./

GENESIS - Learning Outcome & Mini-project Summary Report



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **To be Approved** | **Remarks/Revision Details** |
|  | 19th April 2021 | Shrinidhi V Katti  PS no. 99003751 |  |  |  |

**Details**

Contents

[Contents 3](#_Toc69720033)

[1.0 Mini Project -1 Team SDLC (System Development life cycle) 5](#_Toc69720034)

[1.1 Modules Used 5](#_Toc69720035)

[1.2 Project title: Calculator 5](#_Toc69720036)

[1.3 Topic and Subtopics 5](#_Toc69720037)

[1.4 Objectives & Requirements 5](#_Toc69720038)

[1.4.1 High Level requirement analysis 5](#_Toc69720039)

[1.4.2 Low Level requirement analysis 6](#_Toc69720040)

[1.5 Design 6](#_Toc69720041)

[1.5.1 High level diagram: Deployment Diagram 6](#_Toc69720042)

[1.5.2 LLR Diagram: Composite diagram. 7](#_Toc69720043)

[1.5.3 LLR Diagram: Profile diagram. 8](#_Toc69720044)

[1.5.4 LLR Diagram: Communication diagram. 8](#_Toc69720045)

[1.5.5 LLR Diagram: Object diagram. 9](#_Toc69720046)

[1.5.6 LLR Diagram: Activity diagram. 10](#_Toc69720047)

[1.5.7 LLR Diagram: Deployment diagram. 10](#_Toc69720048)

[1.5.8 LLR Diagram: UML Class diagram. 11](#_Toc69720049)

[1.5.9 LLR Diagram: State diagram. 11](#_Toc69720050)

[1.6 Test Plan 12](#_Toc69720051)

[1.6.1 High Level Test Plan 12](#_Toc69720052)

[1.6.2 Low Level Test Plan 13](#_Toc69720053)

[1.7 Implementation Summary 13](#_Toc69720054)

[1.8 Git Link 13](#_Toc69720055)

[1.9 Git Dashboard 13](#_Toc69720056)

[1.9.1 Badges 13](#_Toc69720057)

[1.9.2 Git Inspector 14](#_Toc69720058)

[1.9.3 Setup for Build 14](#_Toc69720059)

[1.9.4 Outcome of the Build 15](#_Toc69720060)

[1.9.5 Setup for Code Quality 15](#_Toc69720061)

[1.9.6 Outcome of Code Quality 16](#_Toc69720062)

[1.9.7 Setup for Unity Testing 17](#_Toc69720063)

[1.9.8 Outcome of Unity Testing 17](#_Toc69720064)

[1.10 Individual Contribution & Highlights 17](#_Toc69720065)

[1.11 Summary 18](#_Toc69720066)

[1.11.1 Outcomes: 18](#_Toc69720067)

[1.12 Challenges faced and how were they overcome 18](#_Toc69720068)

[2.0 Miniproject -2 [Group] – Embedded C 19](#_Toc69720069)

[2.1 Module: 19](#_Toc69720070)

[2.2 Objectives & Requirements 19](#_Toc69720071)

[2.2.1 OBJECTIVE: 19](#_Toc69720072)

[2.2.2 REQUIREMENTS: 19](#_Toc69720073)

[2.3 Test Plan 19](#_Toc69720074)

[2.4 Implementation Summary 20](#_Toc69720075)

[2.5 Individual Contribution & Highlights 21](#_Toc69720076)

[2.6 Summary 21](#_Toc69720077)

[3.0 Miniproject -3 [Individual] – Python 22](#_Toc69720078)

[3.1 Module 22](#_Toc69720079)

[3.2 Project title: Retrieve data from multiple Excel sheet 22](#_Toc69720080)

[3.3 Topic and Subtopics 22](#_Toc69720081)

[3.4 Objectives: 22](#_Toc69720082)

[3.5 Requirements: 22](#_Toc69720083)

[3.5.1 High Level requirement analysis 22](#_Toc69720084)

[3.5.2 Low Level requirement analysis 22](#_Toc69720085)

[3.6 Design 23](#_Toc69720086)

[3.6.1 Use Case LLR Diagram 23](#_Toc69720087)

[3.6.2 Object HLR Diagram 24](#_Toc69720088)

[3.7 Implementation Summary 24](#_Toc69720089)

[3.8 Git Link 25](#_Toc69720090)

[3.9 Summary – 25](#_Toc69720091)

[3.9.1 Outcomes: 25](#_Toc69720092)

[3.10 Challenges faced and how were they overcome 25](#_Toc69720093)

[4.0 Miniproject -4 [Individual] – Kernel Programming and Device Drivers 26](#_Toc69720094)

[4.1 Module/s: 26](#_Toc69720095)

[4.2 Topic and Subtopics: 26](#_Toc69720096)

[4.3 Objectives & Requirements: 26](#_Toc69720097)

[4.3.1 Requirements: 27](#_Toc69720098)

[4.4 Implementation Summary: 27](#_Toc69720099)

[4.4.1 Hands-on Activity that are implemented are as follow: 27](#_Toc69720100)

[4.4.2 User space code: 27](#_Toc69720101)

[4.4.3 kthread examples: 27](#_Toc69720102)

[4.5 Git Link: 28](#_Toc69720103)

[4.6 Summary: 28](#_Toc69720104)

[4.7 Challenges faced and how were they overcome: 28](#_Toc69720105)

# 1.0 Mini Project -1 Team SDLC (System Development life cycle)

## **1.1 Modules Used**

Modules used in this project are SDLC and C programming.

