./



GENESIS - Learning Outcome & Mini-project Summary Report

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **To be Approved** | **Remarks/Revision Details** |
| 1 | 09.11.2020 | Maneesh S Dani | Lakshmi Narayan S |  |  |
| 2 | 10.11.2020 | Maneesh S Dani | Lakshmi Narayan S |  |  |
| 3 | 11.11.2020 | Maneesh S Dani | Lakshmi Narayan S |  |  |

**Details**

Contents

[Miniproject -1 : C++ - Hospital Management System [Individual] 5](#__RefHeading___Toc3894_419027920)

[Module/s 5](#__RefHeading___Toc3896_419027920)

[Topic and Subtopics 5](#__RefHeading___Toc3898_419027920)

[Objectives & Requirements 5](#__RefHeading___Toc3900_419027920)

[Design: 6](#__RefHeading___Toc3673_1127087468)

[Test Plan: 7](#__RefHeading___Toc3904_419027920)

[Implementation Summary 9](#__RefHeading___Toc3906_419027920)

[Video Summary 9](#__RefHeading___Toc3908_419027920)

[Git Link 9](#__RefHeading___Toc3910_419027920)

[Git Dashboard 9](#__RefHeading___Toc3912_419027920)

[Summary 14](#__RefHeading___Toc3914_419027920)

[Challenges faced and how were they overcome 14](#__RefHeading___Toc3916_419027920)

[Future Scope (If applicable) 14](#__RefHeading___Toc3918_419027920)

[Miniproject -2 : Python [Team] 15](#__RefHeading___Toc1058_2015297489)

[Module/s 15](#__RefHeading___Toc3922_4190279201)

[Topic and Subtopics 15](#__RefHeading___Toc3924_4190279201)

[Objectives & Requirements 15](#__RefHeading___Toc3926_4190279201)

[Design 16](#__RefHeading___Toc3928_4190279201)

[Test Plan 17](#__RefHeading___Toc3930_4190279201)

[Implementation Summary 17](#__RefHeading___Toc3932_4190279201)

[Git Link 18](#__RefHeading___Toc3934_4190279201)

[Video Summary 18](#__RefHeading___Toc3669_1127087468)

[Git Dashboard 18](#__RefHeading___Toc3936_4190279201)

[Individual Contribution & Highlights 23](#__RefHeading___Toc3940_4190279201)

[Summary 23](#__RefHeading___Toc1060_2015297489)

[Challenges faced and how were they overcome 23](#__RefHeading___Toc1062_2015297489)

**Table of Figures**

[Figure 1: Use case diagram 6](#Figure!0|sequence)

[Figure 2: Sequence diagram 6](#Figure!1|sequence)

[Figure 3: Component diagram 7](#Figure!2|sequence)

[Figure 4: Class Diagram 7](#Figure!3|sequence)

[Figure 5: Git dashboard with all the badges 9](#Figure!4|sequence)

[Figure 6: Git Inspector 10](#Figure!15|sequence)

[Figure 7: C/C++ Build using GitHub actions 11](#Figure!5|sequence)

[Figure 8: Codacy Code Quality 12](#Figure!6|sequence)

[Figure 9: Issue raising and closing in GitHub 12](#Figure!7|sequence)

[Figure 10: Unit testing using GitHub actions 13](#Figure!8|sequence)

[Figure 11: Unit testing using .out file 13](#Figure!9|sequence)

[Figure 12: Flow Chart 16](#Figure!10|sequence)

[Figure 13: GitHub Dashboard 18](#Figure!11|sequence)

[Figure 14: Git Inspector 19](#Figure!16|sequence)

[Figure 15: CI/CD Build 20](#Figure!12|sequence)

[Figure 16: Code Quality 21](#Figure!13|sequence)

[Figure 17: Issues 22](#Figure!14|sequence)

Index of Tables

[Table 1: High Level Requirements 5](#Table!0|sequence)

[Table 2: High Level Requirements 5](#Table!1|sequence)

[Table 3: Unit testing 8](#Table!2|sequence)

[Table 4: Integration Testing 8](#Table!3|sequence)

[Table 5: High Level Requirements 15](#Table!4|sequence)

[Table 6: Low Level Requirements 15](#Table!5|sequence)

[Table 7: Unit testing 17](#Table!6|sequence)

[Table 8: Integration testing 17](#Table!7|sequence)

# Miniproject -1 : C++ - Hospital Management System [Individual]

## Module/s

Main module linked to the mini project is C++

### Topic and Subtopics

\* STL concepts (std::list)

\* Google test (unit testing)

\* make file (Cmake and Normal make file)

\* Object Oriented Concepts (Inheritance, Polymorphism, Abstraction)

\* File Input (CSV file)

## Objectives & Requirements

The project is about Hospital Management System in which the receptionist can directly enter or delete required information about the patient rather than making a note of it in the book.

Different information like getting patient details, adding patient by ID or name, discharging or removing the patient and billing details of the respective patient by ID or bill number.

**High level requirements:**

|  |  |
| --- | --- |
| **ID** | **Description** |
| HLR\_01 | Getting patient details |
| HLR\_02 | Adding Patient by the details like ID, Name, Age, Phone number |
| HLR\_03 | Maintaining the bill details of the patient |
|  |  |

Table 1: High Level Requirements

**Low level requirements**

|  |  |
| --- | --- |
| **ID** | **Description** |
| LLR\_01 | Reading data from csv file |
| LLR\_02 | Saving all patient details on list & forwarding it using vector using STL concepts |
| LLR\_03 | Implementation of CI/CD |

Table 2: High Level Requirements

## Design:

**Behavioral Diagrams:**

Use case diagram:

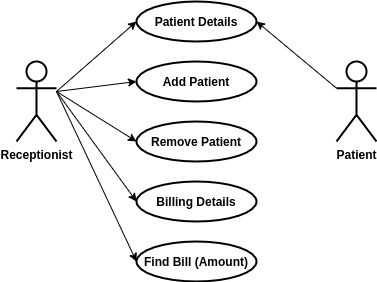
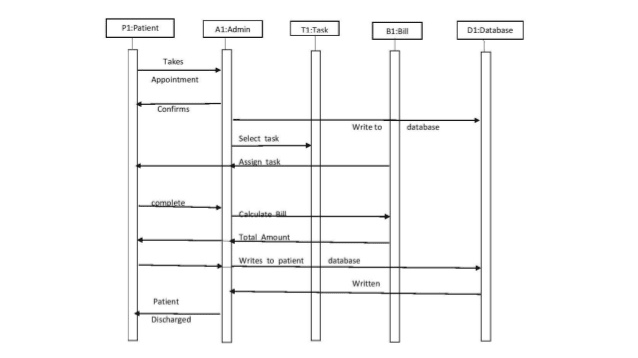


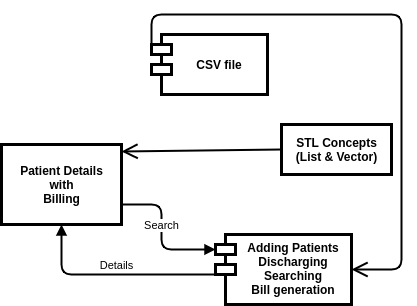
Figure 1: Use case diagram

Sequence Diagram:

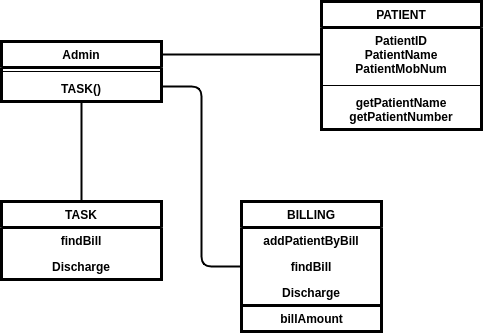
Figure 2: Sequence diagram

**Structural Diagrams:**

Component Diagram:

