./

Learning Report –

Applied SDLC and Software Testing



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **Approved By** | **Remarks/Revision Details** |
|  | 17-09-2020 | Aditi Dutta |  |  | Updation to the previous report |
|  | 18-09-2020 | Aditi Dutta |  |  | Updation to the previous report |
|  | 19-09-2020 | Aditi Dutta |  |  | Updation to the previous report |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Document History**

# 

Contents

[Checklist 5](#_Toc51420347)

[Activity and Tasks 5](#_Toc51420348)

[Activity 1– System/Software Development 5](#_Toc51420349)

[Activity 2 –CI Workflow for C Programming 5](#_Toc51420350)

[Activity 3 – Agile Aspects 5](#_Toc51420351)

[Activity 1– System/Software Development 6](#_Toc51420352)

[ Definition 6](#_Toc51420353)

[ Research 7](#_Toc51420354)

[ Ageing 7](#_Toc51420355)

[ Cost gradation 7](#_Toc51420356)

[Solar modules cost 7](#_Toc51420357)

[Production Cost 7](#_Toc51420358)

[ Definition of System 7](#_Toc51420359)

[ SWOT Analysis 8](#_Toc51420360)

[Strengths 8](#_Toc51420361)

[Weaknesses 8](#_Toc51420362)

[Opportunities 8](#_Toc51420363)

[Threats 8](#_Toc51420364)

[ Detailed Requirements 8](#_Toc51420365)

[ Low level Design /System Level 9](#_Toc51420366)

[ Requirement Based Test Plan 9](#_Toc51420367)

[V Model 10](#_Toc51420368)

[Agile Model 10](#_Toc51420369)

[Activity 2 –CI Workflow for C Programming 11](#_Toc51420370)

[Makefile 11](#_Toc51420371)

[Build 12](#_Toc51420372)

[Code Quality 12](#_Toc51420373)

[Code 13](#_Toc51420374)

[Activity 1– System/Software Development 15](#_Toc51420375)

[ Definition 15](#_Toc51420376)

[ Research 15](#_Toc51420377)

[ Ageing and Cost Degradation 15](#_Toc51420378)

[ Definition of System 15](#_Toc51420379)

[ SWOT Analysis 15](#_Toc51420380)

[Strengths 15](#_Toc51420381)

[Weaknesses 15](#_Toc51420382)

[Opportunities 15](#_Toc51420383)

[Threats 16](#_Toc51420384)

[ Detailed Requirements 16](#_Toc51420385)

[ Requirement Based Test Plan 18](#_Toc51420386)

[Agile Model 18](#_Toc51420387)

[Activity 2 –CI Workflow for C Programming 18](#_Toc51420388)

[Appendix 22](#_Toc51420389)

[References 26](#_Toc51420390)

# Checklist

* Installation of SW on Phone and Desktop
* Additional Aspects …

# Activity and Tasks

## **Activity 1**– System/Software Development

* Sub Tasks
* Complete and Evolve

## **Activity 2** –CI Workflow for C Programming

* Sub Tasks
* Complete and Evolve

## **Activity 3** – Agile Aspects

* …..

## **Activity 1**– System/Software Development

* Section-1-Requirements

# Definition

A solar cell, or photovoltaic cell, is an electrical device that converts the energy of light directly into electricity by the photovoltaic effect, which is a physical and chemical phenomenon. It is a form of photoelectric cell, defined as a device whose electrical characteristics, such as current, [voltage](https://en.wikipedia.org/wiki/Voltage), or [resistance](https://en.wikipedia.org/wiki/Electrical_resistance_and_conductance), vary when exposed to light. Individual solar cell devices are often the electical building blocks of [photovoltaic modules](https://en.wikipedia.org/wiki/Solar_panel), known colloquially as solar panels. The common single junction [silicon](https://en.wikipedia.org/wiki/Silicon) solar cell can produce a maximum [open-circuit voltage](https://en.wikipedia.org/wiki/Open-circuit_voltage) of approximately 0.5 to 0.6 volts.

Solar cells are described as being [photovoltaic](https://en.wikipedia.org/wiki/Photovoltaics), irrespective of whether the source is [sunlight](https://en.wikipedia.org/wiki/Sunlight) or an artificial light. In addition to producing energy, they can be used as a [photodetector](https://en.wikipedia.org/wiki/Photodetector) (for example [infrared detectors](https://en.wikipedia.org/wiki/Infrared_detector)), detecting light or other [electromagnetic radiation](https://en.wikipedia.org/wiki/Electromagnetic_radiation) near the visible range, or measuring light intensity.

The operation of a photovoltaic (PV) cell requires three basic attributes:

* The absorption of light, generating either [electron](https://en.wikipedia.org/wiki/Electron)-[hole](https://en.wikipedia.org/wiki/Electron_hole) pairs or [excitons](https://en.wikipedia.org/wiki/Exciton).
* The separation of [charge carriers](https://en.wikipedia.org/wiki/Charge_carrier) of opposite types.
* The separate extraction of those carriers to an external circuit.

In contrast, a [solar thermal collector](https://en.wikipedia.org/wiki/Solar_thermal_collector) supplies [heat](https://en.wikipedia.org/wiki/Heat) by [absorbing sunlight](https://en.wikipedia.org/wiki/Absorption_(electromagnetic_radiation)), for the purpose of either direct heating or indirect [electrical power generation](https://en.wikipedia.org/wiki/Electrical_power_generation) from heat. A "photoelectrolytic cell" ([photoelectrochemical cell](https://en.wikipedia.org/wiki/Photoelectrochemical_cell" \o "Photoelectrochemical cell)), on the other hand, refers either to a type of photovoltaic cell (like that developed by [Edmond Becquerel](https://en.wikipedia.org/wiki/Edmond_Becquerel) and modern [dye-sensitized solar cells](https://en.wikipedia.org/wiki/Dye-sensitized_solar_cell)), or to a device that [splits water](https://en.wikipedia.org/wiki/Water_splitting) directly into [hydrogen](https://en.wikipedia.org/wiki/Hydrogen) and [oxygen](https://en.wikipedia.org/wiki/Oxygen) using only solar illumination.

# Research

# Ageing

The [photovoltaic effect](https://en.wikipedia.org/wiki/Photovoltaic_effect) was experimentally demonstrated first by French physicist [Edmond Becquerel](https://en.wikipedia.org/wiki/Edmond_Becquerel). In 1839, at age 19, he built the world's first photovoltaic cell in his father's laboratory. [Willoughby Smith](https://en.wikipedia.org/wiki/Willoughby_Smith) first described the "Effect of Light on Selenium during the passage of an Electric Current" in a 20 February 1873 issue of [Nature](https://en.wikipedia.org/wiki/Nature_(magazine)). In 1883 [Charles Fritts](https://en.wikipedia.org/wiki/Charles_Fritts) built the first [solid state](https://en.wikipedia.org/wiki/Solid_state_(electronics)) photovoltaic cell by coating the [semiconductor](https://en.wikipedia.org/wiki/Semiconductor) [selenium](https://en.wikipedia.org/wiki/Selenium) with a thin layer of [gold](https://en.wikipedia.org/wiki/Gold) to form the junctions; the device was only around 1% efficient. Other milestones include:

|  |  |  |
| --- | --- | --- |
| 1888 | first cell | based on the outer [photoelectric effect](https://en.wikipedia.org/wiki/Photoelectric_effect) |
| 1905 | Proposed new quantum theory of light | [photoelectric effect](https://en.wikipedia.org/wiki/Photoelectric_effect) |
| 1941 | discovered p-n-junctions | in [Cu2O](https://en.wikipedia.org/wiki/Copper(I)_oxide) and [Ag2S](https://en.wikipedia.org/wiki/Silver_sulfide) protocells |
| 1946 | modern junction semiconductor solar cell | lead to the [transistor](https://en.wikipedia.org/wiki/Transistor) |
| 1948 |  | photo-voltaic effect |
| 1954 | first practical photovoltaic cell |  |
| 1957 | process of [silicon](https://en.wikipedia.org/wiki/Silicon) [surface passivation](https://en.wikipedia.org/wiki/Surface_passivation) by [thermal oxidation](https://en.wikipedia.org/wiki/Thermal_oxidation) | The surface passivation process has since been critical to [solar cell efficiency](https://en.wikipedia.org/wiki/Solar_cell_efficiency) |
| 1958 | Solar cells | incorporation onto the [Vanguard I](https://en.wikipedia.org/wiki/Vanguard_I) satellite |

# Cost gradation

## Solar modules cost

|  |  |
| --- | --- |
| mid-1970s | $96 per watt |
| 2016 | $68 per watt |

## Production Cost

|  |  |
| --- | --- |
| 2010 | $3.40 per watt |
| late 2011 | $1.09 per watt |
| 2012 | $0.62 per watt |

# Definition of System

Although solar energy can still be collected during cloudy and rainy days, the efficiency of the solar system drops. Solar panels are dependent on sunlight to effectively gather solar energy. Therefore, a few cloudy, rainy days can have a noticeable effect on the energy system. We should also take into account that solar energy cannot be collected during the night. So we need to batteries to store solar energy or any other storage device so that we can use it whenever it will be required. Also we need sensor to detect whether the day is sunny or cloudy. If it is cloudy, it will send a message or notification about the weather condition so that accordingly we can use the solar energy from the solar energy storage device or the batteries.

