./

Learning Report –

SDLC and Testing



NAME : AMIT DAS

PS NO. : 99002591

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **Approved By** | **Remarks/Revision Details** |
| 1. | 17/09/2020 | Amit Das |  |  | V model-HL&LL requirements, UML diagrams, GIT documentation  (TASK 1 and 2) |
| 2. | 18/09/2020 | Amit Das |  |  | Added agile model, Re-mapping requirements and tests into proper formats. |
| 3. | 19/09/2020 | Amit Das |  |  | Moved assignment 3 to appendix, Documented assignment 3, restructured table of contents, updates UML diagrams |

**Document History**

# 

Contents

[Checklist 4](#_Toc51410994)

[Activity and Tasks 4](#_Toc51410995)

[Activity 1– System/Software Development 4](#_Toc51410996)

- Requirements

- System Design and UML diagrams

- Testing

[Activity 2 –AGILE ASPECTS 12](#_Toc51410997)

- Agile Model

[Activity 3 – CI/CD PROJECT 13](#_Toc51410998)

-Requirements

-Testing

-Design and UML diagrams

-Github repository documentation

- Folder structure and badges

- Issues

- Code Quality

- Build logs (C/C++ CI, cppcheck)

-Commit History

**APPENDIX……………………………………………………………………………………………………………… 23**

# Checklist

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

# Activity and Tasks

## **Activity 1**– System Development

* Sub Tasks
* Complete and Evolve

**TASK: REMAP REQUIREMENTS OF PREVIOUS TASK INTO STANDARD TEMPLATE.**

1. REQUIREMENTS:

|  |  |
| --- | --- |
| **ID** | **Description** |
| H\_01 | The device should be able to detect rapid changes in ambient humidity in a closed environment. |
| H\_02 | The sensing circuit has to run all the time. |
| H\_03 | The device should be able to dehumidify a small room with rapid efficacy. |
| H\_04 | The device should be repair friendly. |
| H\_01\_L\_01 | Sensor used should sensitive to detect a 5% change in humidity every second. |
| H\_01\_L\_02 | The sensor sampling frequency should be 1Hz or higher |
| H\_02\_L\_01 | The idling current of the microprocessor should be less than 50uA. |
| H\_02\_L\_02 | The sensor should be a passive transducer type. |
| H\_02\_L\_03 | The active current draw of the system should be less than 100mA. |
| H\_03\_L\_01 | The intake fan should facilitate a filtration rate of a minimum of 50L/min. |
| H\_03\_L\_02 | Air flow vents should have detachable magnetic filters for easy cleaning. |
| H\_04\_L\_01 | The socket and plug used for the device should be standard to the country. |
| H\_04\_L\_02 | The sensing circuit, fan array and dehumidifier array should be independently repairable and modular. |

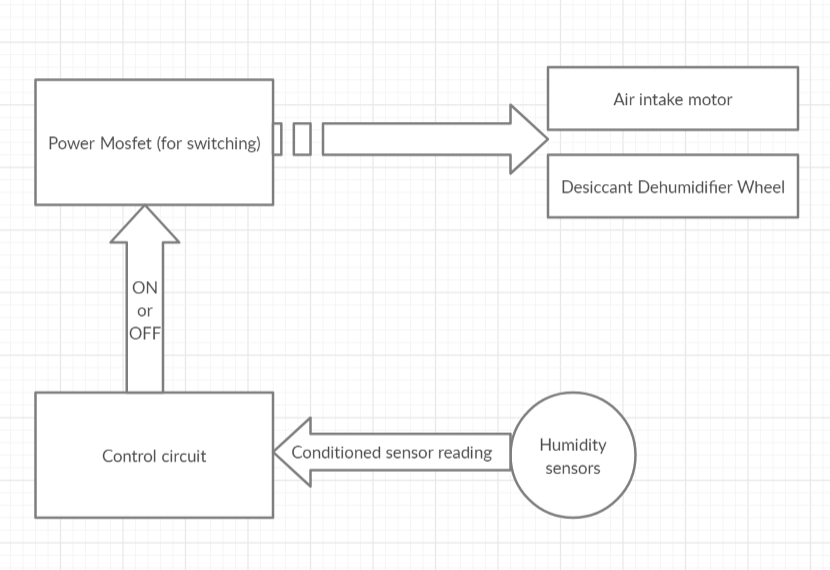
1. TESTING

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Description | Precondition | Expected I/P or state | Expected O/P or state | Actual O/P |
| H\_01\_L\_02\_T\_01 | Check the sensor reading sent to microcontroller | Signal conditioning is active | Device stored in an environment controlled room. | Readings ranging from 0-255 at 1Hz. Gradually readings become stable. |  |
| H\_01\_L\_01\_T\_01 | Check if sensor can detect rapid changes | Signal conditioning is active | The humidity of the room is varied at a more than normal rapid pace. | Readings spike and plummet and do not settle down. Sensor doesn’t reset. |  |
| H\_02\_L\_01\_T\_01 | Check idle power of microprocessor | External ammeter is setup in series with microcontroller input line. | The circuit is powered on and left to steady, sensor is disconnected. | Current draw should be less than 50uA. |  |
| H\_02\_L\_02\_T\_01 | Check if sensor is of passive transducer type | The standalone sensor is connected to a multi-meter | The sensor is placed in a high humidity environment. | The voltage generated across the transducer should spike and become stable eventually. |  |
| H\_02\_L\_03\_T\_01 | Check the current draw of the entire circuit. | The sensing circuit is assembled and ammeter is connected in series | The sensing circuit is kept in a control environment and humidity is varied | The current draw shouldn’t exceed 100mA even in spikes |  |
| H\_03\_L\_01\_T\_01 | Intake fan capacity test | The fan is tested inside the device case | The fan is turned on and the airflow is measured at the case vents | The airflow should be 50L/min or higher |  |
| H\_04\_L\_02\_T\_01 | The entire device is tested on a test bench setup | The device is assembled on a test bench | Physically disabling the sensor from working | The device should function as normal with a simple plug and play replacement of the sensor. |  |

