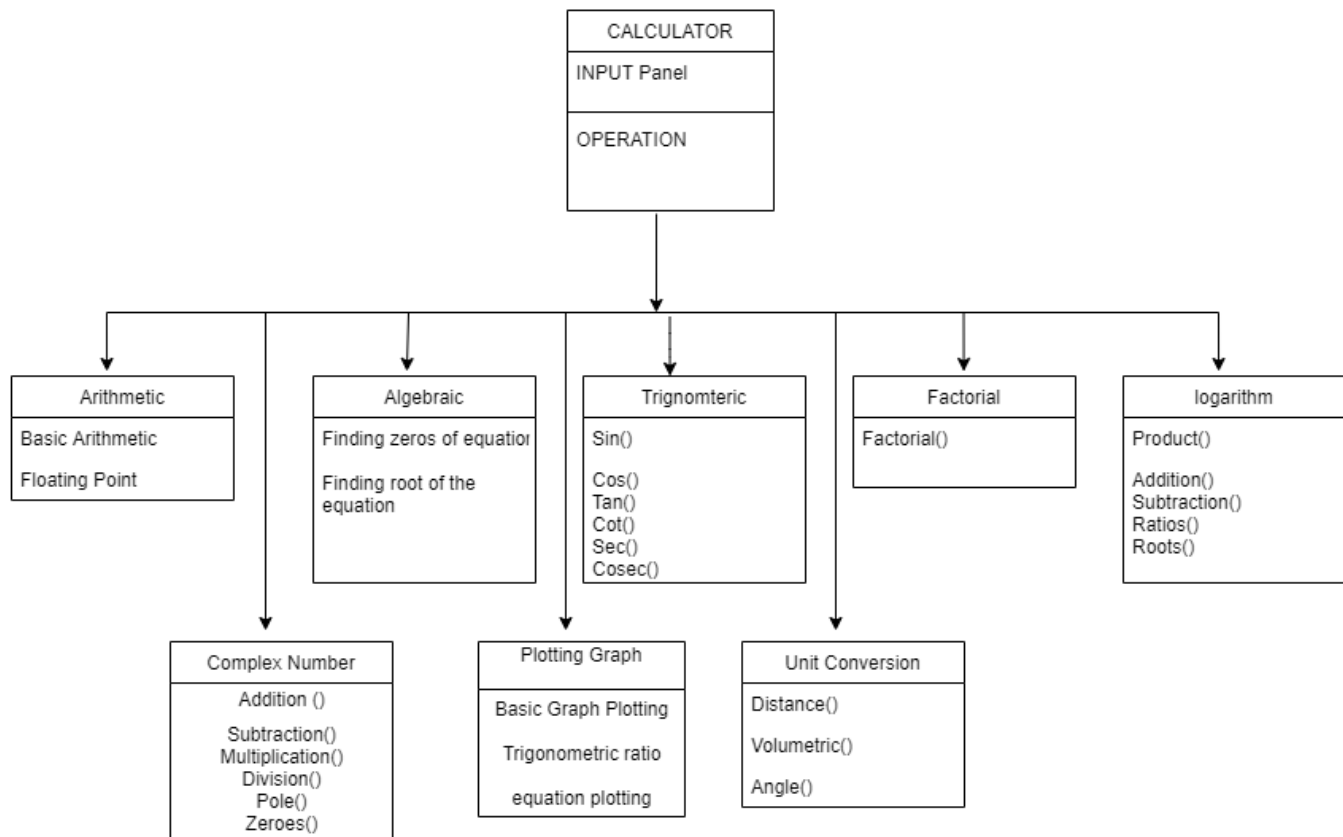
## Cube Root

CODE	REQUIREMENTS	DESCRIPTION	STATUS
LLR 01	INPUT 1	INT	DONE
LLR 02	FUNCTION	CUBE ROOT	DONE
LLR 03	FORMULA OF SQUAREROOT	$(X)^{(1/3)}$	DONE
LLR 04	OUTPUT	INT	DONE

