./

GENESIS - Learning Outcome & Mini-project Summary Report



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **To be Approved** | **Remarks/Revision Details** |
|  |  | Akshansh Mishra  PS No- 99003753 |  |  |  |

Contents

[Contents 3](#_Toc69890254)

[1. Software Development Life Cycle (SDLC) 5](#_Toc69890255)

[1.1 Modules Used 5](#_Toc69890256)

[1.2 Project title : Calculator 5](#_Toc69890257)

[1.3 Topic and Subtopics 5](#_Toc69890258)

[1.4 Objectives & Requirements 5](#_Toc69890259)

[1.4.1 High Level requirement Analysis – 5](#_Toc69890260)

[1.4.1 Low Level Requirement Analysis - 6](#_Toc69890261)

[1.5 Design 6](#_Toc69890262)

[1.5.1 High level diagram: Deployment Diagram 6](#_Toc69890263)

[1.5.2 LLR Diagram: Composite diagram. 7](#_Toc69890264)

[1.5.3 LLR Diagram: Profile Diagram - 8](#_Toc69890265)

[1.5.3 LLR Diagram : Communication Diagram - 8](#_Toc69890266)

[1.5.4 LLR Diagram: Object diagram – 9](#_Toc69890267)

[1.5.5 LLR Diagram: Activity diagram - 10](#_Toc69890268)

[1.5.6 LLR Diagram: Deployment diagram - 11](#_Toc69890269)

[1.5.7 LLR Diagram: UML Class diagram. 11](#_Toc69890270)

[1.5.8 LLR Diagram: State diagram. 12](#_Toc69890271)

[1.6 Test Plan 12](#_Toc69890272)

[1.6.1 High Level Test Plan 12](#_Toc69890273)

[1.6.2 Low Level Test Plan 13](#_Toc69890274)

[1.7 Implementation Summary 13](#_Toc69890275)

[1.8 Video Summary 13](#_Toc69890276)

[1.9 Git Link 13](#_Toc69890277)

[1.10 Git Dashboard 13](#_Toc69890278)

[1.10.1 Badges 13](#_Toc69890279)

[1.10.2 Git Inspector 14](#_Toc69890280)

[1.10.3 Setup for Build 14](#_Toc69890281)

[1.10.4 Outcome of the Build 15](#_Toc69890282)

[1.10.5 Setup for Code Quality 15](#_Toc69890283)

[1.10.6 Outcome of Code Quality 16](#_Toc69890284)

[1.10.7 Setup for Unity Testing 16](#_Toc69890285)

[1.10.8 Outcome of Unity Testing 17](#_Toc69890286)

[1.11 Individual Contribution & Highlights 18](#_Toc69890287)

[1.12 Summary 18](#_Toc69890288)

[1.12.1 Outcomes: 18](#_Toc69890289)

[1.13 Challenges faced and how were they overcome 18](#_Toc69890290)

[2. Python Programming Project 19](#_Toc69890291)

[2.1 Modules Used 19](#_Toc69890292)

[2.2 Project title: Retrieve data from multiple Excel sheet 19](#_Toc69890293)

[2.3 Topic and Subtopics 19](#_Toc69890294)

[2.4 Objectives: 19](#_Toc69890295)

[2.5 Requirements: 19](#_Toc69890296)

[2.5.1 High Level requirement analysis 19](#_Toc69890297)

[2.5.2 Low Level requirement Analysis - 19](#_Toc69890298)

[2.6 Design 20](#_Toc69890299)

[2.6.1 Use Case LLR Diagram 20](#_Toc69890300)

[2.6.2 Object HLR Diagram 20](#_Toc69890301)

[2.7 Implementation Summary 21](#_Toc69890302)

[2.8 Video Summary 21](#_Toc69890303)

[2.9 Git Link 21](#_Toc69890304)

[2.10 Git Dashboard 21](#_Toc69890305)

[2.11 Summary - 21](#_Toc69890306)

[2.11.1 Outcomes: 21](#_Toc69890307)

[2.12 Challenges faced and how were they overcome 22](#_Toc69890308)

[3.Embedded C 23](#_Toc69890309)

[3.1 Module: 23](#_Toc69890310)

[3.2 Objectives & Requirements 23](#_Toc69890311)

[3.2.1 HIGH LEVEL REQUIREMENTS: 23](#_Toc69890312)

[3.2.2 LOW LEVEL REQUIREMENTS: 23](#_Toc69890313)

[3.3 Test Plan 23](#_Toc69890314)

[3.5 Implementation Summary 25](#_Toc69890315)

[3.6 Individual Contribution & Highlights 26](#_Toc69890316)

[3.7 Summary 26](#_Toc69890317)

[4.0 Miniproject -4 [Individual] – Kernel Programming and Device Drivers - 27](#_Toc69890318)

[4.1 Module : 27](#_Toc69890319)

[4.2 Topic and Subtopics: 27](#_Toc69890320)

[4.3 Objectives & Requirements: 27](#_Toc69890321)

[4.3.1 Requirements: 28](#_Toc69890322)

[4.4 Implementation Summary: 28](#_Toc69890323)

[4.4.1 Hands-on Activity that are implemented are as follow: 28](#_Toc69890324)

[4.4.2 User space code: 28](#_Toc69890325)

[4.4.3 kthread examples: 28](#_Toc69890326)

[4.5 Git Link: 29](#_Toc69890327)

[4.6 Git Dashboard 29](#_Toc69890328)

[4.7 Summary: 29](#_Toc69890329)

[4.8 Challenges faced and how were they overcome: 30](#_Toc69890330)

# 1. Software Development Life Cycle (SDLC)

## **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
  1. CPP Check
  2. Valgrind
* Testing
  1. Unity Testing
* Makefile
* V Model
* Agile 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.