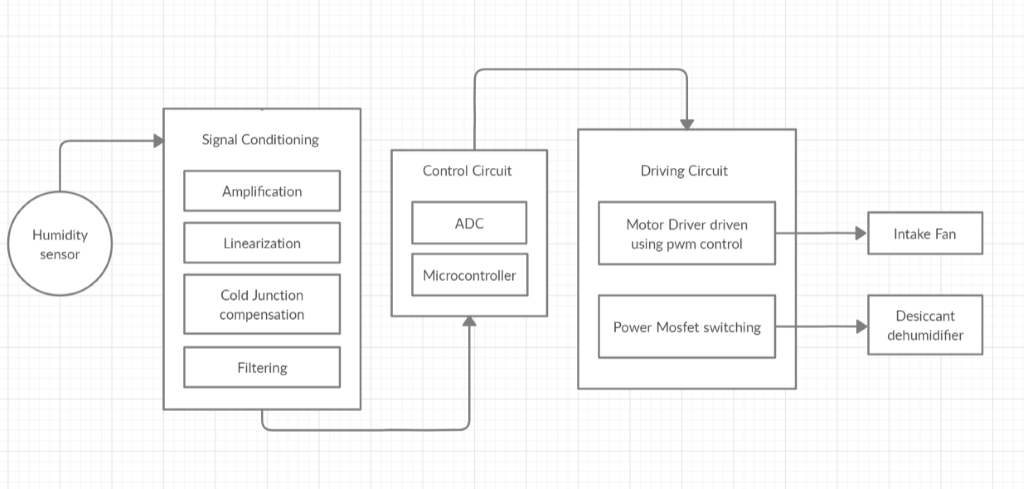
1. DESIGN

HIGH LEVEL DESIGN:

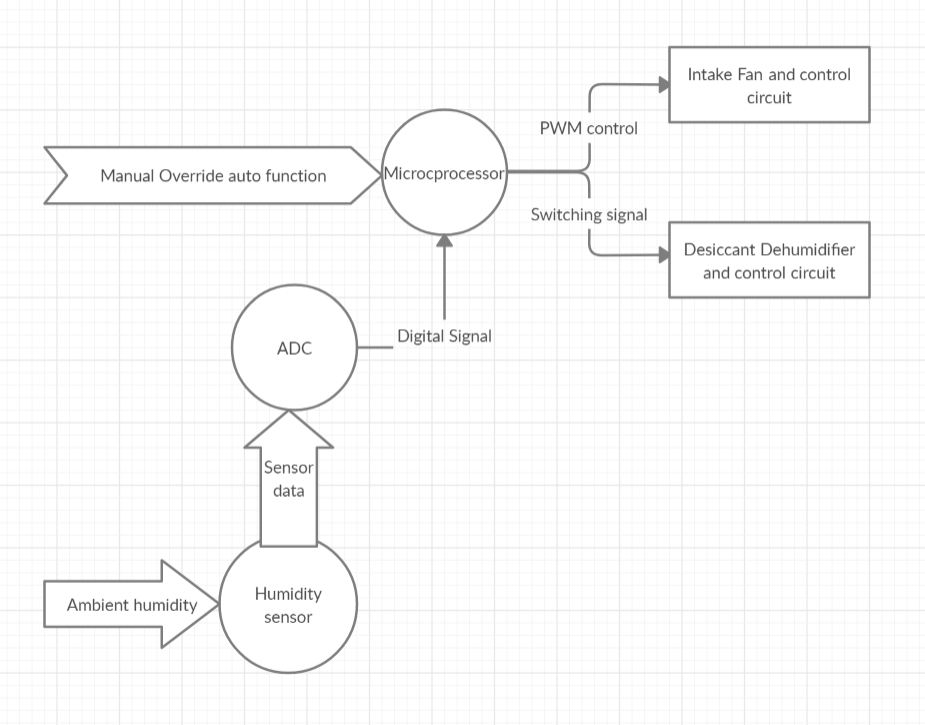
1. Package Diagram (Dependencies between various systems, Structural)



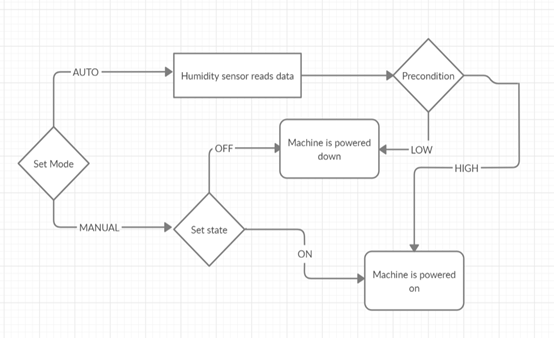
1. Deployment Diagram (Relation of hardware with software in each hardware block, Structural)



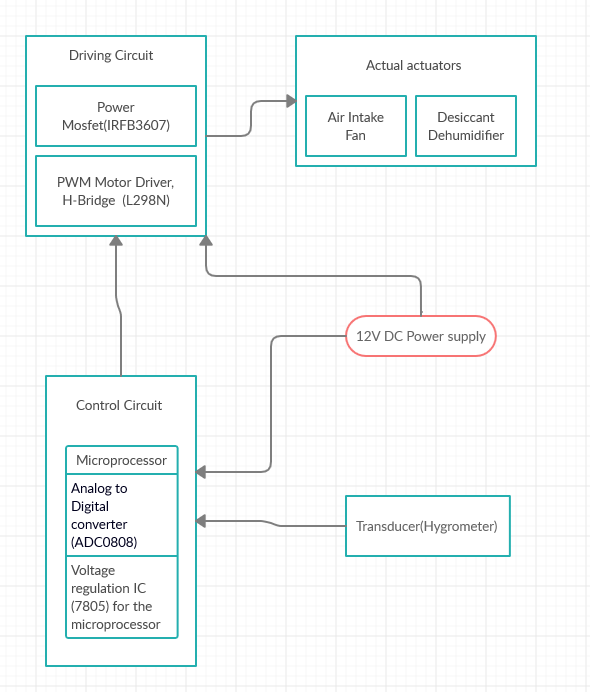
1. Use Case diagram (Main actors and processes involved, Behavioral)