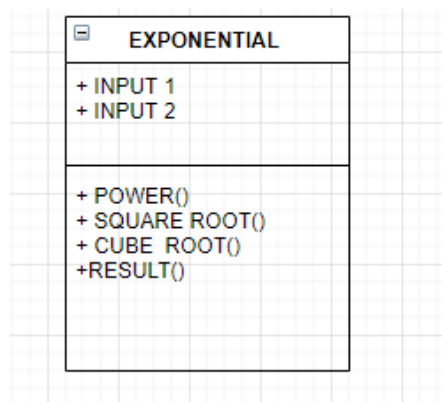
## 1.5 Design

## 1.6 Structural Diagram

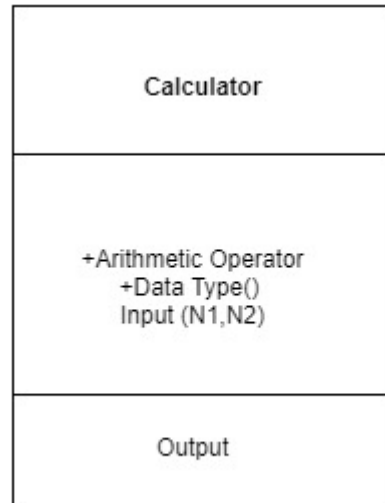
### 1.6.1 HLR of Calculator



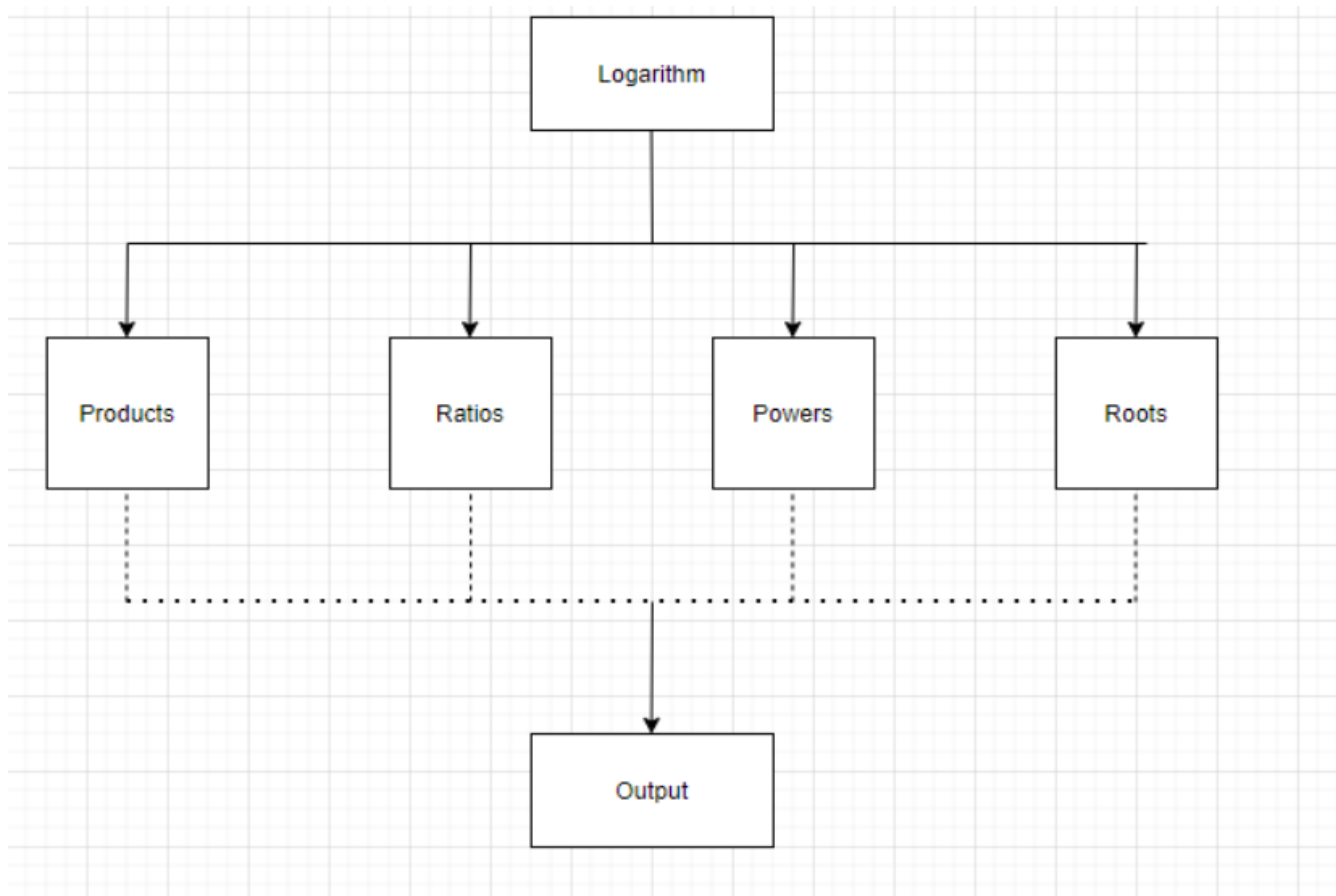
### 1.6.2 LLR Diagram(Exponential)



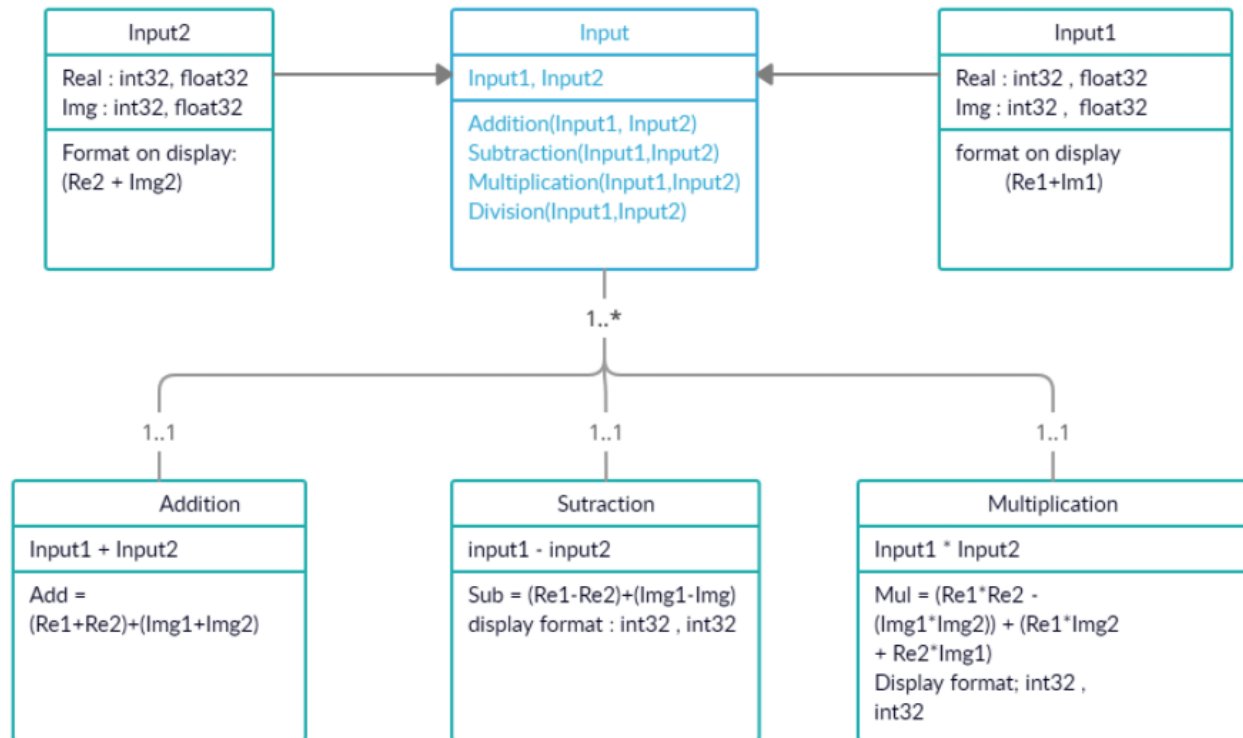
### 1.6.3 LLR Diagram (Calculator)



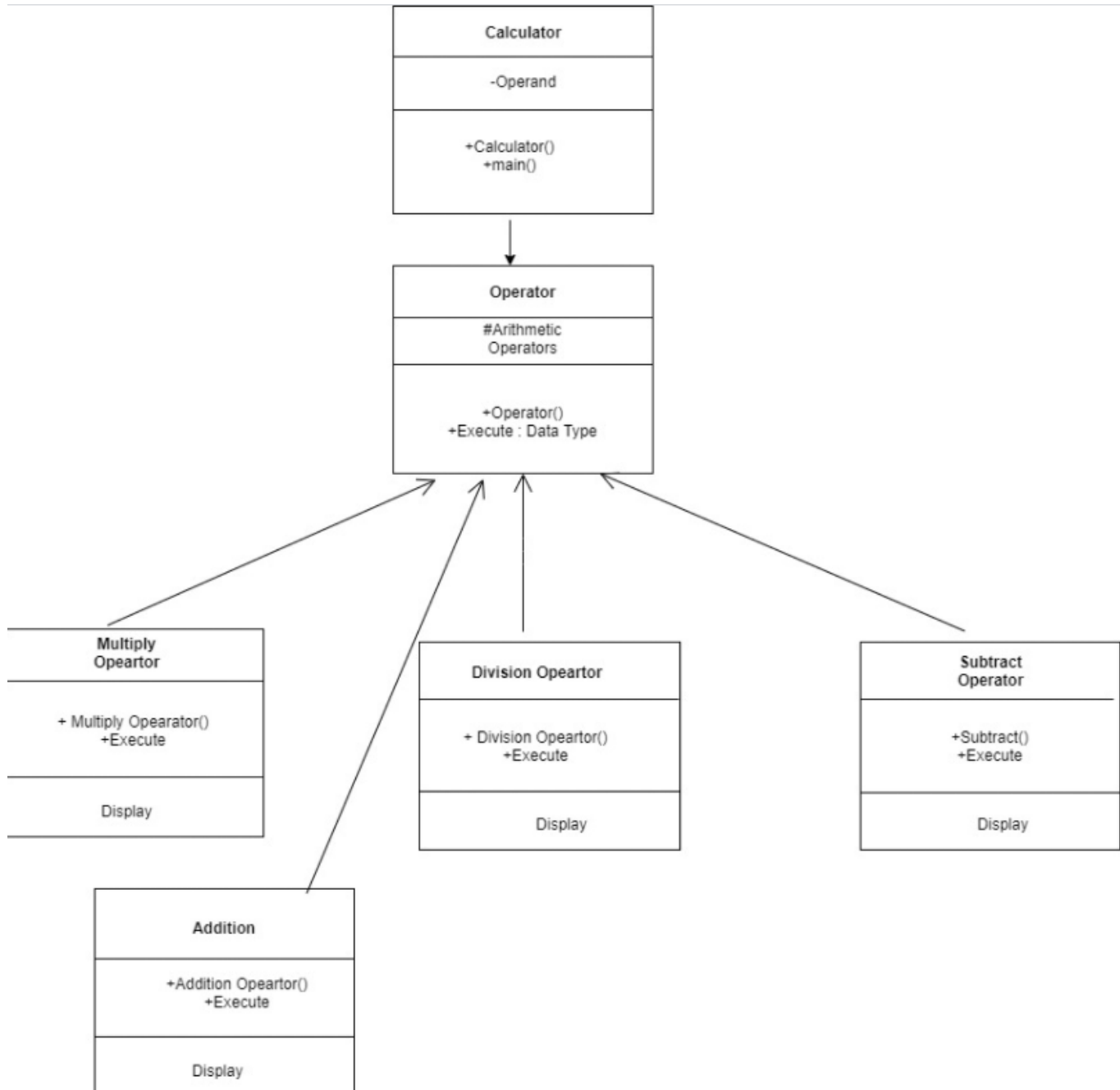
### 1.6.4 HLR Diagram(Logarithm)