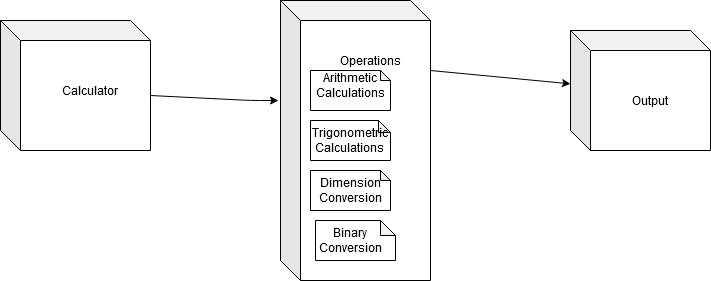
|  |  |  |
| --- | --- | --- |
| ID | Description | Status |
| HLR01 | Basic Arithmetic Calculation | Implemented |
| HLR02 | Trigonometric Calculation | Implemented |
| HLR03 | Dimension Conversion | Implemented |
| HLR04 | Binhttps://www.yammer.com/lnttsgroup.onmicrosoft.com/#/Threads/show?threadId=1165927078649856ary Conversion | Implemented |

### 1.4.1 Low Level Requirement Analysis -

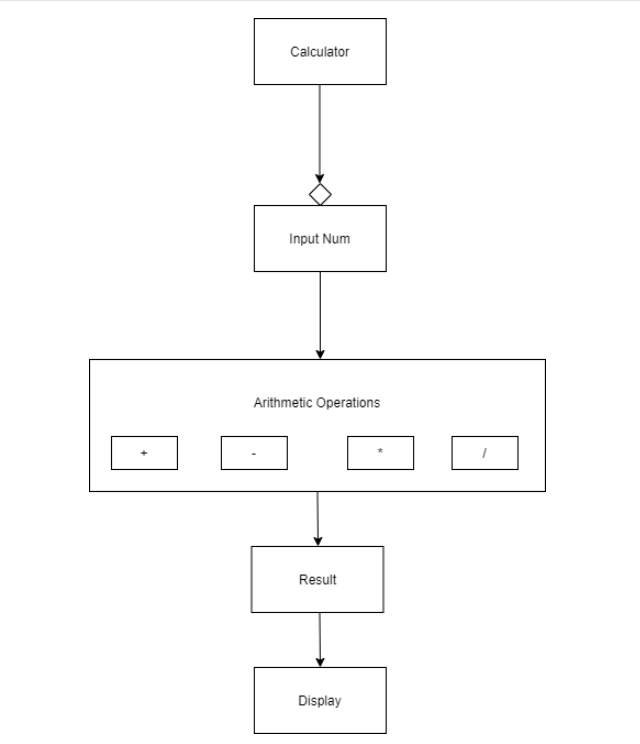
|  |  |  |
| --- | --- | --- |
| ID | Description | Status |
| LLR01 | For Arithmetic Conversion  Addition  Subtraction  Multiplication  Division  modulus | Implemented |
| LLR02 | For Trigonometric Conversion,  Sine, Cosine, Tan, Cot, Sec,  Co-sec. | Implemented |
| LLR03 | For Binary Conversion,  Binary 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**

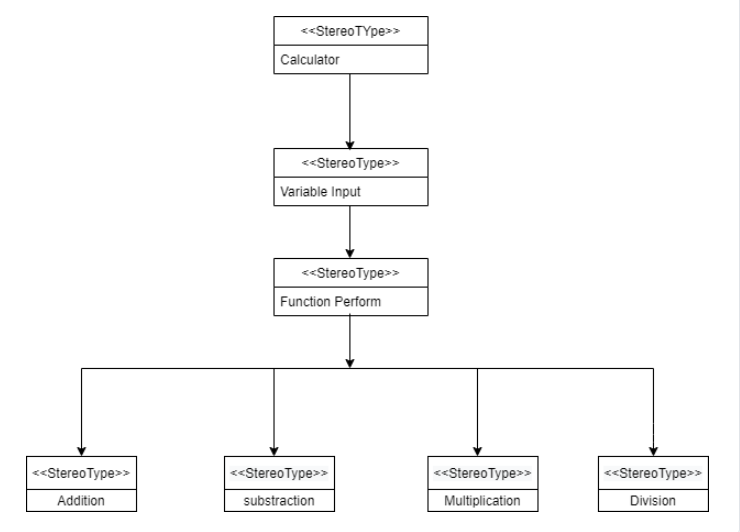
### 1.5.1 High level diagram: Deployment Diagram



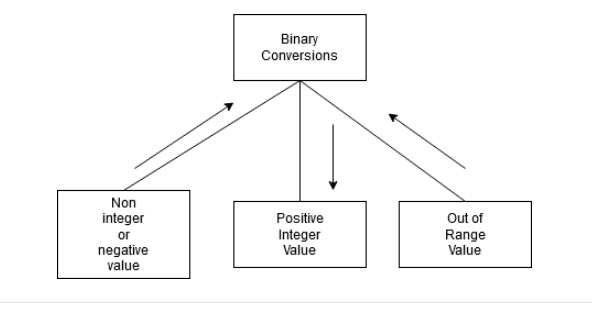
### 1.5.2 LLR Diagram: Composite diagram.



