­./

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 | Asha N |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Details**

Contents

[Miniproject -1 [Team/Individual] 5](#__RefHeading___Toc594_253822950)

[Title : Doctor Appointment System 5](#__RefHeading___Toc677_253822950)

[1. Module/s Used 5](#__RefHeading___Toc596_253822950)

[1.1 Topic and Subtopics 5](#__RefHeading___Toc598_253822950)

[2. Objectives & Requirements 5](#__RefHeading___Toc600_253822950)

[2.1 High level requirements 5](#__RefHeading___Toc1561_3788233749)

[2.2 Low level requirement 6](#__RefHeading___Toc1563_3788233749)

[3. Design 6](#__RefHeading___Toc602_253822950)

[3.1 Class Diagram 6](#__RefHeading___Toc683_253822950)

[3.2 Activity Diagram 7](#__RefHeading___Toc685_253822950)

[3.3 Component Diagram 8](#__RefHeading___Toc4645_2172971112)

[3.4 Use Case Diagram 9](#__RefHeading___Toc4647_2172971112)

[4. Test Plan 9](#__RefHeading___Toc604_253822950)

[4.1 Unit Testing 9](#__RefHeading___Toc4649_2172971112)

[4.2 Integration Testing 10](#__RefHeading___Toc4651_2172971112)

[5. Implementation Summary 10](#__RefHeading___Toc606_253822950)

[5.1 Git link 10](#__RefHeading___Toc610_253822950)

[5.2 Git Dashboard 11](#__RefHeading___Toc612_253822950)

[5.3 Summary 11](#__RefHeading___Toc614_253822950)

[6.Individual Contribution & Highlights 16](#__RefHeading___Toc1559_3788233749)

[7. Challenges faced and how were they overcome 16](#__RefHeading___Toc620_253822950)

[7.1 Future Scope 16](#__RefHeading___Toc622_253822950)

[Miniproject -2 [Team/Individual] 17](#__RefHeading___Toc626_253822950)

[Title : Matrix Multiplication Using Multithreading 17](#__RefHeading___Toc2165_1638596263)

[1. Module/s Used 17](#__RefHeading___Toc628_253822950)

[Mini project is related to linux and OS programming using C programming. 17](#__RefHeading___Toc5703_3408031208)

[1.1 Topic and Subtopics 17](#__RefHeading___Toc630_253822950)

[2. Objectives & Requirements 17](#__RefHeading___Toc632_253822950)

[3. Implementation Summary 17](#__RefHeading___Toc638_253822950)

[3.1 Git Link 17](#__RefHeading___Toc640_253822950)

[3.2 Git Dashboard 18](#__RefHeading___Toc642_253822950)

[4.3 Summary 21](#__RefHeading___Toc2175_1638596263)

[5. Individual Contribution & Highlights 21](#__RefHeading___Toc646_253822950)

[5.1 Issues in Codacy 21](#__RefHeading___Toc1598_3788233749)

[5.2 Challenges faced and how were they overcome 21](#__RefHeading___Toc650_253822950)

[Miniproject -3 [Team/Individual] 22](#__RefHeading___Toc5705_3408031208)

[Title : Top Women Chess Players 22](#__RefHeading___Toc677_2538229501)

[1. Module/s Used 22](#__RefHeading___Toc596_2538229501)

[2. Objectives & Requirements 22](#__RefHeading___Toc600_2538229501)

[2.1 Requirements 22](#__RefHeading___Toc681_2538229501)

[2.1.1 High level requirements 22](#__RefHeading___Toc1600_3788233749)

[2.1.2 Low level requirements 22](#__RefHeading___Toc1602_3788233749)

[3. Design 23](#__RefHeading___Toc602_2538229501)

[3.1 Class Diagram 23](#__RefHeading___Toc683_2538229501)

[3.2 Use Case Diagram 24](#__RefHeading___Toc685_2538229501)

[4. Implementation Summary 24](#__RefHeading___Toc606_2538229501)

[4.1 Git link 24](#__RefHeading___Toc610_2538229501)

[4.2 Git Dashboard 25](#__RefHeading___Toc612_2538229501)

[4.3 Summary 25](#__RefHeading___Toc614_2538229501)

[5. Individual Contribution & Highlights 30](#__RefHeading___Toc616_2538229501)

[5.1 Challenges faced and how were they overcome 30](#__RefHeading___Toc620_2538229501)

[5.2 Future Scope 30](#__RefHeading___Toc622_2538229501)

[6. REFERENCES 30](#__RefHeading___Toc624_2538229501)

**Table of Figures**

Figure 1 : Class Diagram

Figure 2 : Activity Diagram

Figure 3 : Component Diagram

Figure 4 : Use case Diagram

Figure 5 : Git Dashboard

Figure 6 : Git Inspector

Figure 7 : Build

Figure 8 : Badges

Figure 9 : Issues

Figure 10 : Unit Testing

Figure 11 : Git Dashboard

Figure 12 : Git Inspector

Figure 13 : Build

Figure 14 : Issues in Codacy

Figure 15 : Class Diagram

Figure 16 : Use case Diagram

Figure 17 : Git Dashboard

Figure 18 :Git Inspector

Figure 19 : Build

Figure 20 : Code Coverage

Figure 21 : Pylint

# Miniproject -1 [Team/Individual]

## **Title : Doctor Appointment System**

## **1.** **Module/s Used**

Modules linked to the miniproject are Linux, SDLC and C++.

C++:

C++ is a general-purpose programming language created by Bjarne Stroustrup as an extension of the C programming language, or "C with Classes.

SDLC:

Software Development Life Cycle (SDLC) is a framework that defines the steps involved in the development of software at each phase. It covers the detailed plan for building, deploying and maintaining the software.

### 1.1 Topic and Subtopics

* CSV file
* Class and Object-OOP in C++.
* Google test(Unit test)
* Make file
* SDLC and TDLC
* Polymorphism
* Operator overloading
* STL containers.
* Code Quality(Codacy)

## **2.** **Objectives & Requirements**

### 2.1 High level requirements

|  |  |
| --- | --- |
| **ID** | **Description** |
| HL\_01 | Analysis of patient details. |
| HL\_02 | Comparison of different diseases |
| HL\_03 | Highest and lowest aged patients. |
| HL\_04 | Adding new patient details |

### 2.2 Low level requirement

|  |  |
| --- | --- |
| **ID** | **Description** |
| LL\_01 | Reading data from csv file. |
| LL\_02 | Saving all data on list using STL concepts |
| LL\_03 | Implementation of CI/CD. |

