

GENESIS - Learning Outcome & Mini-project Summary Report



LTTTS
GLOBAL
ENGINEERING
ACADEMY



L&T Technology Services



Details

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

Contents

CONTENTS	3
1.0 MINIPROJECT -1 TEAMS DLC (SYSTEM DEVELOPMENT LIFE CYCLE)	5
1.1 MODULES USED.....	5
1.2 PROJECT TITLE: CALCULATOR	5
1.3 TOPIC AND SUBTOPICS.....	5
1.4 OBJECTIVES & REQUIREMENTS.....	5
1.4.1 High Level requirement analysis	5
1.4.2 Low Level requirement analysis.....	6
1.5 DESIGN	6
1.5.1 High level diagram: Deployment Diagram.....	6
1.5.2 LLR Diagram: Composite diagram.	7
1.5.3 LLR Diagram: Profile diagram.	8
1.5.4 LLR Diagram: Communication diagram.	8
1.5.5 LLR Diagram: Object diagram.....	9
1.5.6 LLR Diagram: Activity diagram.	10
1.5.7 LLR Diagram: Deployment diagram.....	10
1.5.8 LLR Diagram: UML Class diagram.....	11
1.5.9 LLR Diagram: State diagram.....	11
1.6 TEST PLAN	12
1.6.1 High Level Test Plan.....	12
1.6.2 Low Level Test Plan.....	13
1.7 IMPLEMENTATION SUMMARY	13
1.8 GIT LINK.....	13
1.9 GIT DASHBOARD.....	13
1.9.1 Badges.....	13
1.9.2 Git Inspector.....	14
1.9.3 Setup for Build	14
1.9.4 Outcome of the Build	15
1.9.5 Setup for Code Quality.....	15
1.9.6 Outcome of Code Quality.....	16
1.9.7 Setup for Unity Testing	17
1.9.8 Outcome of Unity Testing.....	17
1.10 INDIVIDUAL CONTRIBUTION & HIGHLIGHTS.....	17
1.11 SUMMARY.....	18
1.11.1 Outcomes:.....	18
1.12 CHALLENGES FACED AND HOW WERE THEY OVERCOME	18
2.0 MINIPROJECT -2 [GROUP] – EMBEDDED C.....	19
2.1 MODULE:.....	19
2.2 OBJECTIVES & REQUIREMENTS.....	19
2.2.1 OBJECTIVE:.....	19
2.2.2 REQUIREMENTS:.....	19
2.3 TEST PLAN.....	19
2.4 IMPLEMENTATION SUMMARY	20
2.5 INDIVIDUAL CONTRIBUTION & HIGHLIGHTS.....	21
2.6 SUMMARY.....	21
3.0 MINIPROJECT -3 [INDIVIDUAL] – PYTHON.....	22

3.1 MODULE.....	22
3.2 PROJECT TITLE: RETRIEVE DATA FROM MULTIPLE EXCEL SHEET.....	22
3.3 TOPIC AND SUBTOPICS	22
3.4 OBJECTIVES:.....	22
3.5 REQUIREMENTS:.....	22
3.5.1 High Level requirement analysis.....	22
3.5.2 Low Level requirement analysis.....	22
3.6 DESIGN	23
3.6.1 Use Case LLR Diagram.....	23
3.6.2 Object HLR Diagram.....	24
3.7 IMPLEMENTATION SUMMARY	24
3.8 GIT LINK.....	25
3.9 SUMMARY –.....	25
3.9.1 Outcomes:.....	25
3.10 CHALLENGES FACED AND HOW WERE THEY OVERCOME	25
4.0 MINIPROJECT -4 [INDIVIDUAL] – KERNEL PROGRAMMING AND DEVICE DRIVERS.....	26
4.1 MODULE/s:.....	26
4.2 TOPIC AND SUBTOPICS:	26
4.3 OBJECTIVES & REQUIREMENTS:.....	26
4.3.1 Requirements:.....	27
4.4 IMPLEMENTATION SUMMARY:	27
4.4.1 Hands-on Activity that are implemented are as follow:.....	27
4.4.2 User space code:.....	27
4.4.3 kthread examples:.....	27
4.5 GIT LINK:.....	28
4.6 SUMMARY:.....	28
4.7 CHALLENGES FACED AND HOW WERE THEY OVERCOME:.....	28

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.