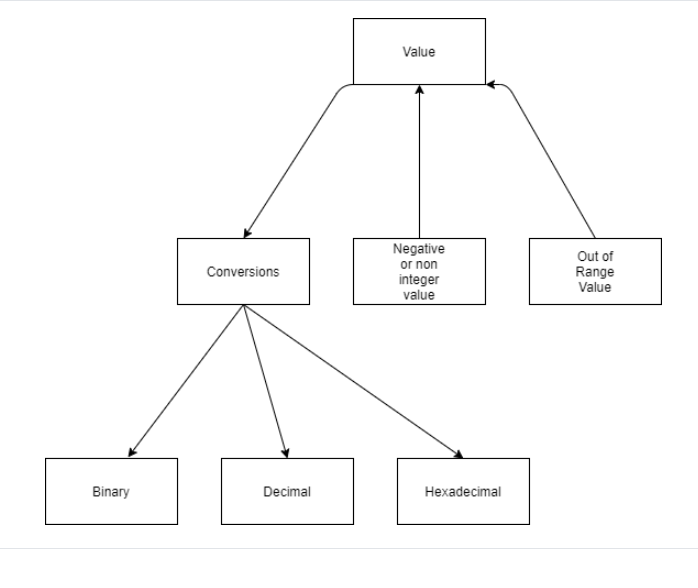
### 1.5.3 LLR Diagram: Profile Diagram -



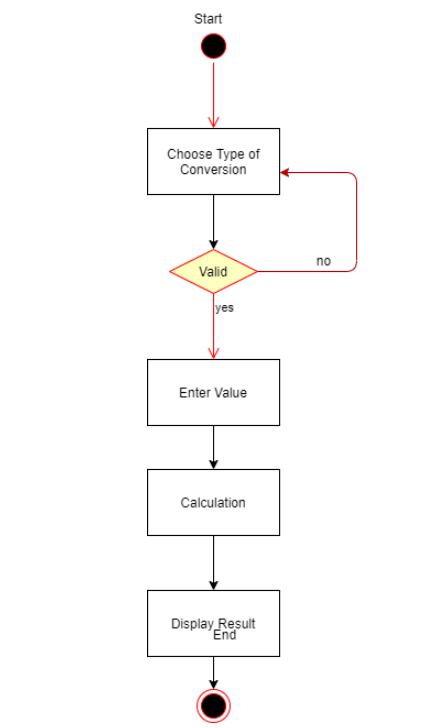
### 1.5.3 LLR Diagram : Communication Diagram -



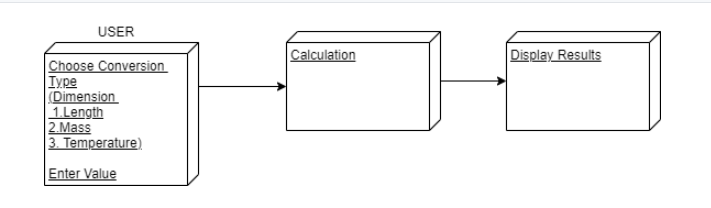
### 1.5.4 LLR Diagram: Object diagram –



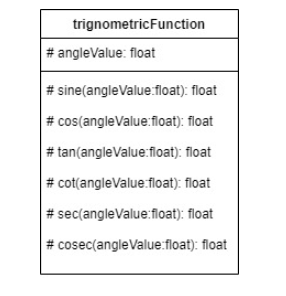
### 1.5.5 LLR Diagram: Activity diagram -



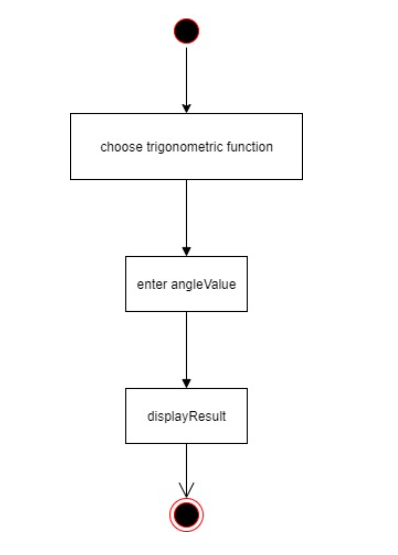
### 1.5.6 LLR Diagram: Deployment diagram -



### 1.5.7 LLR Diagram: UML Class diagram.



### 1.5.8 LLR Diagram: 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**

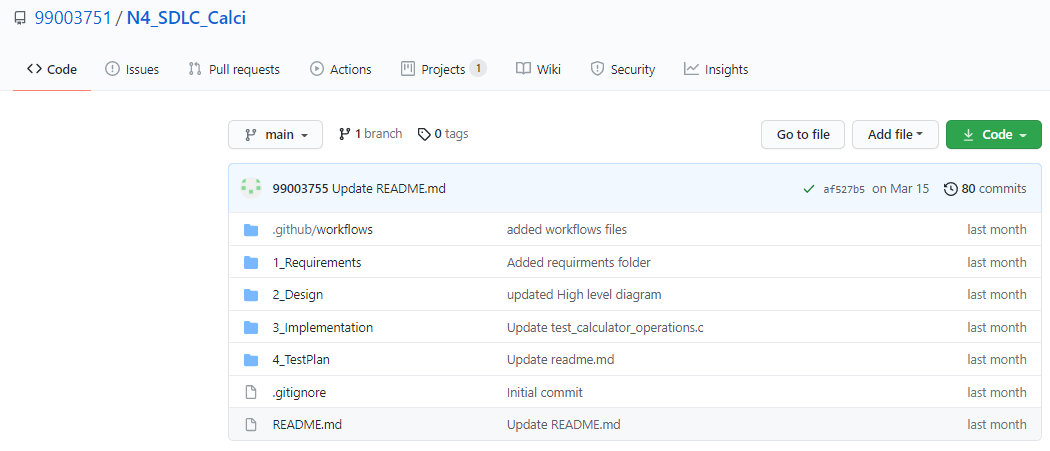
It is a basic calculator that will allow users to perform operations in Mathematics Addition, Subtraction, Multiplication, Division, Trigonometry, Factorial, Area, Volume etc. However, the input has to be in the form "number1 operator1 number2 operator2 number3" (i.e 2+4\*10). The input values can be 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 Video Summary**

