./

Learning Report – Applied SDLC and Software Testing

Course Code: <CODE>



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **To be Approved By** | **Remarks/Revision Details** |
| 1.1 | 17-09-2020 | S P Bala Sirisha | 99002642,  99002636 | Pagala Prithvi Sekhar, Srinivas K | System or Software Development |
| 1.2 | 18-09-2020 | S P Bala Sirisha | 99002642,  99002636 | Pagala Prithvi Sekhar, Srinivas K | Updated Requirements table and added UML diagrams |
| 1.3 | 19-09-2020 | S P Bala Sirisha | 99002642,  99002636 | Pagala Prithvi Sekhar, Srinivas K | CI Workflow |
| 1.4 | 27-09-2020 | S P Bala Sirisha | 99002642,  99002636 | Pagala Prithvi Sekhar, Srinivas K | Updated Document |

**Document History**

Contents

[**SYSTEM/SOFTWARE DEVELOPMENT** 3](#_Toc52149505)

[**AGILE MODEL** 26](#_Toc52149506)

[**ADVANCED SCIENTIFIC CALCULATOR** 29](#_Toc52149507)

**List of Figures:**

|  |  |
| --- | --- |
| **Name** | **Page no:** |
| Low level Activity diagram | 3 |
| High level Activity diagram | 4 |
| Low level State diagram | 5 |
| High level State diagram | 6 |
| Low level Component diagram | 7 |
| High level Component diagram | 7 |
| Low level Deployment diagram | 8 |
| High level Deployment diagram | 9 |

**List of Tables:**

|  |  |
| --- | --- |
| **Name** | **Page no:** |
| Requirements Table |  |
| Priority Encoder Truth Table | 10 |
| High level Test Plan |  |
| Inverter Truth Table | 11 |
| 2 input OR cell Truth Table | 11 |
| 3 input NAND cell Truth Table | 12 |
| 4 input NOR cell Truth Table | 13 |
| Low Level Test Plan |  |

## **SYSTEM/SOFTWARE DEVELOPMENT**

**Consumer product** - Design of Low Power Content Addressable Memory using 45nm technology

**Release date – 20th Sep,2020**

Content Addressable Memory is also called as associative memory which can be frequently used in very high speed searching applications. CAM is one type of functional memory which contains huge amount of stored data where the input search data will be compared with the stored data and the matching data address will be returned. When compared to other hardware and software search systems, CAM is faster because it has a single clock cycle throughput

**Objective:** To reduce the power consumption of content addressable memory.

**Requirements:**

**High level requirements:**

* reduce power by reducing number of transistors
* cadence virtuoso tools for schematic and simulations
* Virtuoso Analog Design Environment
* GPDK 45nm technology libraries
* Cadence Linux OS
* Priority encoder

**Low Level requirements:**

* Nmos and Pmos transistors-nmos1v and pmos1v from GPDK library with variable length and width
* Voltage, VDC= 1V
* VDD=5V, Pulse voltage
* Ground pin, Input and output pins
* Bit, bit bar, match and write pins

**Requirements Table**

|  |  |
| --- | --- |
| ID | Description |
| H\_01\_L\_01\_L\_02\_L\_03\_L\_04 | The virtuoso tool provided by cadence is a schematic and layout editor software |
| H\_02­\_L\_01\_L\_02\_L\_03 | Analog Design Environment is the advanced design and simulation environment for the **Virtuoso** platform. |
| H\_03\_L\_01 | **GPDK**, which stands for Generic Process Design Kit, are a set of rules for designing and creating analog circuits |
| H\_04 | Cadence only runs on Linux operating system. |
| H\_05\_L\_01\_L\_02\_L\_03\_L\_04 | Priority Encoder is used to analyze the output from CAM cell and then return the address of the matched data with highest priority |
| L\_01 | nmos1v and pmos1v from GPDK library with variable length and width |
| L\_02 | Voltage=1V |
| L\_03 | VDD=5V, Pulse voltage |
| L\_04 | Ground pin, Input and output pins  Bit, bit bar, match and write pins |

**Ageing:**

Computer memories-

* RAM-DRAM, SRAM, NVRAM
* ROM-EPROM, EEPROM, Cache Memory, Associate Memory

With respect to data access:

* RAM
* Sequentially Access Memory
* Content Addressable Memory

CAM types:

* Binary CAM
* Ternary CAM

Older-generation CAMs took around 10 n sec for an operation, but currently announced products appear to take 5 n sec, possibly by pipelining parts of the match delay.

Future:

* Development of CAM cells from PRAM and MRAM
* Comparison operation of match line sensing scheme in non-volatile CAM in different fashion