# SWOT Analysis

## Strengths

* Environmentally friendly energy
* 20 years guarantee
* Infinite energy

## Weaknesses

* Seasonal energy
* High upfront cost
* Installation requirements

## Opportunities

* Reduces the energy bill
* Innovative technology

## Threats

* Location sensitivity

# Detailed Requirements

* High Level

We need sensor to detect whether the day is sunny or cloudy. If it is cloudy, it will send a message or notification about the weather condition so that accordingly we can use the solar energy from the solar energy storage device or the batteries.

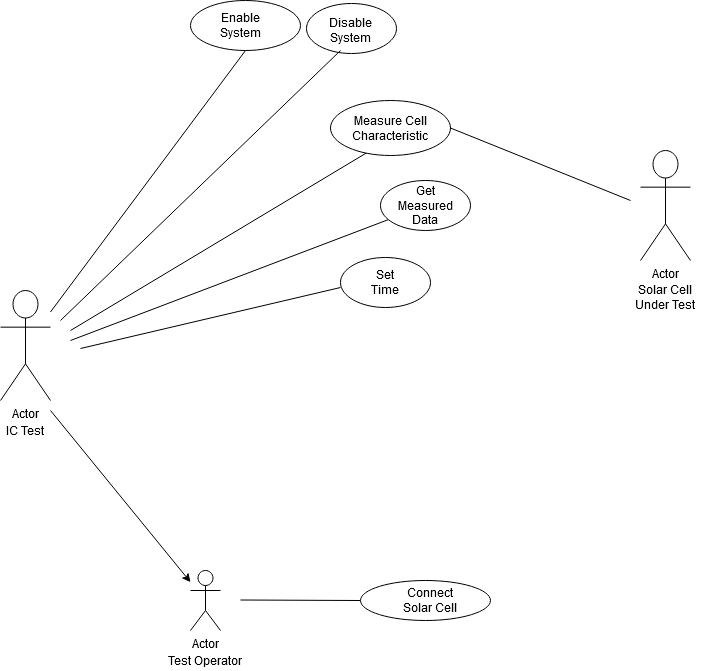
* Low Level

Although solar energy can still be collected during cloudy and rainy days, the efficiency of the solar system drops. Solar panels are dependent on sunlight to effectively gather solar energy. Therefore, a few cloudy, rainy days can have a noticeable effect on the energy system. We should also take into account that solar energy cannot be collected during the night. So, we need to batteries to store solar energy or any other storage device so that we can use it whenever it will be required.

* Section-2-Design

# Low level Design /System Level

* Structural Diagram



* Section-3-Test Plan

# Requirement Based Test Plan

* We need to batteries to store solar energy or any other storage device so that we can use it whenever it will be required.
* We need sensor to detect whether the day is sunny or cloudy. If it is cloudy, it will send a message or notification about the weather condition so that accordingly we can use the solar energy from the solar energy storage device or the batteries.
* System configuration must be working perfectly fine
* **Scenario Based Test Plan**
* At that particular weather condition, the solar cell should be able to store the charge in the battery connected or in the storage device.
* The sensor should be able to send notification so that we can use the stored energy if it is a cloudy day or at night.
* **Boundary Condition Based Test Plan**
* The battery must be connected to the solar cell.
* All other connections must be done properly to the system.

# V Model



**NOTE: To view the details of the excel sheet, double click on the above sheet**

# Agile Model

**Theme:** Solar Cell

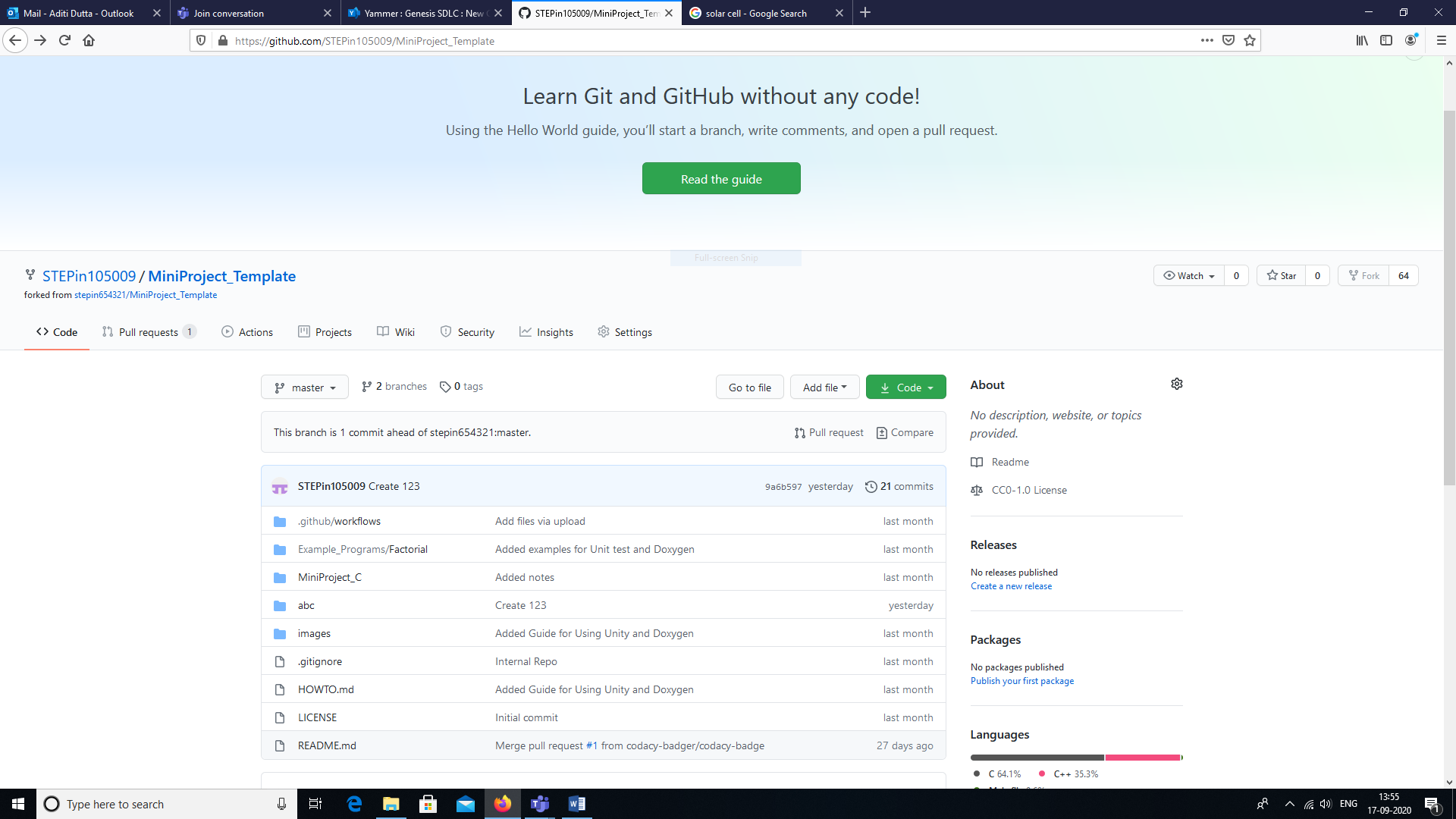
**Epic:** Although solar energy can still be collected during cloudy and rainy days, the efficiency of the solar system drops. Solar panels are dependent on sunlight to effectively gather solar energy. Therefore, a few cloudy, rainy days can have a noticeable effect on the energy system. We should also take into account that solar energy cannot be collected during the night. So, we need to batteries to store solar energy or any other storage device so that we can use it whenever it will be required. Also, we need sensor to detect whether the day is sunny or cloudy. If it is cloudy, it will send a message or notification about the weather condition so that accordingly we can use the solar energy from the solar energy storage device or the batteries.