1.3 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

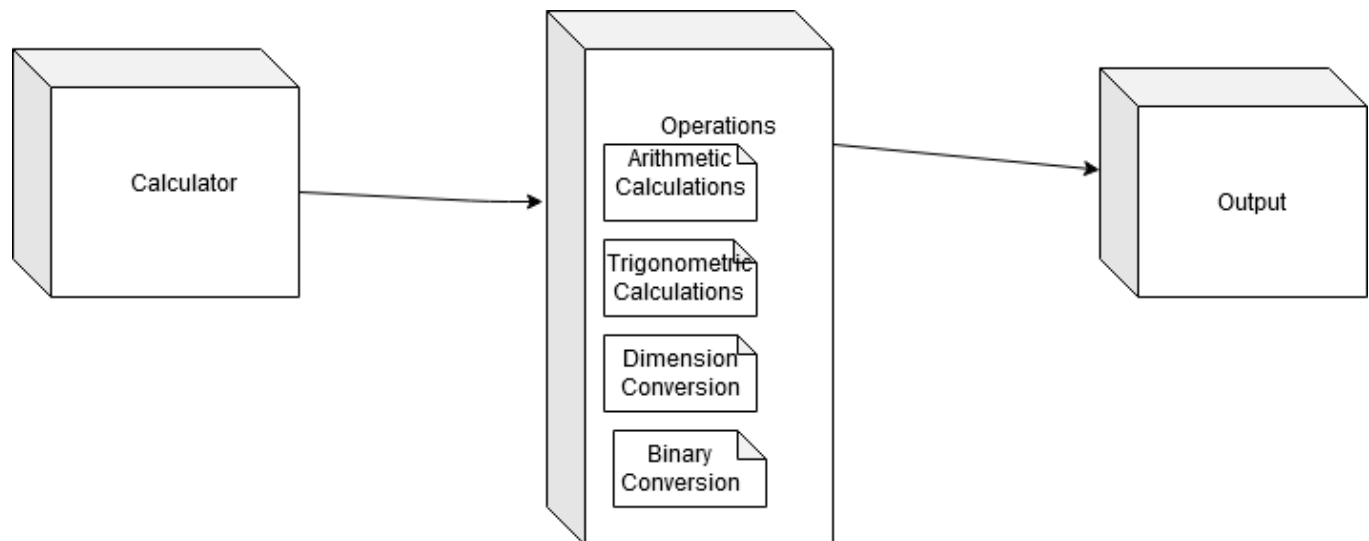


Figure 1: Deployment Diagram.