## **1.2 Project title: Calculator**

Modules linked to the mini project Ex – Linux, SDLC and C.

## **Topic and Subtopics**

* The core steps of SDLC is being implemented.
* The features of Calculator are implemented.
* The testing has been done for each function.
* Introduction about SDLC
* C Programming
* Code Analysis
  + CPP Check
  + Valgrind
* Testing
  + Unity Testing
* Makefile
* V Model
* Git Hub

## **1.4 Objectives & Requirements**

### 1.4.1 High Level requirement analysis

* Any calculator must be efficient.
* Any calculator must have a user-friendly interface.
* It should also be accurate in terms of results.
* It should be able to perform multiple functions.
* It must be cost efficient.

|  |  |  |
| --- | --- | --- |
| ID | Description | Status |
| HLR01 | Basic Arithmetic Calculation | Implemented |
| HLR02 | Trigonometric Calculation | Implemented |
| HLR03 | Dimension Conversion | Implemented |
| HLR04 | Binary Conversion | Implemented |

### 1.4.2 Low Level requirement analysis

|  |  |  |
| --- | --- | --- |
| ID | Description | Status |
| LLR01 | For Arithmetic Conversion  Addition  Subtraction  Multiplication  Division  modulus | Implemented |
| LLR02 | For Trigonometric Conversion,  Sine, Cosine, Tan, Cot, Sec,  Co-sec. | Implemented |
| LLR03 | For Binary Conversion,  Binary to decimal and hex,  Decimal to hex and binary,  Conversion range of word size. | Implemented |
| LLR04 | Dimension Conversion,  Length conversion,  Mass conversion,  Temperature conversion,  Floating values. | Implemented |

## **1.5 Design**

### 1.5.1 High level diagram: Deployment Diagram

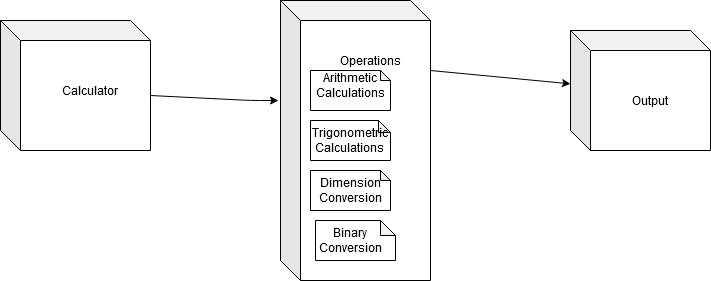


Figure : Deployment Diagram.