**Cost:**

CADENCE- $10500.0

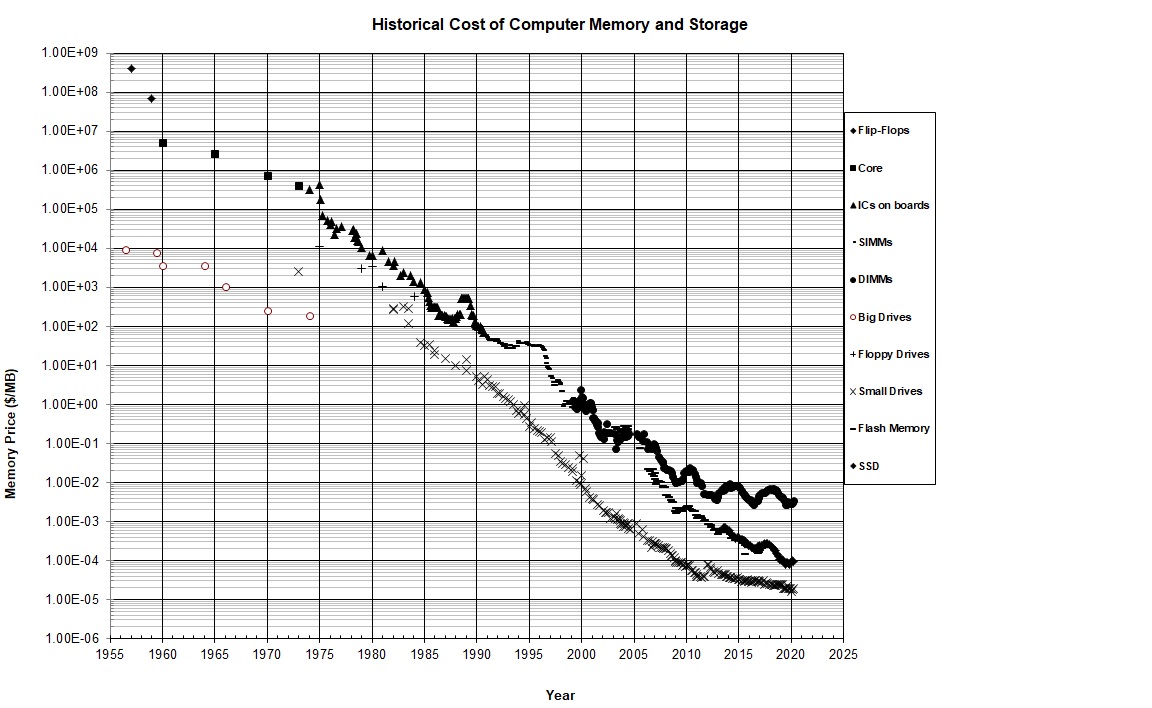


Fig: Cost vs Ageing of CAM

**SWOT Analysis:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Strengths** | **Weakness** | **Opportunities** | **Threats** |
| 1. Low power  2. Search and update in one memory cycle (e.g., 10 n sec) and handle any combination of 100,000 prefixes  3. Density Scaling: One bit in a TCAM requires 10–12 transistors, while an SRAM requires 4–6 transistors  4. Time Scaling: The match logic in a CAM requires all matching rules to arbitrate so that the highest match wins.  5. Used in computer networking devices, network routers, database engine | 1. Errors in connections 2. Costly | 1. Emerging Tech : Processing Power  2.Emerging Tech : Devices and Wires  3.Emerging Tech : Real-Time Data Analysis | 1.Glitches  2.System Failure cannot recover data |