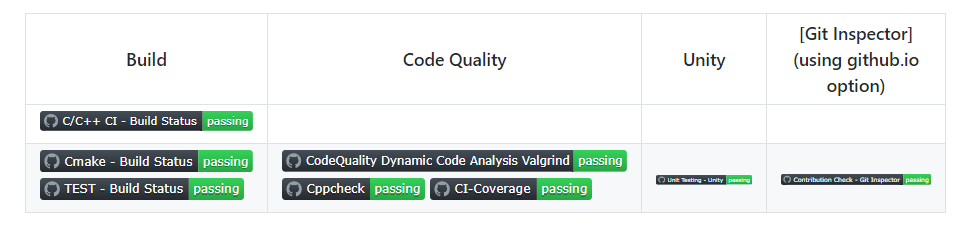
## **1.9 Git Link**

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

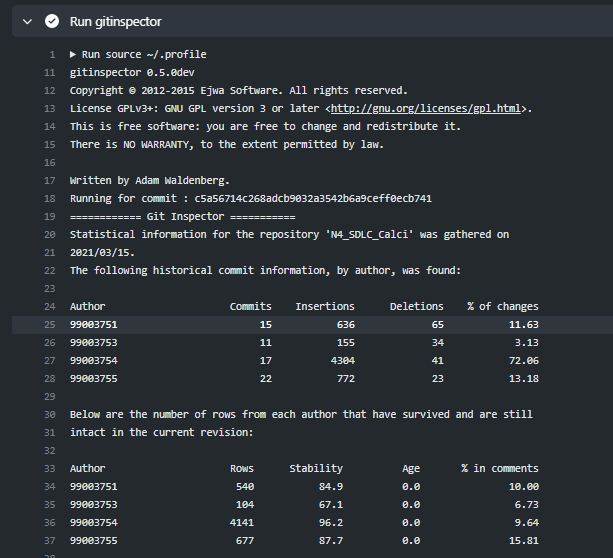
## **1.10 Git Dashboard**



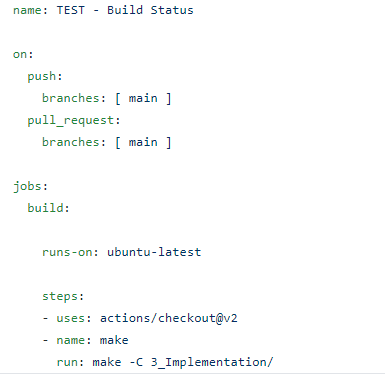
### 1.10.1 Badges



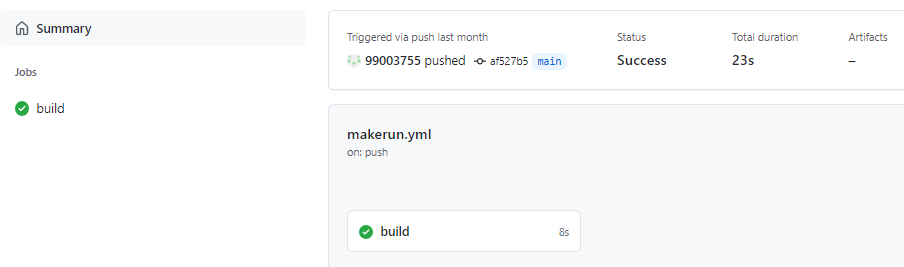
### 1.10.2 Git Inspector



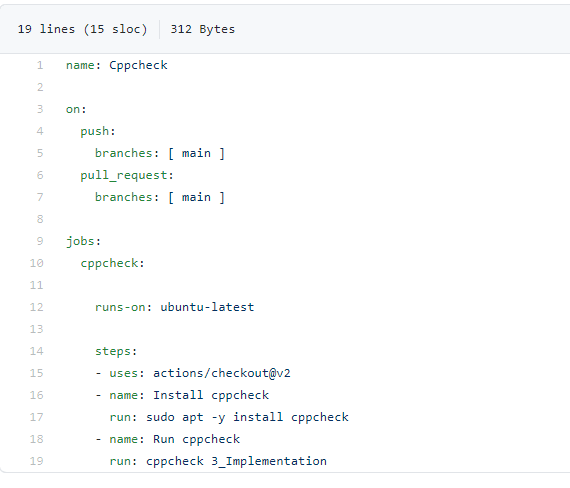
### 1.10.3 Setup for Build



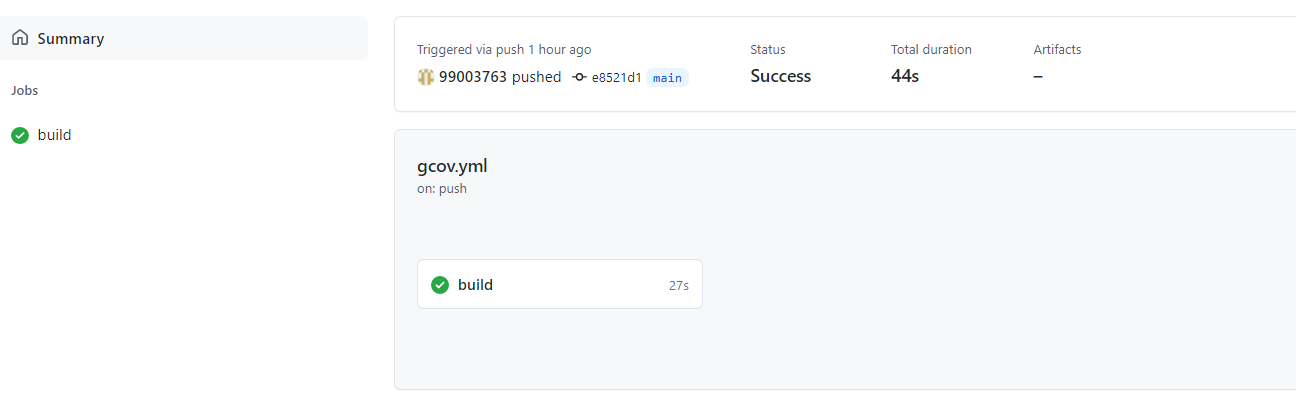
### 1.10.4 Outcome of the Build



### 1.10.5 Setup for Code Quality



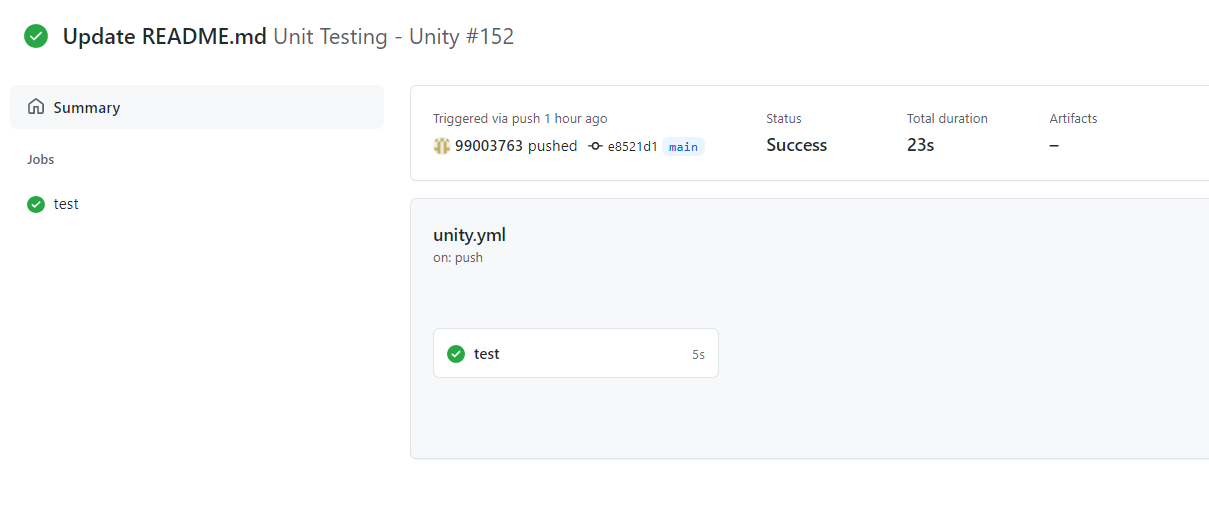