**User Stories:**

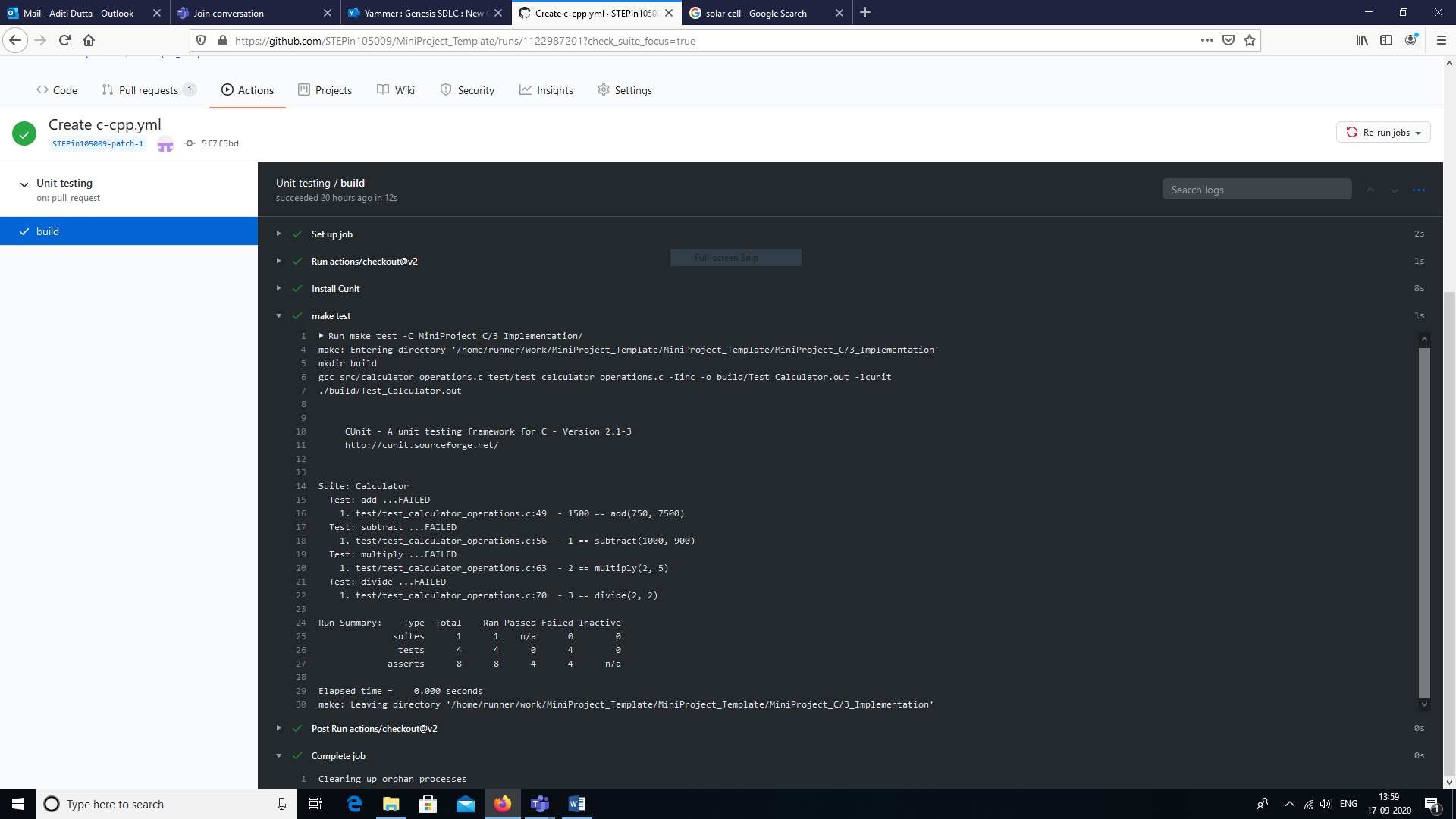
* The solar cell at times becomes location sensitive. To avoid that further implementation must be done.
* The sensor at times gives notification but it may not be appropriate. So, further development to the project must be done.
* The battery or the storage device being used should be capable enough to store sufficient amount of energy in case of extreme conditions.