### 1.5.2 LLR Diagram: Composite diagram.

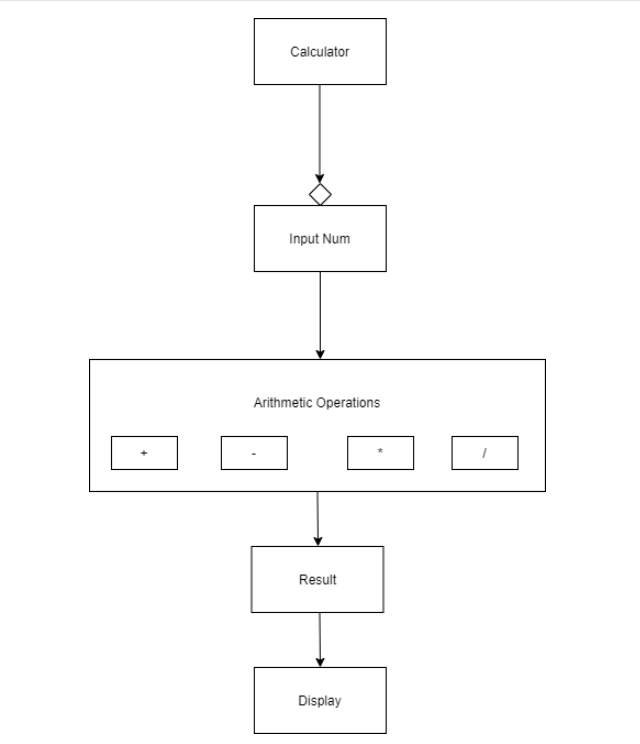


Figure :Composite Diagram

### 1.5.3 LLR Diagram: Profile diagram.

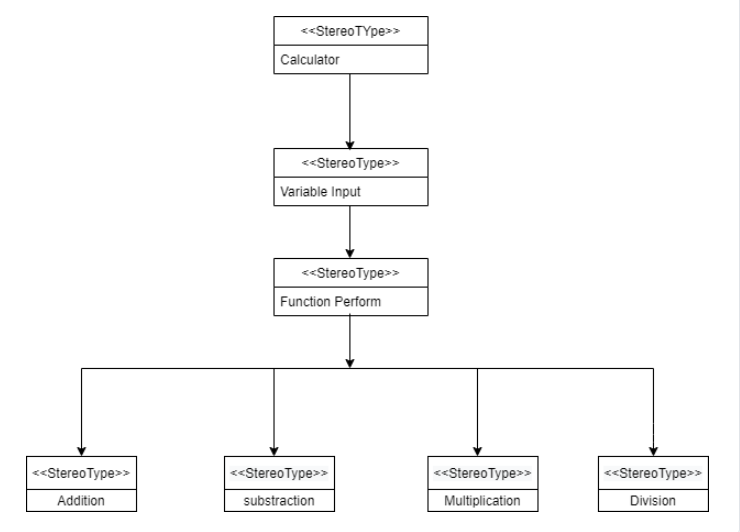


Figure : Profile Diagram

### 1.5.4 LLR Diagram: Communication diagram.

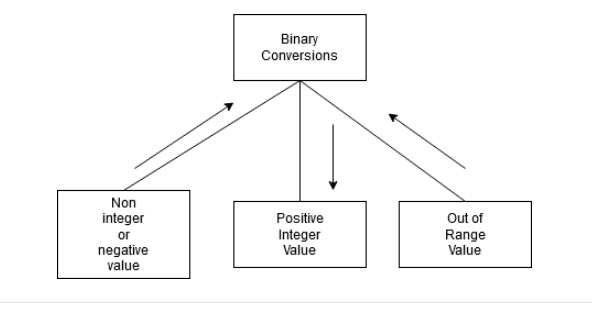


Figure :Communication Diagram

### 1.5.5 LLR Diagram: Object diagram.

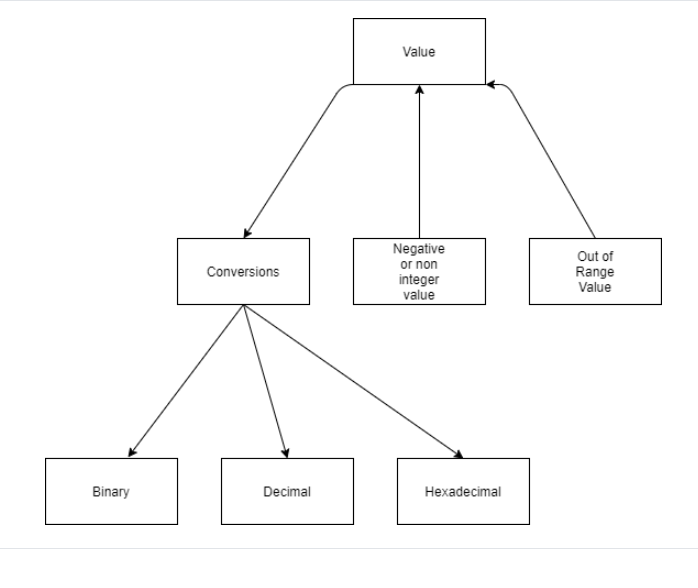


Figure : Object Diagram

### 1.5.6 LLR Diagram: Activity diagram.

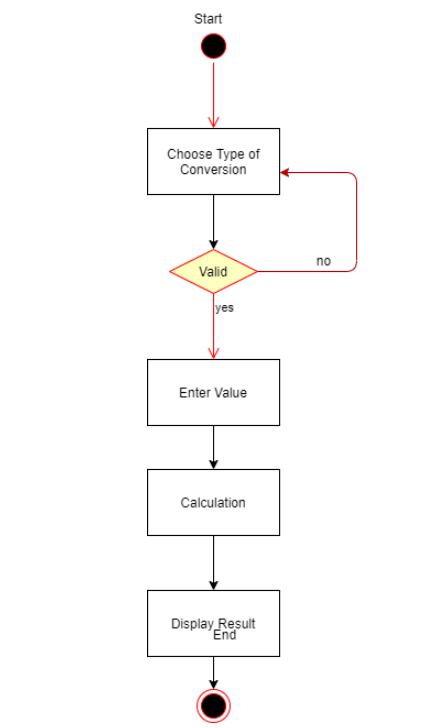


Figure : Activity Diagram

### 1.5.7 LLR Diagram: Deployment diagram.

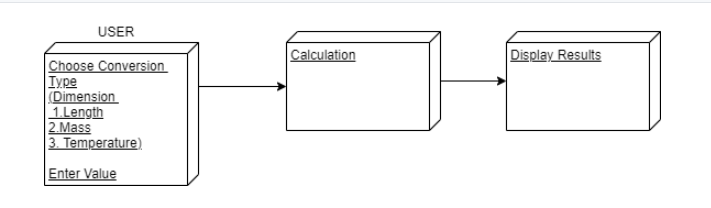


Figure : Deployment Diagram

### 1.5.8 LLR Diagram: UML Class diagram.

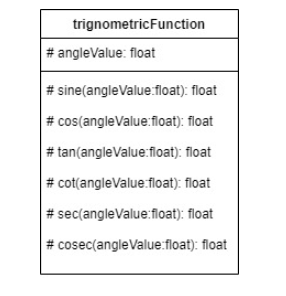


Figure : Class Diagram

### 1.5.9 LLR Diagram: State diagram.

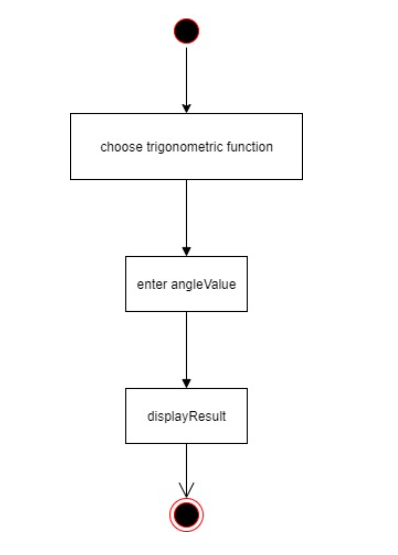


Figure : State Diagram

## 1.6 Test Plan

### 1.6.1 High Level Test Plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Description | Expected input | Expected Output | Actual Output | Type of Test |
| H\_01 | Perform Trigonometric Calculation | 4 | Perform Trigonometric Calculations Based on the Input | Getting right output | Scenario Based |