### 1.10.6 Outcome of Code Quality



### 1.10.7 Setup for Unity Testing

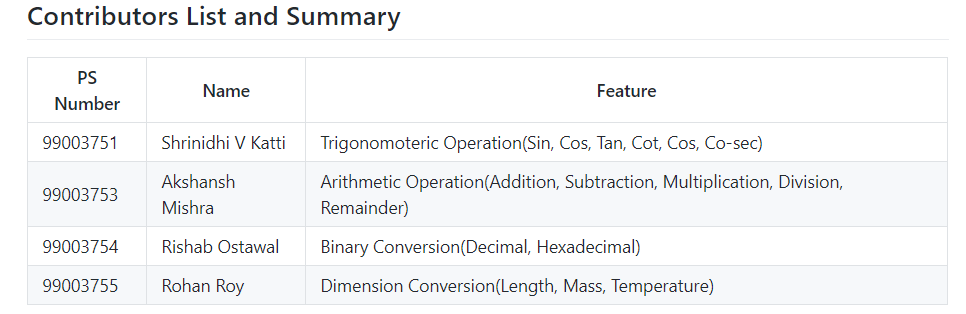


### 1.10.8 Outcome of Unity Testing



## **1.11 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.



## **1.12 Summary**

### 1.12.1 Outcomes:

**Technical**:

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

**Soft skills:**

1. Project management

2. Conflict management.

## **1.13 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. Python Programming Project

## **2.1 Modules Used**

Modules used in this project are Core and Advanced Python.