## **3.** **Design**

### 3.1 **Class Diagram**

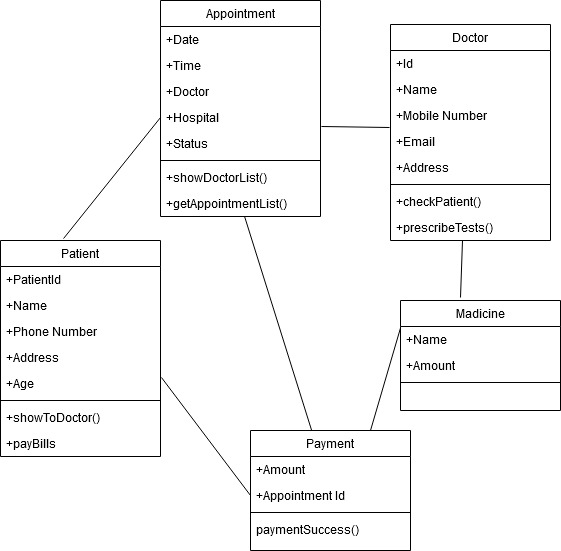


Figure 1 : Class Diagram

### **3.2 Activity Diagram**

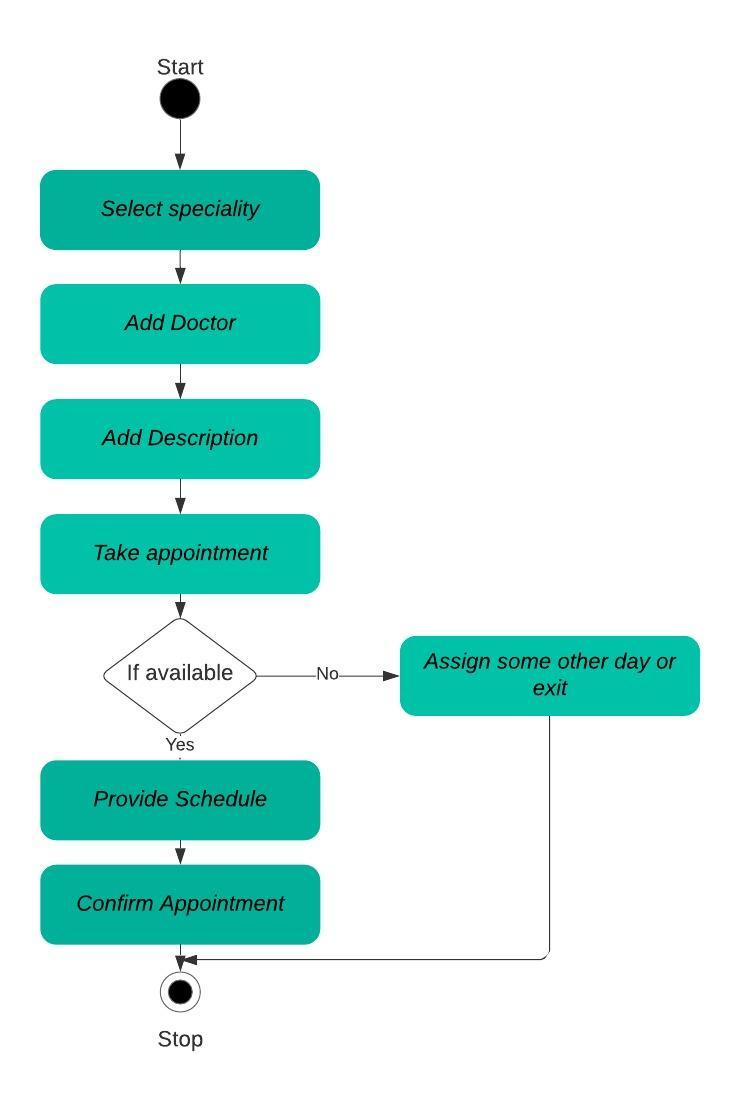


Figure 2 : Activity Diagram

### **3.3 Component Diagram**

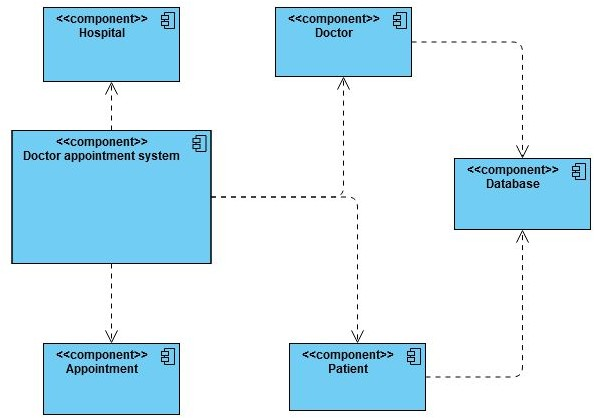


Figure 3 : Component Diagram

### **3.4 Use Case Diagram**

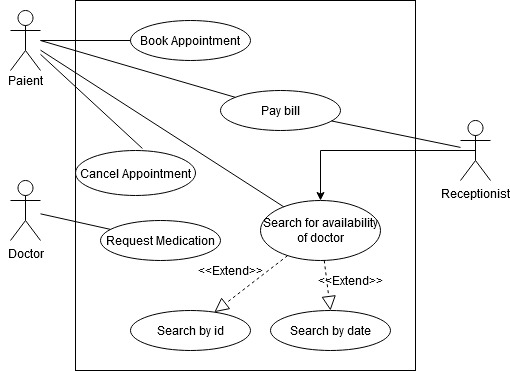


Figure 4 : Use case Diagram

## **4.** **Test Plan**

### **4.1 Unit Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test id | Description | Expected input | Expected output | Actual output |
| HH\_01 | Knowing of patient details. | Adding the data to list | Display of list where patient is added | Patient added |
| HH\_02 | Analysis of different diseases | Checking of different diseases | Printing of different diseases | True |
| HH\_03 | Highest aged patients | Giving patient name | Giving the highest aged patient name. | Year patient name |
| HH\_04 | Adding of new patient | Adding of new patient | Display of list where new patient is added | True |

### **4.2 Integration Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test id | Description | Expected input | Expected output | Actual output |
| LL\_01 | Reading of csv file | Csv file | Adding of all data present in csv to list | Data added to list |
| LL\_02 | Adding data to list using STL concepts | Adding data to list | Data added to list | Display of list |
| LL\_03 | CI/CD | GitHub Actions | Cppcheck, valgrind, unit testing , codacy | Passing all CI/CD |