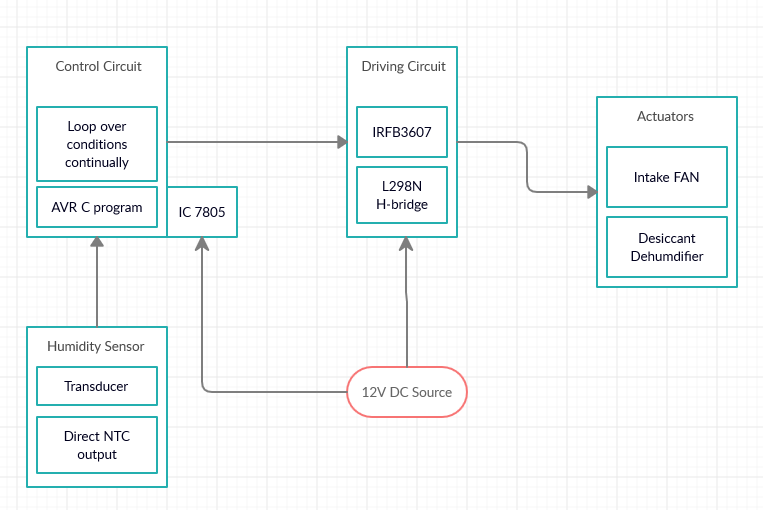


1. Activity Diagram (Workflow, Behavioral)

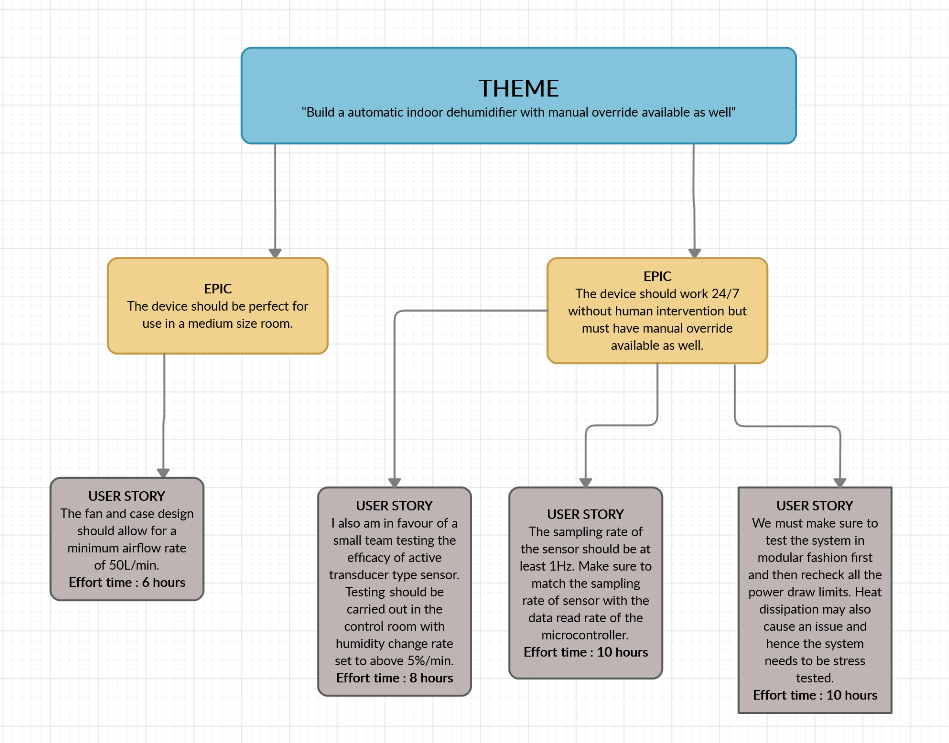


LOW LEVEL DESIGN:

* + - 1. Object Diagram (Real world examples, Structural)
      2. Composite Structures Diagram (Internal structure of a class, Structure)



## **Activity 2** –AGILE ASPECTS



## **Activity 3** – CI/CD PROJECT

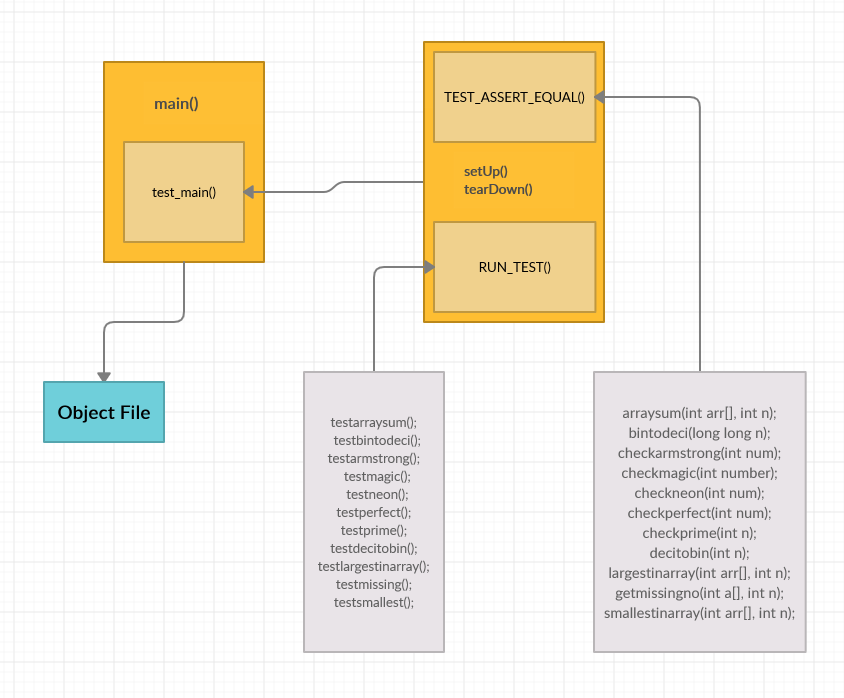
1. REQUIREMENTS

|  |  |
| --- | --- |
| **ID** | **DESCRIPTION** |
| H\_01 | Develop an application to test generic C programming questions. |
| H\_02 | The application should be a multi-file code. |
| H\_03 | The code should be cppcheck complaint. |
| H\_04 | Code quality should be maintained and no unused variable left. |
| H\_05 | Raise and resolve issues on github. |
| H\_01\_L\_01 | The programs chosen for the same ‘sum of numbers in array’, ‘binary to decimal’, ‘armstrong number’, ‘magic number’, ‘neon number’, ‘perfect number’, ‘prime number’, ‘decimal to binary’, ‘largest number in array’, ‘missing number in array’, ‘smallest number in array’ |
| H\_02\_L\_01 | All the functions are stored under ‘src’ and user defined header files are stored under ‘inc’ directory. |
| H\_02\_L\_02 | Make is used to link and compile everything. |
| H\_03\_L\_01 | cppcheck.yml was configured as a workflow |
| H\_04\_L\_01 | Codeacy code quality check was also configured as a workflow |
| H\_04\_L\_02 | Unity throw the switch was used for unit testing |
| H\_05\_L\_03 | TEST\_ASSERT\_EQUAL\_MESSAGE was used to test the accuracy of the code. |