### 1.6.6 LLR Diagram (Complex Number)



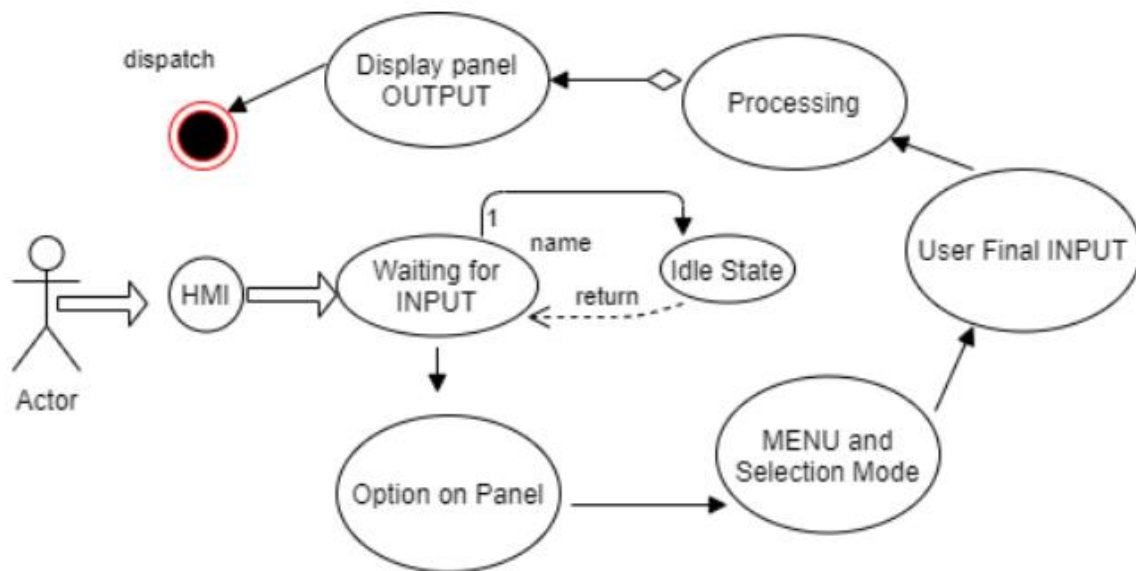
### 1.6.7 LLR Diagram(Arithmetic Operations)