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

## **2.3 Topic and Subtopics**

## **2.4 Objectives:**

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

## **2.5 Requirements:**

### 2.5.1 High Level requirement analysis

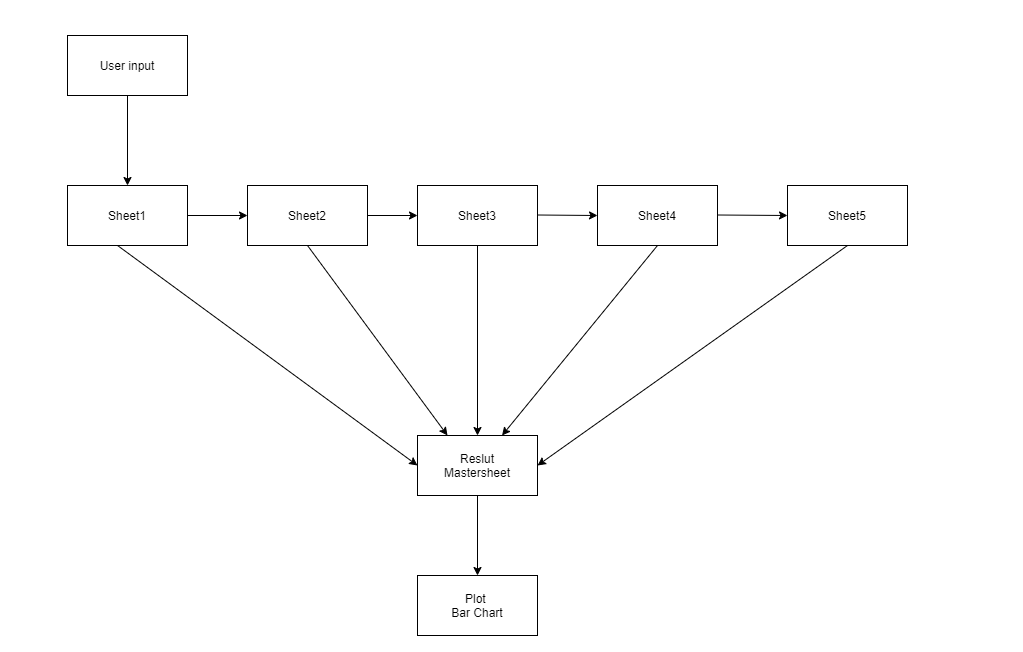
|  |  |  |  |
| --- | --- | --- | --- |
| Id | Requirements | Description | Status |
| HL1 | Searching Data | Search all data from 5 sheets when user defines the PS number to be searched. | Implemented |
| HL2 | writing to excel | Write all the data from different spreadsheet in one master sheet | Implemented |
| 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 |

### 2.5.2 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 |

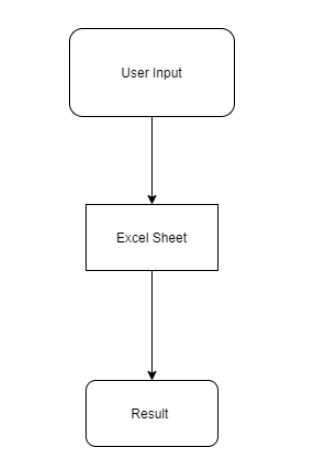
## **2.6 Design**

### 2.6.1 Use Case LLR Diagram



**Figure : LLR Diagram**

### 2.6.2 Object HLR Diagram

**Figure : HLR Diagram**

## **2.7 Implementation Summary**

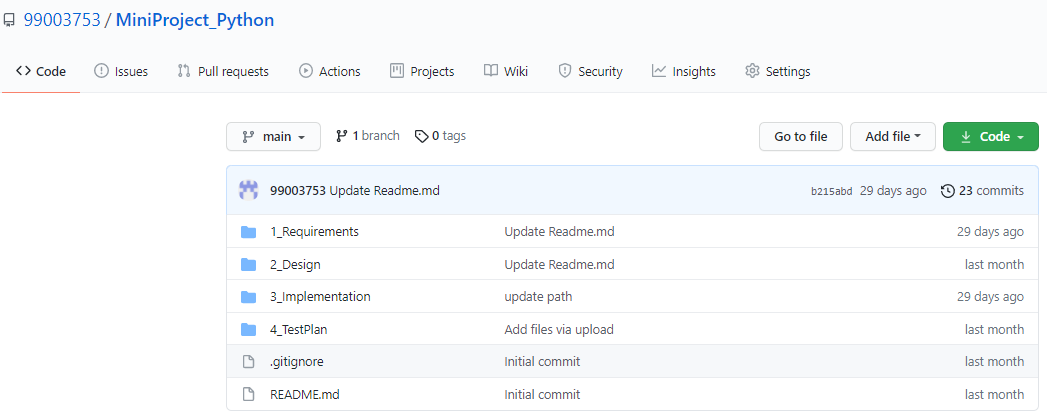
The aim of the project is to extract the data present in different spreadsheets in one excel file as required by the user. The excel sheet consists of 5 spreadsheets. The user defines the data that needs to be searched on the basis of Name, PS Number and Email ID. The python program then reads the data 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.