1.5.2 LLR Diagram: Composite diagram.

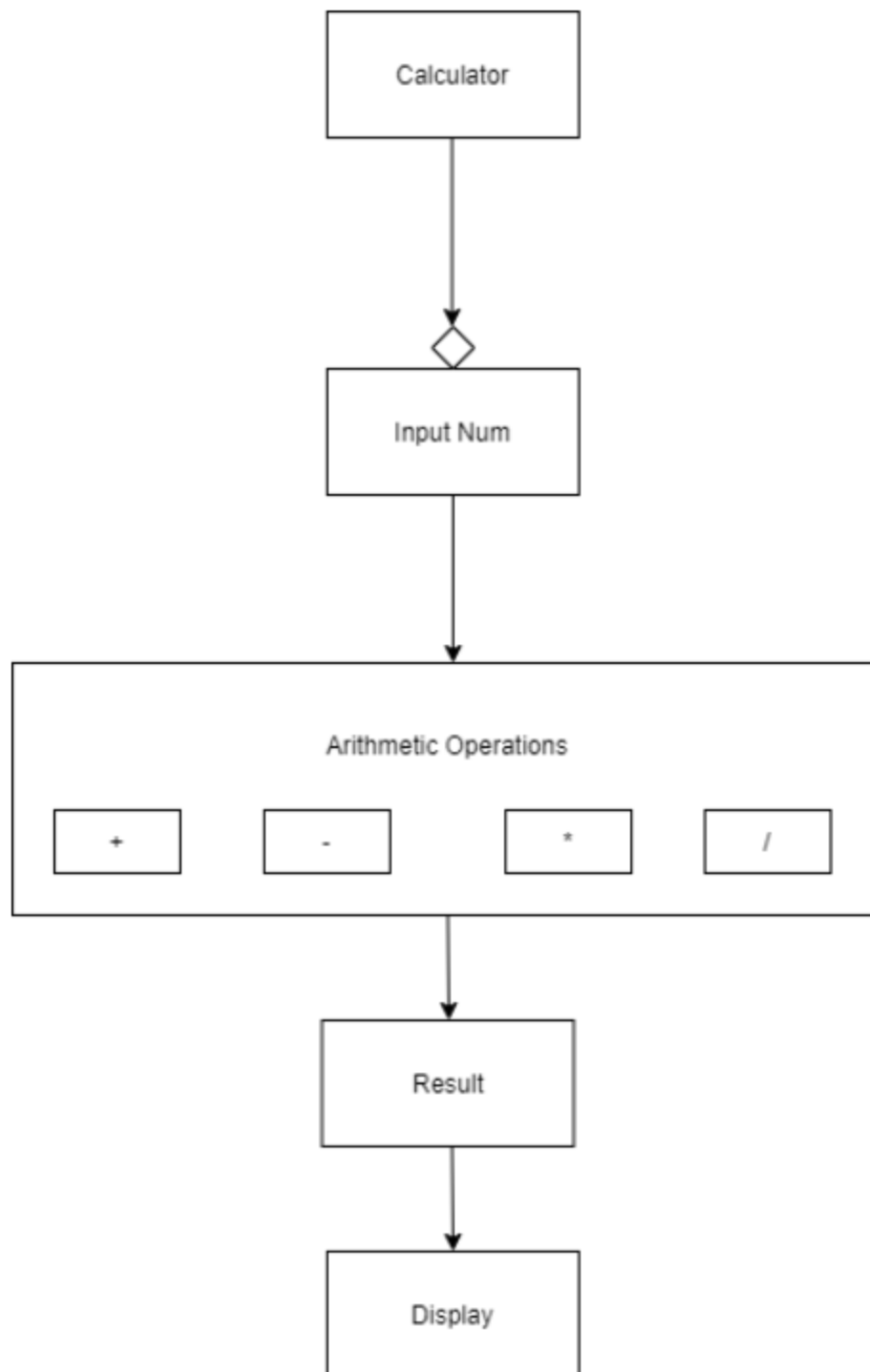


Figure 2:Composite Diagram

1.5.3 LLR Diagram: Profile diagram.

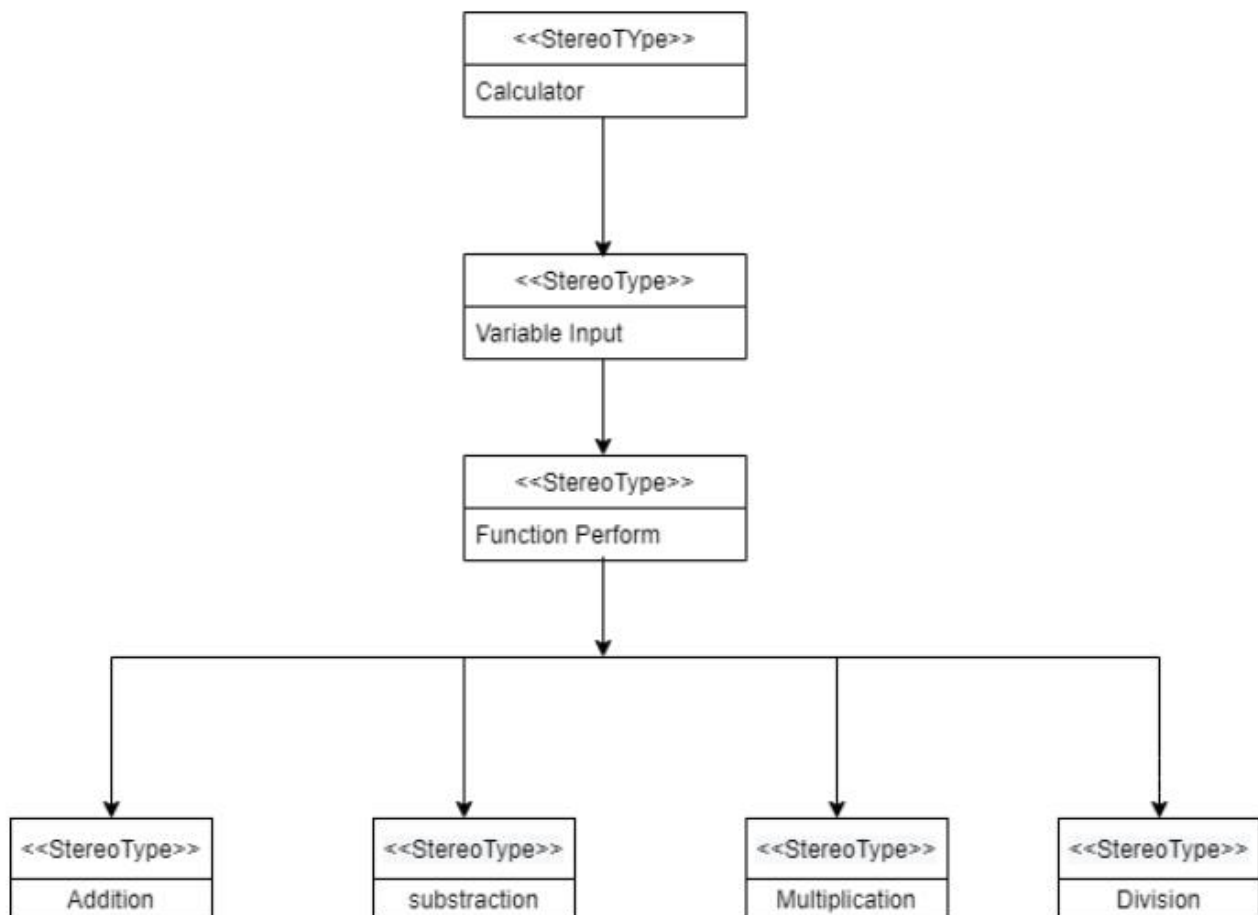


Figure 3: Profile Diagram

1.5.4 LLR Diagram: Communication diagram.