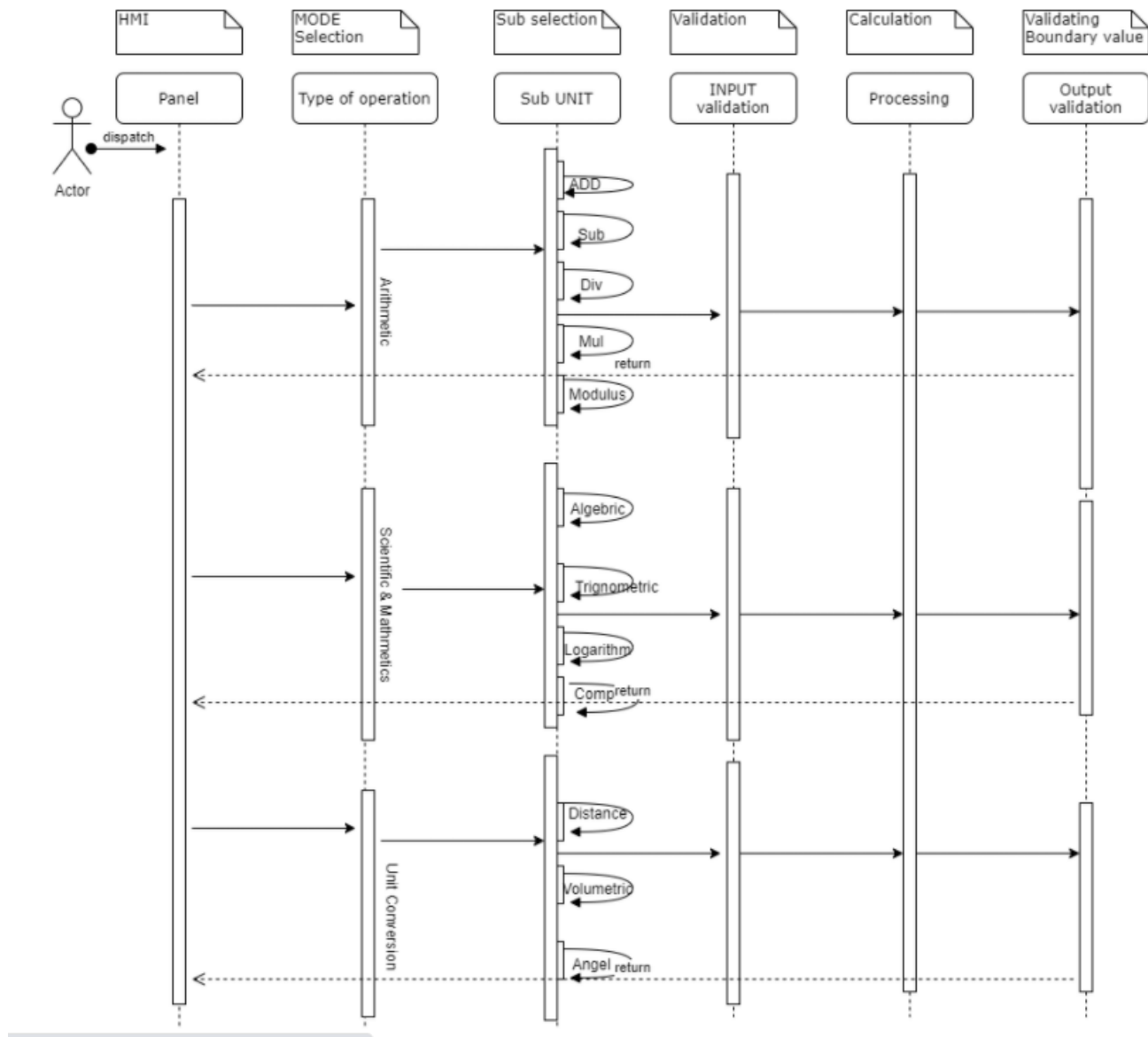


## 1.7 Behavioral Diagram

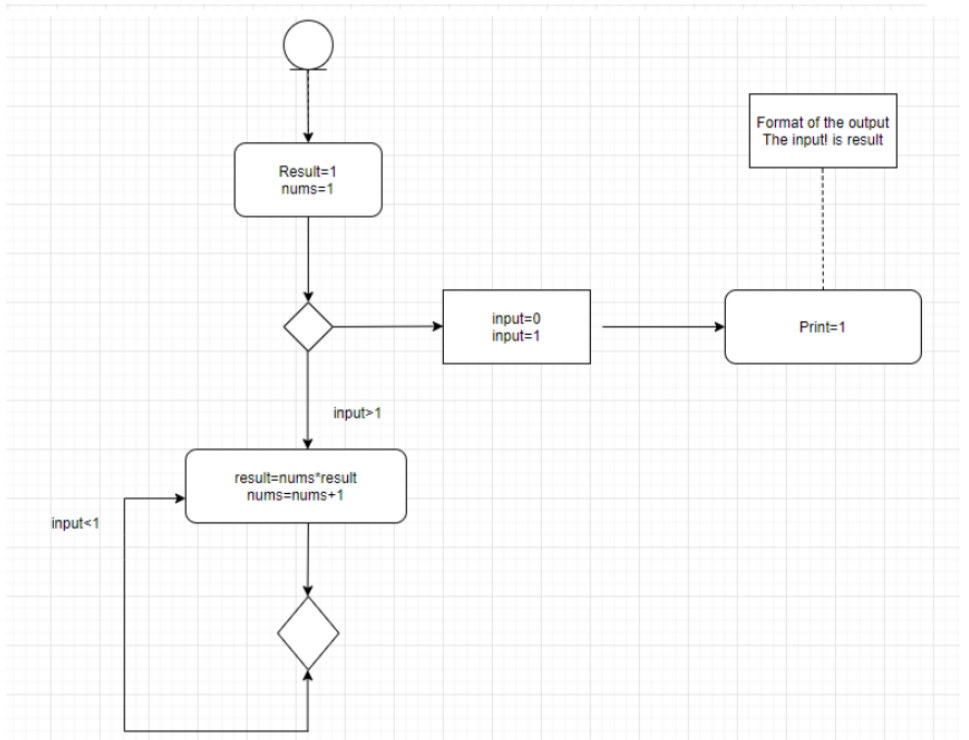
### 1.7.1 HLR Calculator Diagram (Flow control)



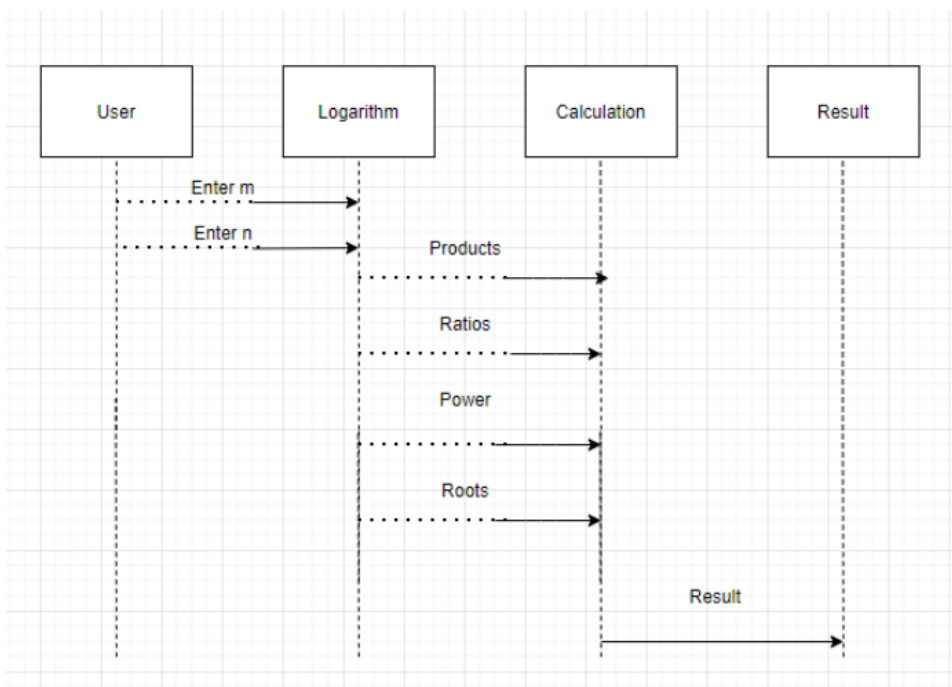
### 1.7.2 HLR Calculator Diagram (Sequence Diagram)



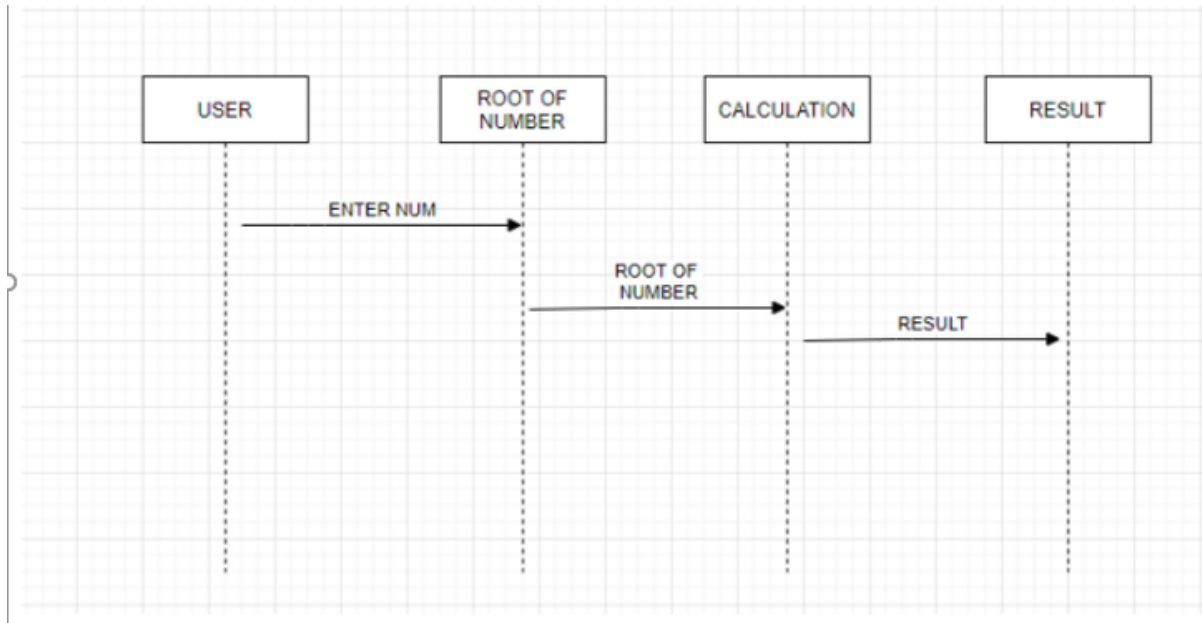
### 1.7.3 LLR Diagram (Factorial)