## **2.8 Video Summary**

## **2.9 Git Link**

<https://github.com/99003753/MiniProject_Python>

## **2.10 Git Dashboard**



## **2.11 Summary -**

### 2.11.1 Outcomes:

Technical:

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

**Soft skills:**

1. Project management

2. Conflict management.

## **2.12 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.

# 

# 3.Embedded C

## **3.1 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.

## **3.2 Objectives & Requirements**

### 3.2.1 HIGH LEVEL REQUIREMENTS:

* + Adjustment of seat with the help of buttons in two direction
  + Forward direction: when the input is 1
  + Reverse direction: when the input is 0

### 3.2.2 LOW LEVEL REQUIREMENTS:

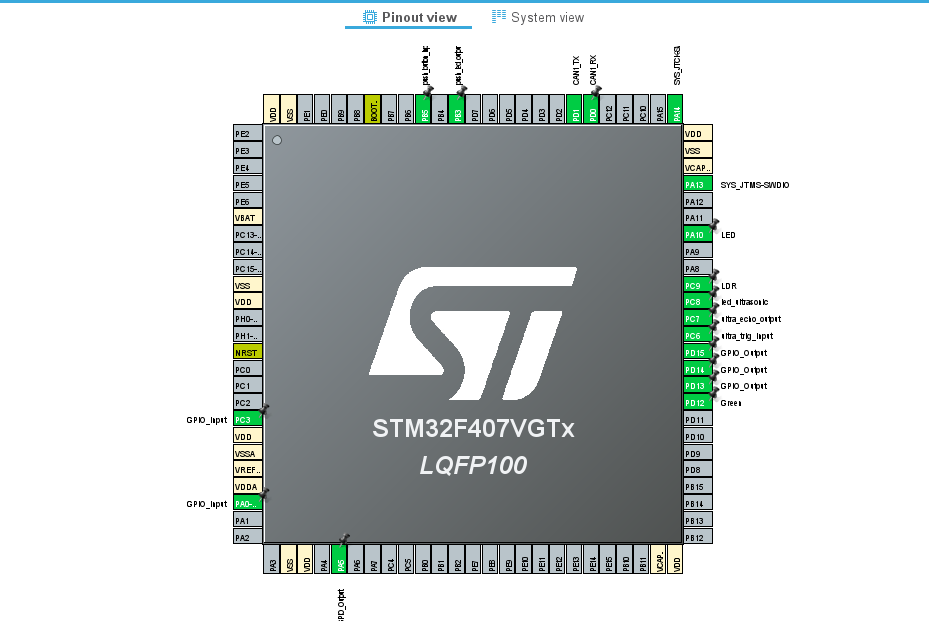
* + When the input is 1 LED glows in clockwise direction
  + When the input is 0 LED glows in anti-clockwise direction