1. TESTING

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Description | Precondition | Expected I/P | Expected O/P | Actual O/P |
| H\_02\_L\_02\_T\_01 | The file should build on the server. | Code has no syntax errors, all files are linked | make | Compiled | Compiled |
| H\_01\_L\_01\_T\_02 | Array sum test, add numbers in 3 element array with positive nos. | Code compiles successfully | [1,2,3], 3 | 6 | 6 |
| H\_01\_L\_01\_T\_03 | Array sum test, add both positive and negative numbers in array | Code compiles successfully | [2,3,6,-10], 4 | 1 | 1 |
| H\_01\_L\_01\_T\_04 | Binary to decimal test, test any number staring with 1 | Code compiles successfully | 110110111 | 439 | 439 |
| H\_01\_L\_01\_T\_05 | Binary to deci test, test any number starting with 0 | Code compiles successfully | 00000110 | 134 | 16  FAIL |
| H\_01\_L\_01\_T\_06 | Armstrong number, check for true and positive | Code compiles successfully | 153 | 1 | 1 |
| H\_01\_L\_01\_T\_07 | Armstrong number, check for false and positive | Code compiles successfully | 200 | 0 | 0 |
| H\_01\_L\_01\_T\_08 | Armstrong number, check for 0 | Code compiles successfully | 0 | 1 | 1 |
| H\_01\_L\_01\_T\_09 | Armstrong number, check for negative number | Code compiles successfully | -21 | -1 | -1 |
| H\_01\_L\_01\_T\_10 | Check for magic number, check for true | Code compiles successfully | 52 | 1 | 0  FAIL |
| H\_01\_L\_01\_T\_11 | Check for magic number, check for false | Code compiles successfully | 12345 | 0 | 0 |
| H\_01\_L\_01\_T\_12 | Check for neon number, check for true | Code compiles successfully | 111 | 0 | 0 |
| H\_01\_L\_01\_T\_13 | Check for neon number, check for false | Code compiles successfully | 9 | 1 | 1 |
| H\_01\_L\_01\_T\_14 | Check for neon number, check for 0 | Code compiles successfully | 0 | 1 | 1 |
| H\_01\_L\_01\_T\_15 | Check for neon number, check for negative no. | Code compiles successfully | -9 | -1 | -1 |
| H\_01\_L\_01\_T\_16 | Check for perfect number, check for true. | Code compiles successfully | 496 | 1 | 1 |
| H\_01\_L\_01\_T\_17 | Check for perfect number, check for false | Code compiles successfully | 400 | 0 | 0 |
| H\_01\_L\_01\_T\_18 | Check for prime no., check for true | Code compiles successfully | 23 | 1 | 1 |
| H\_01\_L\_01\_T\_19 | Check for prime no., check for false | Code compiles successfully | 400 | 0 | 0 |
| H\_01\_L\_01\_T\_20 | Check for prime no., check for negative | Code compiles successfully | -1 | -1 | -1 |
| H\_01\_L\_01\_T\_21 | Check for prime no., check for 0 | Code compiles successfully | 0 | -1 | -1 |
| H\_01\_L\_01\_T\_22 | Check for prime no., check for 1 | Code compiles successfully | 1 | 3 | 3 |
| H\_01\_L\_01\_T\_23 | Check for decimal to binary, check for positive number | Code compiles successfully | 17 | 10001 |  |
| H\_01\_L\_01\_T\_24 | Check for decimal to binary, check for 0 | Code compiles successfully | 0 | 0 | 0 |
| H\_01\_L\_01\_T\_25 | Check for decimal to binary, check for negative | Code compiles successfully | -156 | -10101 | -10101 |
| H\_01\_L\_01\_T\_26 | Check for largest in array, check in array with all positive nos. | Code compiles successfully | [1,2,3], 3 | 3 | 3 |
| H\_01\_L\_01\_T\_27 | Check for largest in array, check in array with positive and negative nos. | Code compiles successfully | [2,3,5,-10], 4 | 6 | 6 |
| H\_01\_L\_01\_T\_28 | Check for smallest in array, check in array with positive and negative nos. | Code compiles successfully | [1,2,3], 3 | 1 | 1 |
| H\_01\_L\_01\_T\_29 | Check for smallest in array, check in array with positive and negative nos. | Code compiles successfully | [2,3,6,-10], 4 | -10 | -10 |
| H\_01\_L\_01\_T\_30 | Check for smallest in array, check in array with all negative nos. | Code compiles successfully | [-1,-3,-19,-81], 4 | -81 | -81 |

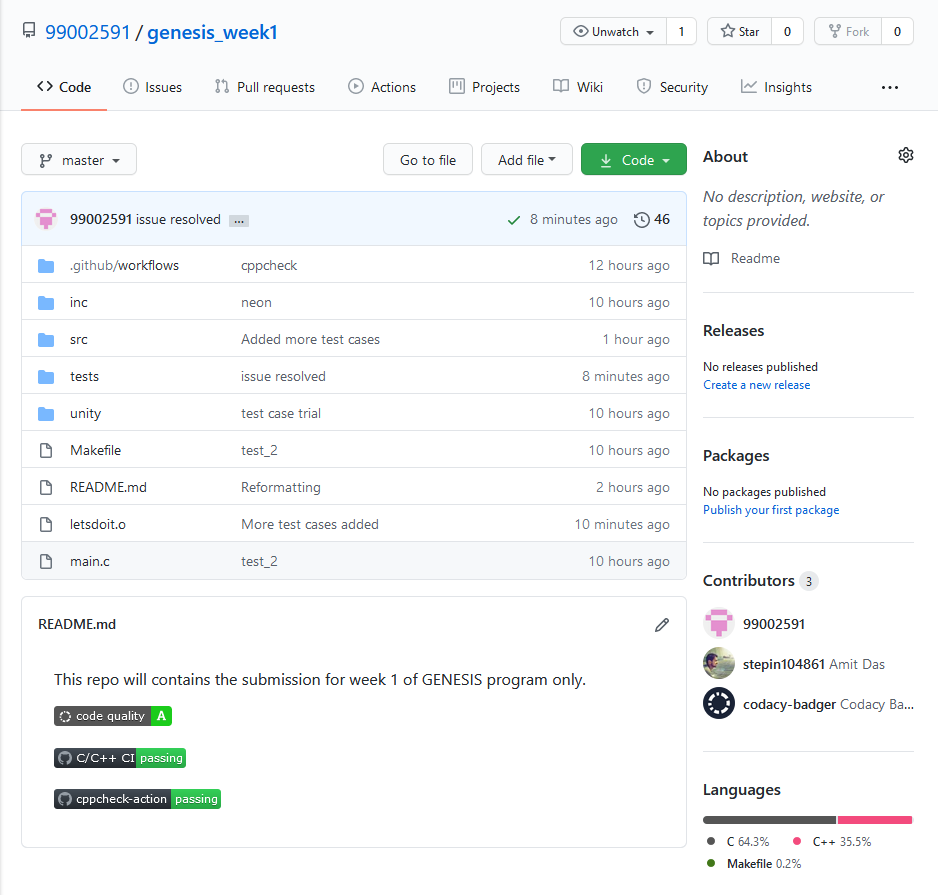
COMPOSITE STRUCTURE DIAGRAM:



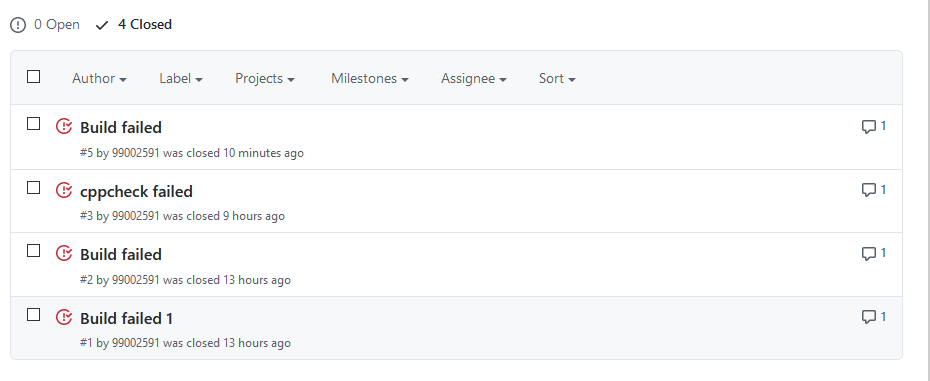
**GITHUB DOCUMENTATION FOR ASSIGNMENT 3**

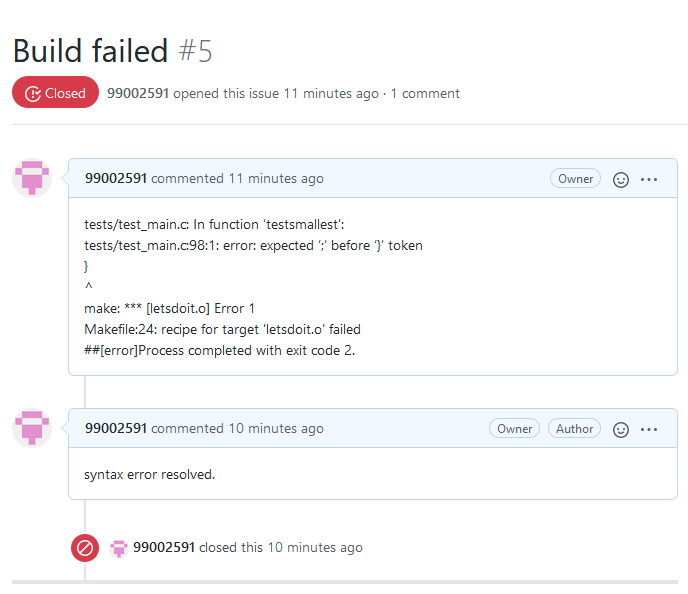
**REPO LINK:** https://github.com/99002591/genesis\_week1