**Behavioural Model:**

**Activity Diagram:**

**Low level Activity diagram:**

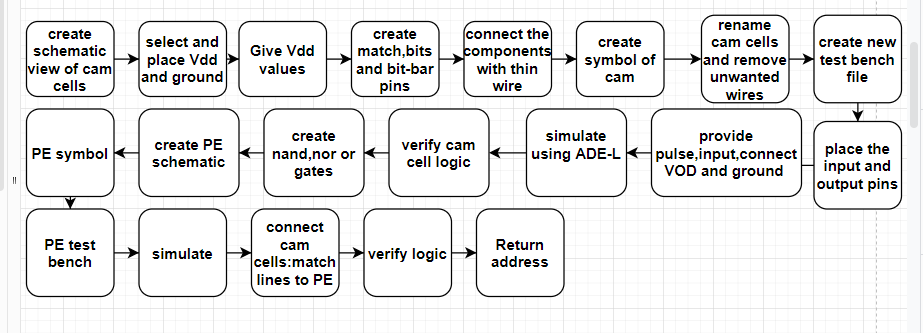
****

Fig: Low level Activity Diagram

**High level activity diagram:**

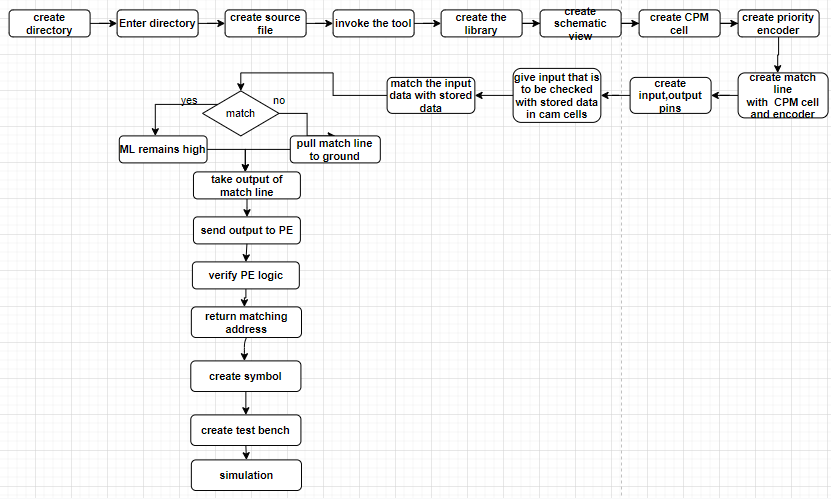
****

Fig: High Level Activity Diagram

**State Chart diagram:**

**Low level state diagram:**

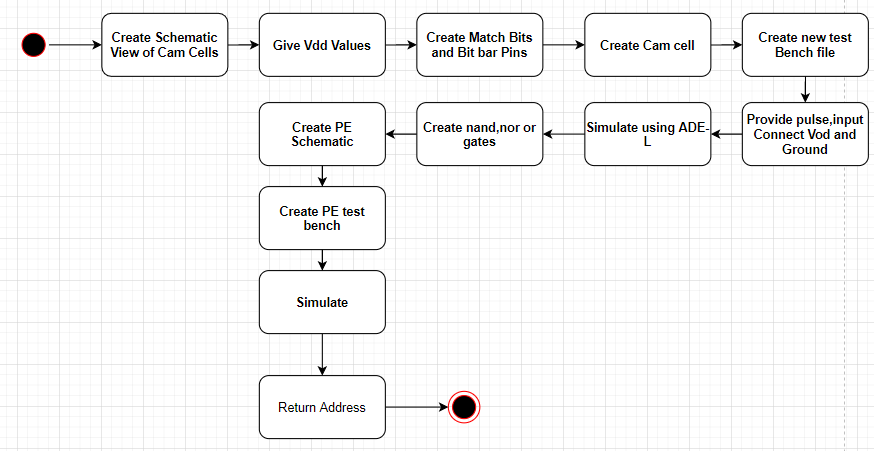
****

Fig: Low Level State Diagram

**High level state diagram**:

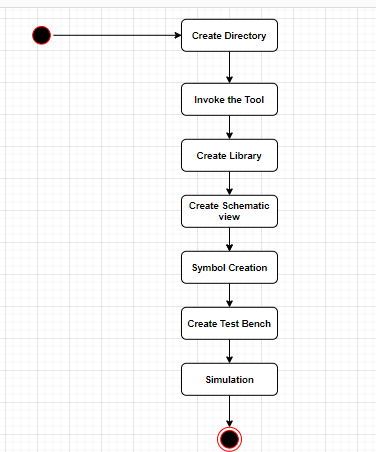
****

Fig: High Level State Diagram

**Structural Model**

**Component Diagram**

**Low level Component Diagram**

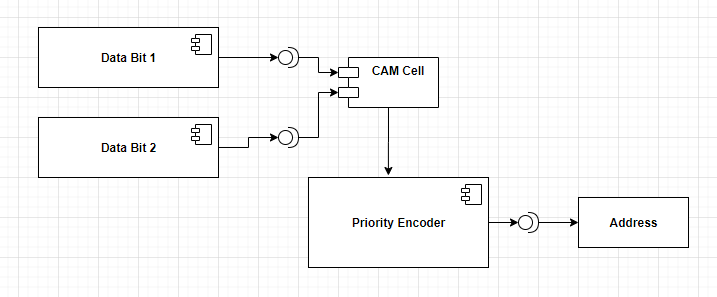
****

Fig: Low Level Component Diagram

**High level Component Diagram**

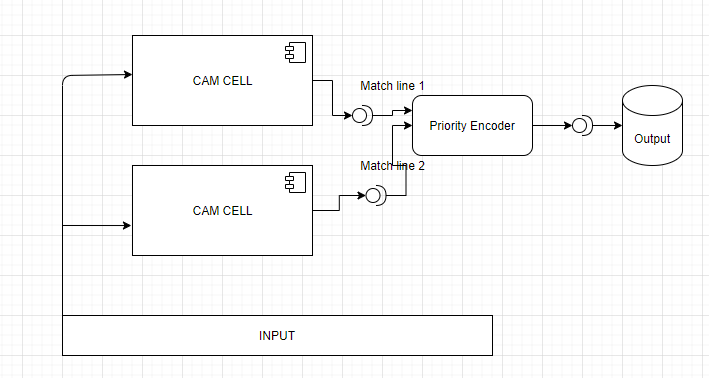
****

Fig: High level component Diagram

**Deployment Diagram**

**Low level Deployment Diagram**

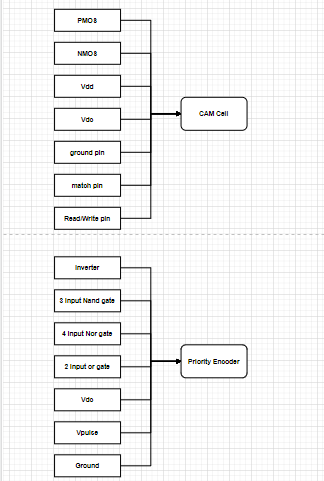
****

Fig: Low Level Deployment Diagram

**High level Deployment Diagram**

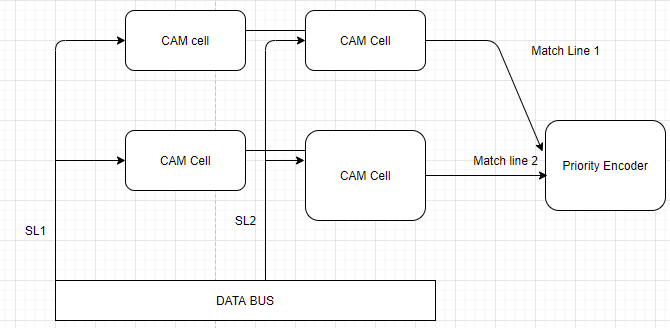
****

Fig: High Level Deployment Diagram

**Test plan for High level requirement:**

**Requirement based:**

The described project is designed for 4 bit content addressable memory and is expandable to n bits. Hence, the requirements would be 4 bits of data. The input data is compared with the stored data and corresponding value of match line output is sent to priority encoder as input.

The inputs and corresponding outputs of Priority Encoder are shown below:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| D0 | D1 | D2 | D3 | Y1 | Y0 | V |
| 0 | 0 | 0 | 0 | X | X | 0 |
| **1** | **0** | **0** | **0** | **0** | **0** | **1** |
| X | 1 | 0 | 0 | 0 | 1 | 1 |
| X | X | 1 | 0 | 1 | 0 | 1 |
| X | X | X | 1 | 1 | 1 | 1 |

**Scenario based:**

As the inputs to the CAM architecture are dynamic, all the outputs should be error free and should be generated as soon as possible to generate a quick output.

**Boundary Conditions:**

The boundary conditions for this cam architecture are that all the inputs are 0 or all the inputs are 1. In either of the cases if the inputs match with the output the corresponding matching address is returned. If the inputs are all 0’s then the output is don’t care.

**High Level Test Plan:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Description | Precondition | Expected Input | Expected Output | Actual Output |
| 1 | CAM Cell | 000 | Bit, nbit, write | Matched bit | Matched bit with glitches at times |
| 2 | CAM Cell | 001 | Bit, nbit, write | Write bit | Write bit |
| 3 | Priority encoder | 0000 | D0,D1,D2,D3 | Y0,Y1 | Priority Based |
| 4 | Priority encoder | 0000 | 1000 | 00 | 00 |
| 5 | Priority encoder | 0000 | X100 | 01 | 01 |
| 6 | Priority encoder | 0000 | XX10 | 10 | 10 |
| 7 | Priority encoder | 0000 | XXX11 | 11 | 11 |