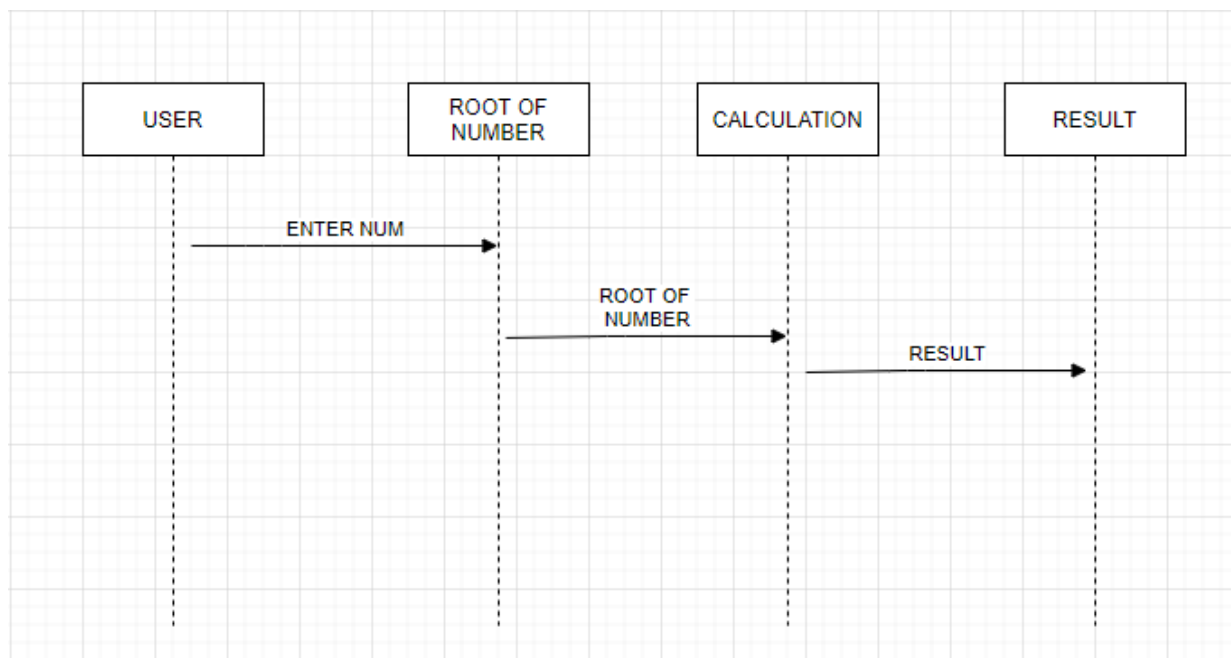
### 1.7.4 LLR Diagram (Logarithm)



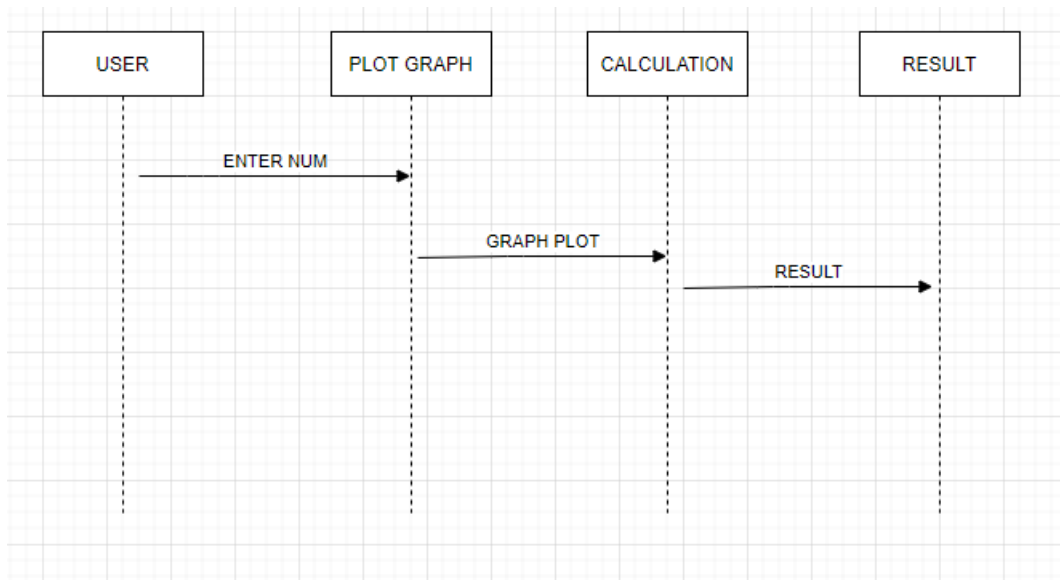
### 1.7.5 LLR Diagram (Finding a Power)



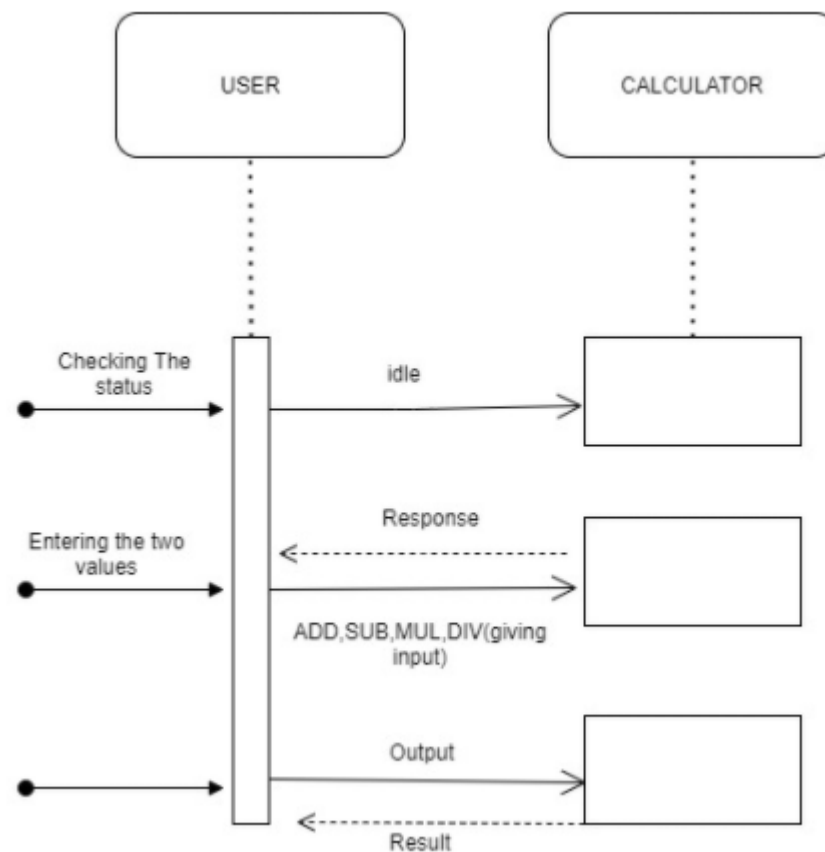
### 1.7.6 LLR Diagram (Finding Root of Number)