## **5.** **Implementation Summary**

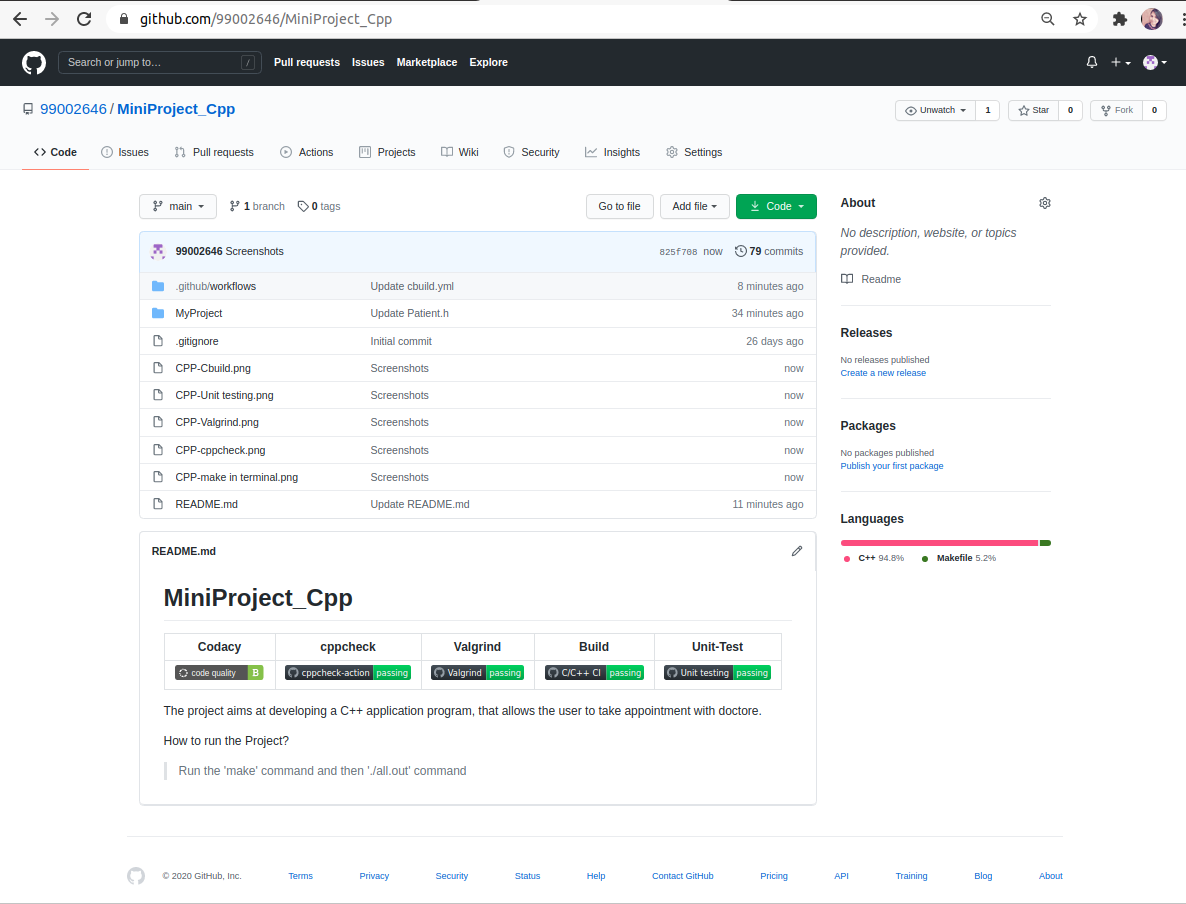
The implementation of this project is updated in the git repository “99002646/MiniProject\_Cpp”

https://github.com/99002646/MiniProject\_Cpp.git

### **5.1** **Git link**

**https://github.com/99002646/MiniProject\_Cpp.git**

### 5.2 Git Dashboard

Figure 5 : Git Dashboard

### 5.3 Summary

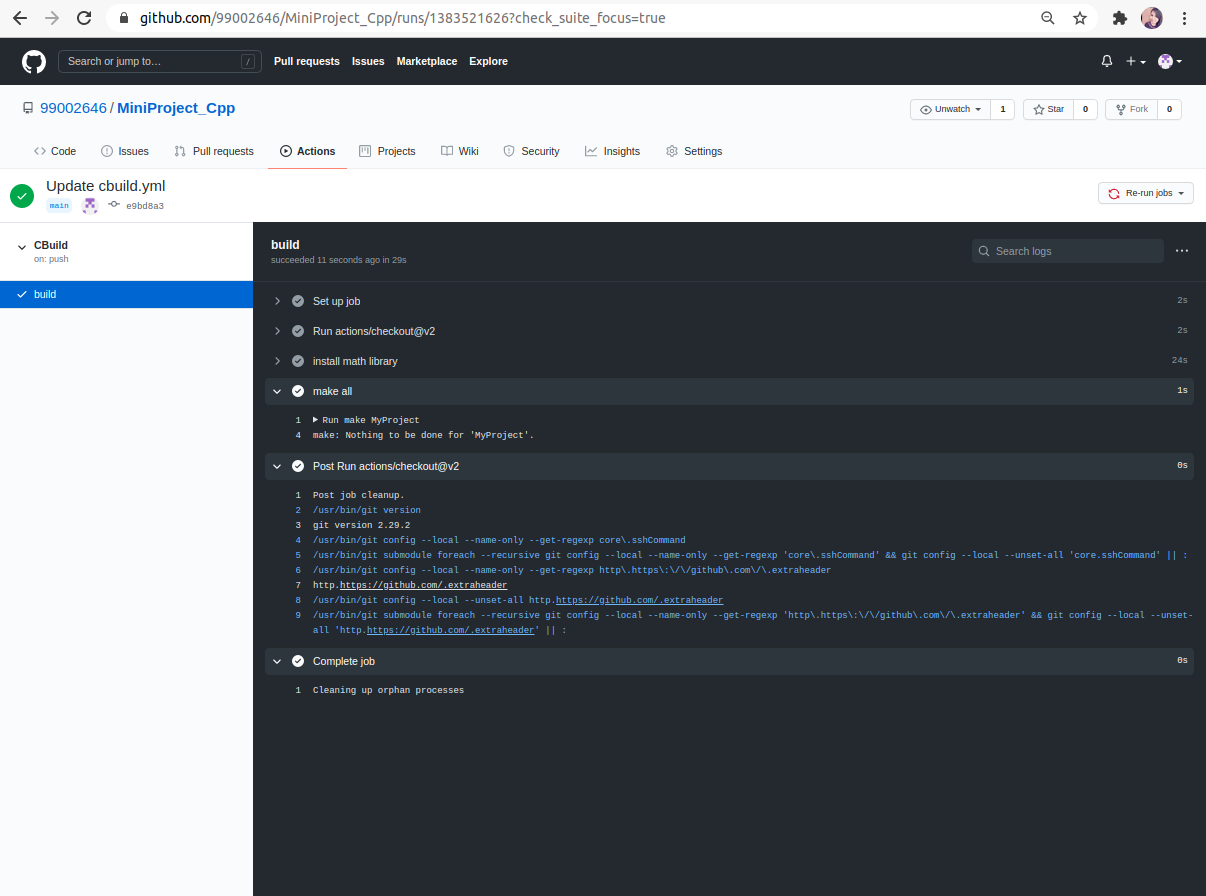
* The Doctor appointment system project is implemented using the C++.
* STL Concepts is used to work with the csv file.