## **Activity 2 –CI Workflow for C Programming**

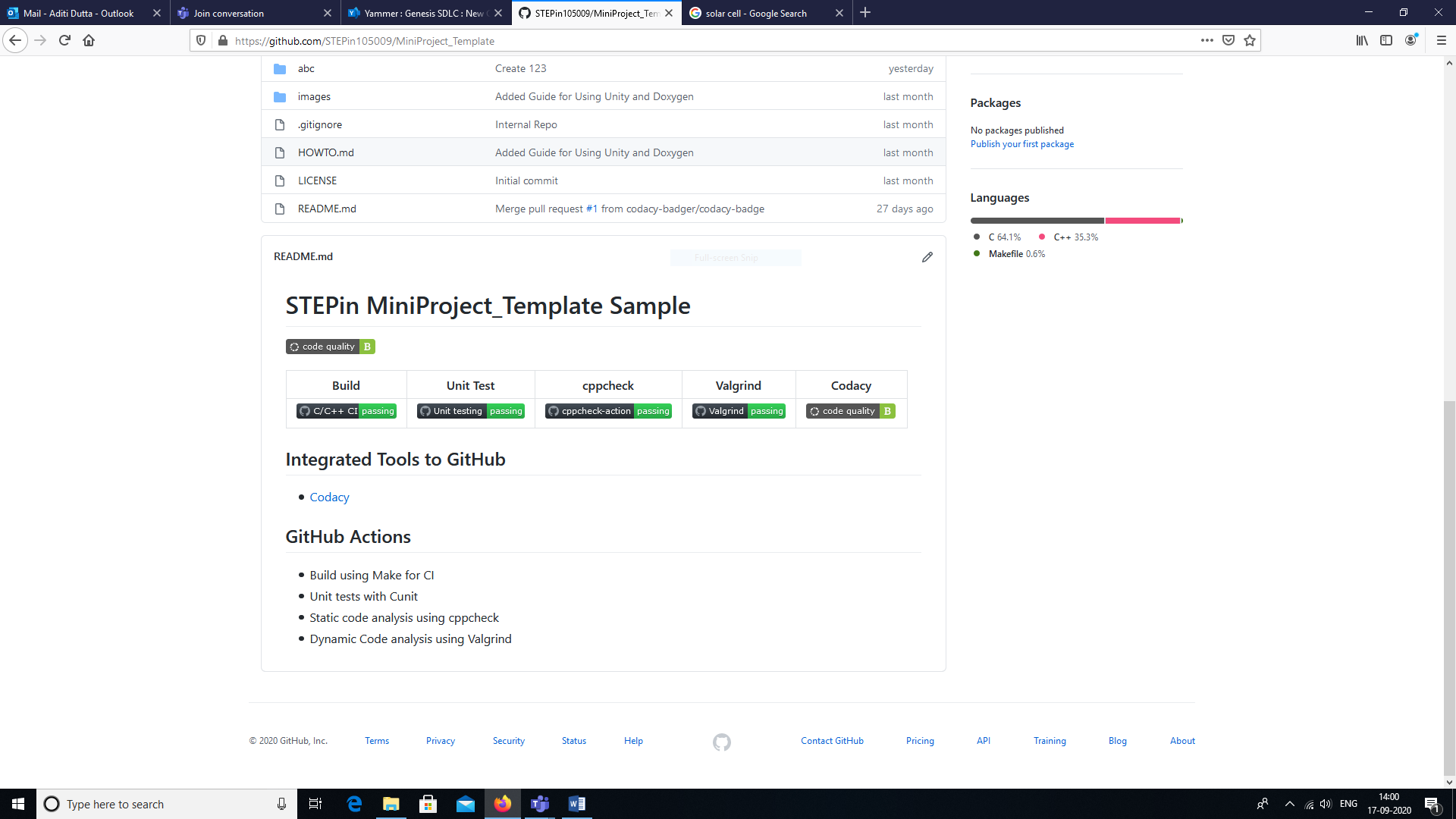
## Makefile

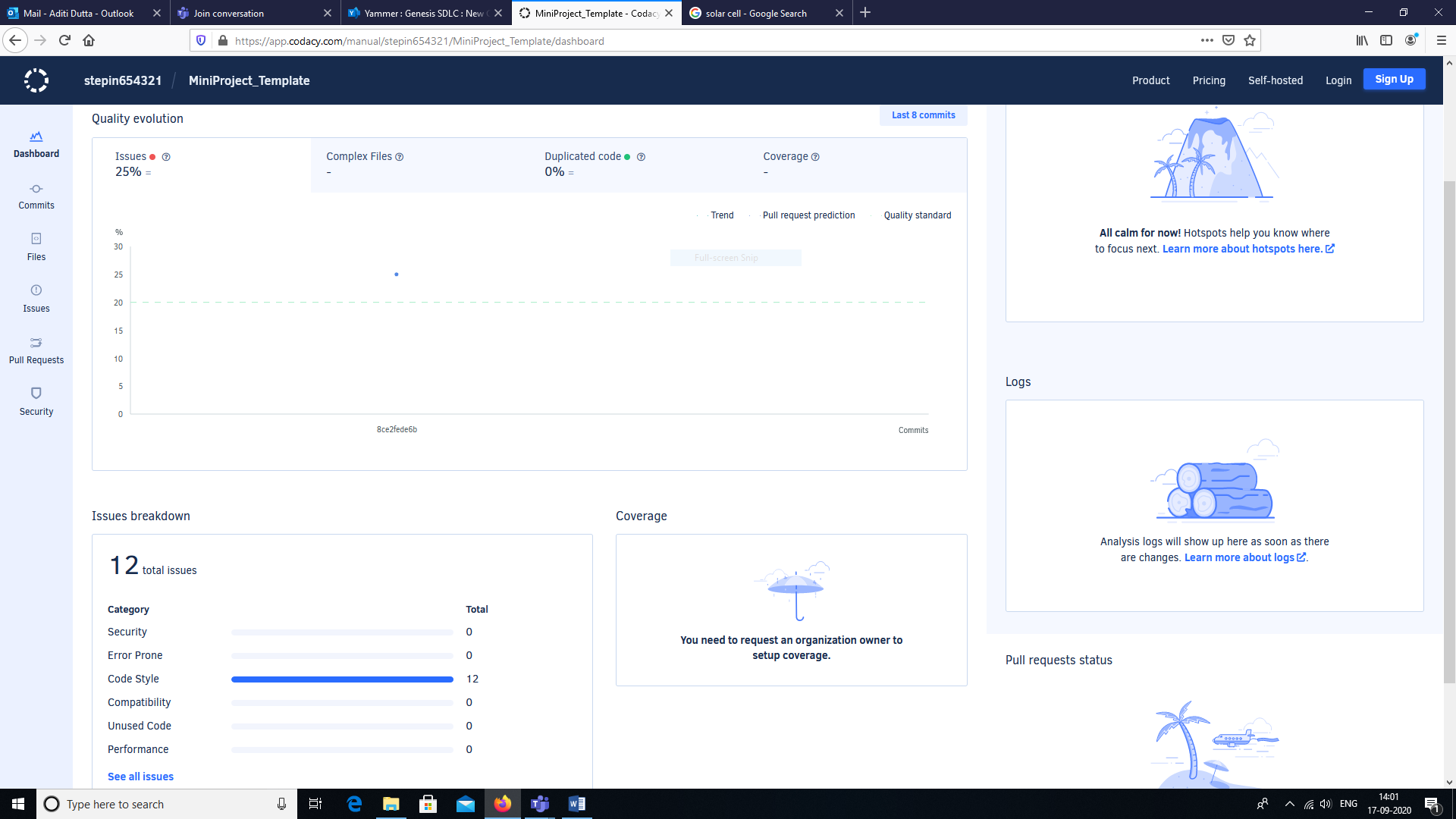


## Build

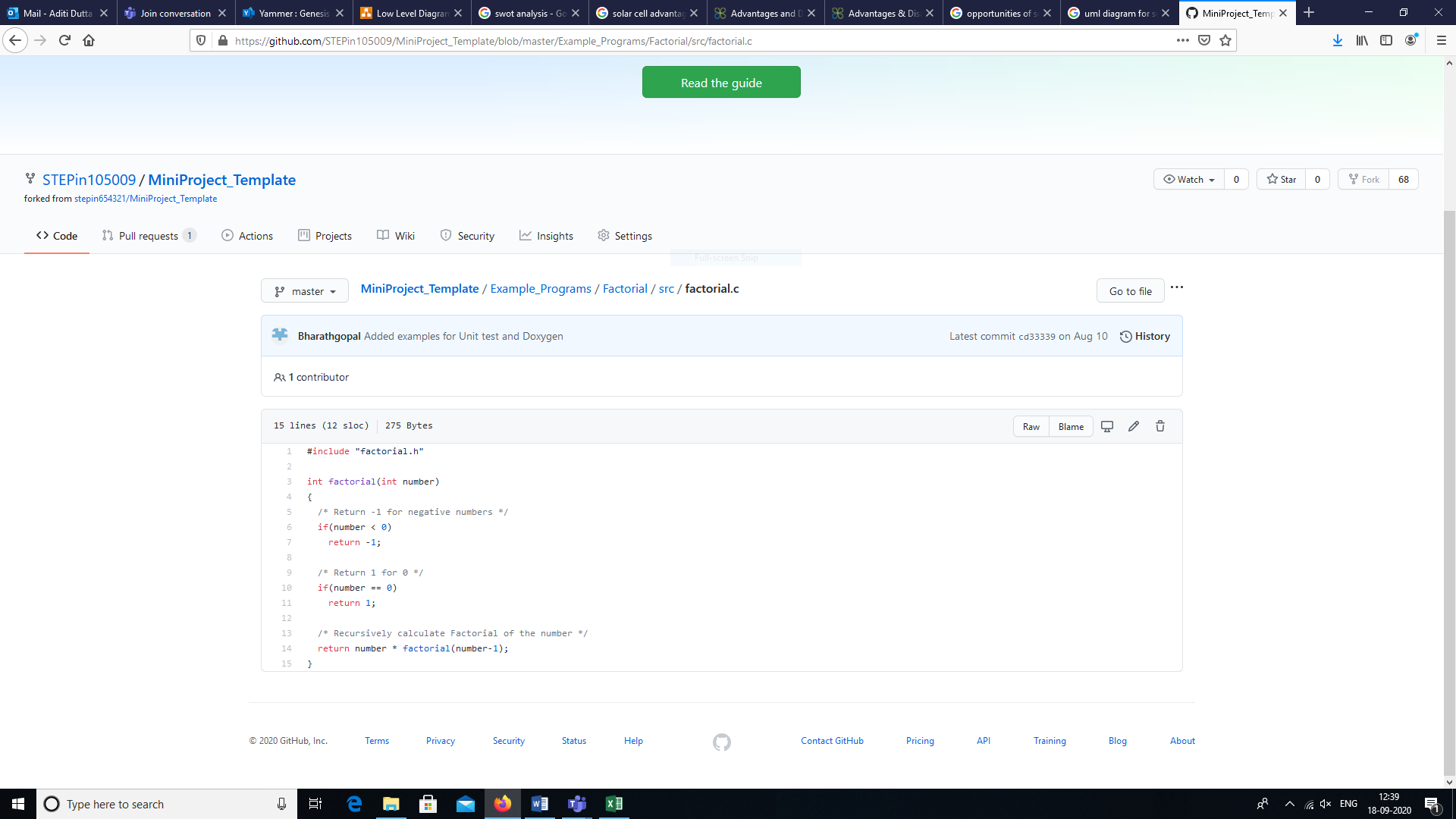


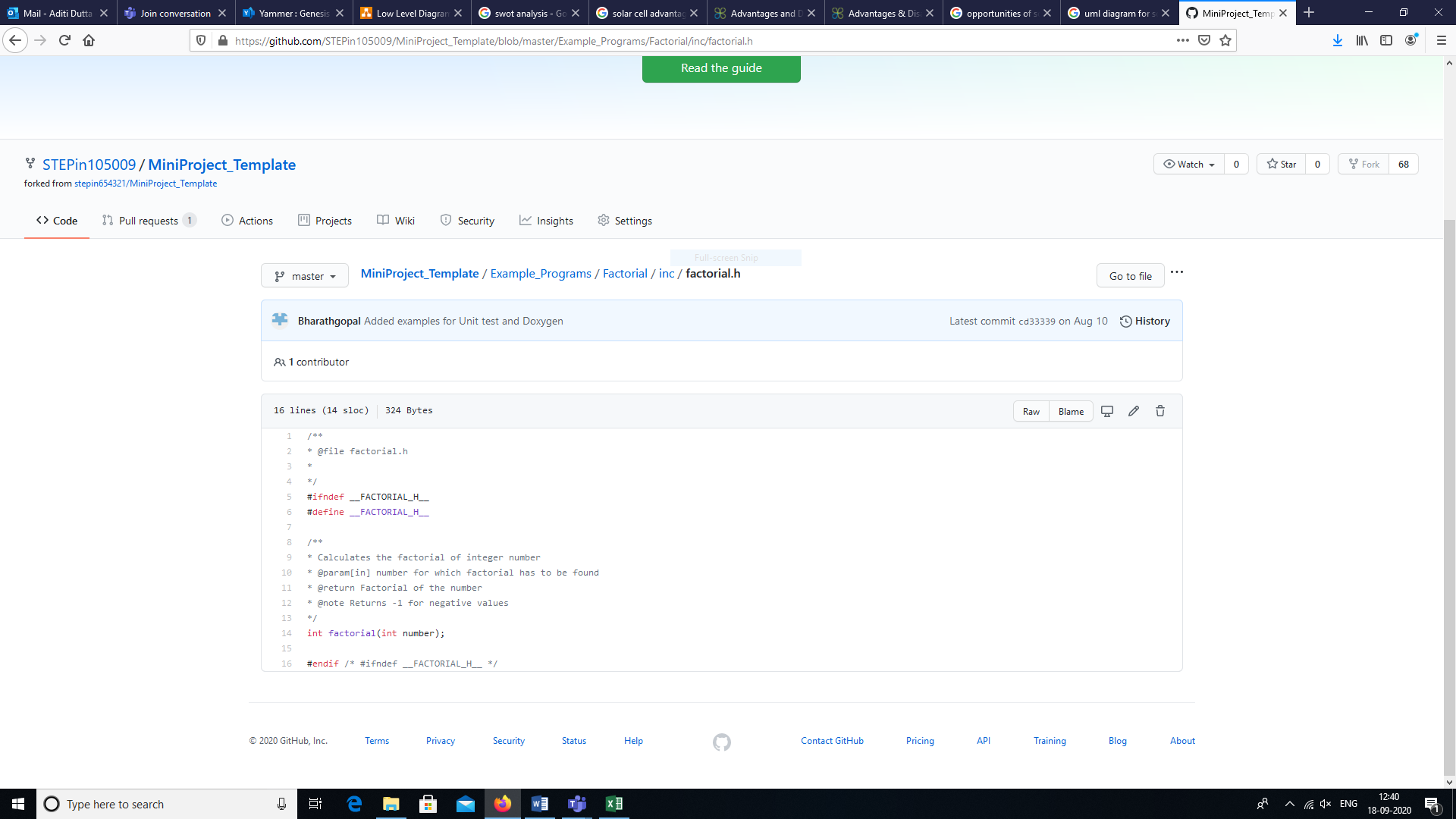
## Code Quality





## Code





LINK: <https://github.com/STEPin105009/Solar-Cell>

## **Activity 1**– System/Software Development

* Section-1-Requirements

# Definition

Calculator is something used for making mathematical calculations, in particular a small electronic device with a keyboard and a visual display.

# Research

# Ageing and Cost Degradation

Over the years, the prices have decreased for calculators. This is mainly due to the fact that nowadays people mostly use mobile phones and an application is already installed in it. Mainly, it is used by school students and college students only. So, we see a degradation in the prices of the calculator over the years.