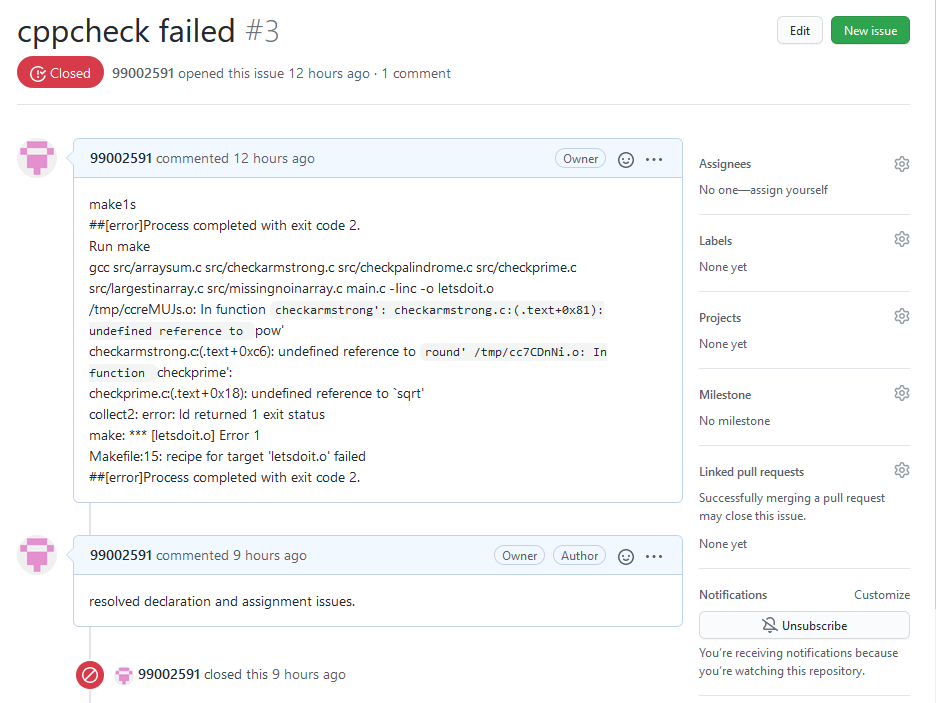
1. FOLDER STRUCTURE AND BADGES

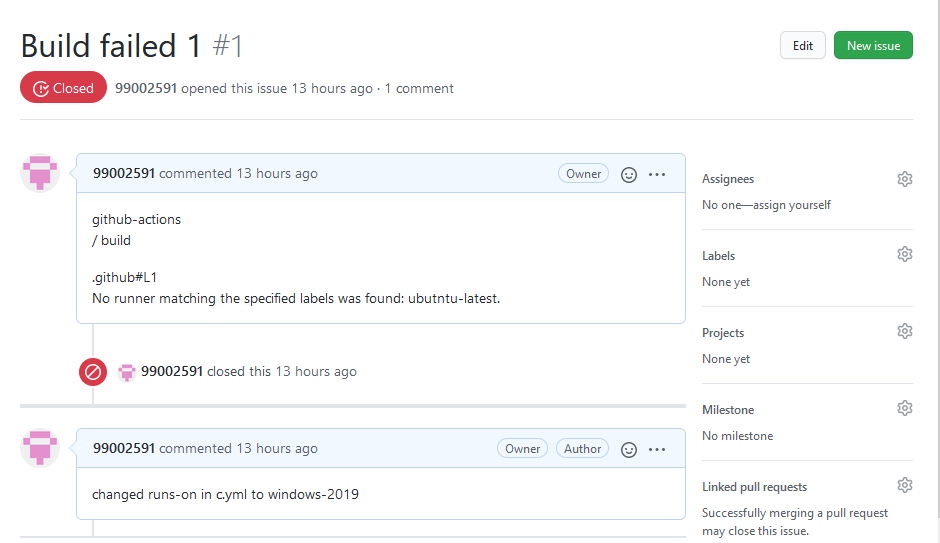


1. ISSUES

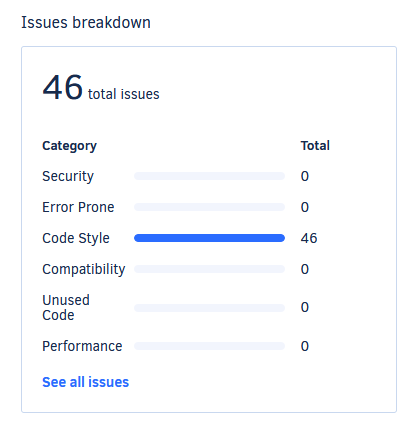


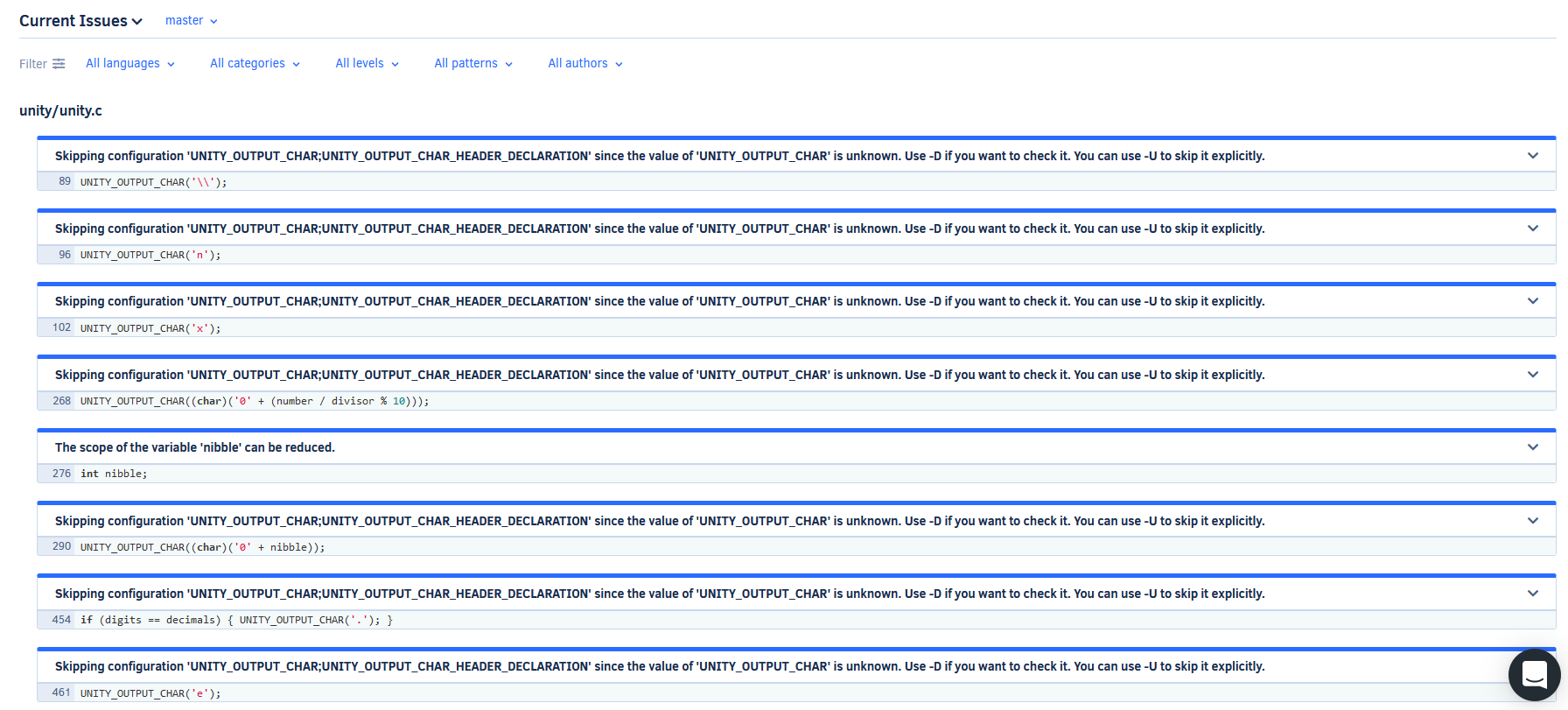






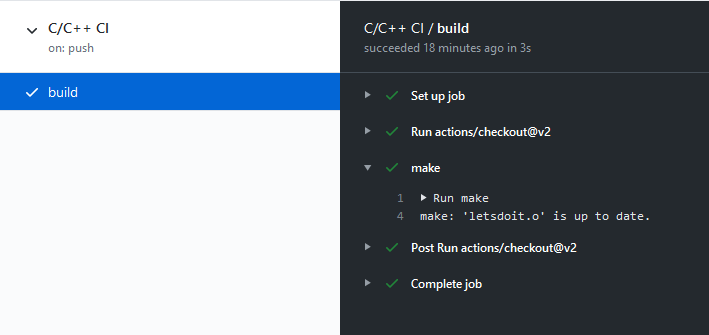