#### Git inspector summary

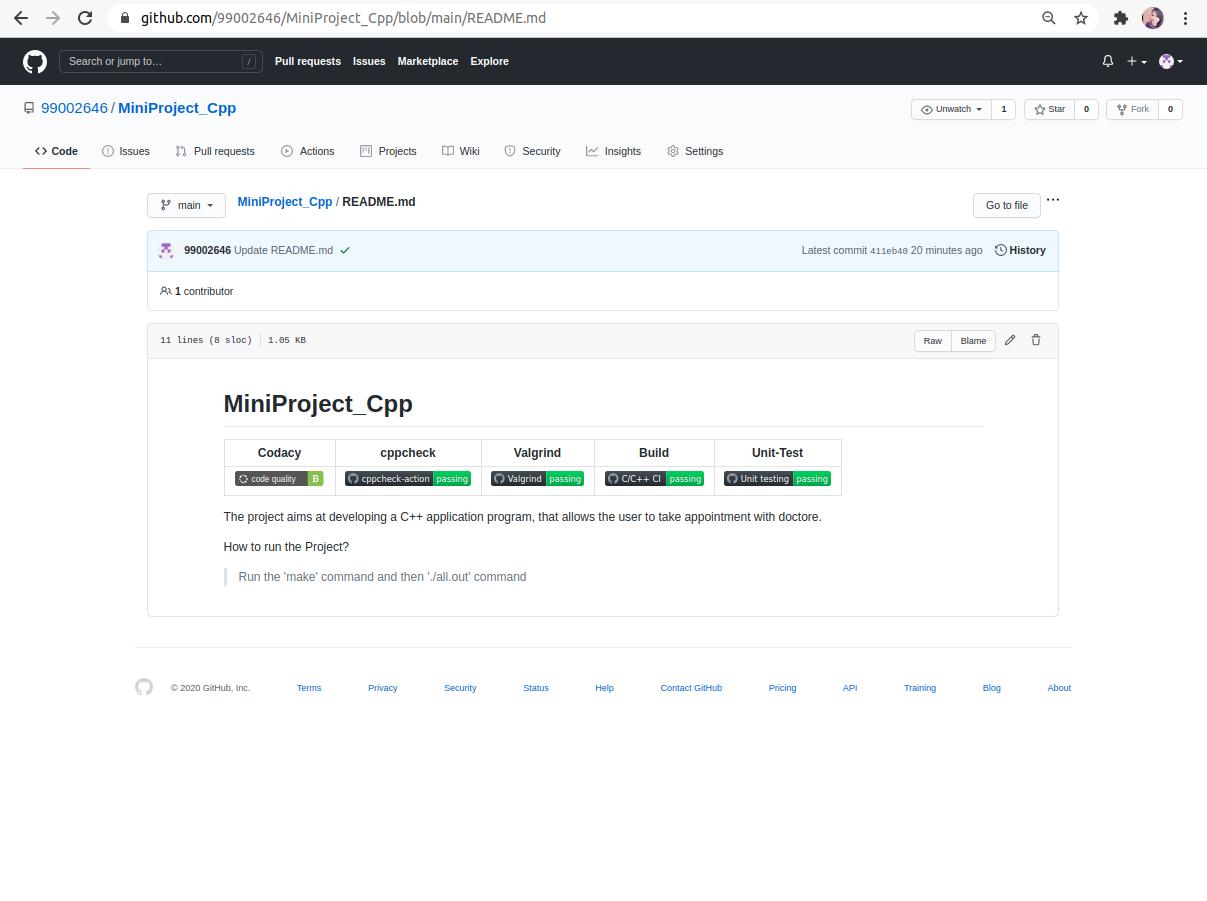
Figure 6 : Git Inspector

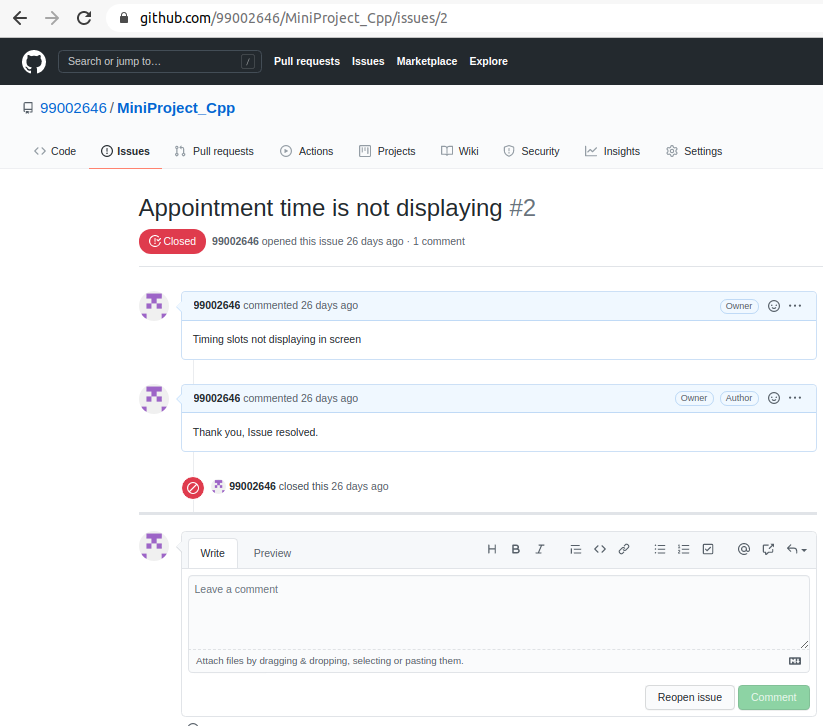
#### Build

* Use Cmake to link all the files and generate the Makefile
* Execute generated Makefile
* run ./executeTests
* run valgrind ./executeTests to check for memory leaks