# Definition of System

The application developed here is basically a calculator which performs the mathematical operations like addition, subtraction, multiplication, division and power function. The user has to input two operands and choose a valid operation to be performed. Then, the result will be displayed. If an invalid choice is entered, it will exit from the menu.

# SWOT Analysis

## Strengths

* Environmentally friendly application
* Can perform operations anytime and every time

## Weaknesses

* Does not perform all operations
* There may be some incorrect results due to the invalid or wrong user input

## Opportunities

* Saves human power to calculate manually
* Innovative technology

## Threats

* When some invalid user input is given, it may end the program

# Detailed Requirements

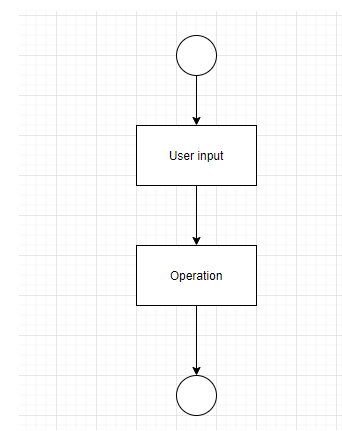
* High Level

All mathematical operations like addition, subtraction, multiplication, division and power function needs to be performed when a user will give two operands as input.

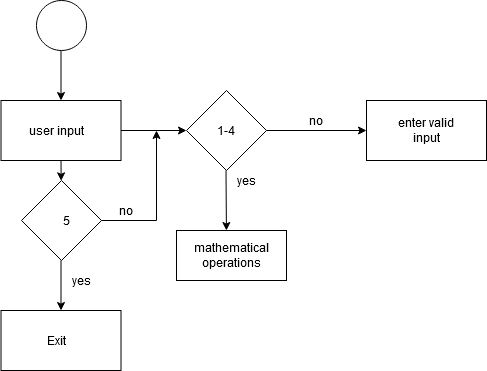
* Low Level

The results obtained or the operations performed needs to be correct or error-free.

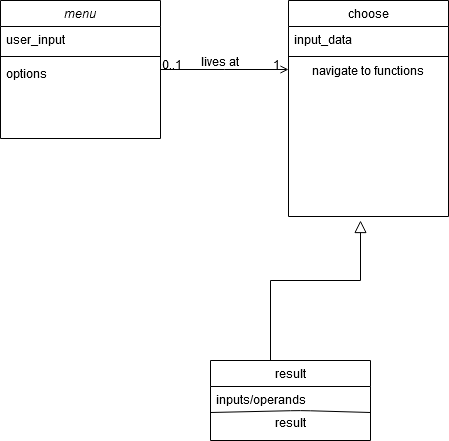
* Section-2-Design



**State level UML for Calculator**



**Activity UML for Calculator**



**Class UML for Calculator**

* Section-3-Test Plan

# Requirement Based Test Plan

* We need the application to perform well.
* There should not be any discrepancies in the results.
* **Scenario Based Test Plan**
* At a particular moment, there should be a particular operation working perfectly fine.
* If a user gives invalid input, then, it should properly terminate or exit the program.
* **Boundary Condition Based Test Plan**
* There should be a limiting value for the user to input as the two operands.
* User should not give too large values as operands to perform the operations.

# Agile Model

**Theme:** Calculator

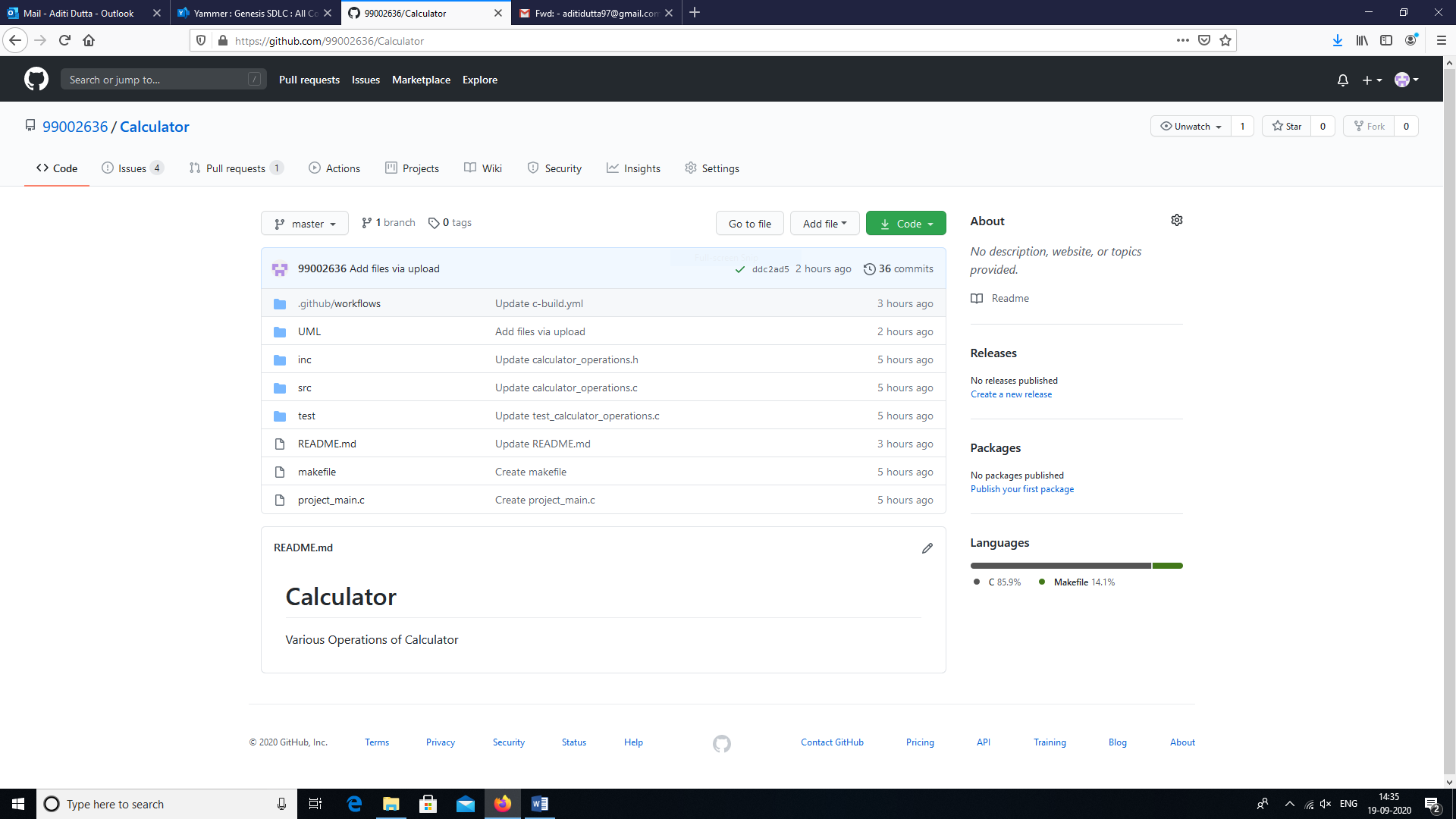
**Epic:** The application developed here is basically a calculator which performs the mathematical operations like addition, subtraction, multiplication, division and power function. The user has to input two operands and choose a valid operation to be performed. Then, the result will be displayed. If an invalid choice is entered, it will exit from the menu.

**User Stories:**

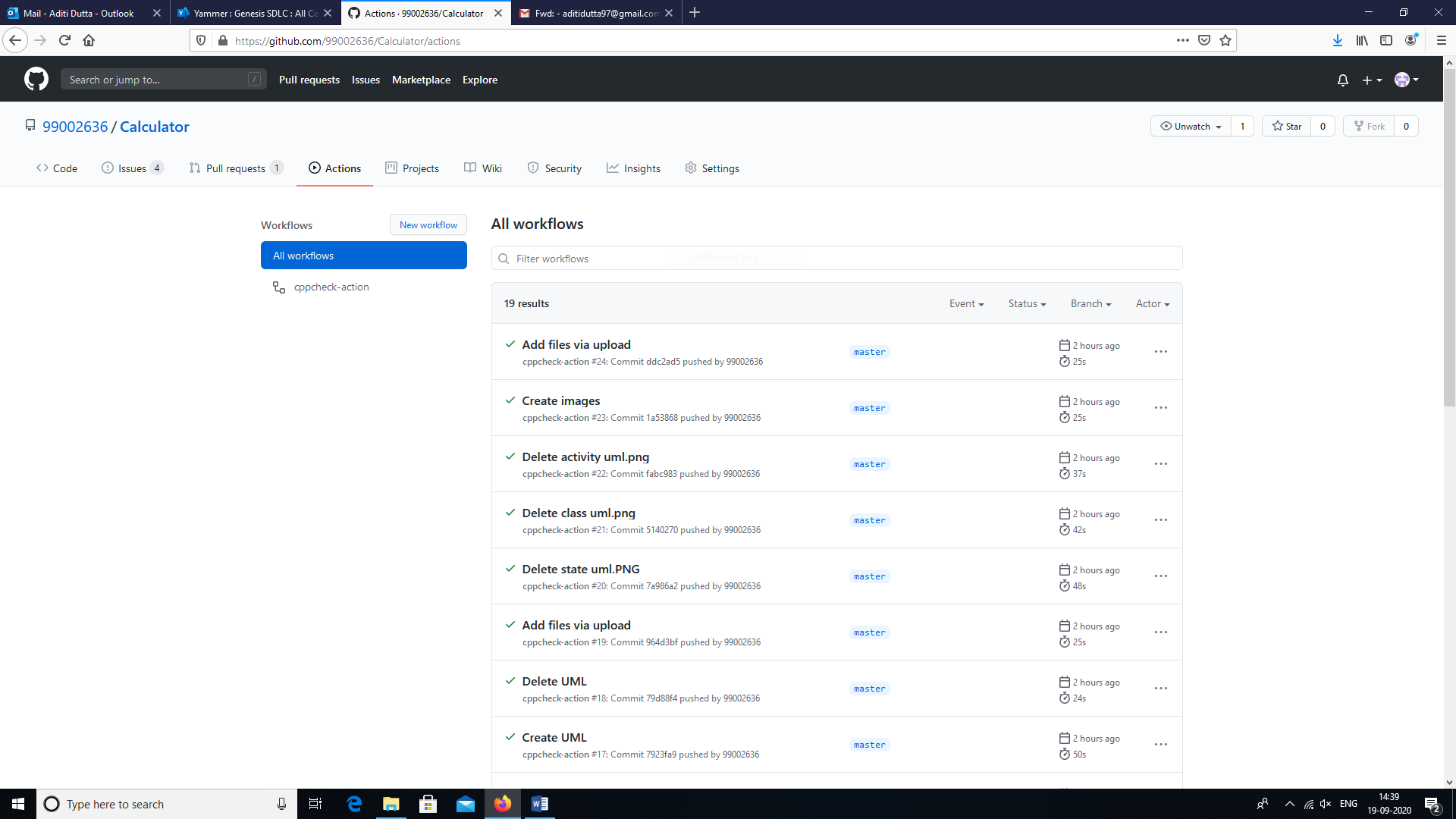
* While the user gives invalid input, there should be a strong message prompted on the screen telling what is invalid.
* There was a discrepancy in the result for subtraction when a large value was given as input.
* The program needs to be developed more so that more operations can be implemented further in it.