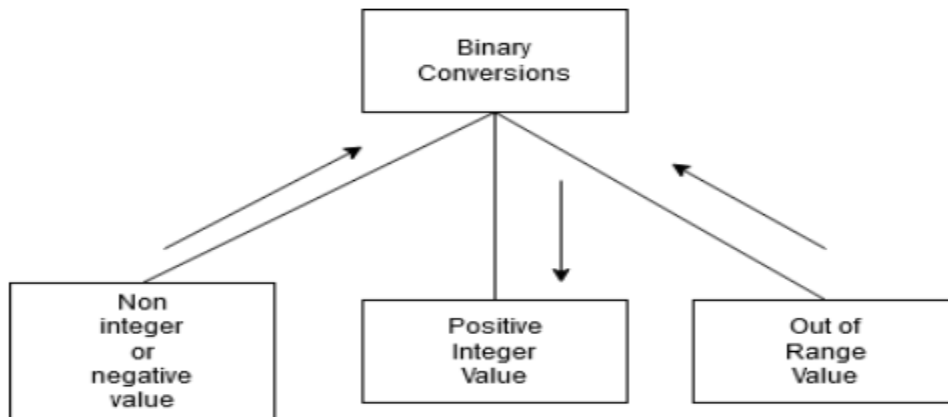


Figure 4:Communication Diagram

1.5.5 LLR Diagram: Object diagram.

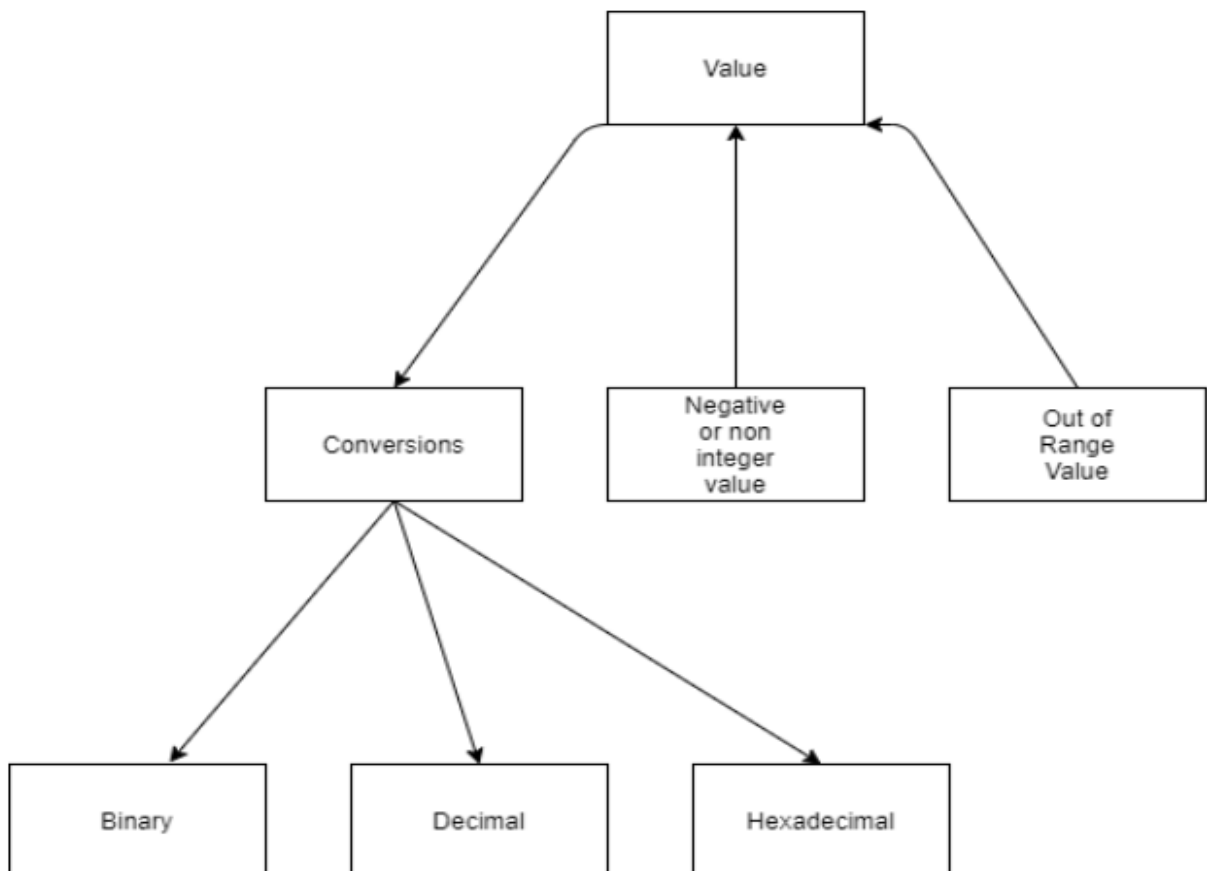


Figure 5: Object Diagram

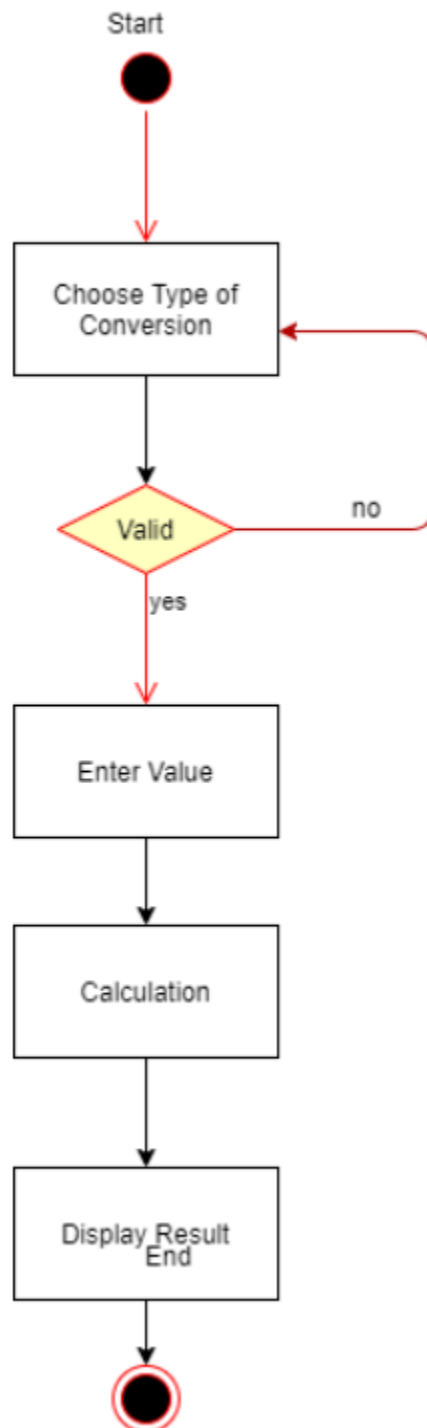
1.5.6 LLR Diagram: Activity diagram.