Figure 3: Component diagram

Class Diagram:

Figure 4: Class Diagram

## Test Plan:

**Unit testing:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test id** | **Description** | **Expected input** | **Expected output** | **Actual output** |
| HLR\_01 | Getting patient details | Name, Number | Display of Patient Name, Number |  |
| HLR\_02 | Adding patient dteails | Name, ID, Age | Adding into the patient records |  |
| HLR\_03 | Bill details | Adding amount, bed charges | Adding or removing bill of respective patient |  |
| HLR\_04 | Discharging of patient | Find by ID, discharge after paying amount | Bill cleared & patient details deleted from active patients list |  |

Table 3: Unit testing

**Integration testing:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test\_id** |  | **Description** |  | **Expected input** |  | **Expected output** | **Actual output** |
| LLR\_01 |  | Reading from csv file |  | CSV file |  | Adding of all details present in CSV to list |  |
| LLR\_02 |  | Adding details to list using STL concepts |  | Adding data to list |  | Details added to list from CSV |  |
| LLR\_03 |  | CI/CD |  | GitHub Actions |  | Cpp-Check, Valgrind, UnitTest, Codacy |  |

Table 4: Integration Testing

### **Implementation** Summary

The implementation of the above mini project is updated in the GitHub repository (99002519).The link for the same is provided below

https://github.com/99002519/Miniproject\_Cpp.git

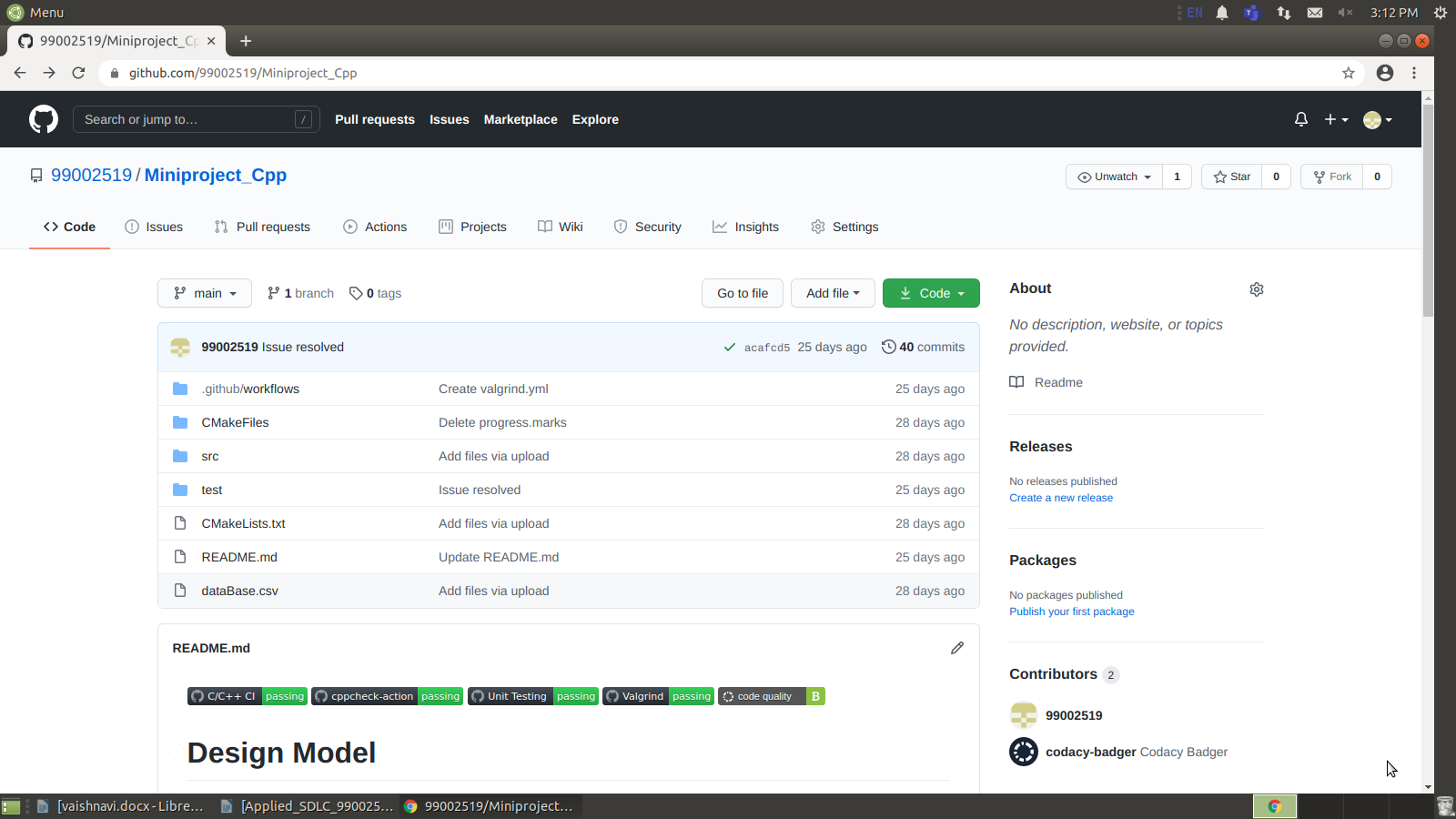
### Video Summary

Video link - [VideoSummary](https://github.com/99002519/Miniproject_Cpp/tree/main/VideoSummary)