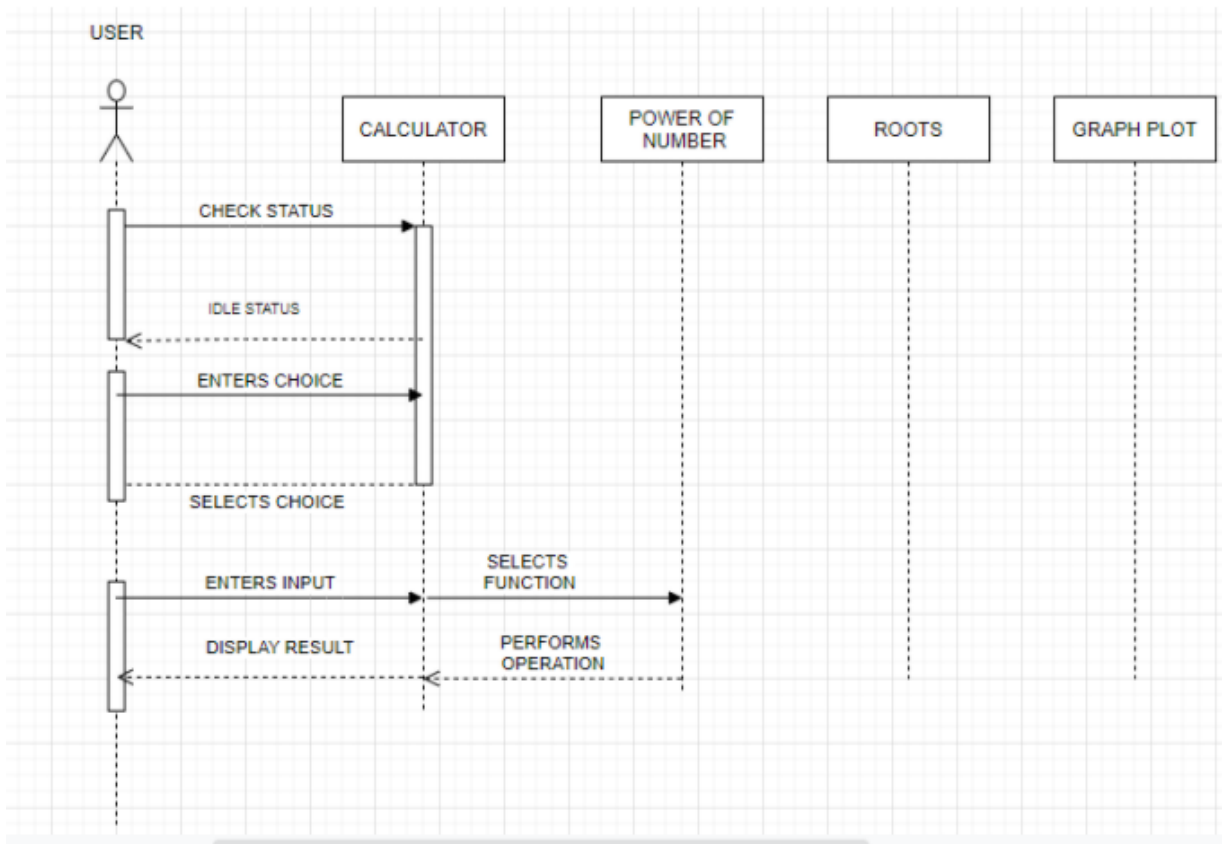
### 1.7.7 LLR Diagram (Plotting Graph)



### 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

Test ID	Description	Expected input	Expected Output	Actual Output	Type of Test
H_01	Perform Complex Number	3	Perform complex number Calculations Based on the Input	Getting right output	Requirement Based and Scenario Based
H_02	Factorial and Logarithm	4	Perform factorial and logarithm Calculations Based on the Input	Getting right output	Requirement Based and Scenario Based
H_03	Basic Arithmetic Operations	5	Perform basic arithmetic Calculations Based on the Input	Getting right output	Requirement Based and Scenario Based
H_04	Power and Roots	4	Perform power and roots Calculations Based on the Input	Getting right output	Requirement Based and Scenario Based

### 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 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

### 1.11 Git Link

[https://github.com/99003738/AppliedSDLC\\_N3.git](https://github.com/99003738/AppliedSDLC_N3.git)

## 1.12 Git Dashboard

### 1.12.1 Badges:-

Build	Cppcheck	Unity	[Git Inspector] (using github.io option)	Code Quality Dynamic	Code Coverage
					

## Folder Structure

Folder	Description
1_Requirements	Documents detailing requirements and research
2_Design	Documents specifying design details
3_Implementation	All code and documentation
4_Test_plan	Documents with test plans and procedures

## Contributors List and Summary

PS No.	Name	Features	Issues Raised	Issues Resolved	No Test Cases	Test Case Pass
99003737	Jeshwanth Kumar ega	Feature: Aithmetic Operations FLoating point airthmetic	2	2	5	5
99003738	Abhishek Mishra	Feature: Complex Number, Trigonometric Function	4	3	8	8
99003743	Yandapalli Priya Mansa	Feature: Index and Shurds, Basic Graph Plotting	2	2	5	5
99003746	Nitin N Shetty	Features: Factorial, Logarithm	4	4	9	9

### 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

## Contributors Git Inspector Result

```

19 ===== Git Inspector =====
20 Statistical information for the repository 'AppliedSDLC_NB' was gathered on
21 2021/03/15.
22 The following historical commit information, by author, was found:
23
24 Author                Commits    Insertions    Deletions    % of changes
25 99003737                41         212          128          4.77
26 99003738                82        5142         507         79.28
27 99003743                54         257          62          4.48
28 99003746                39         533         284         11.47
29
30 Below are the number of rows from each author that have survived and are still
31 intact in the current revision:
32
33 Author                Rows      Stability    Age          % in comments
34 99003737                128        60.4         1.0          12.50
35 99003738               4414        85.8         0.1           9.27
36 99003743                183        71.2         1.6          24.04
37 99003746                239        44.8         1.1           7.53
38
39 The following history timeline has been gathered from the repository:
40
41 Author                2021W06    2021W07    2021W10    2021W11
42 99003737                .          ++         -+
43 99003738               ++++++++  --+++++   ----++++  -++++
44 99003743                .          ++++      +         -+++++
45 99003746                .          ----++++  --++++
46 Modified Rows:         4829       640       1621       35
47