### 1.6.2 Low Level Test Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Id | Input | Expected output | Actual output | Status(pass/fail) |
| T1 | Sine 30 | 0.5 |  |  |
| T2 | 111(in binary) | 7(decimal), 7(hex) |  |  |
| T3 | gm to Kg(1000g) | 1 Kg |  |  |
| T4 | Addition (15,8) | 23 |  |  |
| T5 | Division (18,9) | 2 |  |  |

## **1.7 Implementation Summary**

It is a basic calculator that will allow users to perform operations in Mathematics Addition, Subtraction, Multiplication, Division, Trigonometry, Factorial, Area, Volume etc. However, the input has to be in the form "number1 operator1 number2 operator2 number3" (i.e. 2+4\*10). The input values can be from any integer to even a number with decimals. Moreover, this calculator is smart enough to operate multiplication/division before addition/subtraction, in another word it is implemented with the order of precedence logic.

## **1.8 Git Link**

<https://github.com/99003751/N4_SDLC_Calci.git>

## **1.9 Git Dashboard**

### 1.9.1 Badges

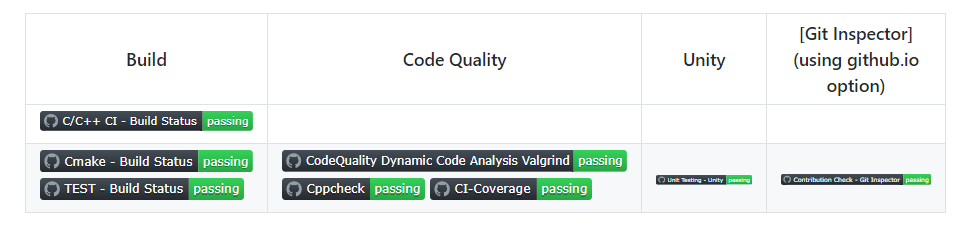


Figure : Badges