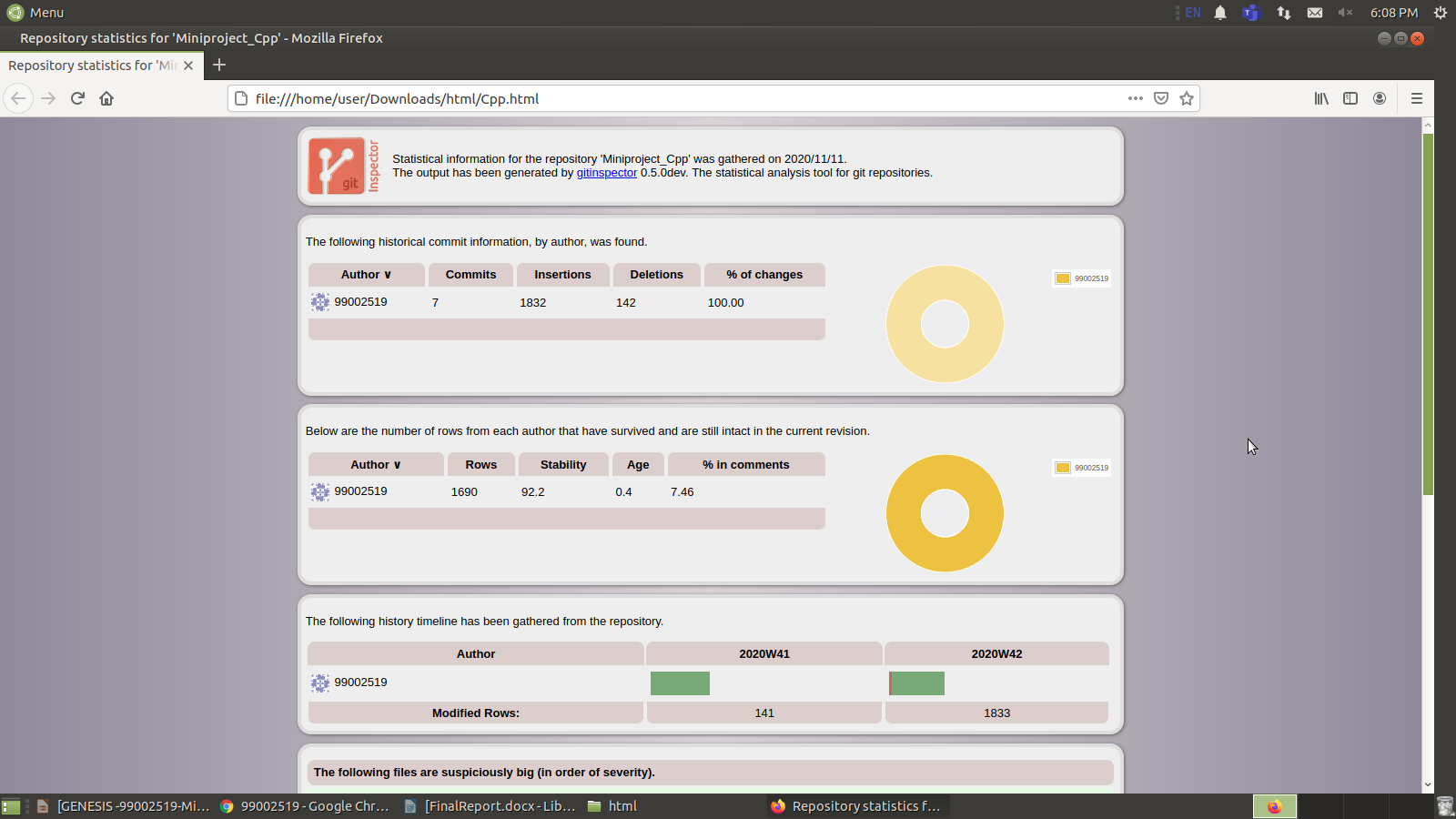
### Git Link

<https://github.com/99002519/Miniproject_Cpp.git>

### Git Dashboard

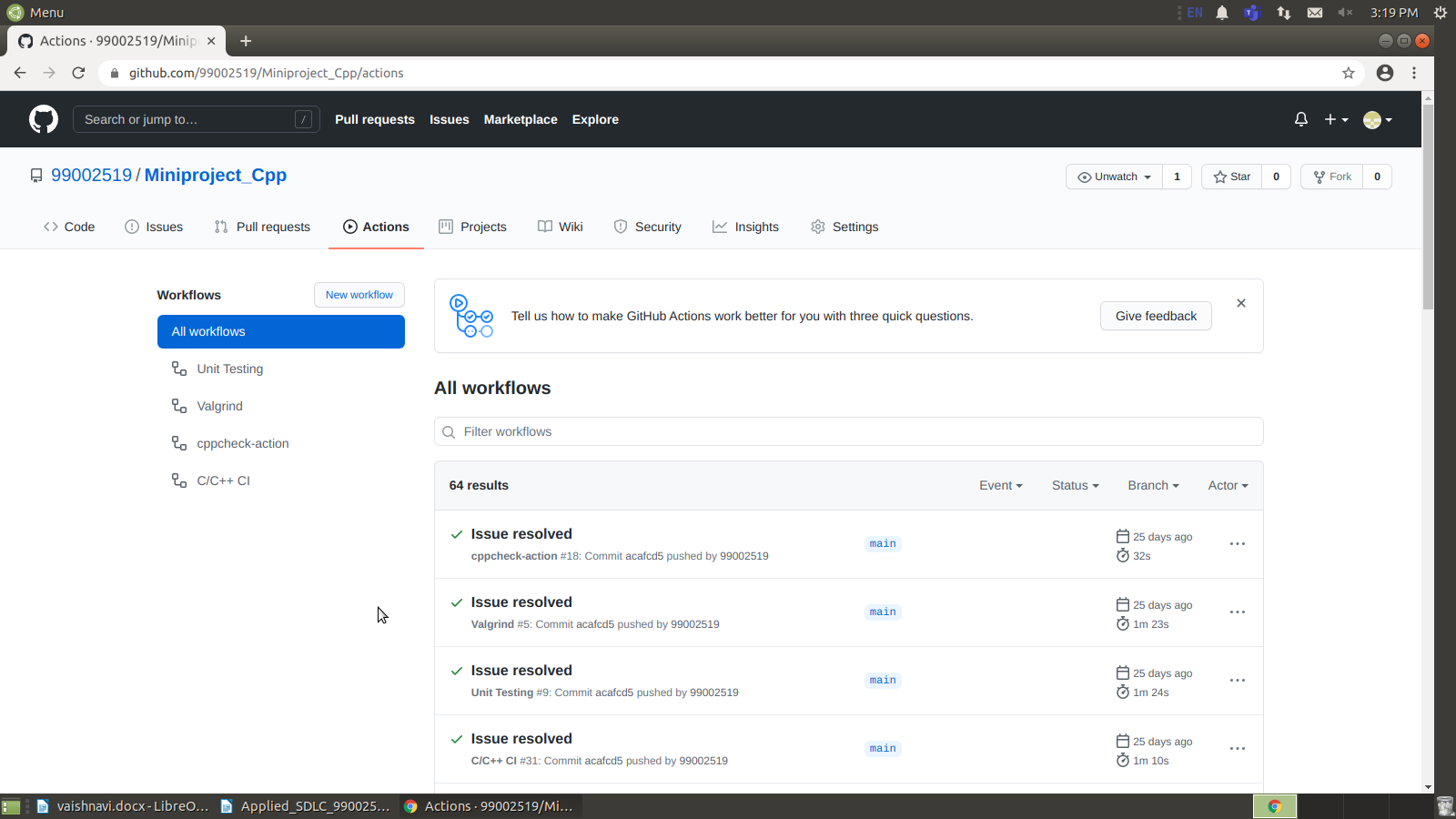
Figure 5: Git dashboard with all the badges

#### Git inspector summary

Figure 6: Git Inspector

#### Build

The mini-project is implemented using c++ language and the test units are implemented using the google test plan , a make file is created to build the project and verify whether the test units are cleared , In git action plans the C/C++ CI framework is setup to build the make file.