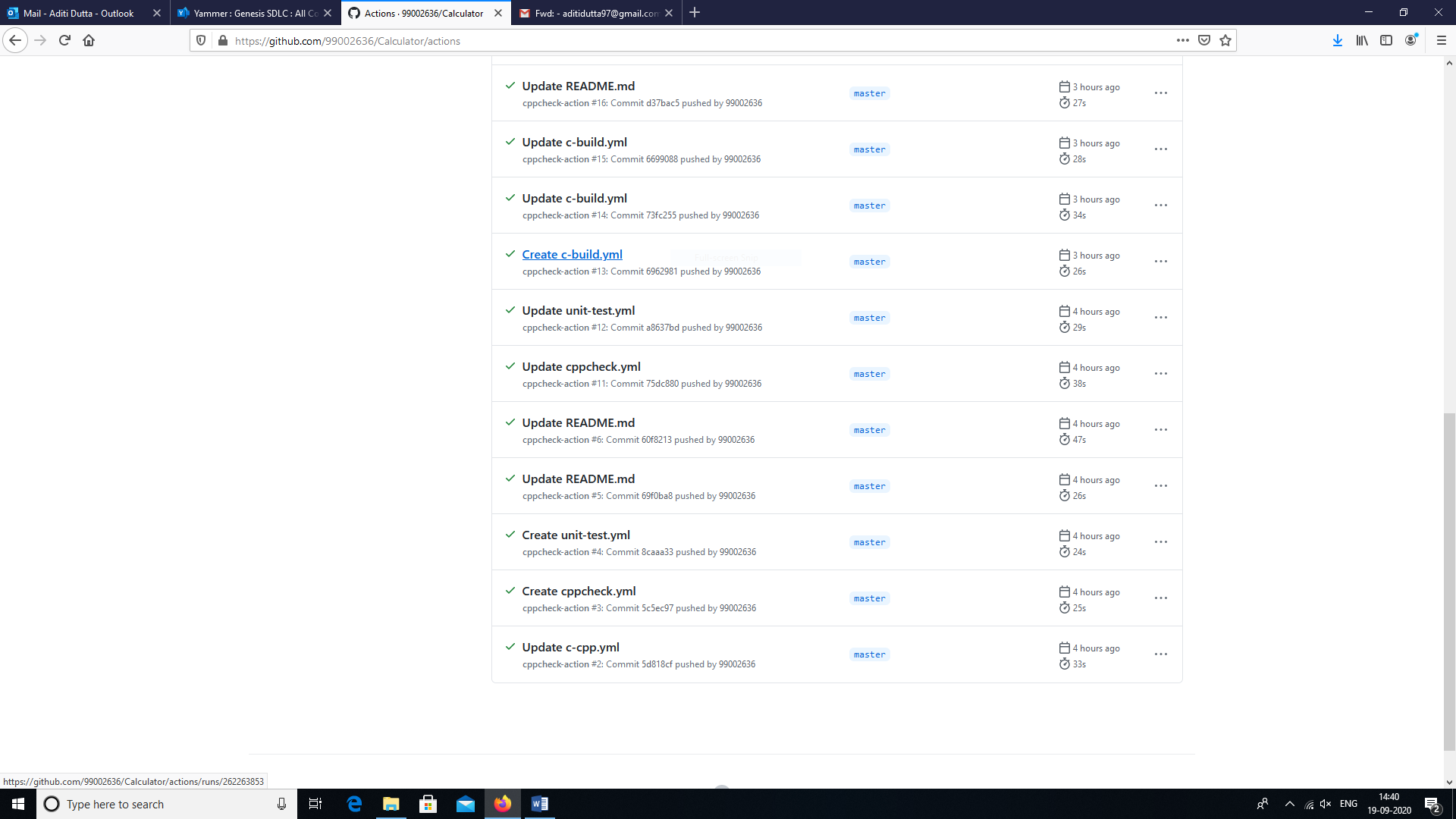
## **Activity 2 –CI Workflow for C Programming**