1. CODE QUALITY

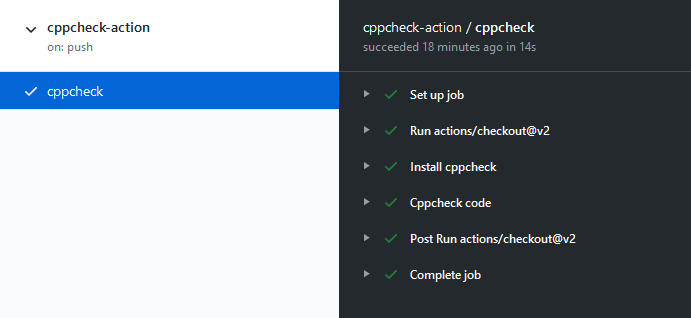


1. BUILD LOGS

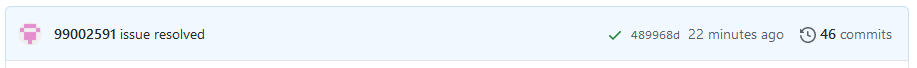
C/C++ CI

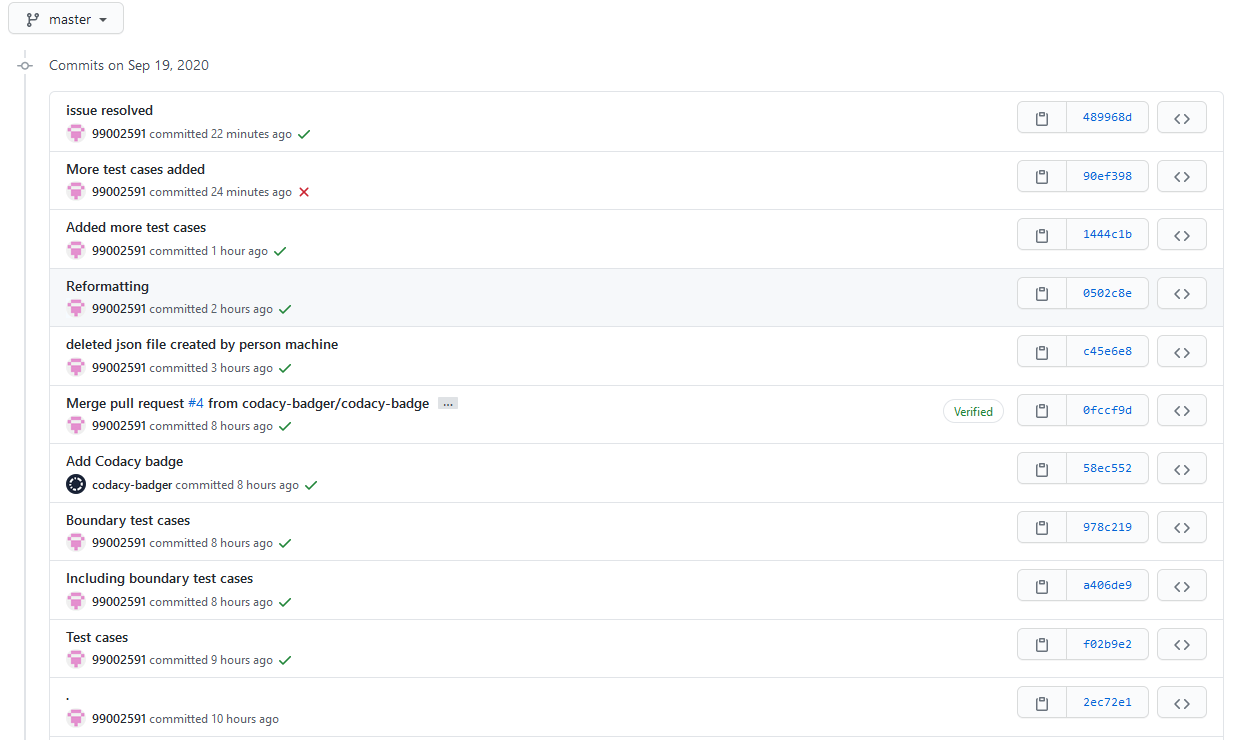


cppcheck



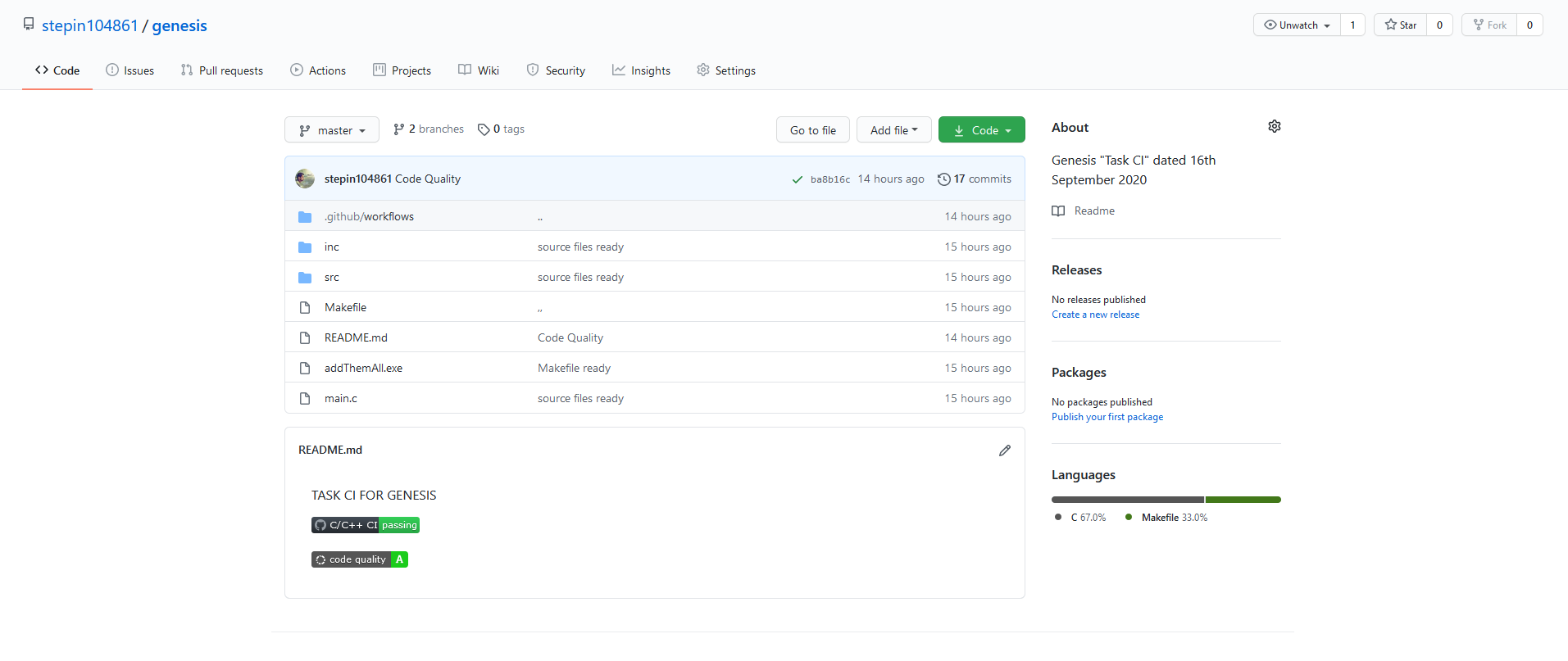