Figure 6: Activity Diagram

1.5.7 LLR Diagram: Deployment diagram.

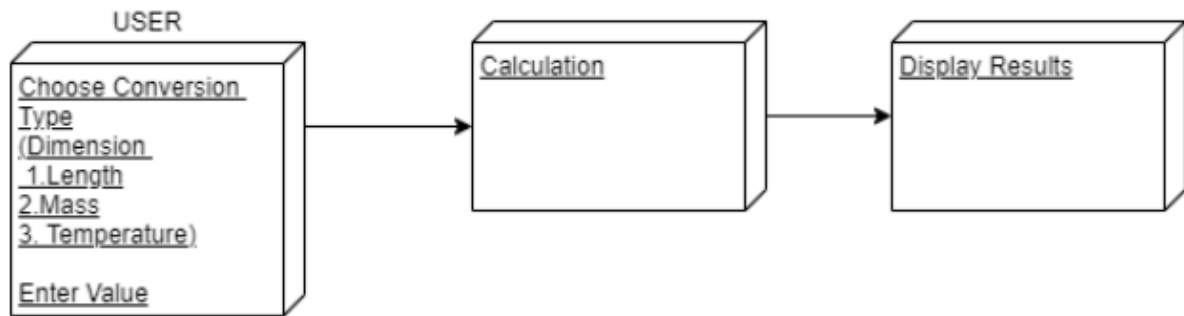


Figure 7: Deployment Diagram

1.5.8 LLR Diagram: UML Class diagram.



Figure 8: Class Diagram

1.5.9 LLR Diagram: State diagram.

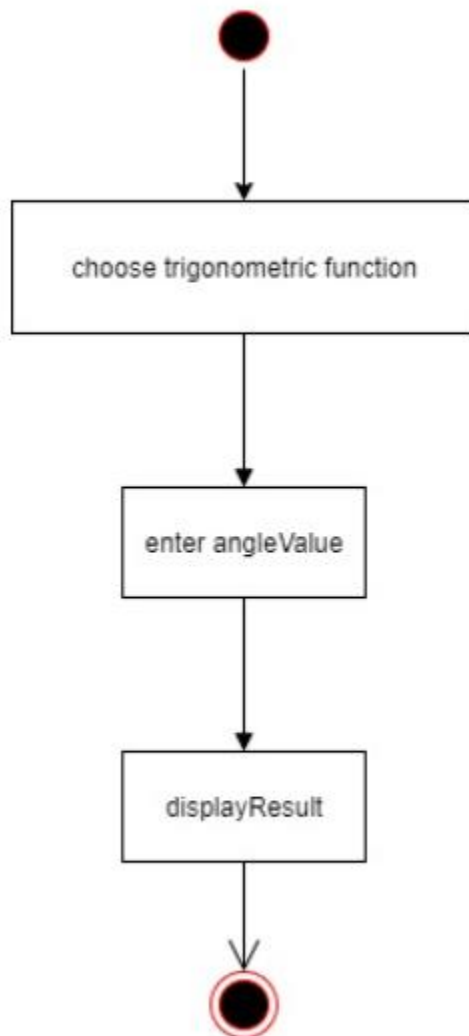


Figure 9: State Diagram

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


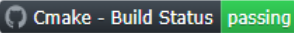
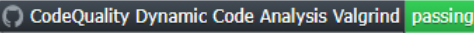
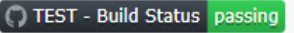

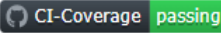
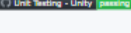
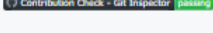
Build	Code Quality	Unity	[Git Inspector] (using github.io option)
			
			
	 		