Figure 7: C/C++ Build using GitHub actions

#### Code quality and Issues or Bug Tracking

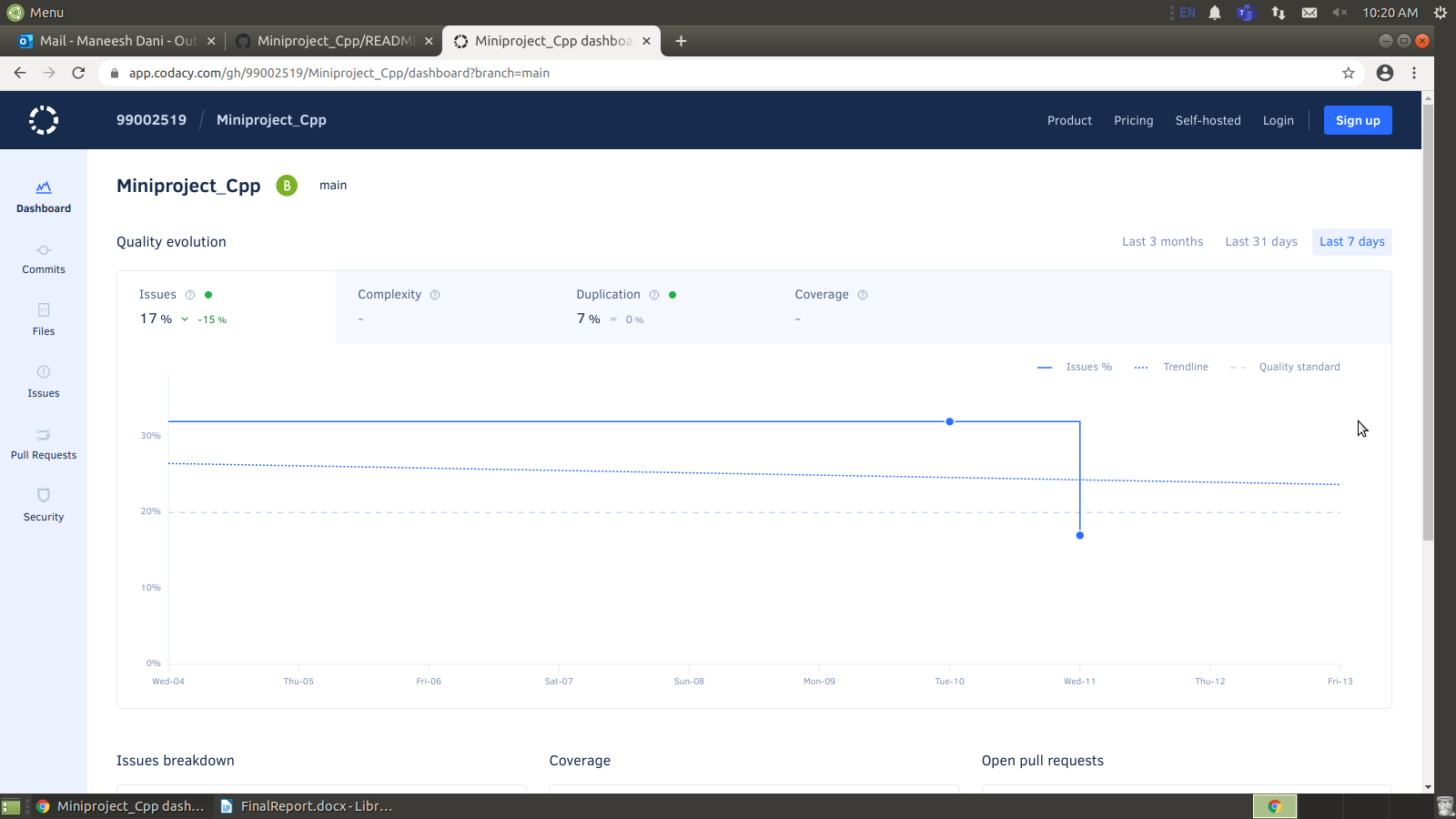
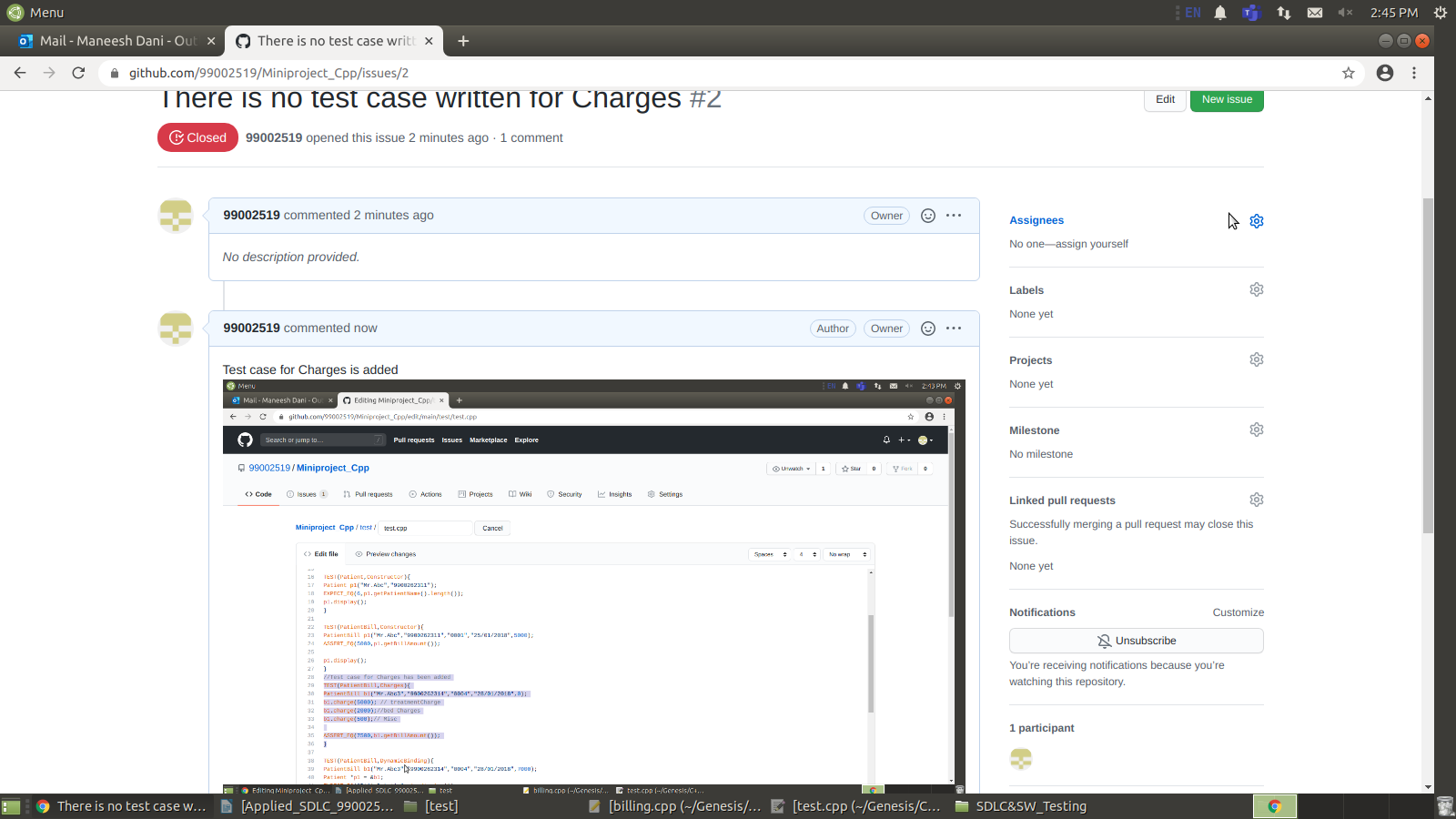