**Test Plan for Low level requirements:**

**Requirement based:**

To get the correct output from priority encoder, check the outputs for inverter, 3 input nand cell, 4 input nor cell, 2 input OR cell, Inverter.

**Inverter**

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| 0 | 1 |
| 1 | 0 |

**2 Input OR Gate**

|  |  |  |
| --- | --- | --- |
| A | B | Y |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**3 Input NAND Gate**

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | X |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

**4 Input NOR Gate**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | D | Y |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 |

**Scenario based:**

If any condition goes beyond the voltage levels, glitches will occur. However, addition of a sense amplifier or a buffer can help to amplify and stabilize the binary level.

**Boundary Conditions:**

Boundary conditions for the pmos and nmos are 0 V or 1V. In case of deviations, there will be glitches in the output waveform.

**Low Level Test Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | Inverter | 0 | 0 | 1 | 1 |
| 2 | Inverter | 0 | 1 | 0 | 0 |
| 3 | 2 Input Or Gate | 00 | 00 | 0 | 0 |
| 4 | 2 Input Or Gate | 00 | 01 | 1 | 1 |
| 5 | 2 Input Or Gate | 00 | 10 | 1 | 1 |
| 6 | 2 Input Or Gate | 00 | 11 | 1 | 1 |
| 7 | 3 Input NAND Gate | 000 | 000 | 1 | 1 |
| 8 | 3 Input NAND Gate | 000 | 001 | 1 | 1 |
| 9 | 3 Input NAND Gate | 000 | 010 | 1 | 1 |
| 10 | 3 Input NAND Gate | 000 | 011 | 1 | 1 |
| 11 | 3 Input NAND Gate | 000 | 100 | 1 | 1 |
| 12 | 3 Input NAND Gate | 000 | 101 | 1 | 1 |
| 13 | 3 Input NAND Gate | 000 | 110 | 1 | 1 |
| 14 | 3 Input NAND Gate | 000 | 111 | 0 | 0 |
| 15 | 4 Input NOR Gate | 0000 | 0000 | 1 | 1 |
| 16 | 4 Input NOR Gate | 0000 | 0001 | 0 | 0 |
| 17 | 4 Input NOR Gate | 0000 | 0010 | 0 | 0 |
| 18 | 4 Input NOR Gate | 0000 | 0011 | 0 | 0 |
| 19 | 4 Input NOR Gate | 0000 | 0100 | 0 | 0 |
| 20 | 4 Input NOR Gate | 0000 | 0101 | 0 | 0 |
| 21 | 4 Input NOR Gate | 0000 | 0110 | 0 | 0 |
| 22 | 4 Input NOR Gate | 0000 | 0111 | 0 | 0 |
| 23 | 4 Input NOR Gate | 0000 | 1000 | 0 | 0 |
| 24 | 4 Input NOR Gate | 0000 | 1001 | 0 | 0 |
| 25 | 4 Input NOR Gate | 0000 | 1010 | 0 | 0 |
| 26 | 4 Input NOR Gate | 0000 | 1011 | 0 | 0 |
| 27 | 4 Input NOR Gate | 0000 | 1100 | 0 | 0 |
| 28 | 4 Input NOR Gate | 0000 | 1101 | 0 | 0 |
| 29 | 4 Input NOR Gate | 0000 | 1110 | 0 | 0 |
| 30 | 4 Input NOR Gate | 0000 | 1111 | 0 | 0 |
| 31 | VDD | 5V | 5V | 5V | 5V |
| 32 | VDD | 5V | Any voltage other than 5V | Glitches | Glitches |
| 33 | Vdc | 1V | 1V | 1V | 1V |
| 34 | Vdc | 1V | Any voltage other than 5V | Error | Error |
| 35 | Ground | 0V | 0V | 0V | 0V |
| 37 | Bit | 0 | 0 or 1 | Bit | 0 or 1 |
| 38 | N-Bit | 0 | 0 or 1 | Inverted output of bit | 1 or 0 |