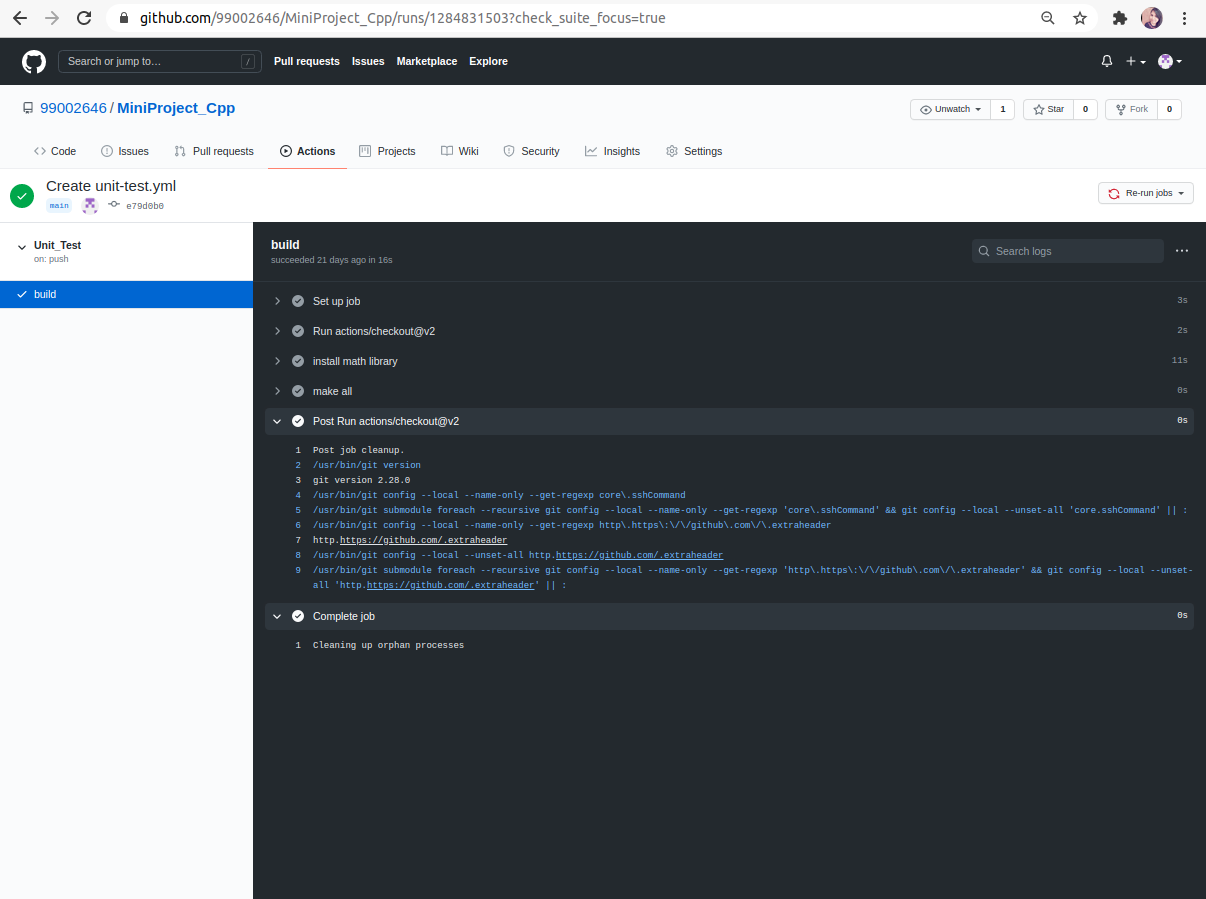
Figure 7 : Build

#### Code quality and Issues or Bug Tracking

Figure 8 : Badges

Figure 9 : Issue

#### Unit Testing

Figure 10 :Unit Testing

## **6.****Individual Contribution & Highlights**

Not Applicable – Individual Project; Sole Collaborator.

## **7.** **Challenges faced and how were they overcome**

* While implementing the STL concepts like list, vector we have faced an issue to display the data from the Data set .
* We have overcome this problem by implementing it using the list as the containers.

### 7.1 Future Scope

* This project can be implemented further by upgrading the appointment schedule and analyzing the shows.

# Miniproject -2 [Team/Individual]

## **Title : Matrix Multiplication Using Multithreading**

## **1.** **Module/s Used**

## Mini project is related to linux and OS programming using C programming.

### 1.1 Topic and Subtopics

* Process, Threads and IPC
* File handling
* Multi threading

## **2.** **Objectives & Requirements**

* To perform the multiplication in multiple threads
* To decide the user to create number of threads
* reading the input from .txt file.

## **3.** **Implementation Summary**

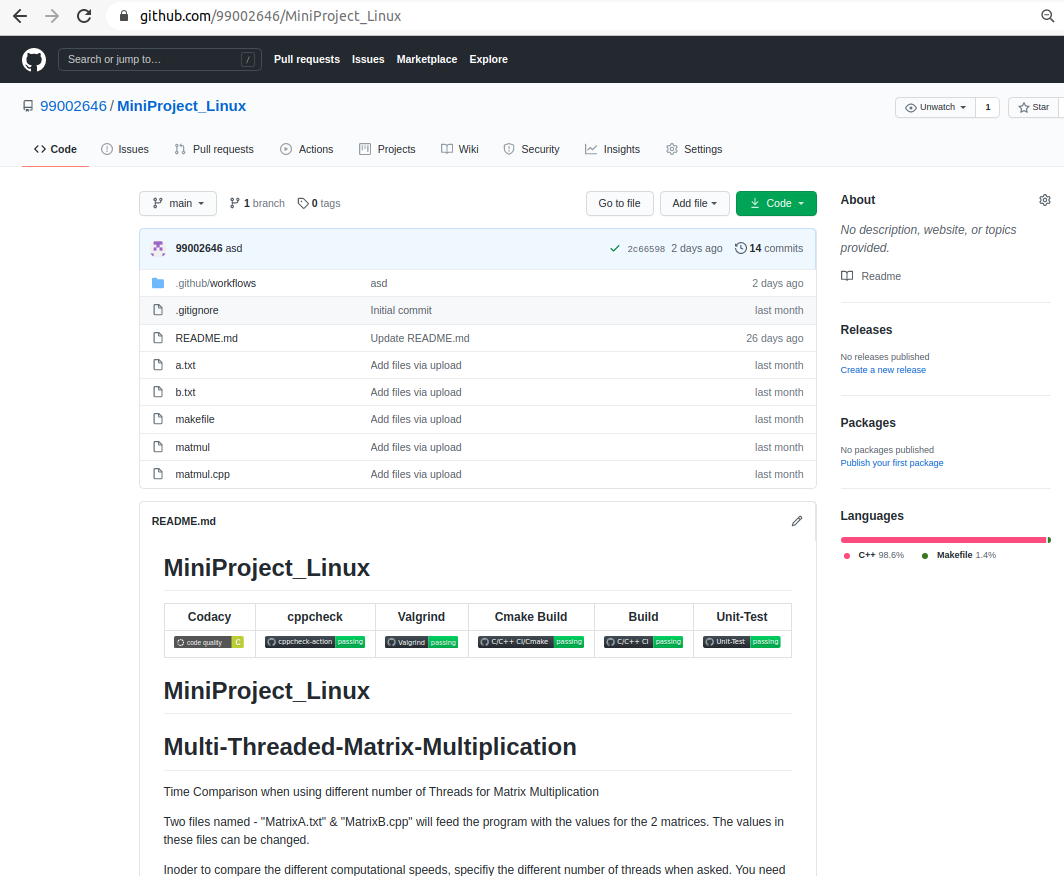
The linux and OS programming project is updated in the git repository “99002646/MiniProject\_Linux”