Figure 8: Codacy Code Quality

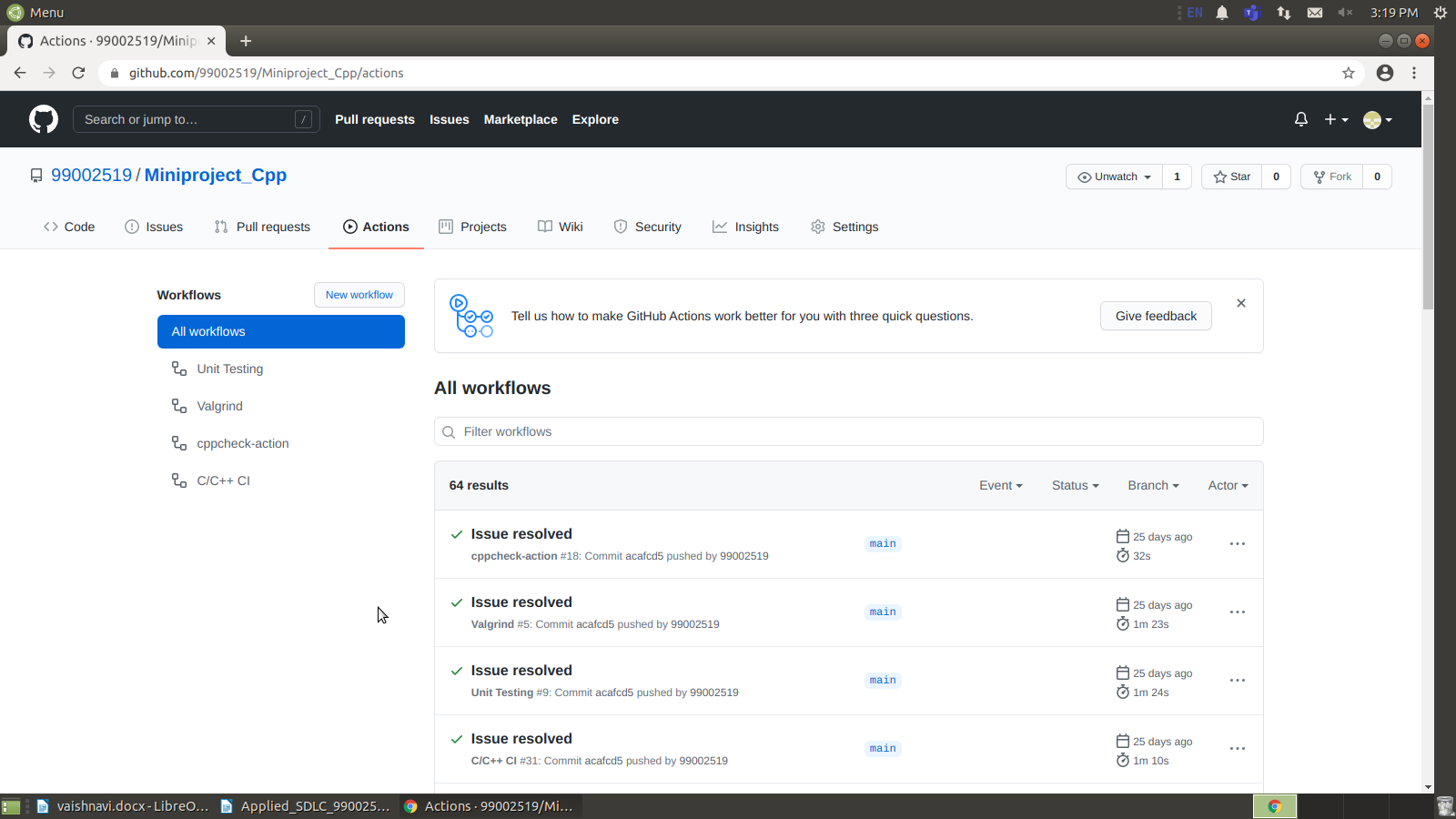
Figure 9: Issue raising and closing in GitHub

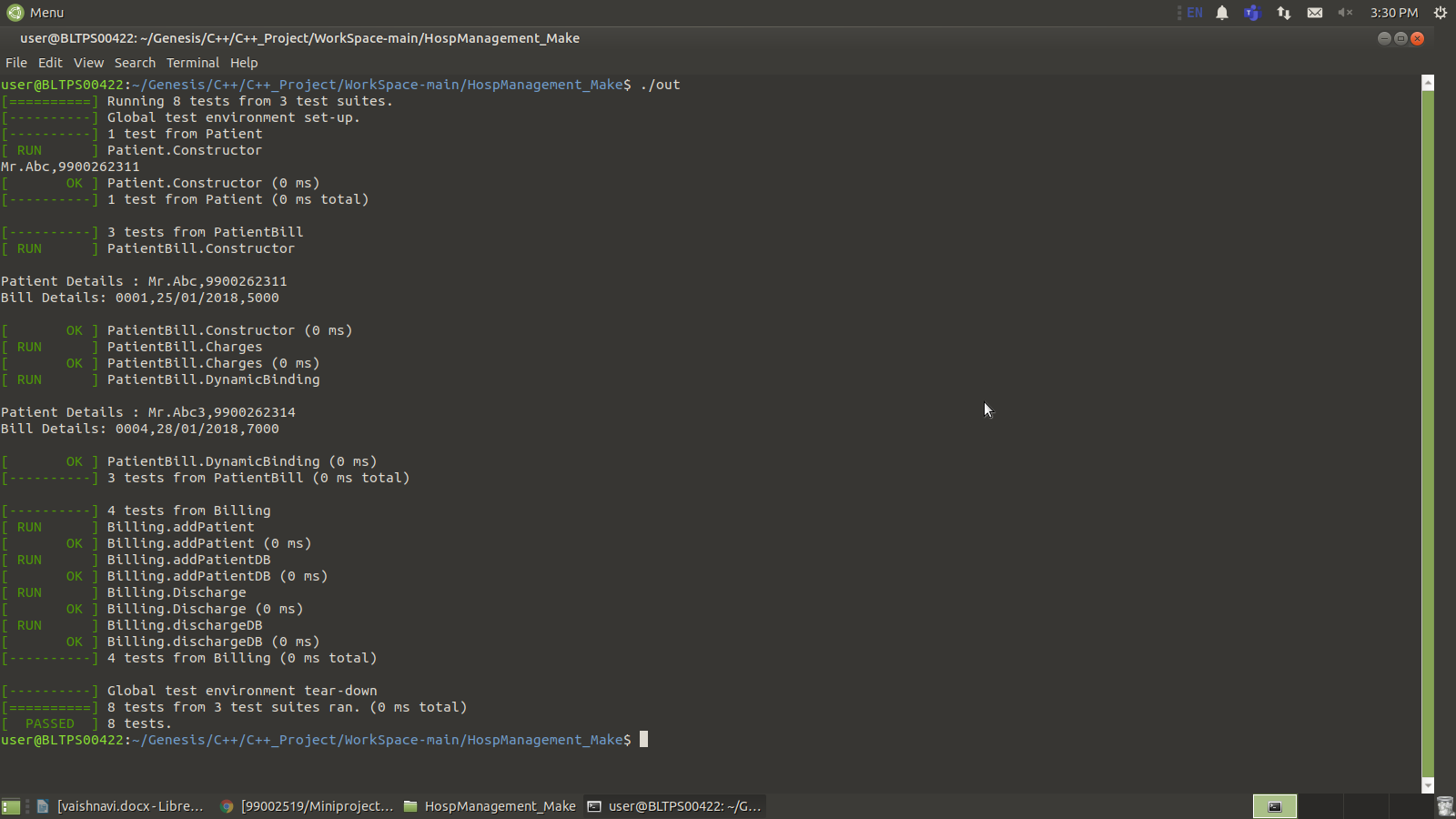
#### Unit Testing

Unit testing means running and validating the program for each basic functionality that can be done by the program, unit testing is performed in two ways:

1.Unit testing using the GitHub actions.(Fig 11)

2.unit testing using the customized MakeFile with the help of googletest library.(Fig 12)

Figure 10: Unit testing using GitHub actions

Figure 11: Unit testing using .out file

### Summary

* The proposed project is a analysis tool implemented using the various concepts of oops such as abstract classes, inheritance, virtual methods and STL, in C++ .The input to the tool is the CSV file which is fed with the data of patients. Features provided are:
* The Hospital Management System is used to get all the details of the patients and also remove the patient details after paying the bill.
* Here, patient details are added at the desk which helps to keep a track of the respective patient.
* The billing also can be precise as the bill details is attached with the patient details.

### Challenges faced and how were they overcome

* The challenges faced were fetching the data from the CSV file (i.e excel sheet) which was resolved by using the file operations - getLine() function and StringStream class.

### Future Scope (If applicable)

* This particular mini project can be enhanced to make it a real time application from where the patient can be monitored remotely by either doctor or nurse using cloud data.
* This can improved by using machine learning concepts like analyzing heart beat.