#### Git inspector summary

### 1.9.2 Git Inspector

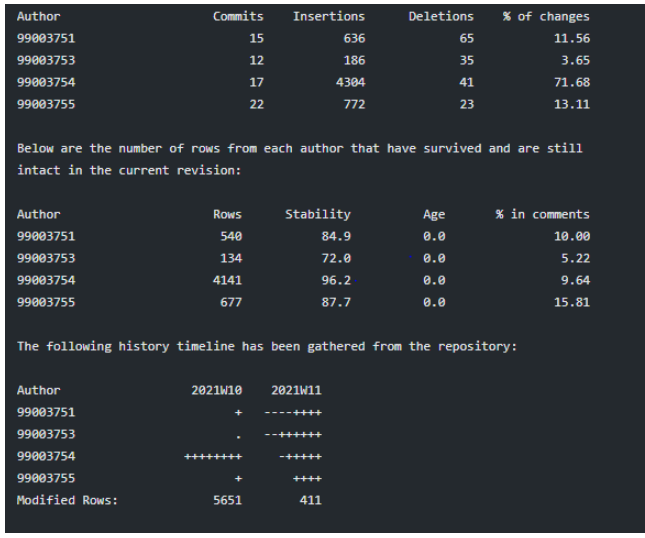


Figure : Git inspector

#### Build

### 1.9.3 Setup for Build

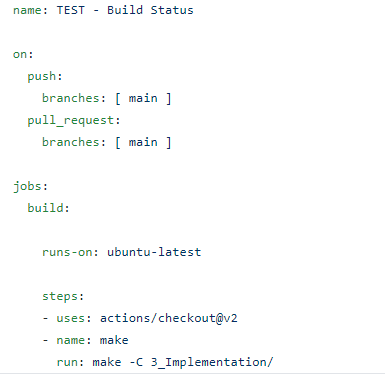


Figure : Build

### 1.9.4 Outcome of the Build

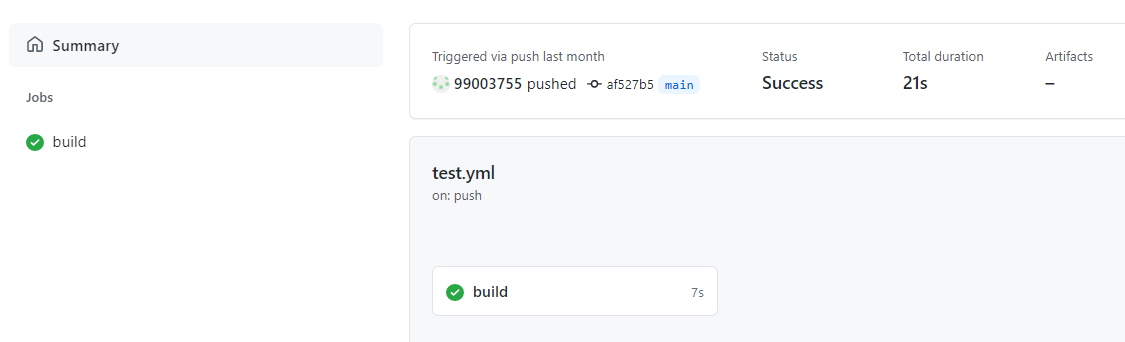


Figure : Build results

#### 1.14 Code quality and Issues or Bug Tracking

### 1.9.5 Setup for Code Quality

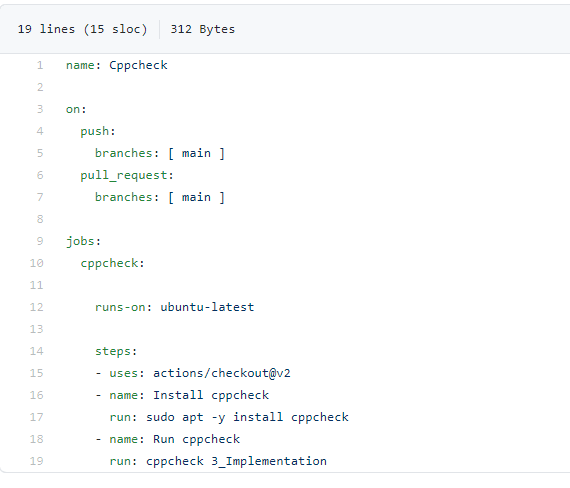


Figure :Code quality

### 1.9.6 Outcome of Code Quality

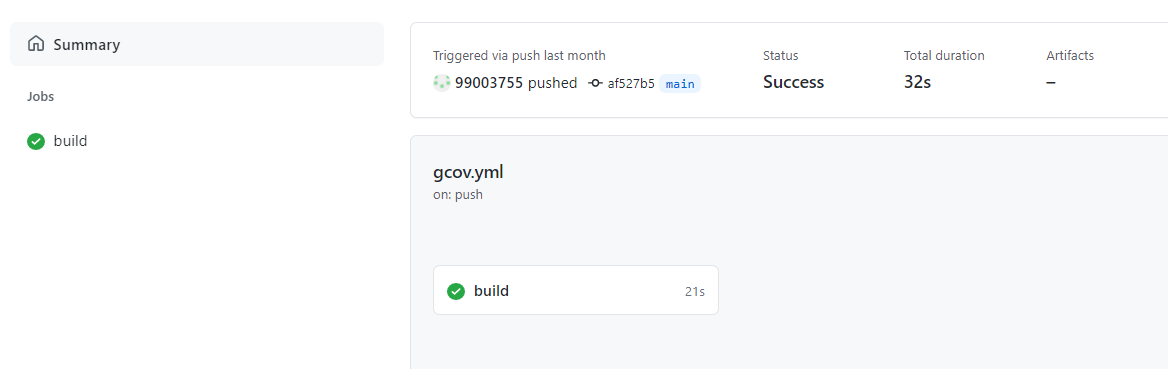


Figure : code quality results

#### 1.15 Unit Testing

### 1.9.7 Setup for Unity Testing



Figure : Unity testing

### 1.9.8 Outcome of Unity Testing

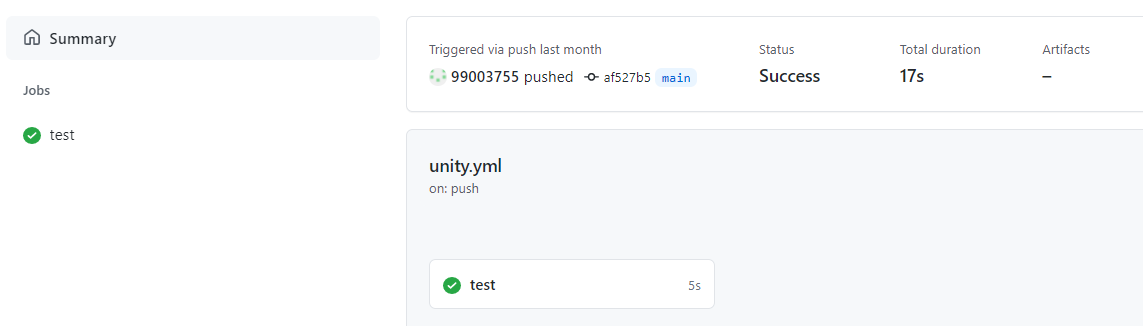
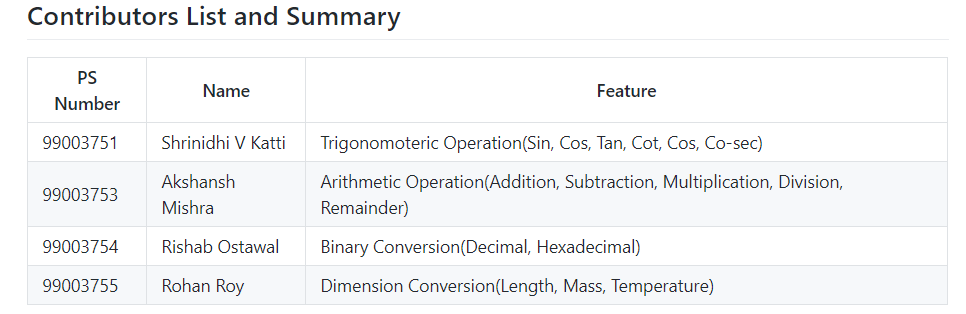


Figure : unity testing results

## **1.10 Individual Contribution & Highlights**

* Trigonometry functionalities implemented.
* Test case for the same is implemented.
* High level and low-level test cases is implemented for the same.
* Issue raised and the issue was solved.
* Helped during the workflow's implementation of the project.