<https://github.com/99002646/MiniProject_Linux.git>

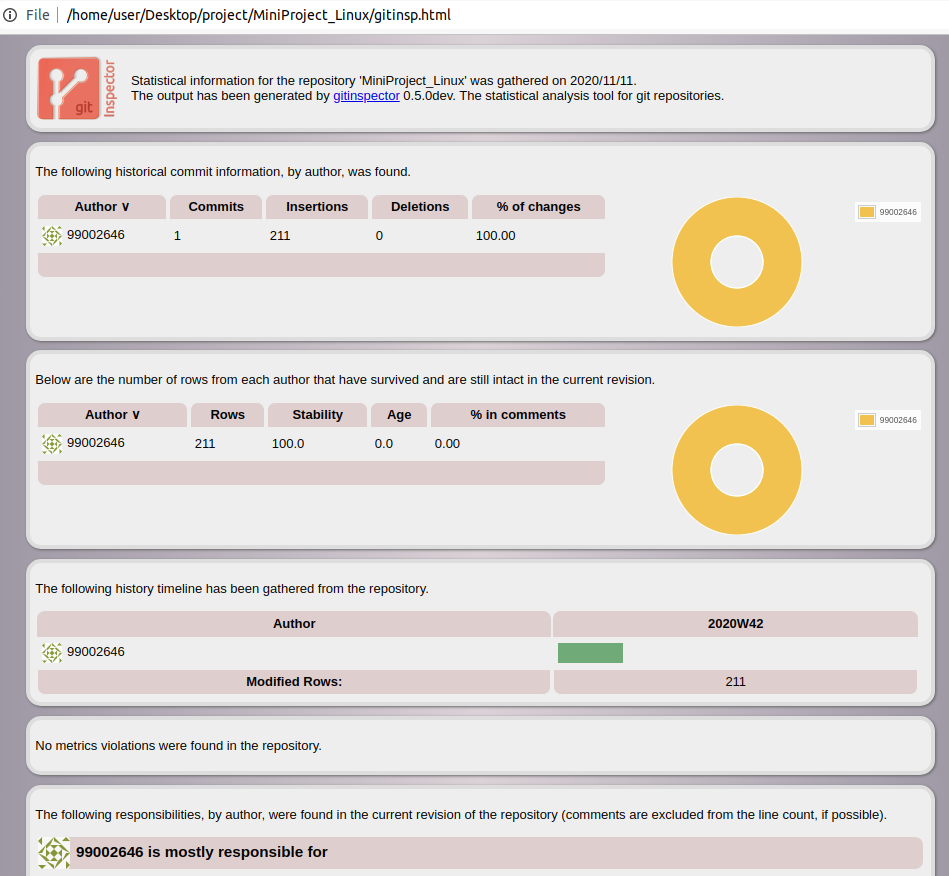
### 3.1 Git Link

<https://github.com/99002646/MiniProject_Linux.git>

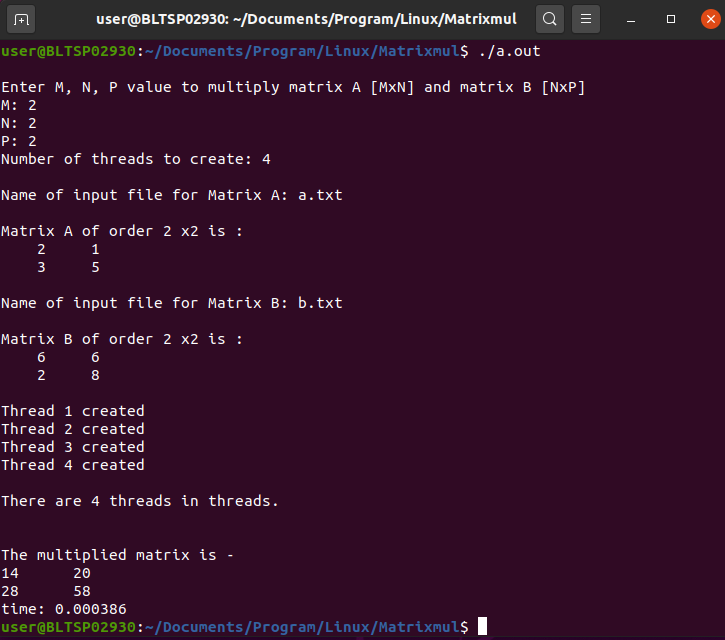
### 3.2 Git Dashboard

Figure 11 : Git Dashboard

#### Git inspector summary

Figure 12 : Git Inspector

#### Build

Figure 13 : Build

#### Code quality

* Code Quality: Codacy: B Grade
* CppCheck: Passing
* C/C++ CI Build: Passing

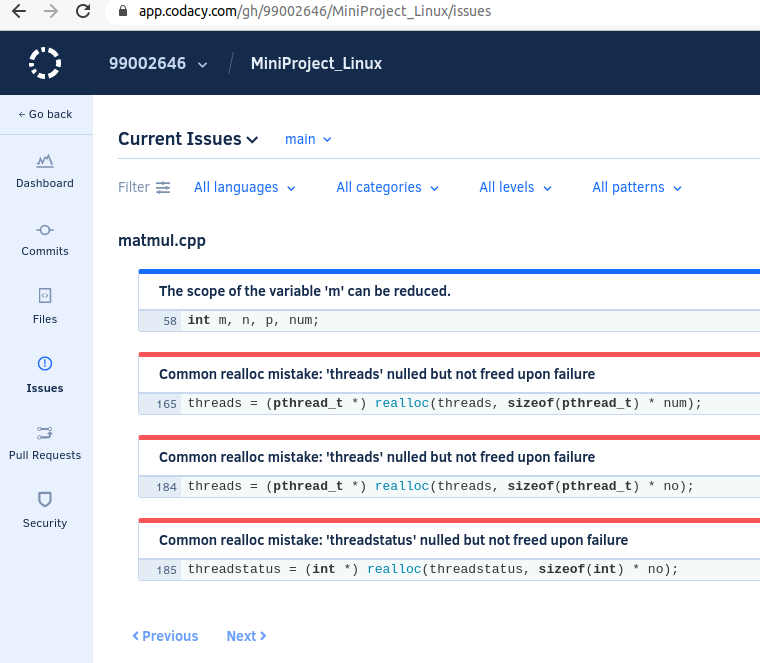
## **4.** **Summary**

In this project we have not used any synchronization in future it can be implemented while serving the burger with many customer to maintain the process accurately.

## **5.** **Individual Contribution & Highlights**

Not Applicable – Individual Project; Sole Collaborator.

### 5.1 Issues in Codacy

Figure 14 : Issues in Codacy

### 5.2 Challenges faced and how were they overcome

While implementing the file handling concept we have faced some issues like reading error, file not found error, then resolved the issue.

# Miniproject -3 [Team/Individual]

## **Title : Top Women Chess Players**

## **1.** **Module/s Used**

Mini project is related to python.

## **2.** **Objectives & Requirements**

* The top women chess players analysis mini project has fide id, name of the player, title, date of birth, federation, standard rating, rapid rating, blitz rating columns.
* which helps to find the players by their fide id.
* It sorts the names, titles with the ascending and descending order.
* It calculates the standard rating, rapid rating, blitz rating and finds the maximum and minimum rating for the individual players.
* Counts the number of players in the given data set and group the players by their respective federation and region.

### 2.1 Requirements

### 2.1.1 High level requirements

-To Compute the minimum and maximum rating of the Top Chess players with standard rating, rapid rating and blitz rating.

-To Count the players by the Federation.

-To Compute the Maximum rating of the players.

### 2.1.2 Low level requirements

-To sort the names, titles of the players in the ascending and descending order.

-To find women players by the date of birth and federation name.