Figure 10: Badges

Git inspector summary

1.9.2 Git Inspector

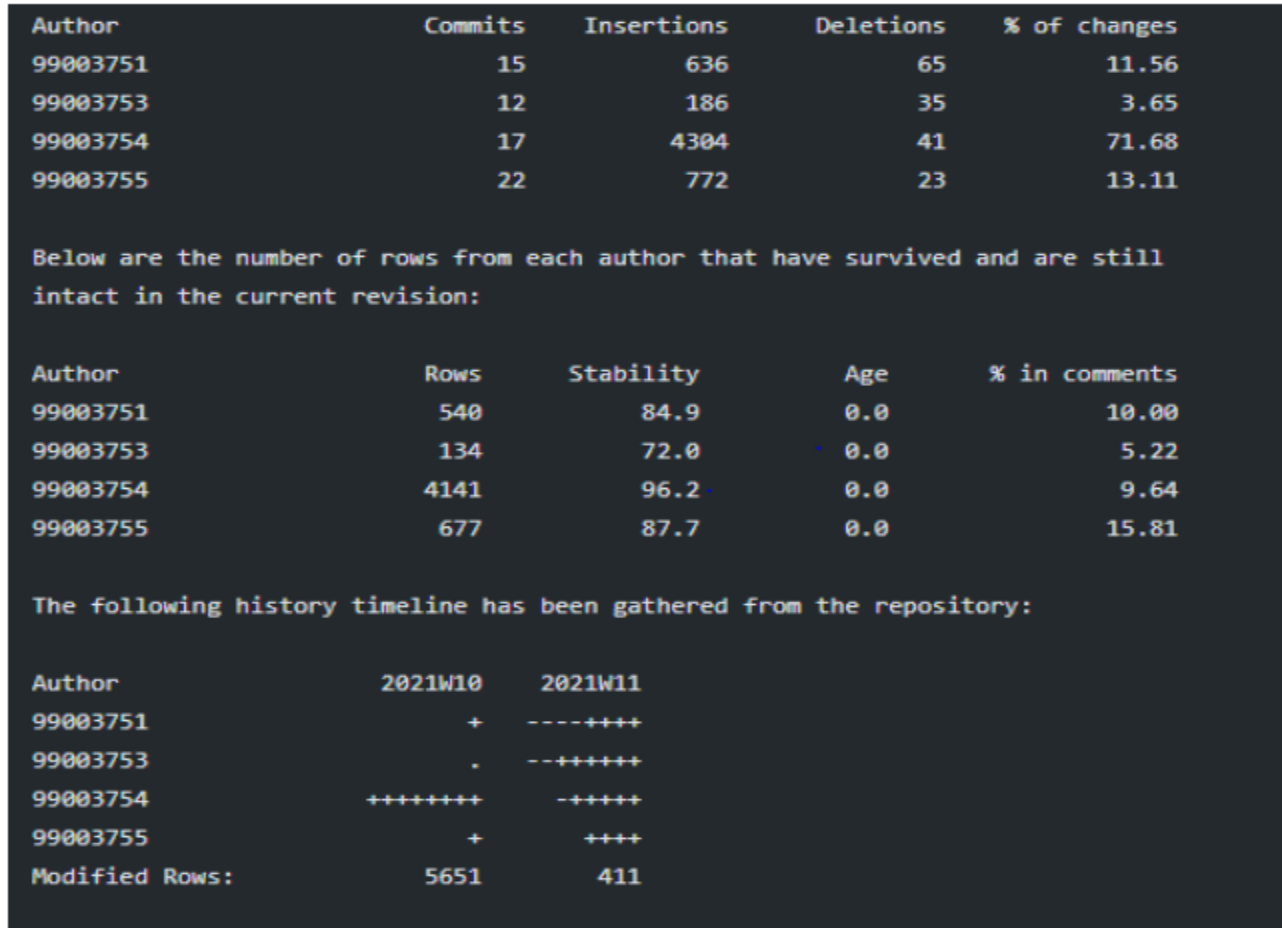


Figure 11: Git inspector

Build

1.9.3 Setup for Build

```
name: TEST - Build Status

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

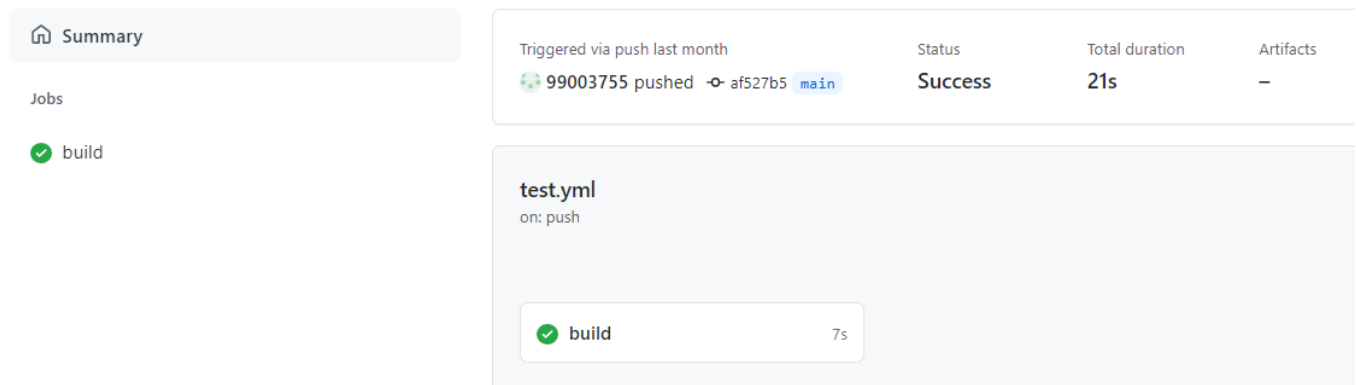
jobs:
  build:

    runs-on: ubuntu-latest

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

Figure 12: Build

1.9.4 Outcome of the Build



The screenshot shows the GitHub Actions interface for a workflow named 'test.yml'. On the left, a sidebar lists 'Summary' (selected), 'Jobs', and 'build' (marked with a green checkmark). The main area displays the workflow details: 'Triggered via push last month', '99003755 pushed' (commit af527b5 on the main branch), 'Status: Success', 'Total duration: 21s', and 'Artifacts: -'. Below this, a section titled 'test.yml' shows the workflow configuration: 'on: push' and a job 'build' with a green checkmark and a duration of 7s.

Figure 13: Build results

1.14 Code quality and Issues or Bug Tracking

1.9.5 Setup for Code Quality

```
19 lines (15 sloc) | 312 Bytes

1  name: Cppcheck
2
3  on:
4    push:
5      branches: [ main ]
6    pull_request:
7      branches: [ main ]
8
9  jobs:
10   cppcheck:
11
12     runs-on: ubuntu-latest
13
14     steps:
15     - uses: actions/checkout@v2
16     - name: Install cppcheck
17       run: sudo apt -y install cppcheck
18     - name: Run cppcheck
19       run: cppcheck 3_Implementation
```

Figure 14:Code quality

1.9.6 Outcome of Code Quality

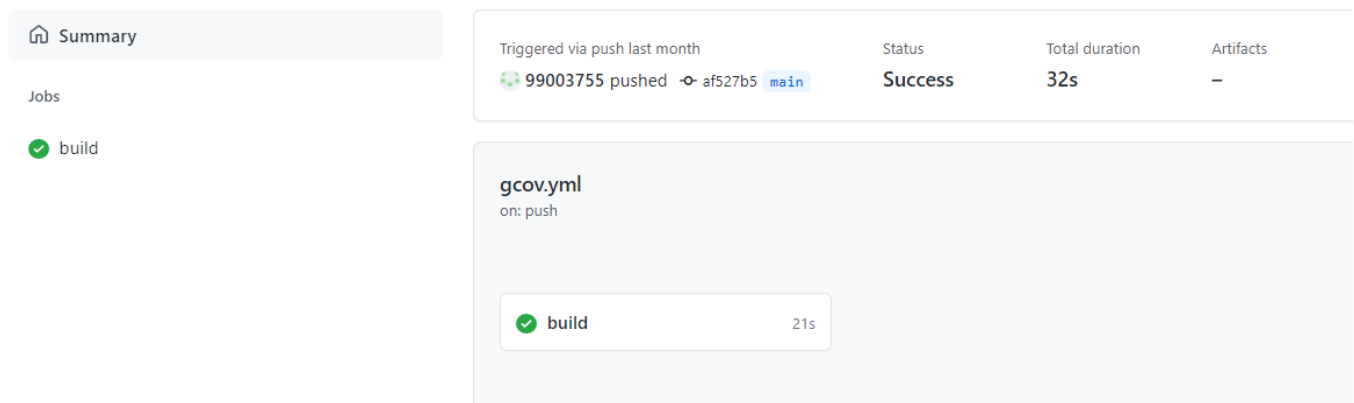


Figure 15: code quality results

1.15 Unit Testing


1.9.7 Setup for Unity Testing

```
17 lines (13 sloc) | 248 Bytes


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

Figure 16: Unity testing



1.9.8 Outcome of Unity Testing

 Summary

Jobs

 test

Triggered via push last month

 99003755 pushed  af527b5 main

Status

Success

Total duration


17s

Artifacts

—

unity.yml

on: push

 test

5s

Figure 17: 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.