# Miniproject -2 : Python [Team]

## Module/s

* Socket Programming
* Mutex
* Regular Expressions
* Mulit-Threading
* PIP Packages
* inheritance and Polymorphism and Abstract Classes
* File handling

### Topic and Subtopics

* Analysis of the data of smartphones
* Reading of CSV Files

## Objectives & Requirements

**High level requirements:**

|  |  |
| --- | --- |
| **Req ID** | **Description** |
| HL\_01 | To connect the Server (Producer) and Client (Customer) |
| HL\_02 | To give the user, the data analysis of Mobile Phones based on previous Phones models data |
| HL\_03 | To know about the specific model of the Mobile phone |
| HL\_04 | The owner is able to add and delete the specific Company Mobile phone data to list |

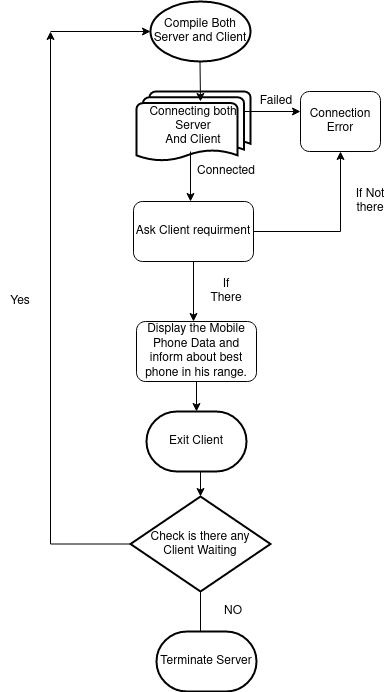
Table 5: High Level Requirements

**Low level requirements:**

|  |  |
| --- | --- |
| **Req ID** | **Description** |
| LL\_1.1 | The user should able to know on which Mobile phone is better on which company |
| LL\_2.1 | The user should able to know on which is the best phone on certain company |
|  |  |
| LL\_3.1 | The user will be able know about the specifications and specific data of that mobile company |

Table 6: Low Level Requirements

## Design

Figure 12: Flow Chart

## Test Plan

**Unit testing:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test\_Id** | **Description** | **Expected input** | **Expected output** | **Actual output** |
| T-01 | Establish Connection between server and client | Run server and client | Connecting the server and client | Passed |
| T-02 | Add Data | Specification of the mobile phone added | Add the Mobile phone data from list | Passed |
| T-03 | Display | Ask for display of data of Mobile phone . | Display the data present in List about the mobile phone | Passed |
| T\_04 | Number of elements in list | Count All | Display the Total Count of Mobile phones present in list | Passed |

Table 7: Unit testing

**Integration Testing:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test\_Id** | **Description** | **Expected Input** | **Expected output** | **Actual Output** |
| T\_01 | Mobile phone data | Specific grain name | Display of Grain data | Passed |
| T\_02 | Mobile phone RAM,ROM, Prices | Verifying the specific Product RAM,ROM and Prices and there Processor | Estimating which is best based on RAM, ROM , Processor and price | Passed |
| T\_03 | Knowing the RAM, ROM and Processor and Price | Knowing the specific phone RAM, ROM and Processor and price | Display of the Phone RAM , ROM and processor and Price | Passed |

Table 8: Integration testing

## Implementation Summary

Nowadays, because of the rapid changing technologies in the smartphone industries, it is very difficult to choose better smartphones based on the requirements. This Micro Project titled “Smartphone Analysis” is a simple project which helps the customer to compare smartphones and come to the decision of buying a better smartphone based on their requirements.