Makefile

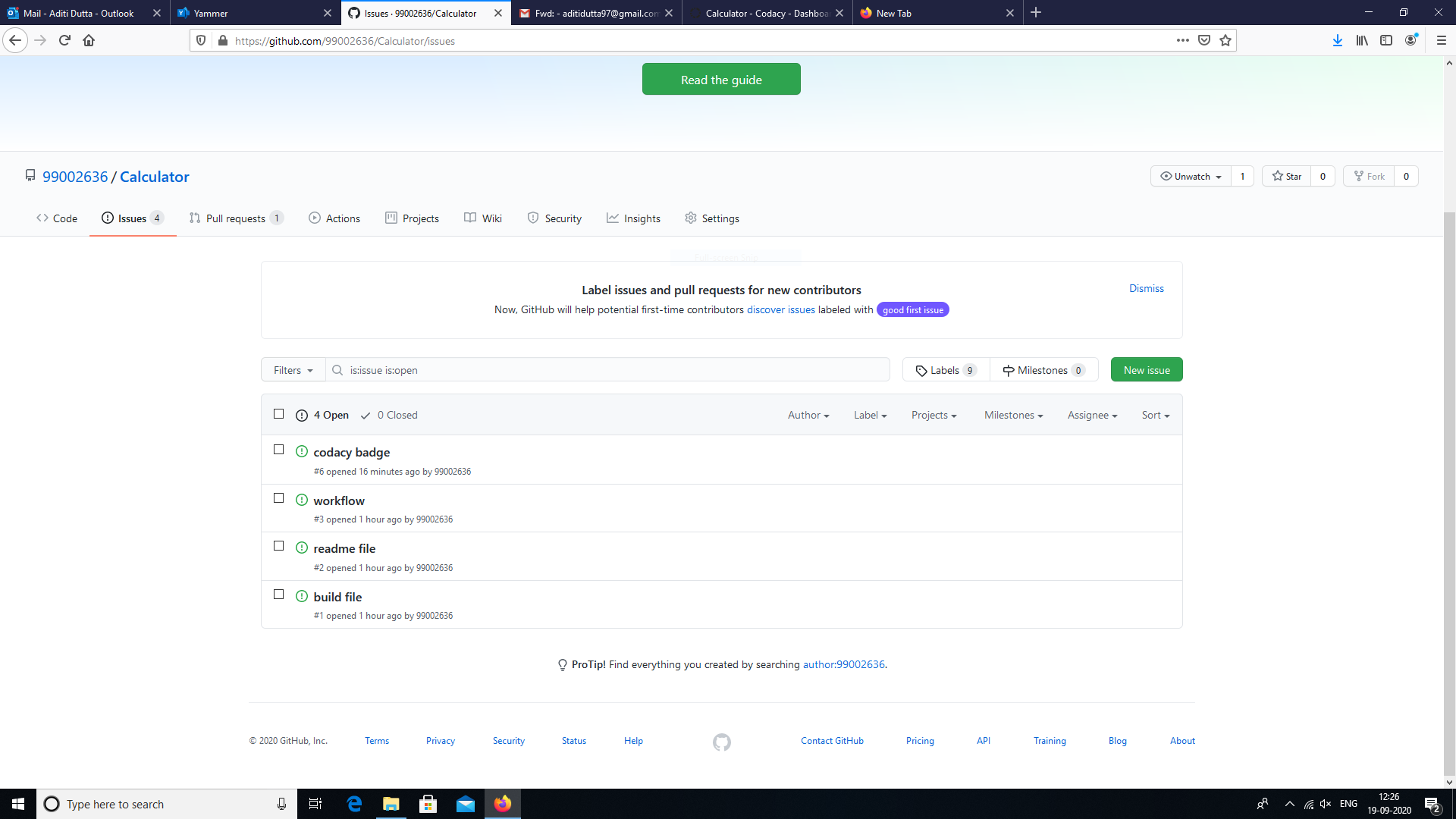


Build

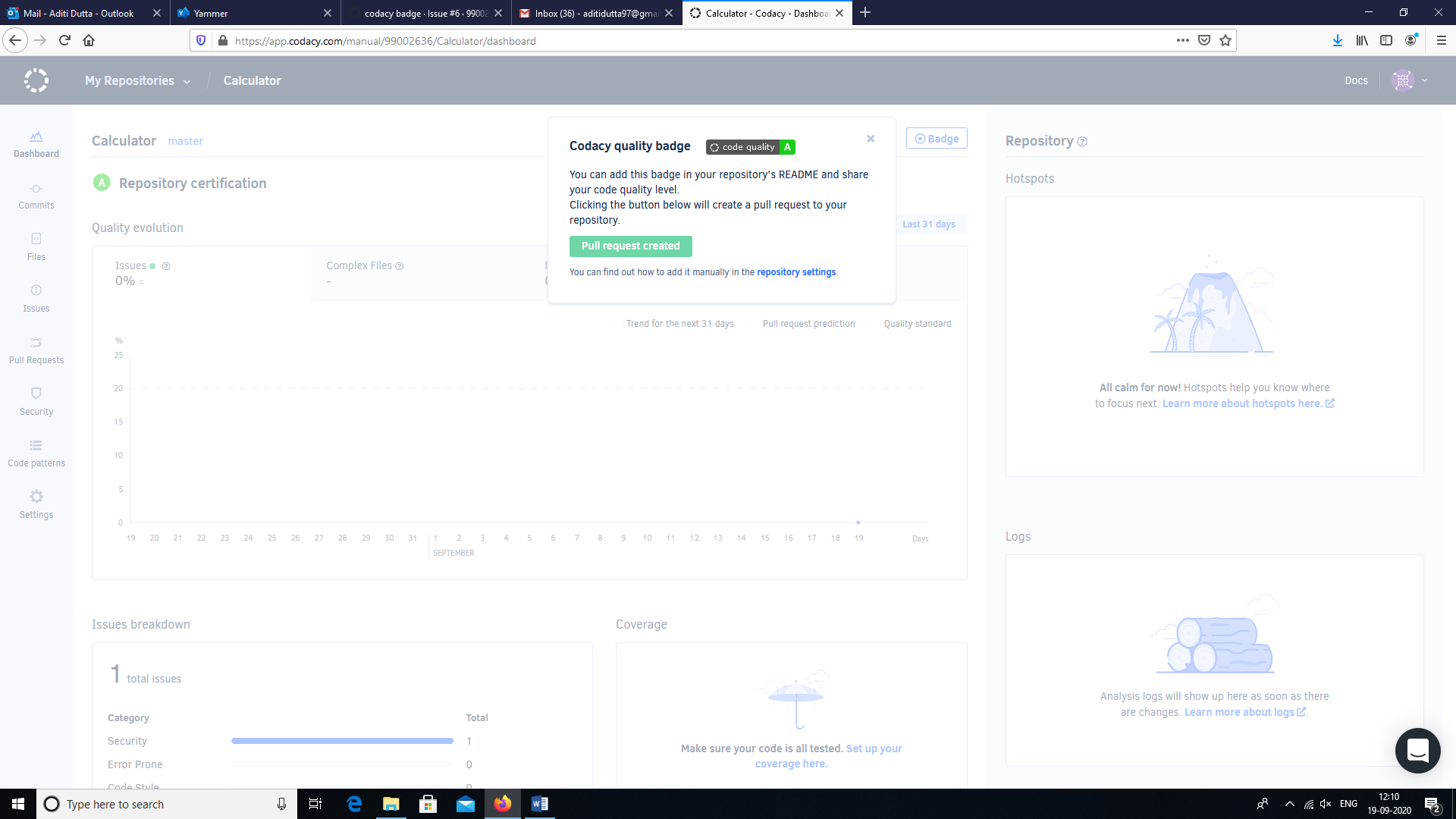


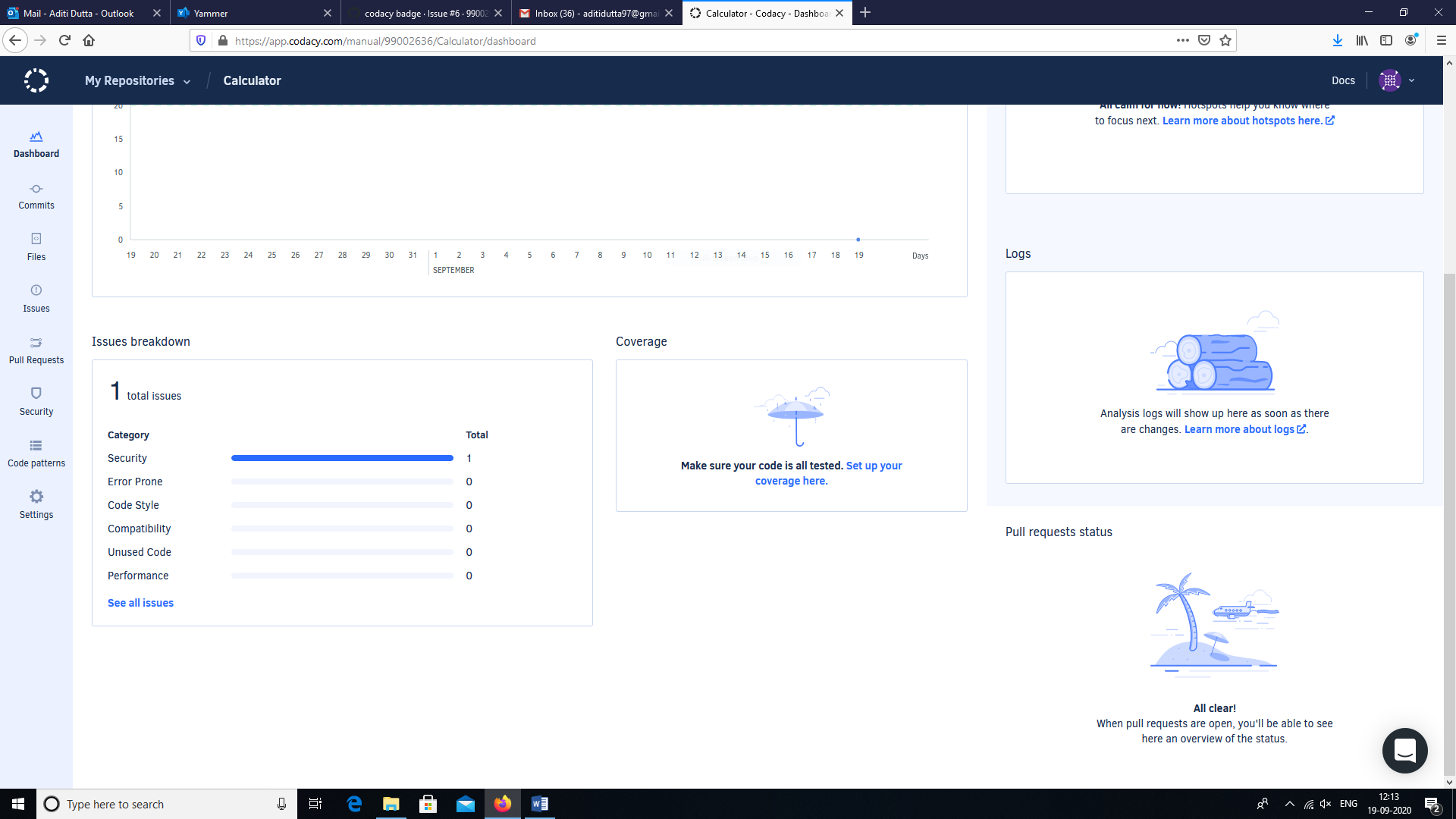


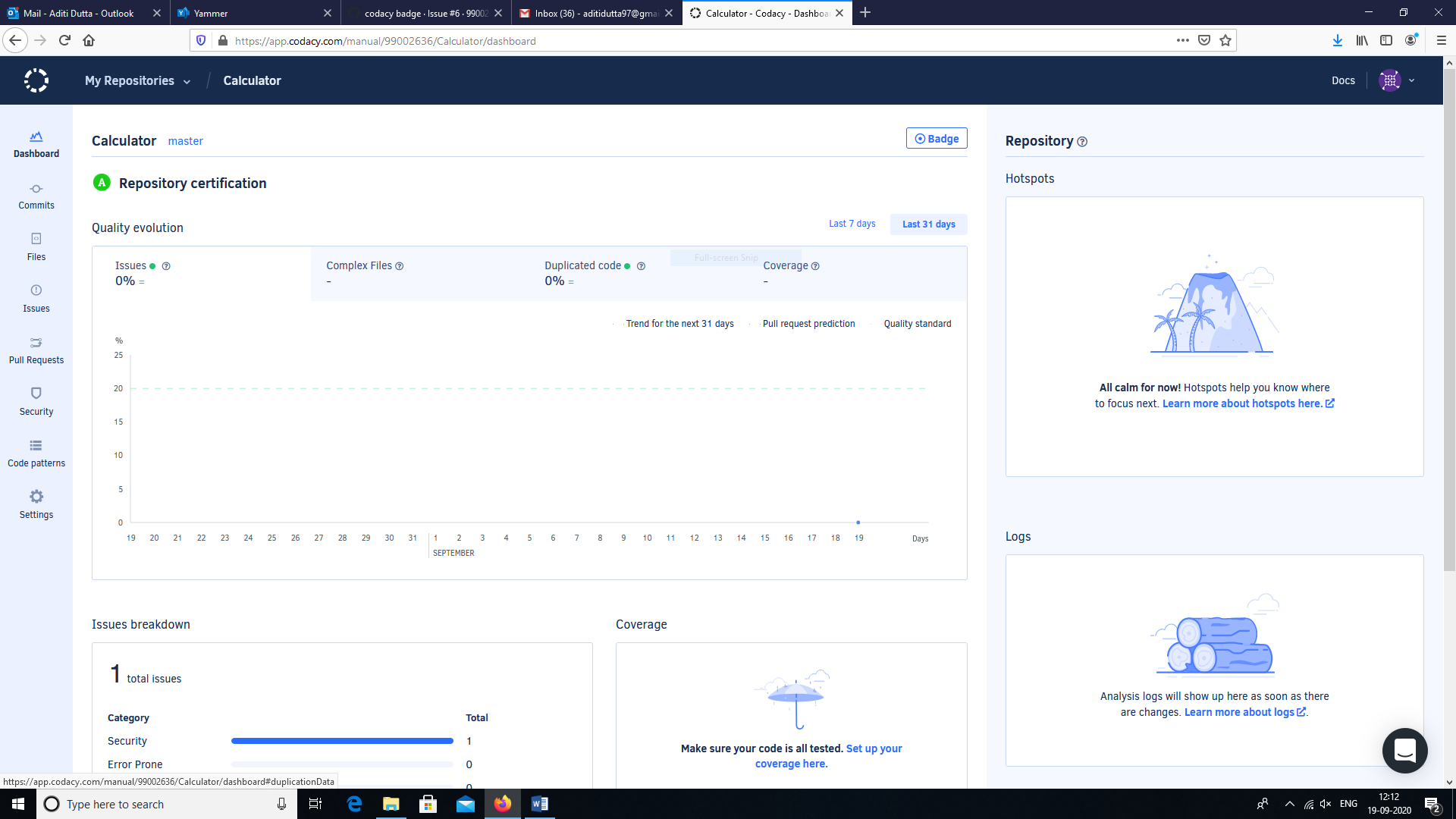
Issues



Code Quality







# Appendix

|  |
| --- |
| #include <calculator\_operations.h> |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| /\* Status of the operation requested \*/ |
|  |

|  |
| --- |
| #define VALID (1) |
|  |

|  |
| --- |
| #define INVALID (0) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| /\* Calculator operation requested by user\*/ |
|  |

|  |
| --- |
| unsigned int calculator\_operation = 0; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| /\* Operands on which calculation is performed \*/ |
|  |

|  |
| --- |
| int calculator\_operand1 = 0; |
|  |

|  |
| --- |
| int calculator\_operand2 = 0; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| /\* Valid operations \*/ |
|  |

|  |
| --- |
| enum operations{ ADD=1, SUBTRACT, MULTIPLY, DIVIDE, EXIT }; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| /\* Display the menu of operations supported \*/ |
|  |

|  |
| --- |
| void calculator\_menu(void); |
|  |

|  |
| --- |
| /\* Verifies the requested operations validity \*/ |
|  |

|  |
| --- |
| int valid\_operation(int operation); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| /\* Start of the application \*/ |
|  |

|  |
| --- |
| int main(int argc, char \*argv[]) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| printf("\n\*\*\*\*Calculator\*\*\*\*\n"); |
|  |

|  |
| --- |
| while(1) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| calculator\_menu(); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| void calculator\_menu(void) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| printf("\nAvailable Operations\n"); |
|  |

|  |
| --- |
| printf("\n1. Add\n2. Subtract\n3. Multiply\n4. Divide\n5. Exit"); |
|  |

|  |
| --- |
| printf("\n\tEnter your choice\n"); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| \_\_fpurge(stdin); |
|  |

|  |
| --- |
| scanf("%d", &calculator\_operation); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| if(EXIT == calculator\_operation) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| printf("\nThank you. Exiting the Application\n"); |
|  |

|  |
| --- |
| exit(0); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| if(INVALID != valid\_operation(calculator\_operation)) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| printf("\n\tEnter your Numbers with space between them\n"); |
|  |

|  |
| --- |
| \_\_fpurge(stdin); |
|  |

|  |
| --- |
| scanf("%d %d", &calculator\_operand1, &calculator\_operand2); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| else |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| printf("\n\t---Wrong choice---\nEnter to continue\n"); |
|  |

|  |
| --- |
| \_\_fpurge(stdin); |
|  |

|  |
| --- |
| getchar(); |
|  |

|  |
| --- |
| return; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| switch(calculator\_operation) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| case ADD: |
|  |

|  |
| --- |
| printf("\n\t%d + %d = %d\nEnter to continue", |
|  |

|  |
| --- |
| calculator\_operand1, |
|  |

|  |
| --- |