## **3.** **Design**

### 3.1 **Class Diagram**

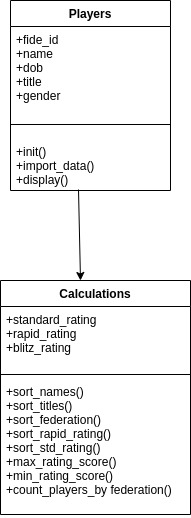


Figure 15 : Class Diagram

### **3.2 Use Case Diagram**

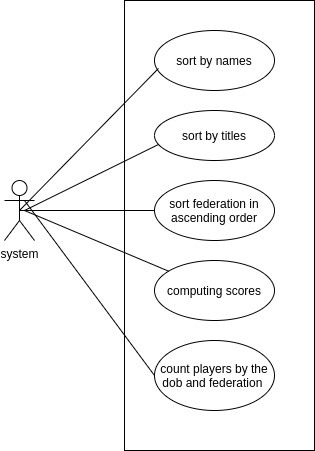


Figure 16 : Use case Diagram

## **4.** **Implementation Summary**

The implementation of this project is updated in the git repository “99002646/MiniProject\_Python”

[https://github.com/99002646/MiniProject\_Python.git](https://github.com/99002646/MiniProject_Cpp)

### **4.1** **Git link**

[https://github.com/99002646/MiniProject\_Python.git](https://github.com/99002646/MiniProject_Cpp)

### 4.2 Git Dashboard

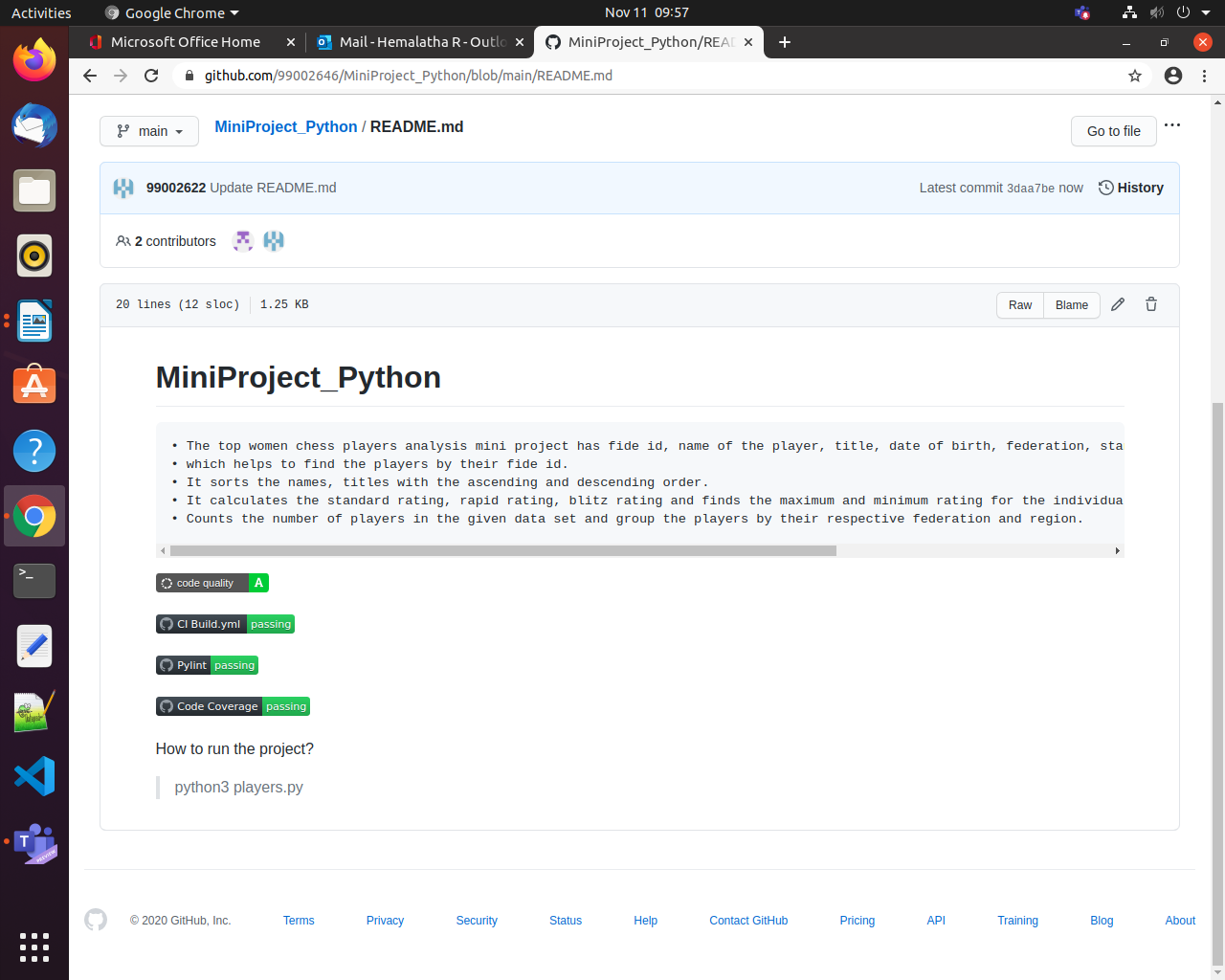


Figure 17 : Git Dashboard

### 4.3 Summary

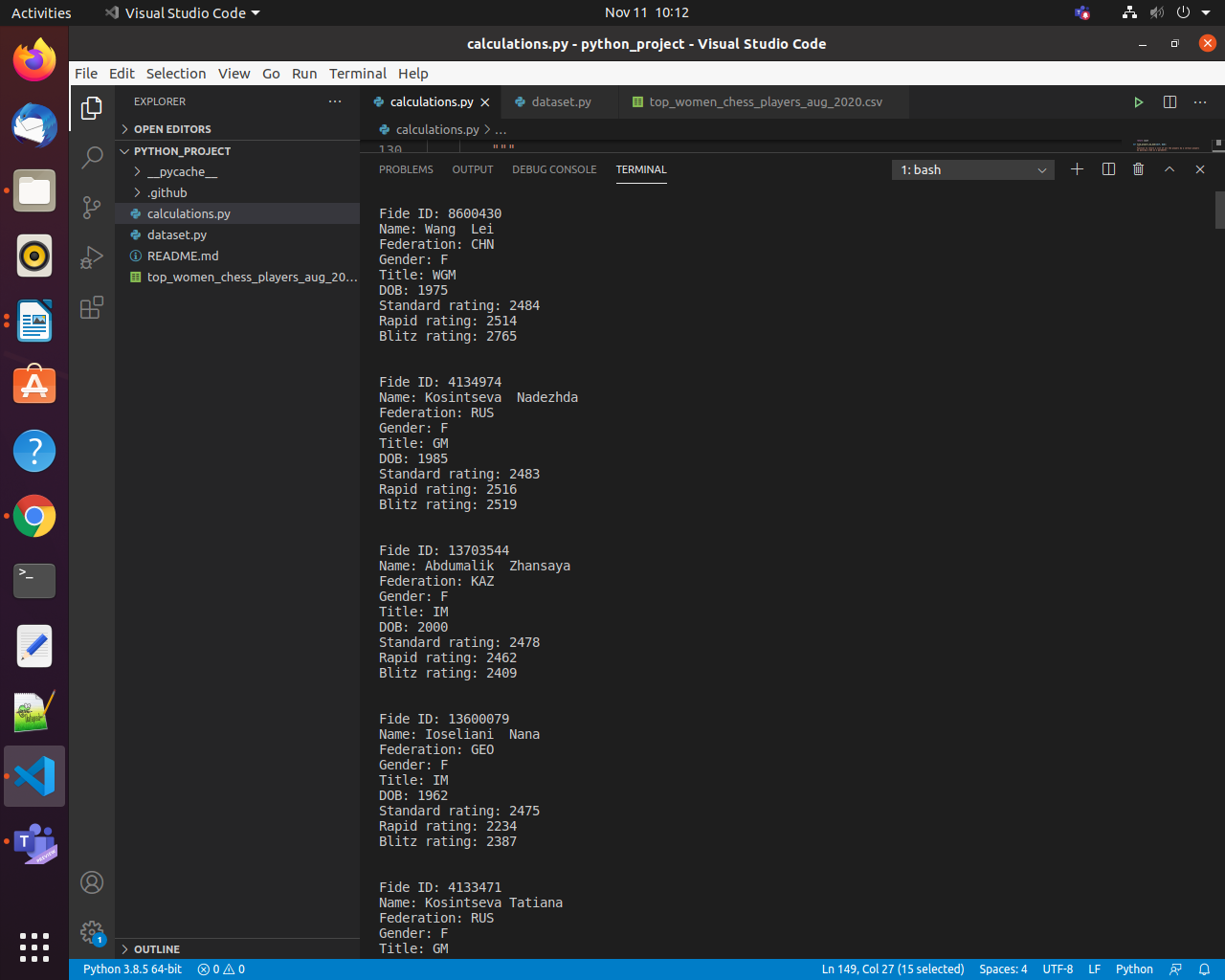
* The Top women chess player analysis Project is implemented using the C++ which displays the players name, fide id, date of birth, gender, federation, Titles and ratings.
* STL Concepts is used to retrieve the DataSet.
* Here, we perform some of the functions like computing the ratings of all the three different ratings and displaying the maximum ratings of all the players.
* By using the Federation as the key displaying the players within that federation.
* Identifying the player using the Fide ID by using the Id as parameter.
* Finding total number of women players by using the count function in the dataset.