1. COMMIT HISTORY



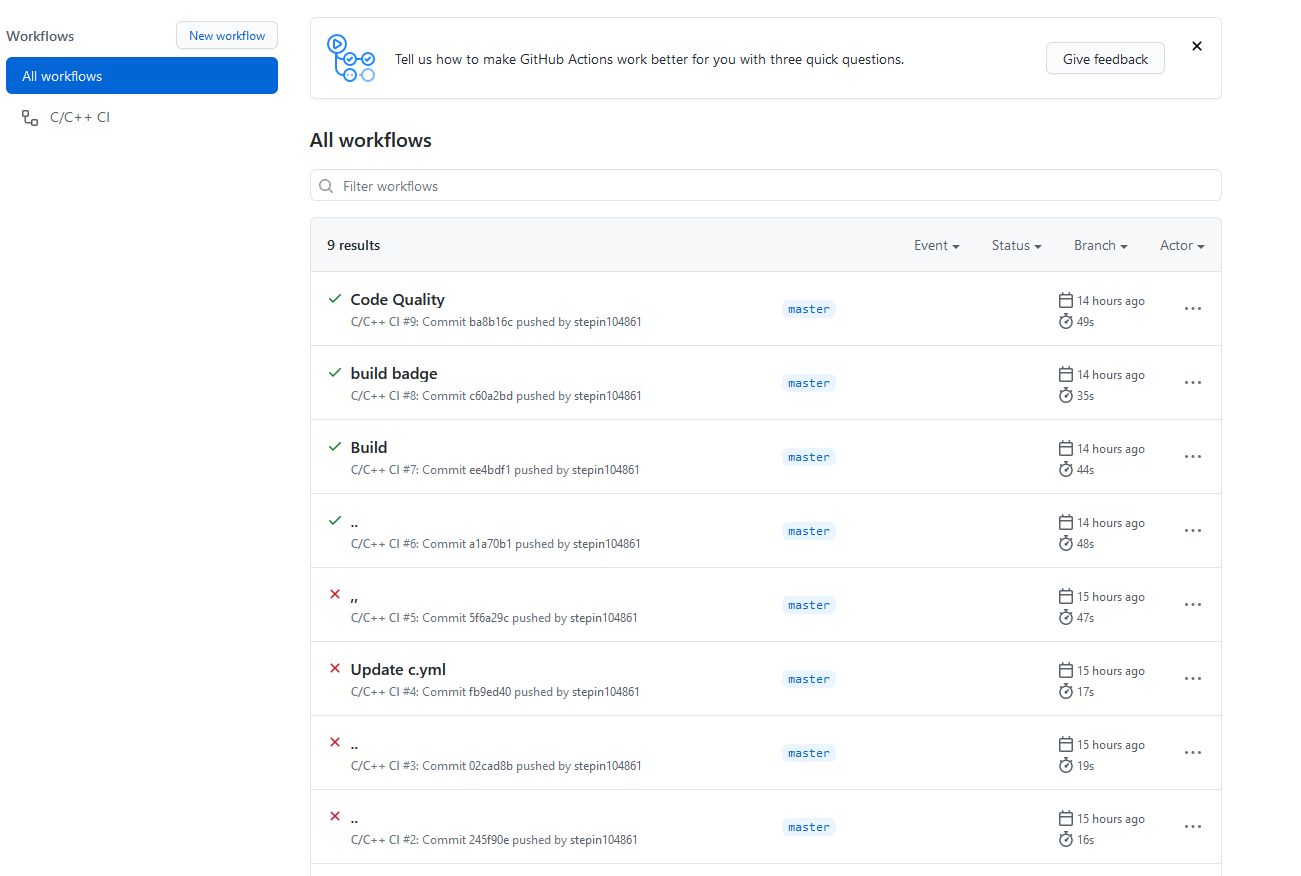


* **APPENDIX**
* GITHUB REPOSITORY LINK: https://github.com/stepin104861/genesis

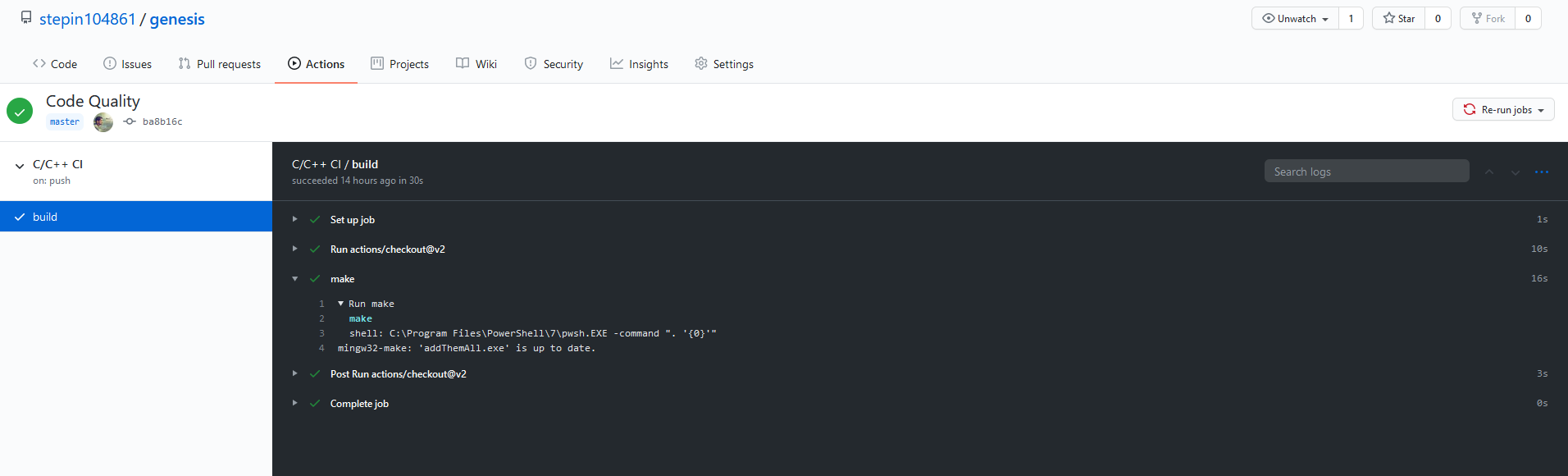
BADGES



COMMIT HISTORY



BUILD LOG



ISSUES



FILE STRUCTURE