## **1.11 Summary**

### 1.11.1 Outcomes:

Technical:

* Improved implementation of ‘C’ concepts.
* Practical implementation of SDLC lifecycle.
* Source code management. (Github )

Soft skills:

1. Project management

2. Conflict management.

## **1.12 Challenges faced and how were they overcome**

* Differentiation of high level and low level.
* Committing to GitHub, pull and push in GitHub.
* Converting pictures & tables into readme file.
* Cpp check and Unity testing.

# 2.0 Miniproject -2 [Group] – Embedded C

## **2.1 Module:**

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

**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.

## **2.2 Objectives & Requirements**

### 2.2.1 OBJECTIVE:

To implement body control module functionalities using STM32 development board.

### 2.2.2 REQUIREMENTS:

* + STM32 development board.
  + LDR sensor
  + Push buttons
  + Bread board
  + buzzer
  + Jumper wires
  + LEDs

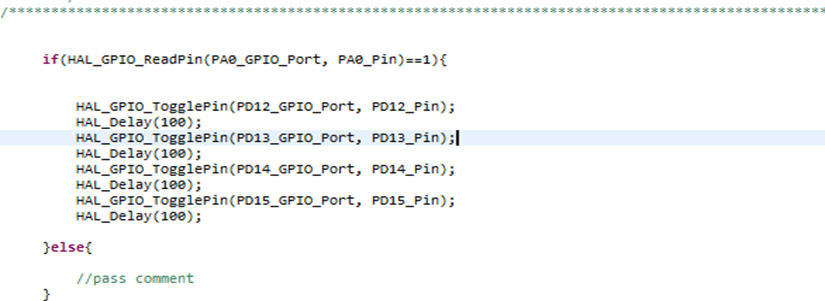
## **2.3 Test Plan**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SL No** | **TEST\_ID** | **Testing function** | **Expected input** | **Expected Output** |
| **1** | **ID\_1** | Wiper control system. | When switch is pressed | Wiper starts |
| **2.** | **ID\_2** | Interior door light | When push button is pressed | Door light turns off indicating all doors are locked. |
| **3.** | **ID\_3** | Power window module | When switch is pressed | Window opens |
| **4.** | **ID\_4** | Seat belt warning system | When switch is open | Buzzer will be on |
| **5.** | **ID\_5** | Side mirror control system. | When button is pressed | Side mirror rotates (inwards and outwards) |
| **6.** | **ID\_6** | Automatic head light system. | Light intensity to LDR sensor | Head light turns on. |

**Table 3: High Level Test plans**

## **2.4 Implementation Summary**

The code for two functionalities was written by each team member respectively and finally was integrated at the end. The integrated code was dumped on the STM32 board.