**CI/CD Task**

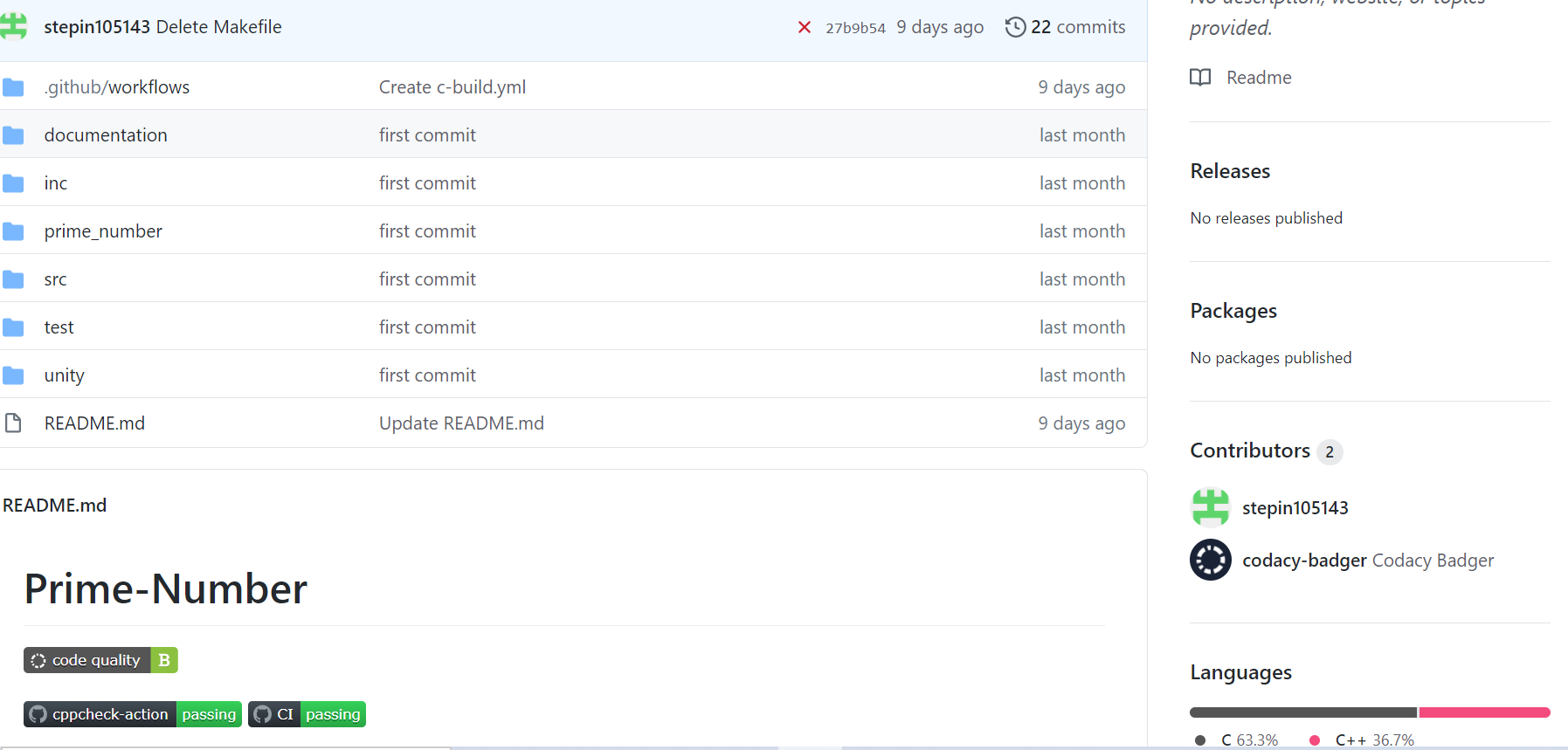
**GIT Summary**

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.

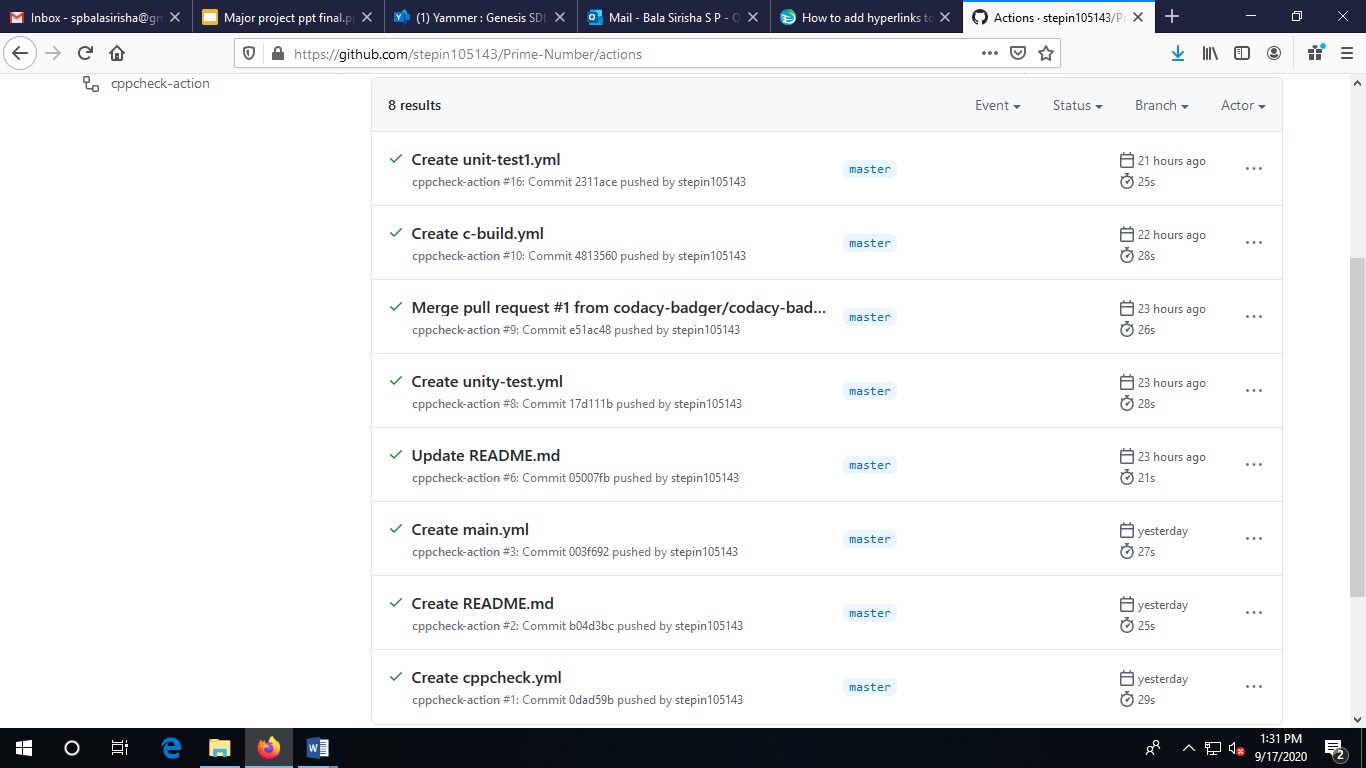
The basic operations in GIT involve:

1. Create a repository
2. Create a branch
3. Create Pull Request
4. Make Pull Request
5. Commit Changes

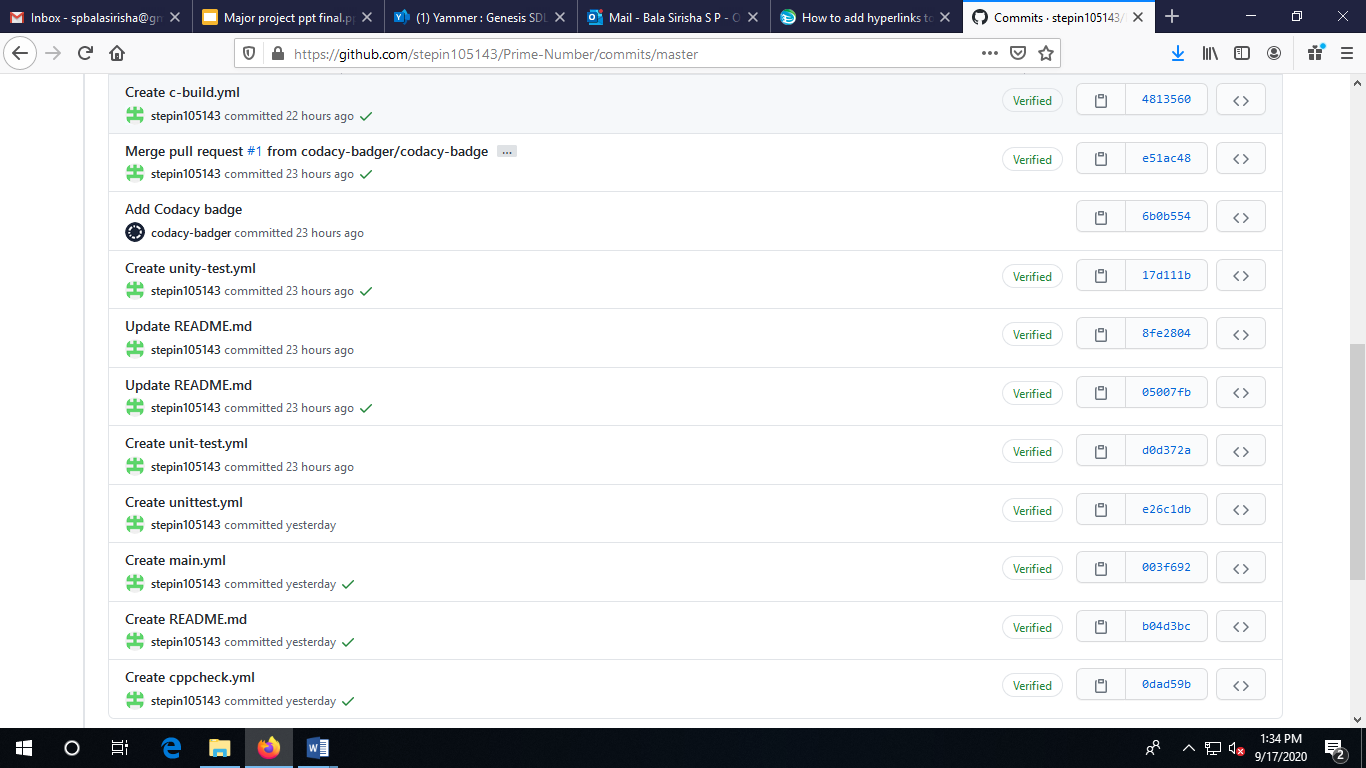
**Badges**

****

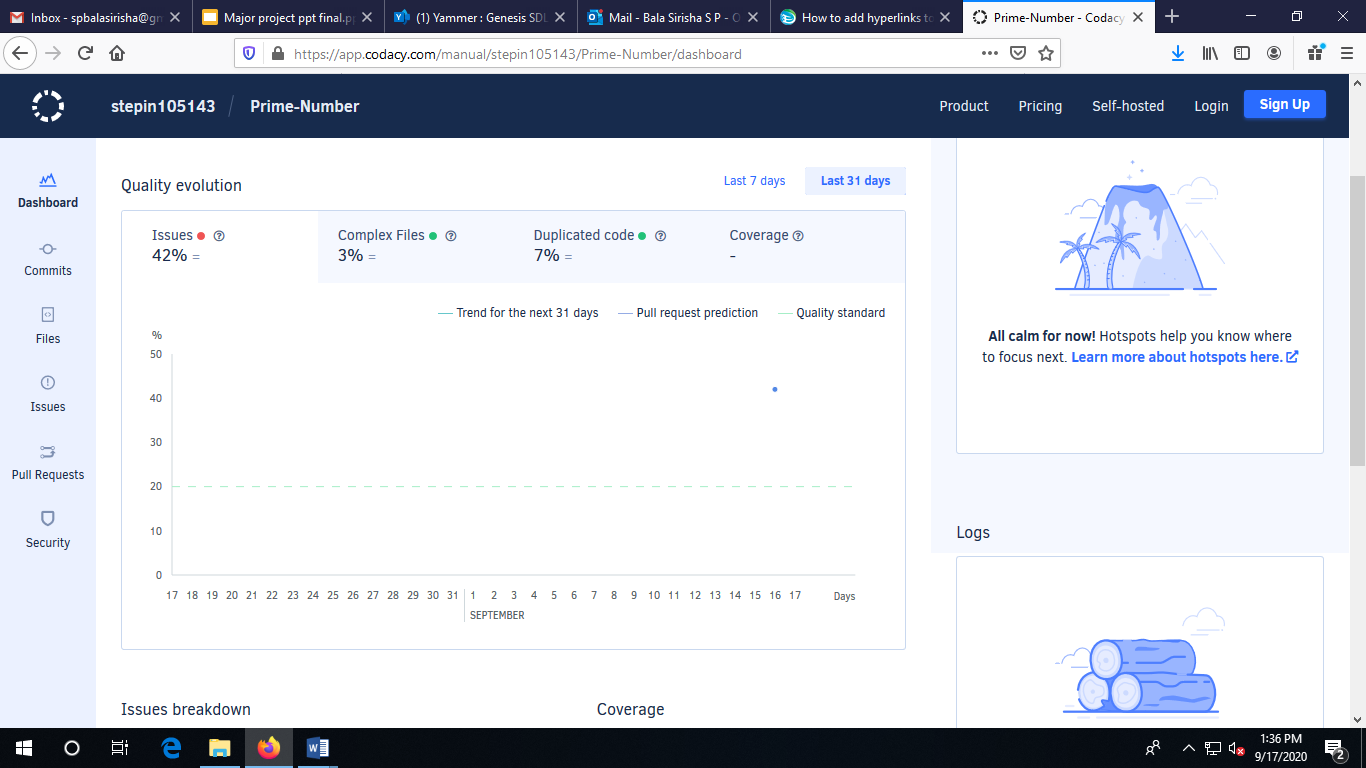
**Actions**



**Commit History**



**Issues**



**Appendix:**

[**https://github.com/stepin105143/Prime-Number**](https://github.com/stepin105143/Prime-Number)

[**https://github.com/stepin105143**](https://github.com/stepin105143)

**REFERENCES:**

[1] T. Kohonen, Content-Addressable Memories, 2nd ed. New York: Springer-Verlag, 1987.

[2] L. Chisvin and R. J. Duckworth, “Content-addressable and associative memory: alternatives to the ubiquitous RAM,” IEEE Computer, vol. 22, no. 7, pp. 51–64, Jul. 1989.

[3] K. E. Grosspietsch, “Associative processors and memories: a survey,” IEEE Micro, vol. 12, no. 3, pp. 12–19, Jun. 1992.

[4] I. N. Robinson, “Pattern-addressable memory,” IEEE Micro, vol. 12, no. 3, pp. 20–30, Jun. 1992.

[5] S. Stas, “Associative processing with CAMs,” in Northcon/93 Conf. Record, 1993, pp. 161–167.

## **AGILE MODEL**

**What is Agile Model?**

Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like −

* Planning
* Requirements Analysis
* Design
* Coding
* Unit Testing and
* Acceptance Testing.

At the end of the iteration, a working product is displayed to the customer and important stakeholders.

**The Agile Process Flow**

1. **Concept** - Projects are envisioned and prioritized
2. **Inception** - Team members are identified, funding is put in place, and initial environments and requirements are discussed
3. **Iteration/Construction** - The development team works to deliver working software based on iteration requirements and feedback
4. **Release** - QA (Quality Assurance) testing, internal and external training, documentation development, and final release of the iteration into production
5. **Production** - Ongoing support of the software
6. **Retirement** - End-of-life activities, including customer notification and migration

**A typical iteration process flow can be visualized as follows:**

* **Requirements** - Define the requirements for the iteration based on the product backlog, sprint backlog, customer and stakeholder feedback
* **Development** - Design and develop software based on defined requirements
* **Testing** - QA (Quality Assurance) testing, internal and external training, documentation development
* **Delivery** - Integrate and deliver the working iteration into production
* **Feedback** - Accept customer and stakeholder feedback and work it into the requirements of the next iteration

**Agile User Stories:**

**Operating Systems:**

Consider the case of windows operating system in our computers. Initially, the user comes up with a user story as follows:

I want to have a system software that manages computer hardware, software resources, and provides common services for computer programs. For hardware functions such as input and output and memory allocation, it should acts as an intermediary between programs and the computer hardware. It should be compatible with a system with **CPU of** 1GHz (32bit) and 2GHz (64bit); **RAM of**1GB (32bit) 2GB (64bit); **Disk Space of** 16GB (32bit) 20GB (64bit); **Video with** Direct X9 with WDDM 1.0 or higher driverand **Media type of** DVD ROM within a stipulated time frame of 2 months.

To begin with, the developers come up with an action plan to prepare the business model canvas and develop the operating system. They document the requirements, design and develop it in concrete build form. After testing each of the build, they release it into the market. As the customers are the end users of the system, they continuously take the feedback from them. Later, they introduce the additional features into the system in the form of updates. These updates are given to either introduce special features or to fix bugs and this makes it an iterative as well as incremental process.

**ATM User Story**

Let us assume that a customer wants to access his bank accounts and use the ATM. The customer would come up with the following user story:

As a Customer, I want to Login to my account using card and PIN code, so that I can perform the transactions. The acceptance criteria would ensure that the system validates the card and pin code. The card number must not exceed 16 0r 19 digits depending on the bank and the first digits of the card would comply with a designated bank. In case Customer enters wrong pin code three times then the system must lock the card.

Initially, the product was launched in order to leverage the process of withdrawal of cash and checking account balance. Later on, the product evolved to offer multiple facilities like open or withdraw a fixed deposit, recharge the mobile, pay income tax, deposit cash, pay insurance premium, apply for personal loan, transfer cash, Pay the bills.

**HR Manager User Story**

For instance, the HR manager of a firm puts forward the following user story:

As a HR Manager, I need to view a candidate’s status so that I can manage their application process throughout the recruiting phases. The acceptance Criteria clinches that the HR Manager is able to log in to the virtual job openings board system, view / edit / add the status for job candidates, update for each phase (e.g. Phone Screening Completed, In-person Interview Scheduled, Background Check in-progress, etc.),send email communication to staff regarding candidate.

In this case, the HR manager is the end user and his feedback plays a crucial role in updating the product to ameliorate the customer experience. The incessant feedback makes certain that new features are added to the end product as per the viability and unique proposition value.

**References:**

For agile process flow:

<https://www.smartsheet.com/understanding-agile-software-development-lifecycle-and-process-workflow>.

## **ADVANCED SCIENTIFIC CALCULATOR**

The advanced scientific calculator performs various operations ranging from calculating the sum, difference, product, division, bitwise Or, bitwise and, xor of two numbers to calculating the factorial, power, square root, sine,cosine,tan,asine,acos,atan,sinh,tanh,cosh,ceil,floor,logarithmic values of a given number.

It is equipped with a special feature to calculate the sum of all numbers till the given input number and also calculate the factorial of a given number.