## **3.3 Test Plan**

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

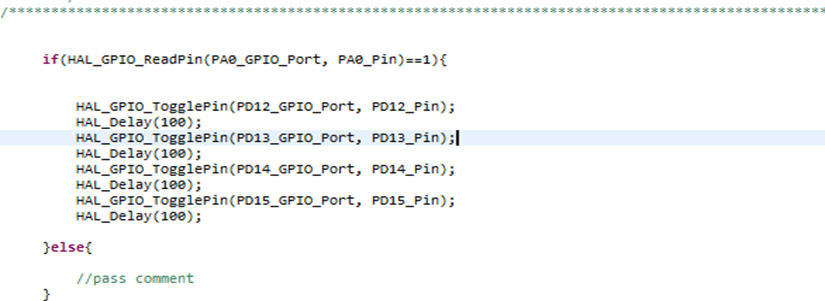
**Table 3: High Level Test plans**

**3.4 Pin Config –**

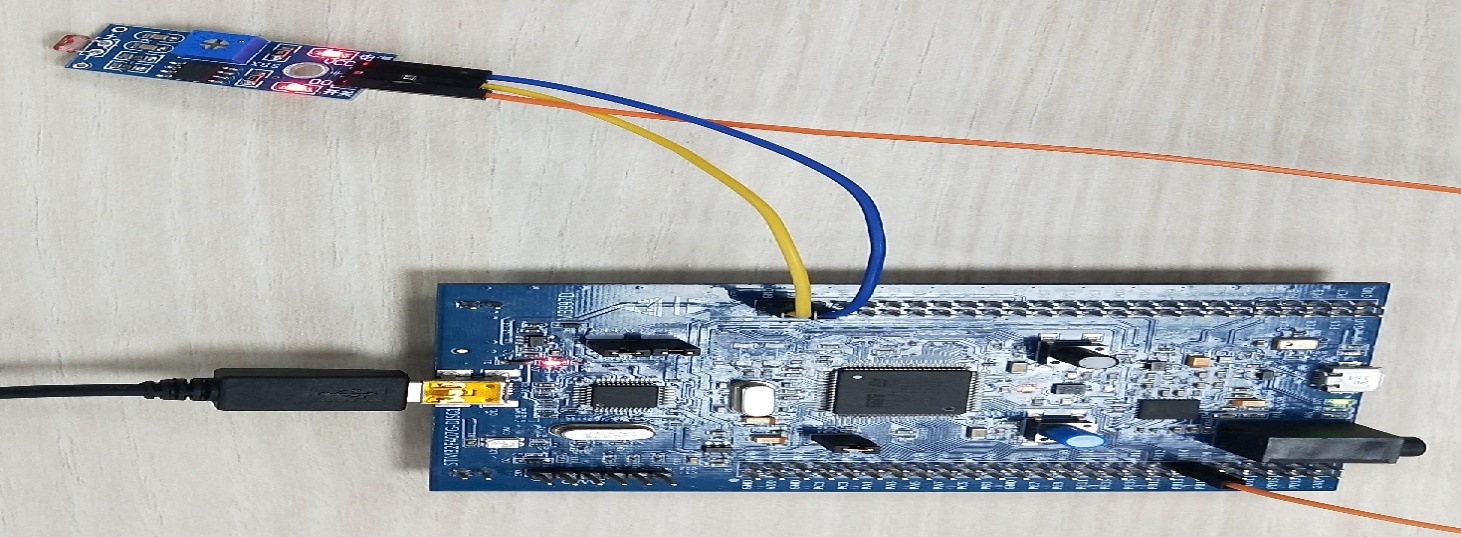
****

## **3.5 Implementation Summary**

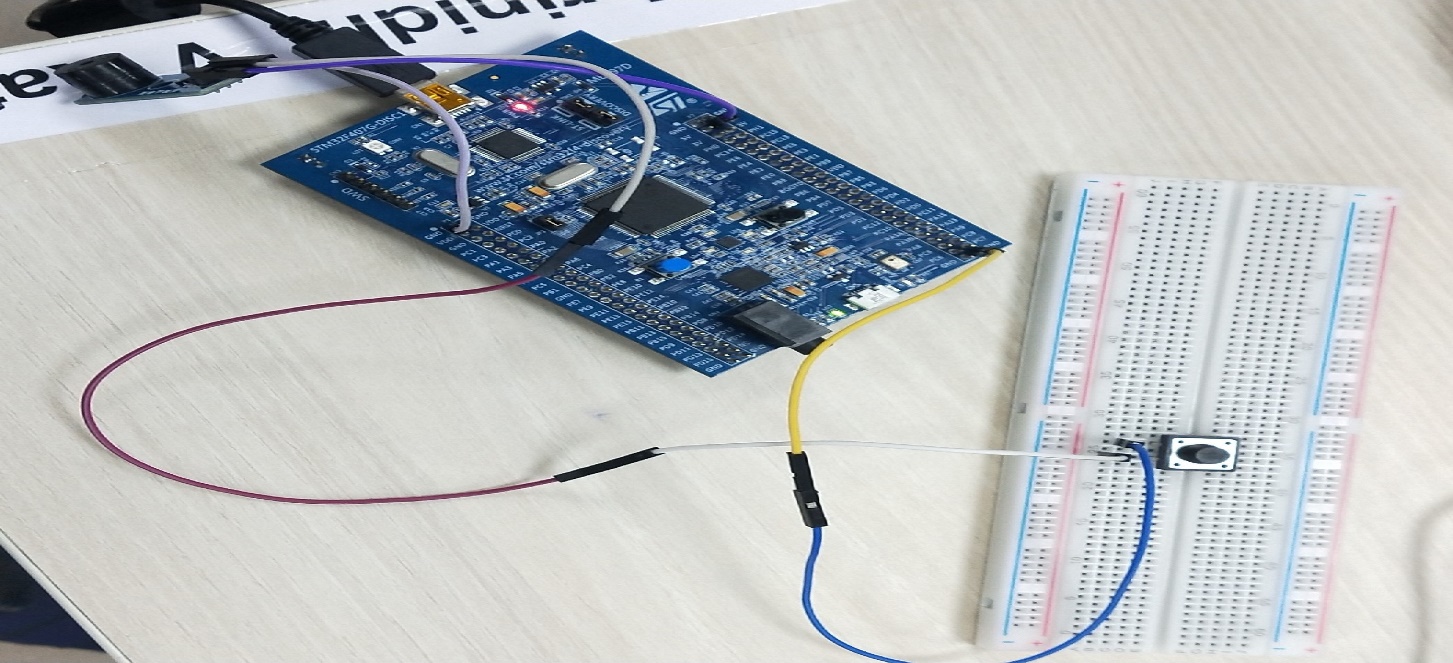
The code for seat system, Window system and lighting system was written by each team member respectively and finally was integrated at the end. The integrated code was dumped on the STM32 board. The individual control of the systems was working properly.



**Figure : Basic logic for Implementation**



**Figure : Use of the LDR sensor**



**Figure : Switch and push button the Implementation**

## **3.6 Individual Contribution & Highlights**

1. Interior door light system.
2. Wiper Control module.
3. Power window.
4. Seat belt warning system.
5. Side mirror control system.
6. Automatic head light control system.

## **3.7 Summary**

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**

* + Controlling the function of Windows through code.
  + With less components showing the output was challenging.

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

## **4.1 Module :**

The modules used in this are Linux and Kernel Device drivers. Kernel programming is an advanced topic that requires in‐depth study of the source code for the Linux kernel.

## **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

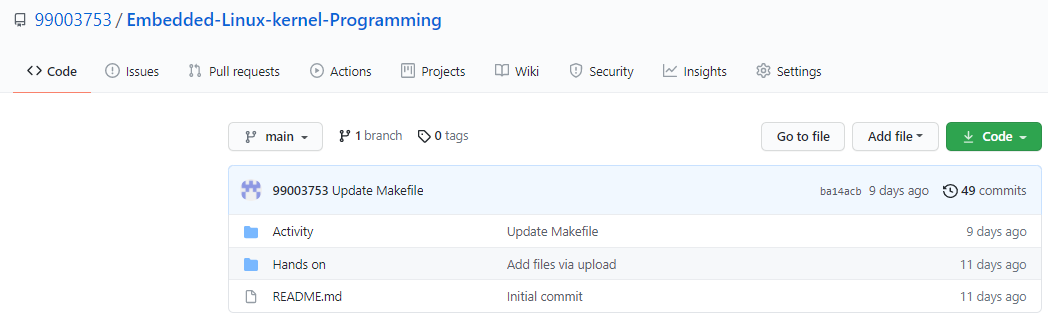
### 4.4.3 kthread examples:

* 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/99003753/Embedded-Linux-kernel-Programming>

## **4.6 Git Dashboard**



## **4.7 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.

## **4.8 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.