```

## Build

---


### 1.12.4 Set up for build

17 lines (13 sloc) | 251 Bytes

```
1  name: C/C++ CI - Build Status
2
3  on:
4    push:
5      branches: [ main ]
6    pull_request:
7      branches: [ main ]
8
9  jobs:
10   build:
11
12     runs-on: ubuntu-latest
13
14     steps:
15       - uses: actions/checkout@v2
16       - name: make
17         run: make -C 3_Implementation/ all
```

### 1.12.5 Set up for build


✓ Update README.md C/C++ CI - Build Status #152

 Summary

Jobs

✓ build

Triggered via push last month

99003743 pushed  b6ca1a4 main

Status

Success

Total duration

19s

Artifacts

-

c-cpp.yml

on: push

✓ build

5s

### Code quality and Issues or Bug Tracking

### 1.12.6 Set up for code Quality

```

name: CodeQuality Dynamic Code Analysis Valgrind
on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  test:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: apt install dependency
      run: |
        sudo apt-get -y install valgrind
        sudo apt-get -y install libcunit1 libcunit1-doc libcunit1-dev
    - name: make test
      run: make -C 3_Implementation/ test
    - name: Valgrid
      run: valgrind ./3_Implementation/build/Test_Calculator_N3.out

```

### 1.12.7 Outcome of code quality

✓ Update README.md CodeQuality Dynamic Code Analysis Valgrind #24

Summary

Jobs

✓ test

Triggered via push last month

99003743 pushed  b6ca1a4 [main](#)

Status

Success

Total duration

39s

Artifacts

—

CodeQuality\_Dynamic.yml

on: push

✓ test

28s

### Unit Testing

#### 1.12.8 Set up for unity testing

```
name: Unity - Unit Testing

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  test:

    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2
      - name: make
        run: make -C 3_Implementation/ test
```



### 1.12.9 Outcome for unity testing

#### ✓ Update README.md Unity - Unit Testing #152

##### Summary

Jobs

✓ test

Triggered via push last month

99003743 pushed → b6ca1a4 main

Status

Success

Total duration

16s

Artifacts

—

unity.yml

on: push

✓ test

4s

### 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- 2. 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

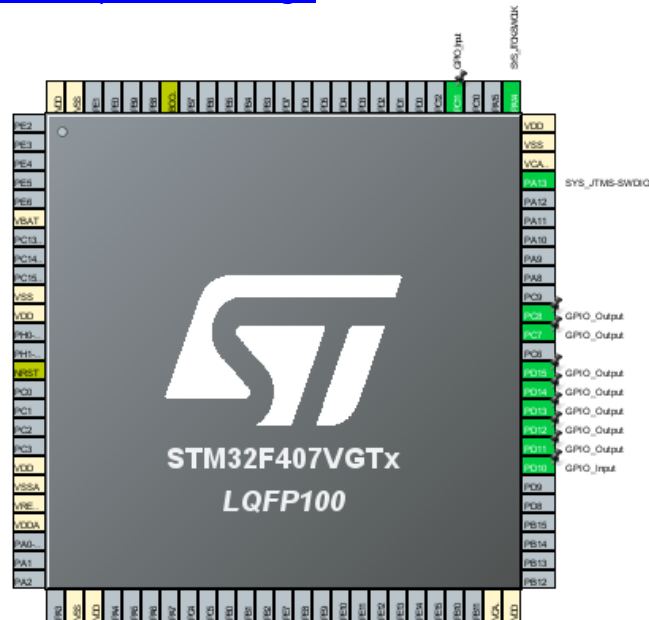
### 2.4 Test Plan

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

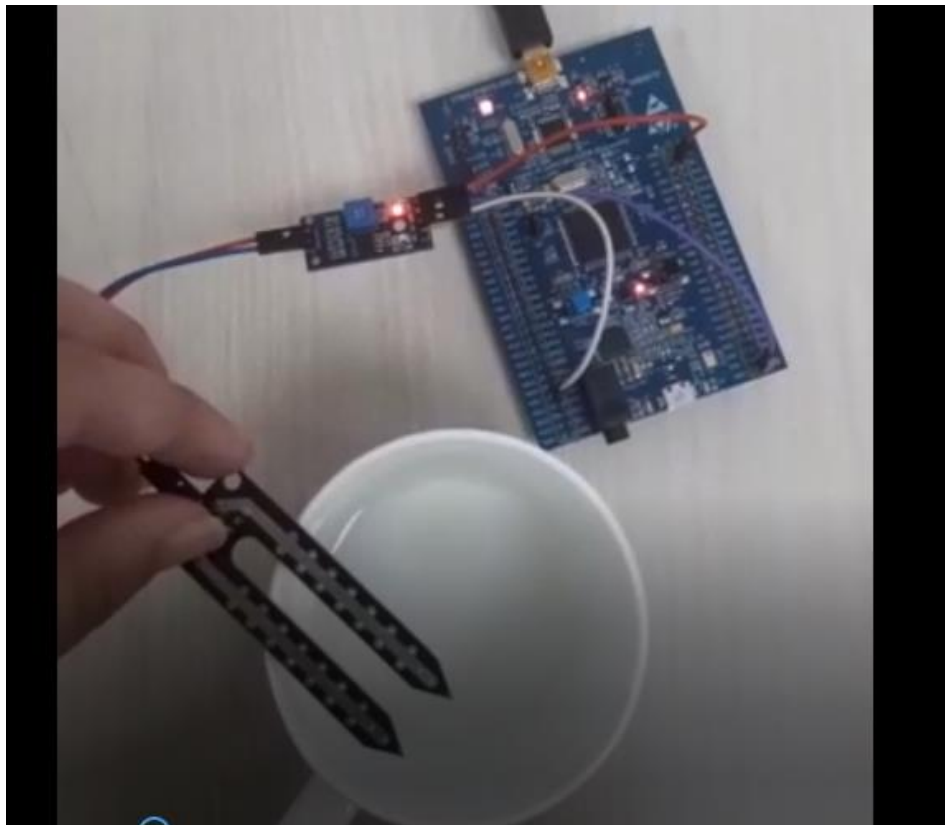
## 2.6 GIT Link

[https://github.com/99003755/Embedded\\_C.git](https://github.com/99003755/Embedded_C.git)

## 2.7 Design



### 2.7.1 Wiper Control



## 2.7.2 Headlight Control



## 2.8 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.

## 2.9 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 detect 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.

## **2.10 Summary**

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

## **2.11 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 data from different sheets and adding retrieved data in Master Sheet.