**Requirements Table:**

|  |  |
| --- | --- |
| ID | Description |
| H\_01 | Sum of two numbers |
| H\_02 | Diﬀerence of two numbers |
| H\_03 | Product of two numbers |
| H\_04 | Division of two numbers |
| L\_01 | Square of number |
| L\_02 | Cube of Number |
| L\_03 | Factorial of a number |
| L\_04 | Power of number |
| L\_05 | Square root of number |
| L\_06 | sine of number |
| L\_07 | cosine of number |
| L\_08 | tan of number |
| L\_09 | bitwise or of two numbers |
| L\_10 | bitwise and of two numbers |
| L\_11 | xor of two numbers |
| L\_12 | logarithmic value of number |
| L\_13 | log base 10 value of number |
| L\_14 | sinh of number |
| L\_15 | cosh of number |
| L\_16 | tanh of number |
| L\_17 | exponent of number |
| L\_18 | ﬂoor value of number |
| L\_20 | absolute value of number |

**Design**

**Activity Diagram**

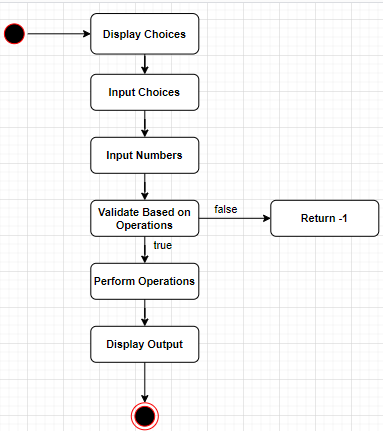


Fig: Activity Diagram

**Use Case Diagram:**

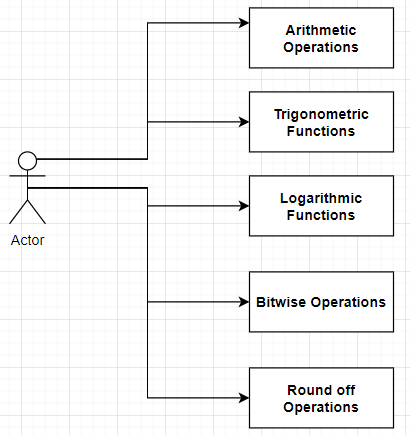
****

Fig: Use Case Diagram

**Test Plan Table:**

**High Level Test Plan:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Description | Precondition | Expected Input | Expected Output | Actual Output |
| 1 | SUM | 0,0 | num1,num2 | sum of two numbers | num1+num2 |
| 2 | Diﬀerence | 0,0 | num1,num2 | Diﬀerence of two numbers | num1-num2 |
| 3 | Product | numbers cannot be zero | num1,num2 | Product of two numbers | num1\*num2 |
| 4 | Division | num2!=0 | num1,num2 | Division of two numbers | num1/num2 |

**Low Level Test Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | Square | 0 | num1 | square of number | num1\*num1 |
| 2 | Cube | 0 | num1 | cube of number | num1\*num1  \*num1 |
| 3 | Factorial | 1 | num1 | factorial of number | num1! |
| 4 | Power | numbers cannot be zero | num1,num2 | power of numbers | pow(num1,n um2) |
| 5 | Square Root | number cannot be zero | num1 | square root of number | sqrt(num1) |
| 6 | Sine | 0 | num1 | sine of number | sin(num1) |
| 7 | Cosine | 0 | num1 | cosine of number | cos(num1) |
| 8 | Tan | 0 | num1 | tan of number | tan(num1) |
| 9 | Bitwise  OR | 0,0 | num1,nu m2 | bitwise or of two numbers | num1||num 2 |
| 10 | Bitwise AND | 0,0 | num1,nu m2 | bitwise and of two numbers | num1&&num 2 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 11 | XOR | 0,0 | num1,nu m2 | xor of two numbers | num1^num2 |
| 12 | Logarithmic value | 0 | num1 | logarithmi c value of number | ln(num1) |
| 13 | Log base 10 | 0 | num1 | log base 10 value of number | log(num1) |
| 14 | sinh | 0 | num1 | sinh of number | sinh(num1) |
| 15 | cosh | 0 | num1 | cosh of number | cosh(num1) |
| 16 | tanh | 0 | num1 | tanh of number | tanh(num1) |
| 17 | Exponent | 0 | num1 | exponent of number | exp(num1) |
| 18 | ﬂoor | 0 | num1 | ﬂoor value of number | ﬂoor(num1) |
| 19 | Ceil | 0 | num1 | ceil value of number | ceil(num1) |
| 20 | Absolute | 0 | num1 | absolute value of number | abs(num1) |