**Figure : Basic logic for Implementation**

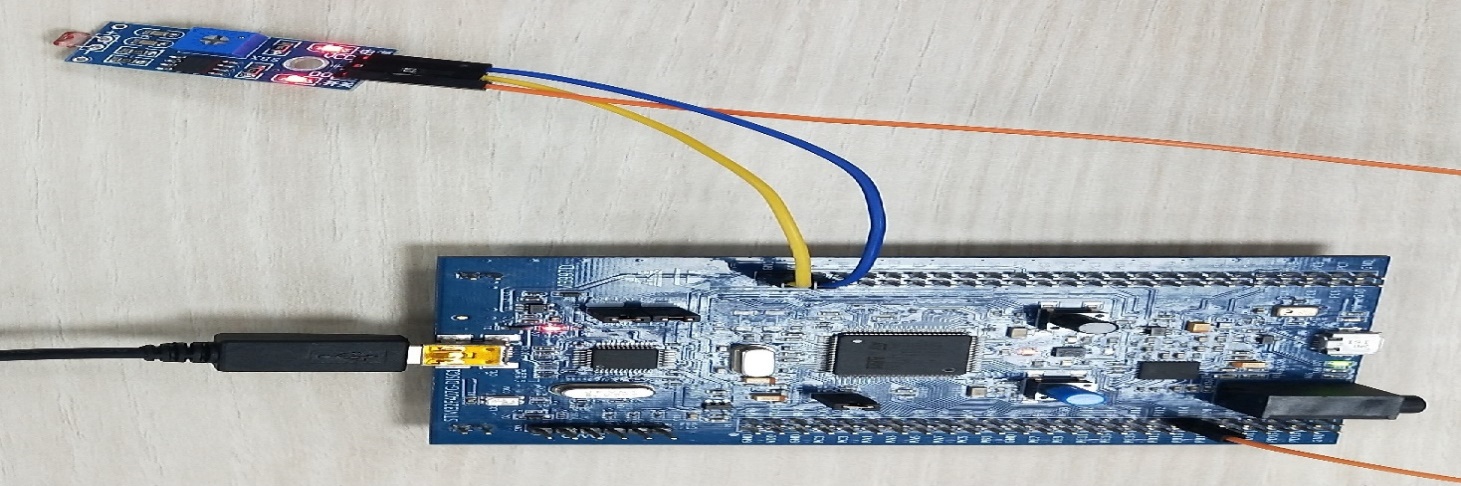


Figure : Implementation

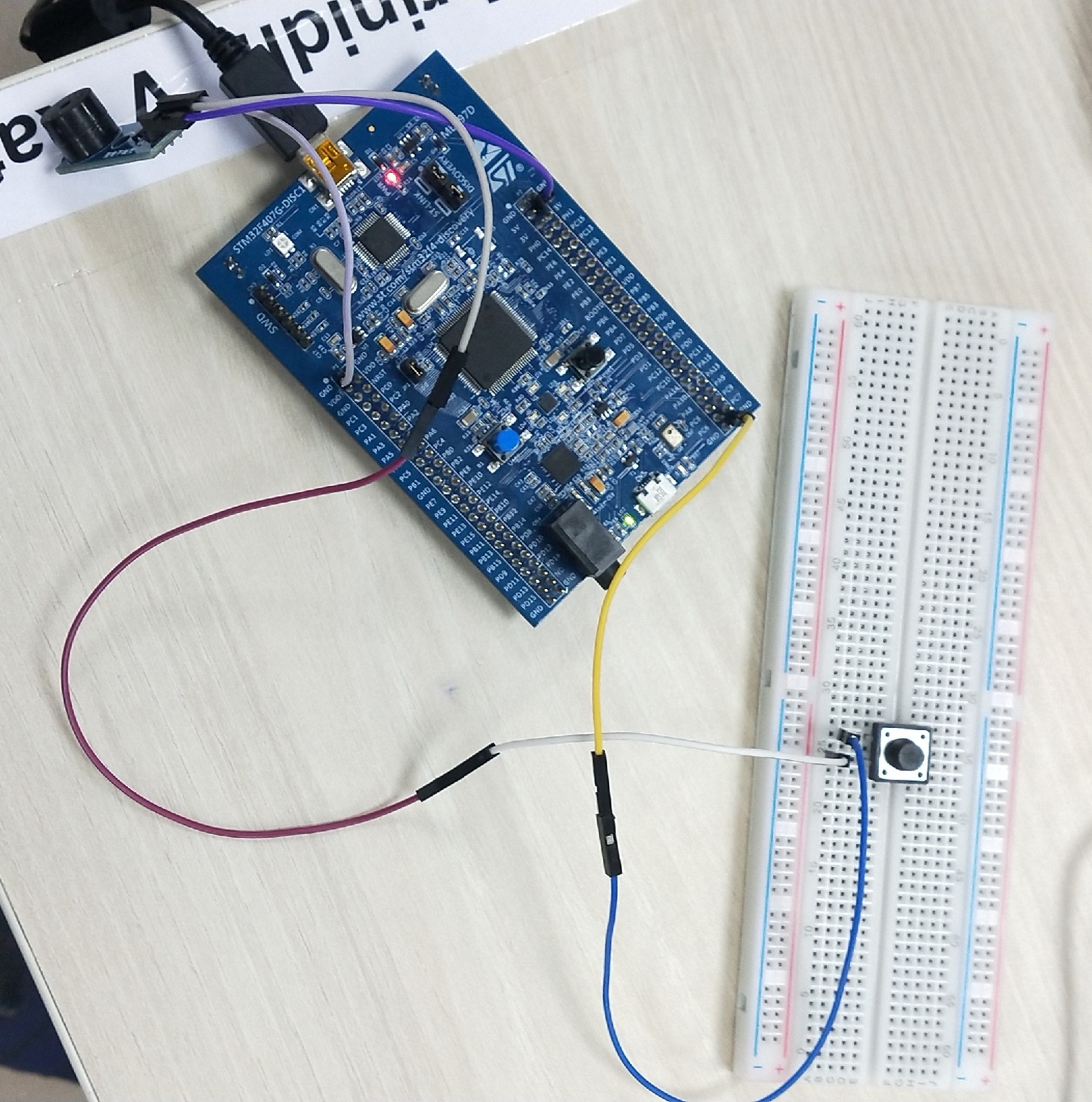


Figure : Switch Implementation

## **2.5 Individual Contribution & Highlights**

1. Interior door light system.
2. Wiper Control module.

## **2.6 Summary**

Learning outcomes:

* Programming the STM32 board.
* Designing embedded application using HAL API.

The code was written individually and integrated together by one of the teammate and code was dumped on the hardware. The output of each system was show in the form of LEDs.

**Challenges faced and how were they overcome**

* With less components showing the output was challenging.
* Code integration.

# 3.0 Miniproject -3 [Individual] – Python

## **3.1 Module**

Modules used in this project are Core and Advanced Python.

## **3.2 Project title:** Retrieve data from multiple Excel sheet

## **3.3 Topic and Subtopics**

* Openpyxl
  + Read excel file
  + Write excel file
* Barchart
* functions

## **3.4 Objectives:**

To extract the data present in different spreadsheets in one excel file as required by the user.