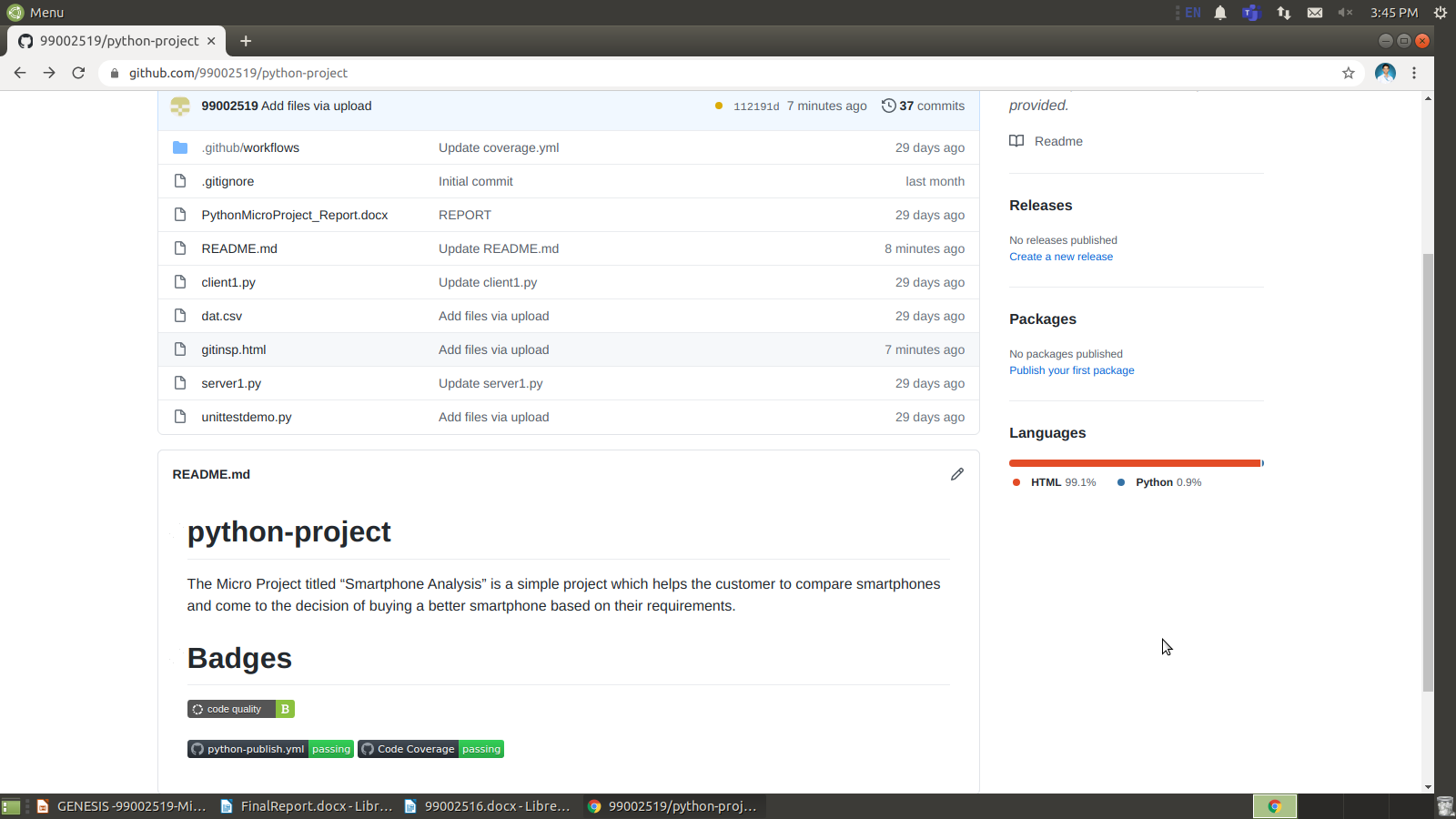
### Git Link

GitHub Link - https://github.com/99002519/python-project.git

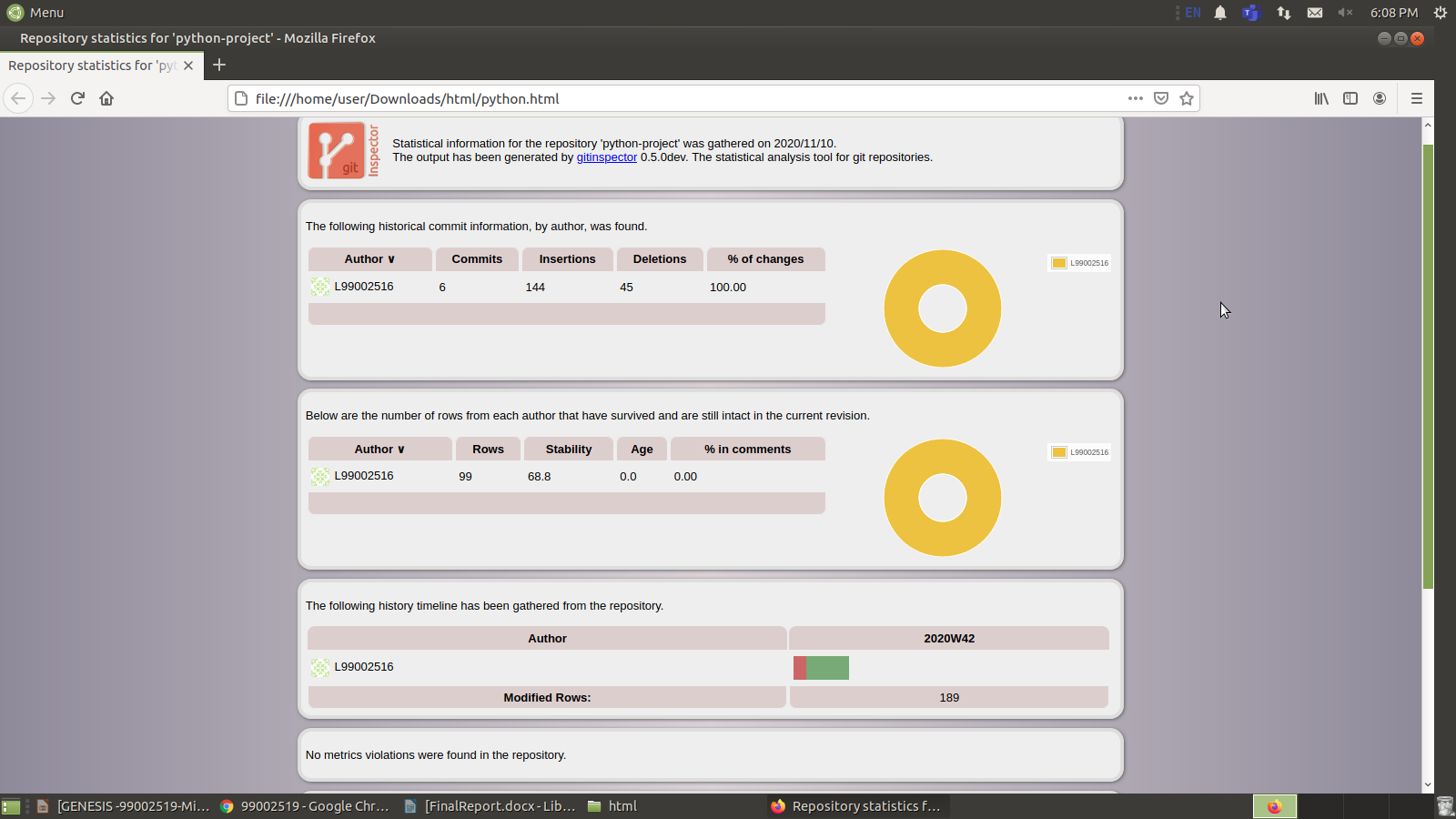
### Video Summary

Video link - [VideoSummary](https://github.com/99002519/python-project/tree/main/VideoSummary)