| calculator\_operand2, |
|  |

|  |
| --- |
| add(calculator\_operand1, calculator\_operand2)); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| \_\_fpurge(stdin); |
|  |

|  |
| --- |
| getchar(); |
|  |

|  |
| --- |
| break; |
|  |

|  |
| --- |
| case SUBTRACT: |
|  |

|  |
| --- |
| printf("\n\t%d - %d = %d\nEnter to continue", |
|  |

|  |
| --- |
| calculator\_operand1, |
|  |

|  |
| --- |
| calculator\_operand2, |
|  |

|  |
| --- |
| subtract(calculator\_operand1, calculator\_operand2)); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| \_\_fpurge(stdin); |
|  |

|  |
| --- |
| getchar(); |
|  |

|  |
| --- |
| break; |
|  |

|  |
| --- |
| case MULTIPLY: |
|  |

|  |
| --- |
| printf("\n\t%d \* %d = %d\nEnter to continue", |
|  |

|  |
| --- |
| calculator\_operand1, |
|  |

|  |
| --- |
| calculator\_operand2, |
|  |

|  |
| --- |
| multiply(calculator\_operand1, calculator\_operand2)); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| \_\_fpurge(stdin); |
|  |

|  |
| --- |
| getchar(); |
|  |

|  |
| --- |
| break; |
|  |

|  |
| --- |
| case DIVIDE: |
|  |

|  |
| --- |
| printf("\n\t%d / %d = %d\nEnter to continue", |
|  |

|  |
| --- |
| calculator\_operand1, |
|  |

|  |
| --- |
| calculator\_operand2, |
|  |

|  |
| --- |
| divide(calculator\_operand1, calculator\_operand2)); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| \_\_fpurge(stdin); |
|  |

|  |
| --- |
| getchar(); |
|  |

|  |
| --- |
| break; |
|  |

|  |
| --- |
| case 5: |
|  |

|  |
| --- |
| exit(0); |
|  |

|  |
| --- |
| break; |
|  |

|  |
| --- |
| default: |
|  |

|  |
| --- |
| printf("\n\t---It should never come here---\n"); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| int valid\_operation(int operation) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| /\* Check if the operation is a valid operation \*/ |
|  |

return ((ADD <= operation) && (EXIT >= operation)) ? VALID: INVALID;

<https://github.com/99002636/Calculator>

<https://github.com/STEPin105009/MiniProject_Template>

# References

* <https://en.wikipedia.org/wiki/Solar_cell>
* <https://www.greenmatch.co.uk/blog/2015/06/advantages-and-disadvantages-of-solar-cells>
* <https://www.greenmatch.co.uk/blog/2014/08/5-advantages-and-5-disadvantages-of-solar-energy>
* <https://www.quora.com/What-are-advantages-and-disadvantages-of-solar-cells>
* [weakness of a calculator - Google Search](https://www.google.com/search?client=firefox-b-d&q=weakness+of+a+calculator)
* [calculator cost degradation - Google Search](https://www.google.com/search?client=firefox-b-d&sxsrf=ALeKk039ERAIgbWOhMxP4gmho07m8FMobw%3A1600500259069&ei=IrJlX5biOsWNoASu05foDA&q=calculator+cost+degradation&oq=calculator+cost+degra&gs_lcp=CgZwc3ktYWIQARgAMgUIIRCgAToECAAQRzoICAAQsQMQkQI6CwgAELEDEIMBEJECOgIIADoHCAAQFBCHAjoGCAAQFhAeOggIIRAWEB0QHjoICAAQCBANEB5Q1LwwWIbpMGCU-zBoAHACeACAAaICiAHVE5IBBDItMTCYAQCgAQGqAQdnd3Mtd2l6yAEIwAEB&sclient=psy-ab)
* [decrease in price of calculator over time - Google Search](https://www.google.com/search?client=firefox-b-d&sxsrf=ALeKk021_WQ6KoZjxK8saIL9GRNV5HM0IA:1600501159030&q=decrease+in+price+of+calculator+over+time&spell=1&sa=X&ved=2ahUKEwiLoa-42_TrAhWvzIsBHQLJDi0QBSgAegQIDRAo&biw=1760&bih=886)