#### 3.3 Topic and Subtopics

- |                  |                            |
|------------------|----------------------------|
| <b>Subtopics</b> | ➤ Basic Python             |
|                  | ➤ 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

#### High-Level Requirement

Code	Requirements	Description	Status
HLR-001	Excel Sheets	Collecting Data's from 5 Different Sheets	Implemented
HLR-002	Processing	Processing input Data from Sheets	Implemented
HLR-003	Write down Data	Write Correspond Data to Master Sheet	Implemented
HLR-004	Plotting Graph	Plotting Bar Graph	Implemented

### 3.5.3 Low Level Requirement

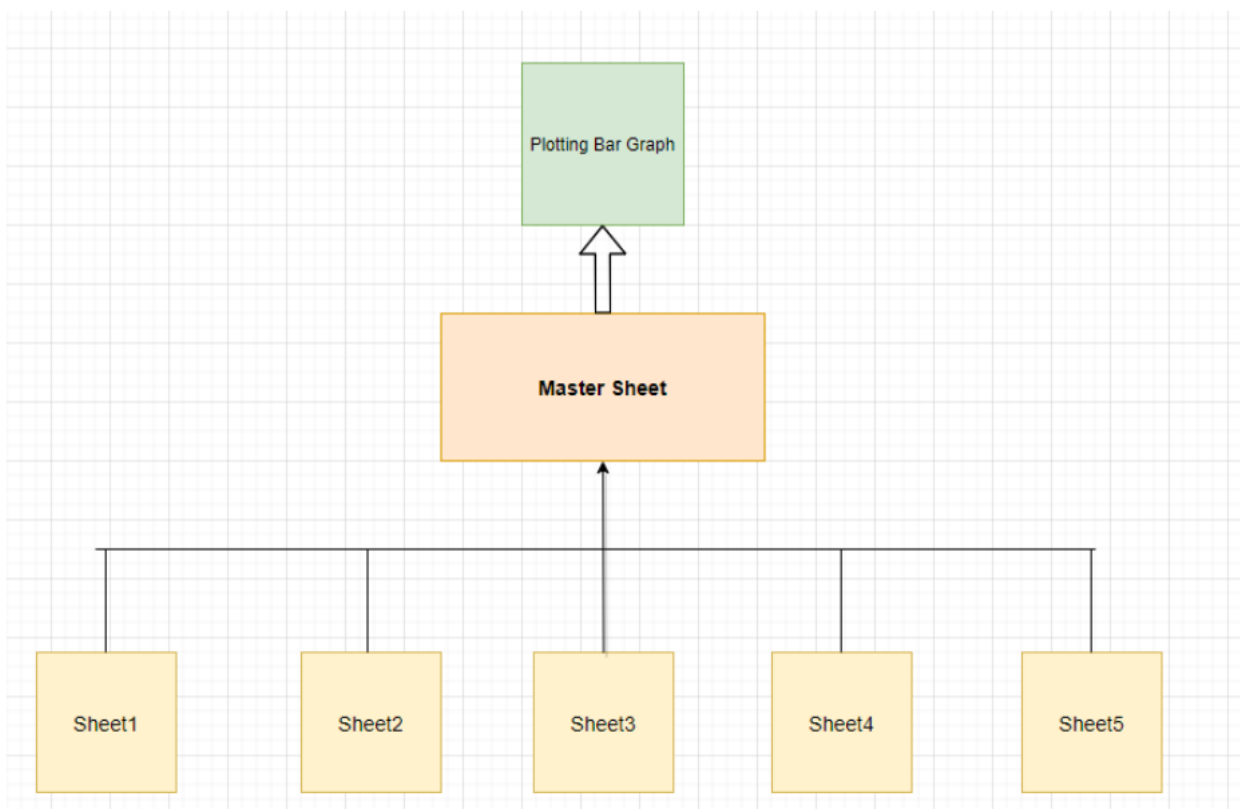
#### Low-Level Requirements

Code	Requirements	Description	Status
LLR-001	Input (Searching Multiple of Data From 5 Sheets)	Need to search a Particular data from Five Different Sheets	Implemented
LLR-002	Validating Data from Different Sheets	Reading Different Sheets of corresponding Data which is present in Excel-Sheet	Implemented
LLR-003	After Taking user input write down particular data in Master-Sheet	Using openpyxl library as per user input collect all the different data which is present in 5 sheets write down in Master-Sheet	Implemented
LLR-004	Store all the data's in allocate sheet	After Successfully writing Data in Master Sheet as per user requirement store it or display all the Details	Implemented
LLR-005	Plotting Bar Graph	After Storing Data in Master Sheet, representing data by plotting Bar graph	Implemented

## 3.6 Design

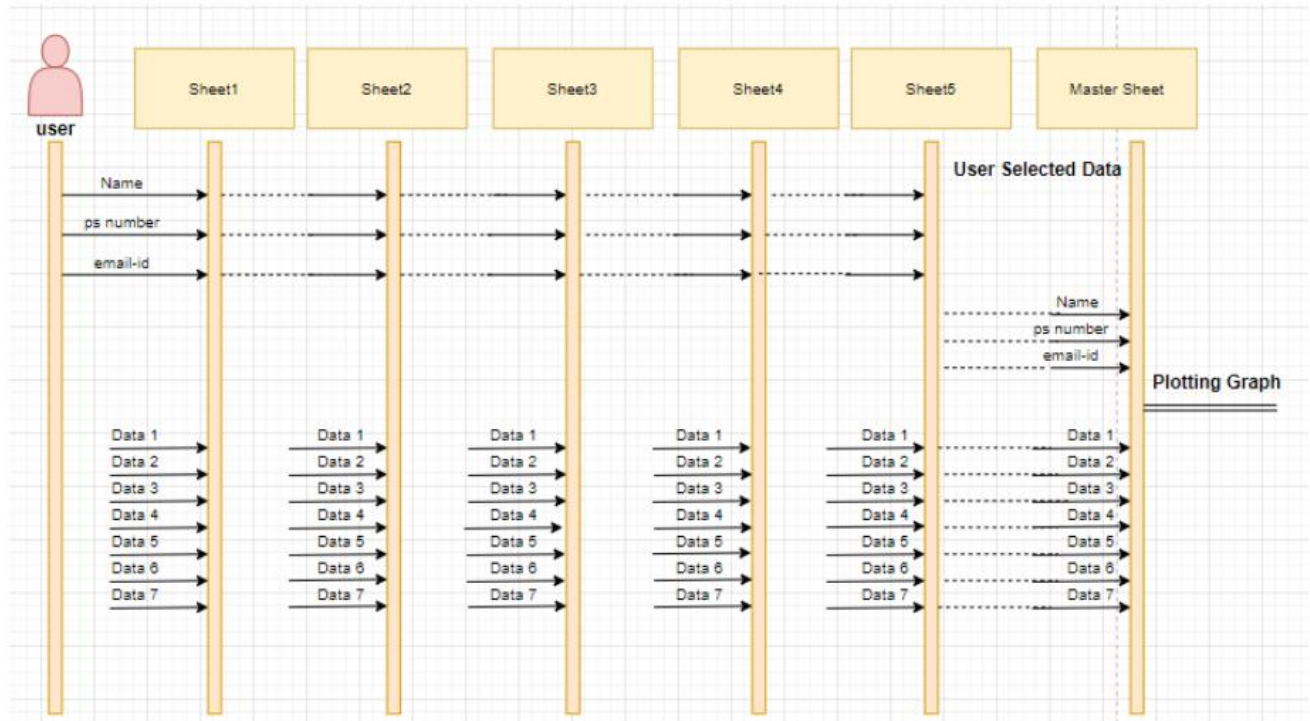
### 3.6.1 High Level Design

#### Object Diagram



### 3.6.2 Low Level Design

#### Sequence Diagram



### 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](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/mmc" 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 system call). System call displays a required output.

#### 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.