### Git Dashboard

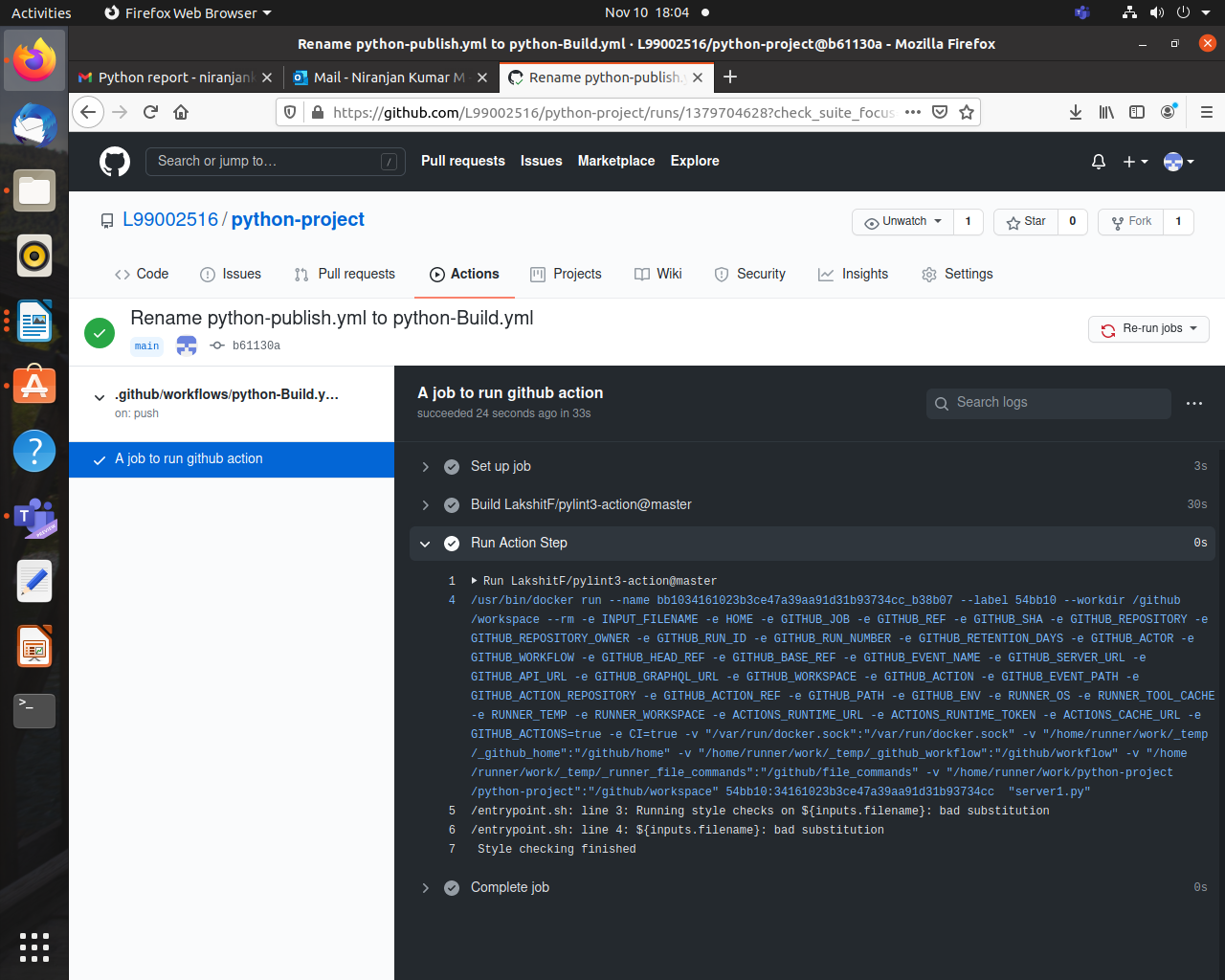
Figure 13: GitHub Dashboard

#### Git inspector summary

Figure 14: Git Inspector

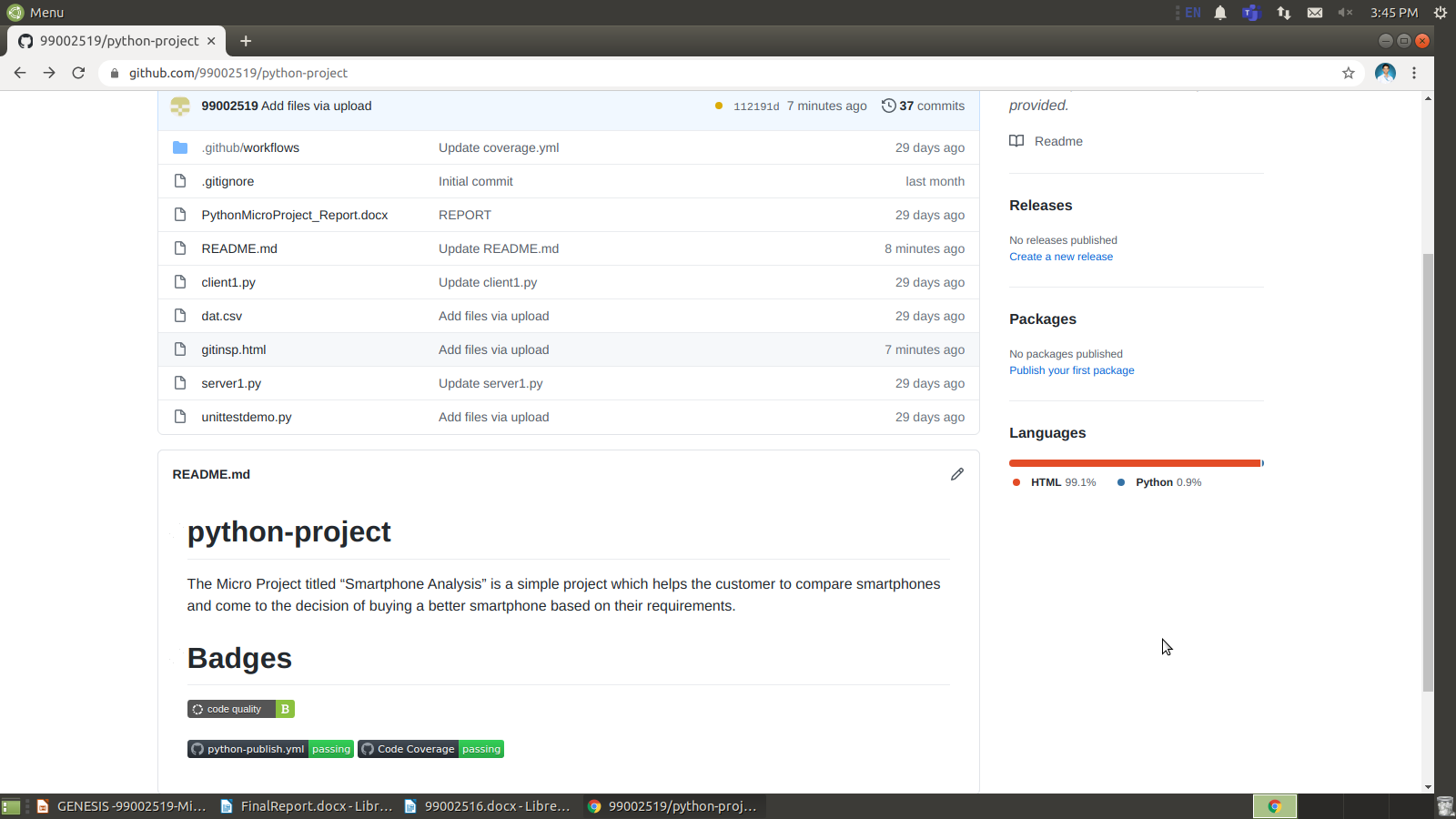
#### Build

For the Mini\_Project, we have used python language to code and with the help of Pytest software I have verified the all test cases. I have implemented the Python FrameWork and Code Coverage and codacy code Quality.

Figure 15: CI/CD Build

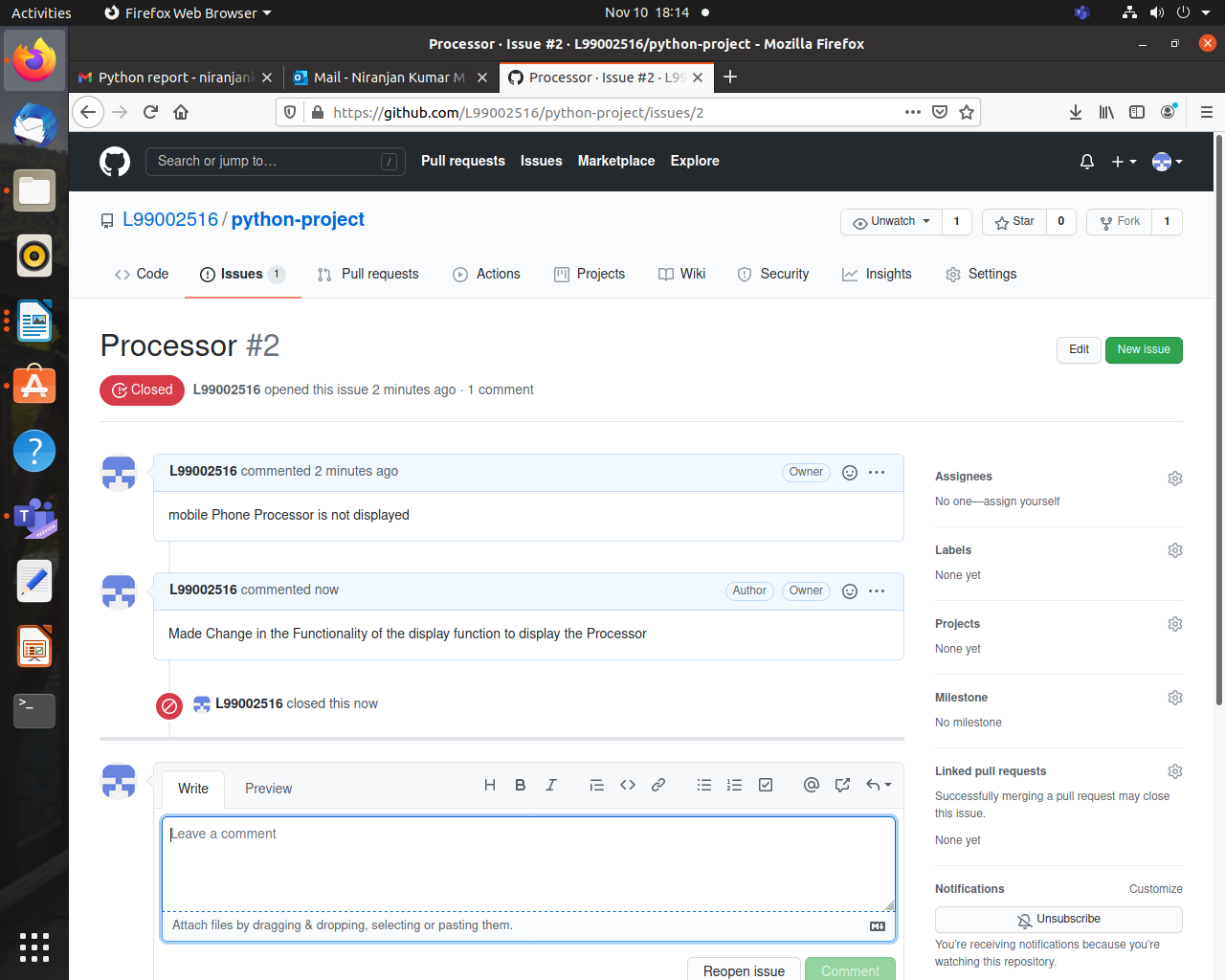
#### Code quality

#### Unit Testing

Figure 16: Code Quality

Code coverage is done in python and uploaded the batch in GitHub repository.

#### Issues

Figure 17: Issues

## Individual Contribution & Highlights

Niranjan Kumar M(99002516)

Writing the core logic of the project with Multithreading, Regular Expressions and executed test cases and git action( codacy and build/CI)

Maneesh Dani(99002519)

Socket programming and git action( code coverage).

### Summary

* Smart Analysis of the Mobile Phone data which was given in CSV file.
* Connecting the server as Consumer and Client as Customer .
* As per request of the Client Server need to display the specific Mobile Phone Data.
* Here , We Have Used the Mutex Concepts for server and client Connection.

### Challenges faced and how were they overcome

* Reading the CSV File in Server and displaying that in Client.
* Analysis of the Mobile phone data.
* Storing the Mobile phone data.