Contributors List and Summary

PS Number	Name	Feature
99003751	Shrinidhi V Katti	Trigonometric Operation(Sin, Cos, Tan, Cot, Cos, Co-sec)
99003753	Akshansh Mishra	Arithmetic Operation(Addition, Subtraction, Multiplication, Division, Remainder)
99003754	Rishab Ostawal	Binary Conversion(Decimal, Hexadecimal)
99003755	Rohan Roy	Dimension Conversion(Length, Mass, Temperature)

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.

```

/*****

if(HAL_GPIO_ReadPin(PA0_GPIO_Port, PA0_Pin)==1){

    HAL_GPIO_TogglePin(PD12_GPIO_Port, PD12_Pin);
    HAL_Delay(100);
    HAL_GPIO_TogglePin(PD13_GPIO_Port, PD13_Pin);|
    HAL_Delay(100);
    HAL_GPIO_TogglePin(PD14_GPIO_Port, PD14_Pin);
    HAL_Delay(100);
    HAL_GPIO_TogglePin(PD15_GPIO_Port, PD15_Pin);
    HAL_Delay(100);

}else{
    //pass comment
}

```

Figure : Basic logic for Implementation

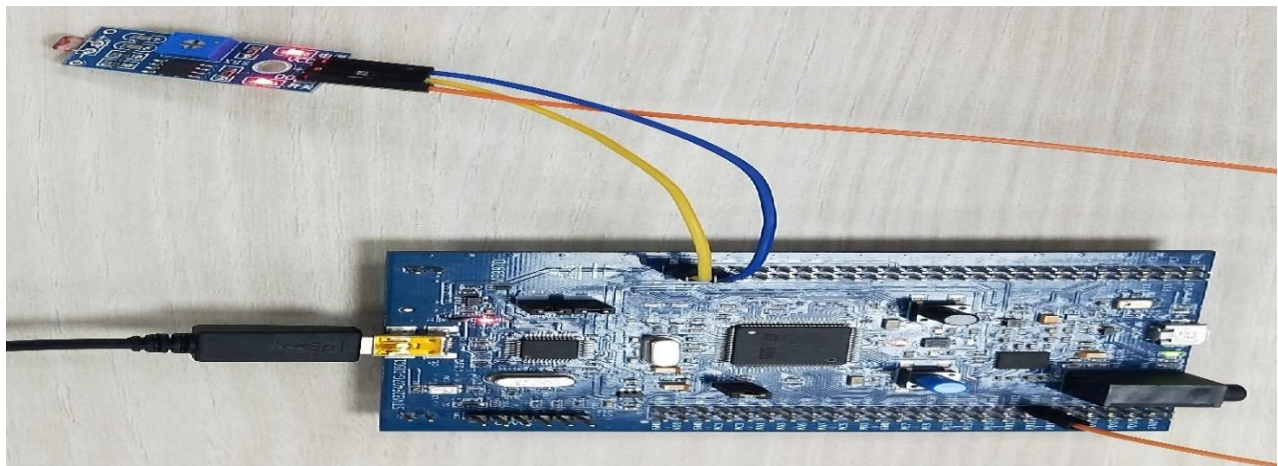


Figure 18: Implementation

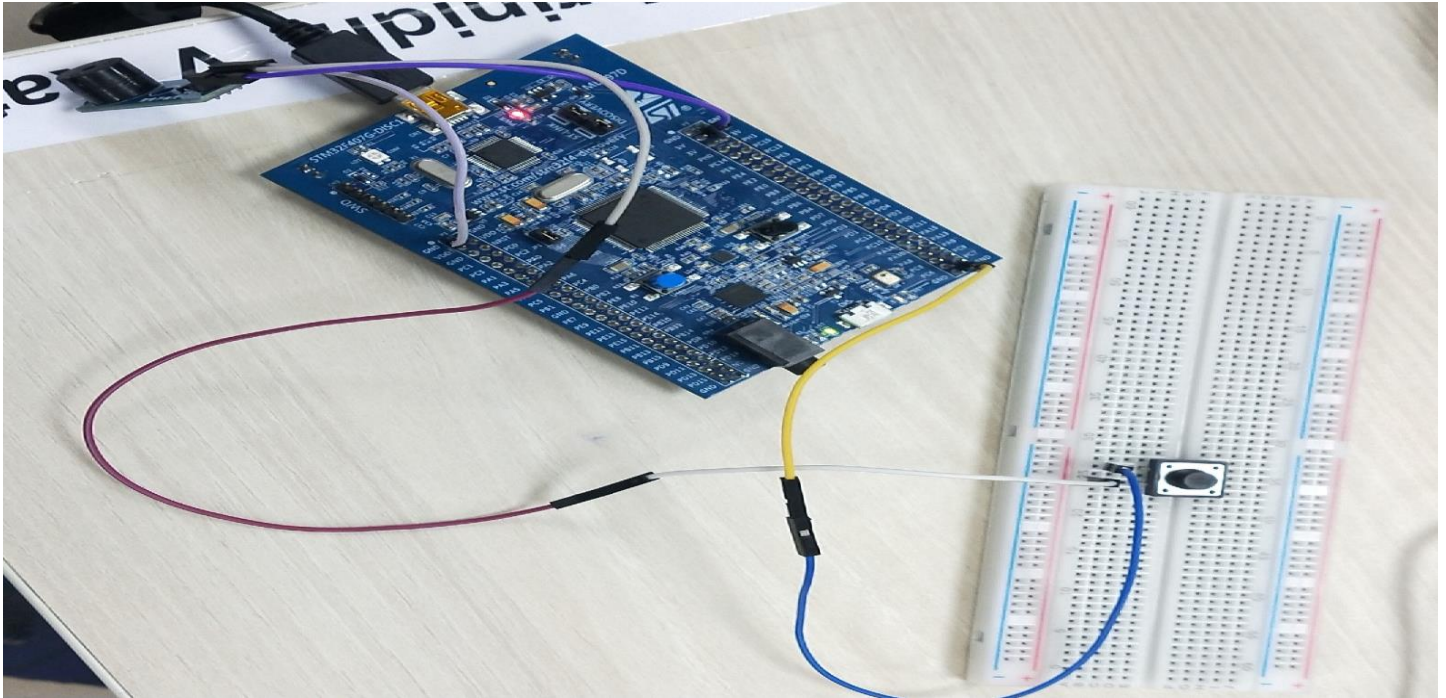


Figure 19: Switch Implementation

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

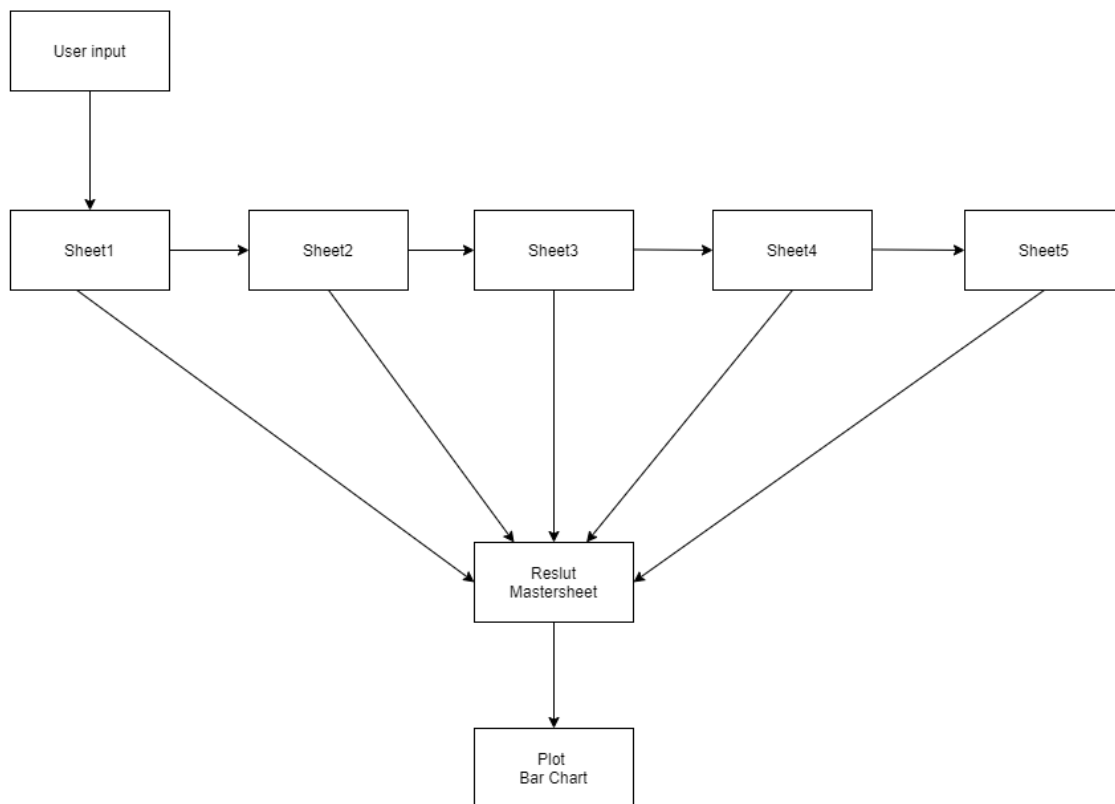


Figure 20: Use case diagram

3.6.2 Object HLR Diagram

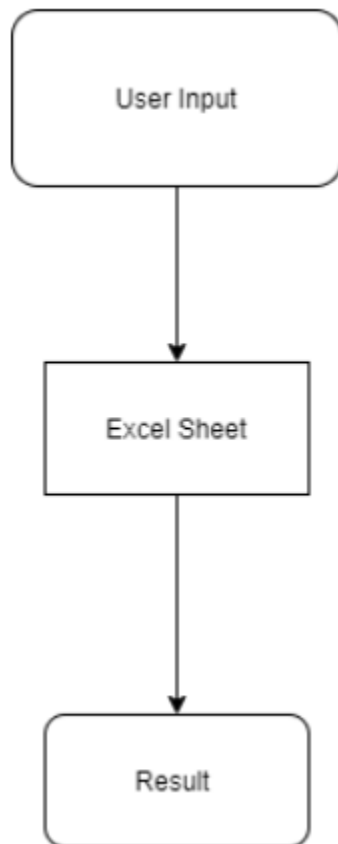


Figure 21: Object diagram

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.