## **3.5 Requirements**:

### 3.5.1 High Level requirement analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Requirements | Description | Status |
| HL1 | Searching Data | Search all data from 5 sheets when user defines the PS number to be searched. | Implemented |
| HL2 | writing to excel | Write all the data from different spreadsheet in one master sheet | Implemented |
| HL3 | extracting user defined data | Write required data in the excel file. | Implemented |
| HL4 | plotting the bar chart | plot the bar chart of the data present in the master sheet. | Implemented |

### 3.5.2 Low Level requirement analysis

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **Requirements** | **Description** | **Status** |
| LL1 | Search Parameters | The user defines the Name or PS Number or Email Id of the data to be searched | Implemented |
| LL2 | Searching Data in excel file in every sheet | The data to be searched is defined by the user. | Implemented |
| LL3 | writing the data into master sheet | Data defined by user has to be extracted from 5 different spreadsheets and put into one master sheet. | Implemented |
| LL4 | Plotting the bar chart | plot the bar chart of the data present in the master sheet considering rows and columns. | Implemented |

## 3.6 **Design**

### 3.6.1 Use Case LLR Diagram

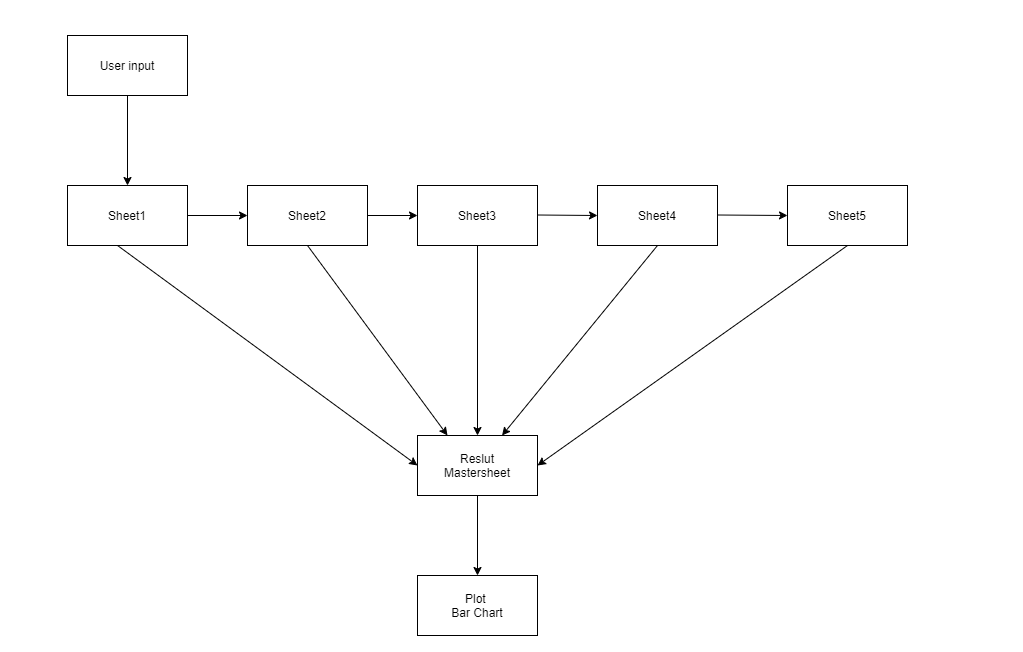


Figure : Use case diagram

### 3.6.2 Object HLR Diagram

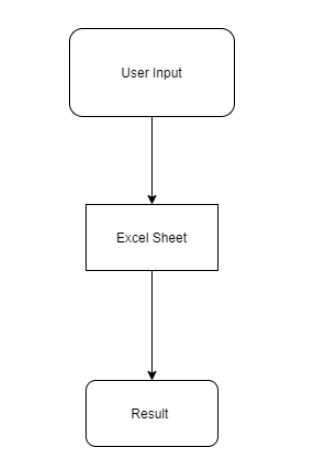


Figure : Object diagram

## **3.7 Implementation Summary**

The aim of the project is to extract the data present in different spreadsheets in one excel file as required by the user. The excel sheet consists of 5 spreadsheets. The user defines the data that needs to be searched on the basis of Name, PS Number and Email ID. The python program then reads the data corresponding to the particular data from different spreadsheets of excel. It then creates a master sheet and adds the data from all the sheets to it. In the end, it will plot the bar graph from the data present in the master sheet.

## **3.8 Git Link**

<https://github.com/99003763/Python_Mini_Project>

## **3.9 Summary –**

As per the objective, firstly I created an excel file that has five sheets. Then I started populating the sheets with data like NAME, PS NUMBER, Email, D.O.B, etc. The NAME, EMAIL ID and PS NUMBER column is common throughout all the five sheets.

Now coming to the code, here I have used openpyxl library. I have also followed Object Oriented Programming approach and used classes, object and functions in order to implement the code. For example the user is giving the input of NAME, PS NO and EMAIL ID and if it matches then it will print all the data of the candidate available in all the 5 sheets combined in the master sheet 1 of the masterbook.

### 3.9.1 Outcomes:

Technical:

* Improved implementation of Python concepts.
* Practical implementation of SDLC lifecycle.
* Source code management. (GitHub)

Soft skills:

1. Project management

2. Conflict management.

## **3.10 Challenges faced and how were they overcome**

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

# 

# 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:**

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

* 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/99003751/Embedded-Linux_and_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 user space 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.