**GITHUB**

**Link to Repository**:

<https://github.com/99002653>

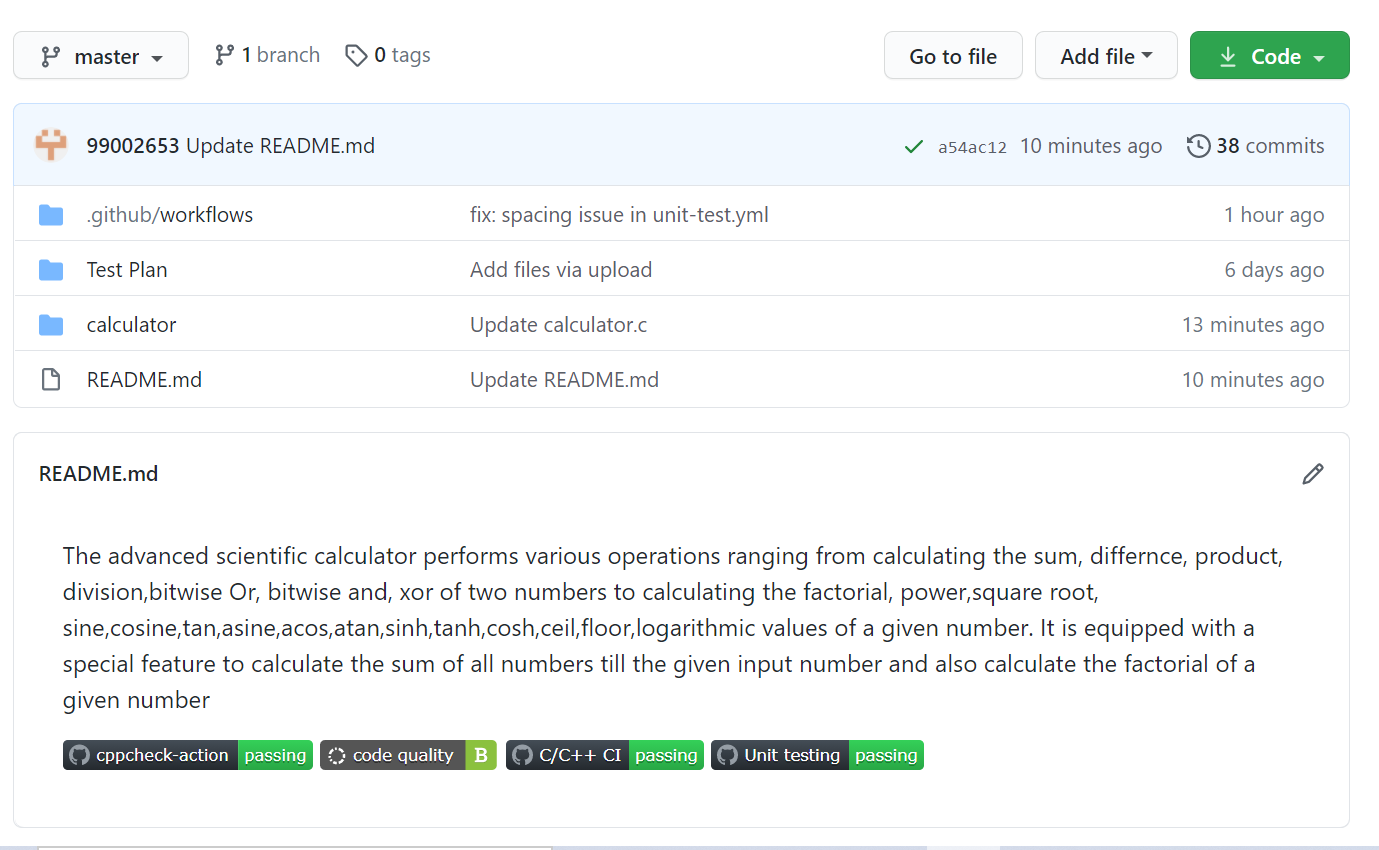
**GIT SUMMARY**

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.

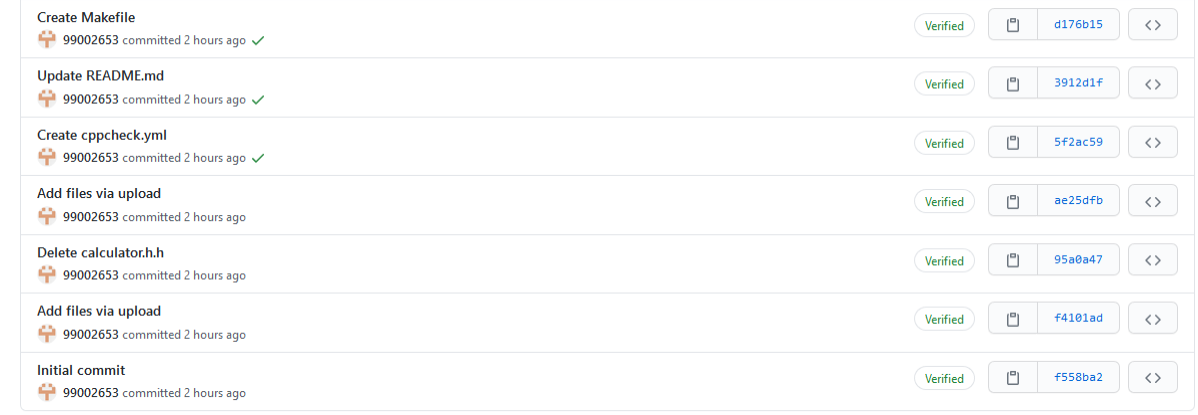
The basic operations in GIT involve:

1. Create a repository
2. Create a branch
3. Create Pull Request
4. Make Pull Request
5. Commit Changes

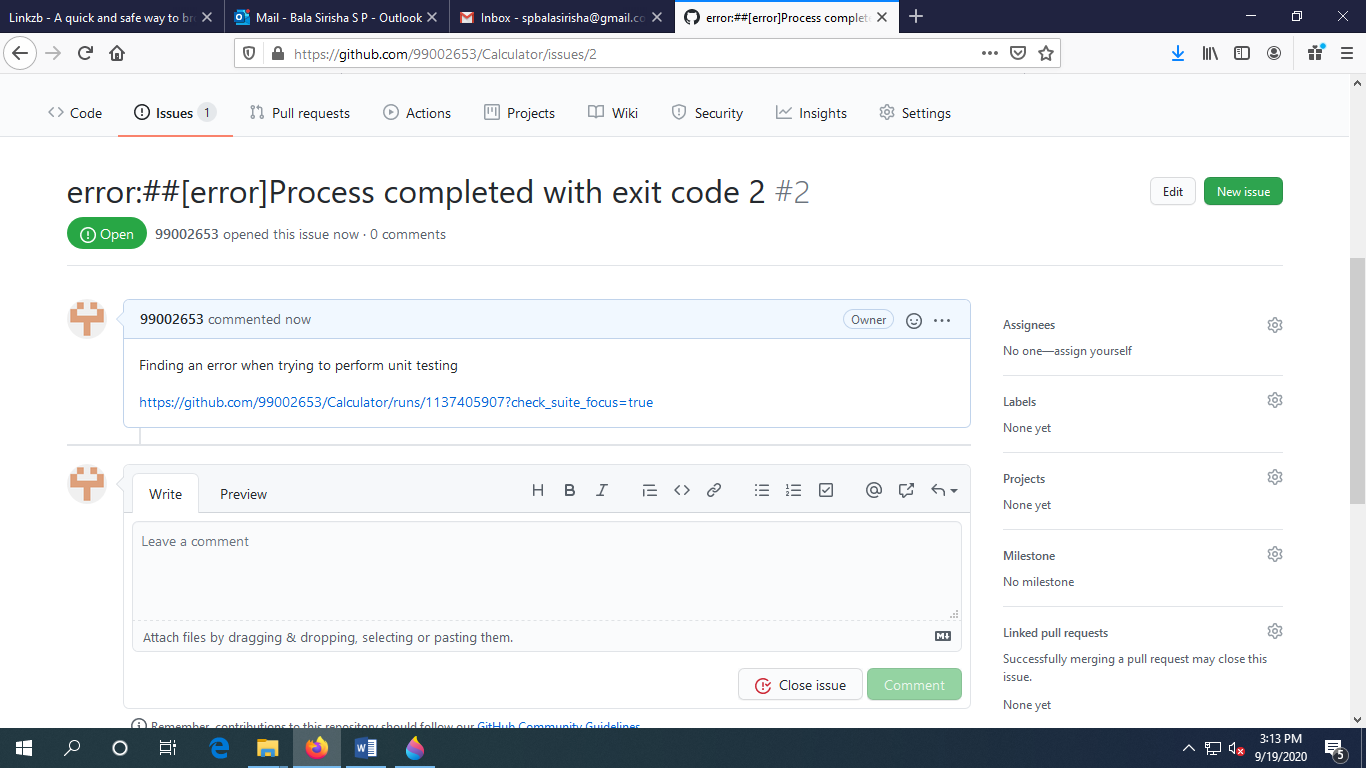
**Badges**

****

**Commit History**

****

**Raising Issues**



**Issues shown via Codacy**