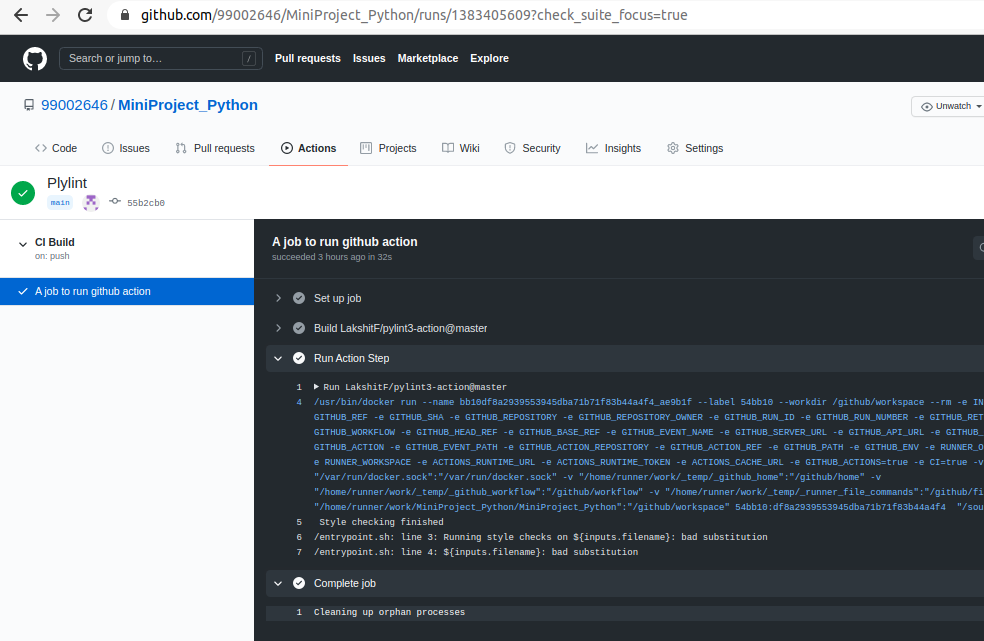
#### Git inspector summary



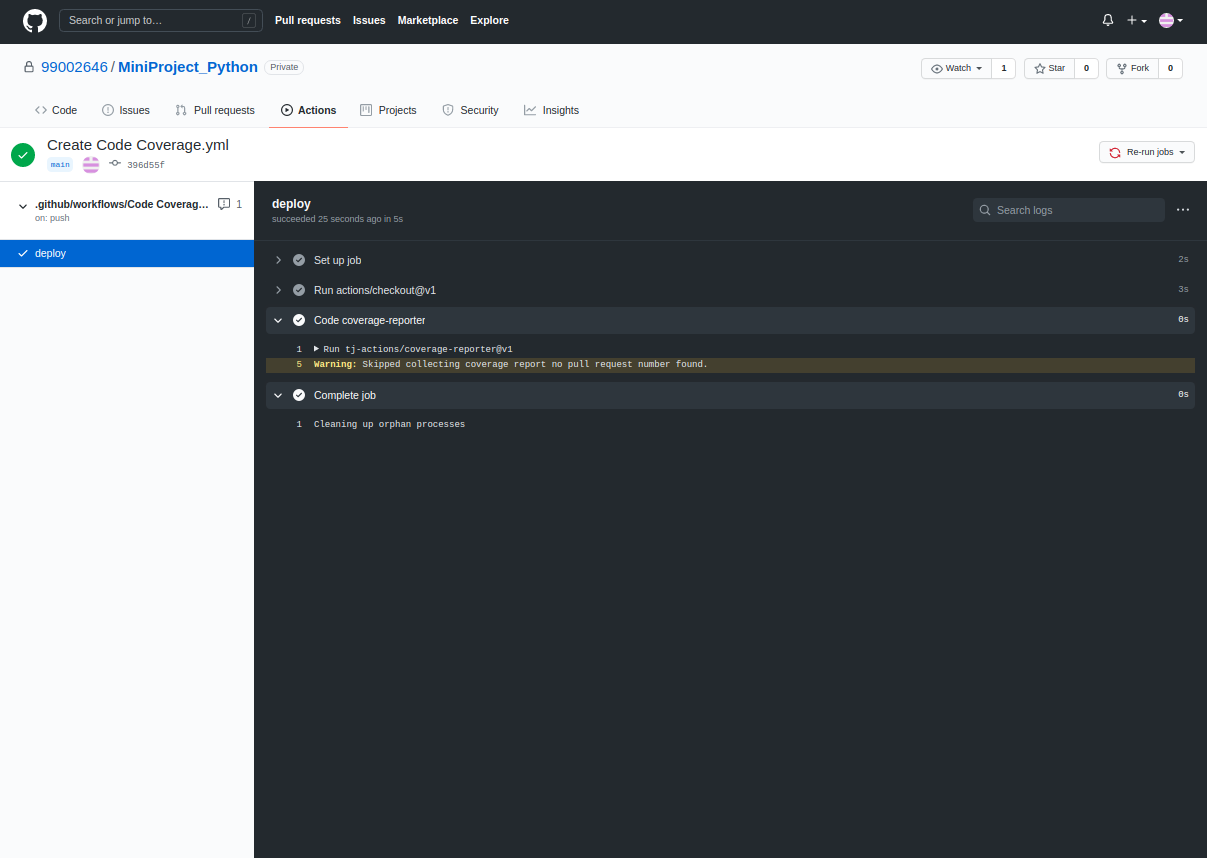
Figure 18 : Git Inspector

#### Build



Figure 19 : Bulid

#### Code quality and Issues or Bug Tracking



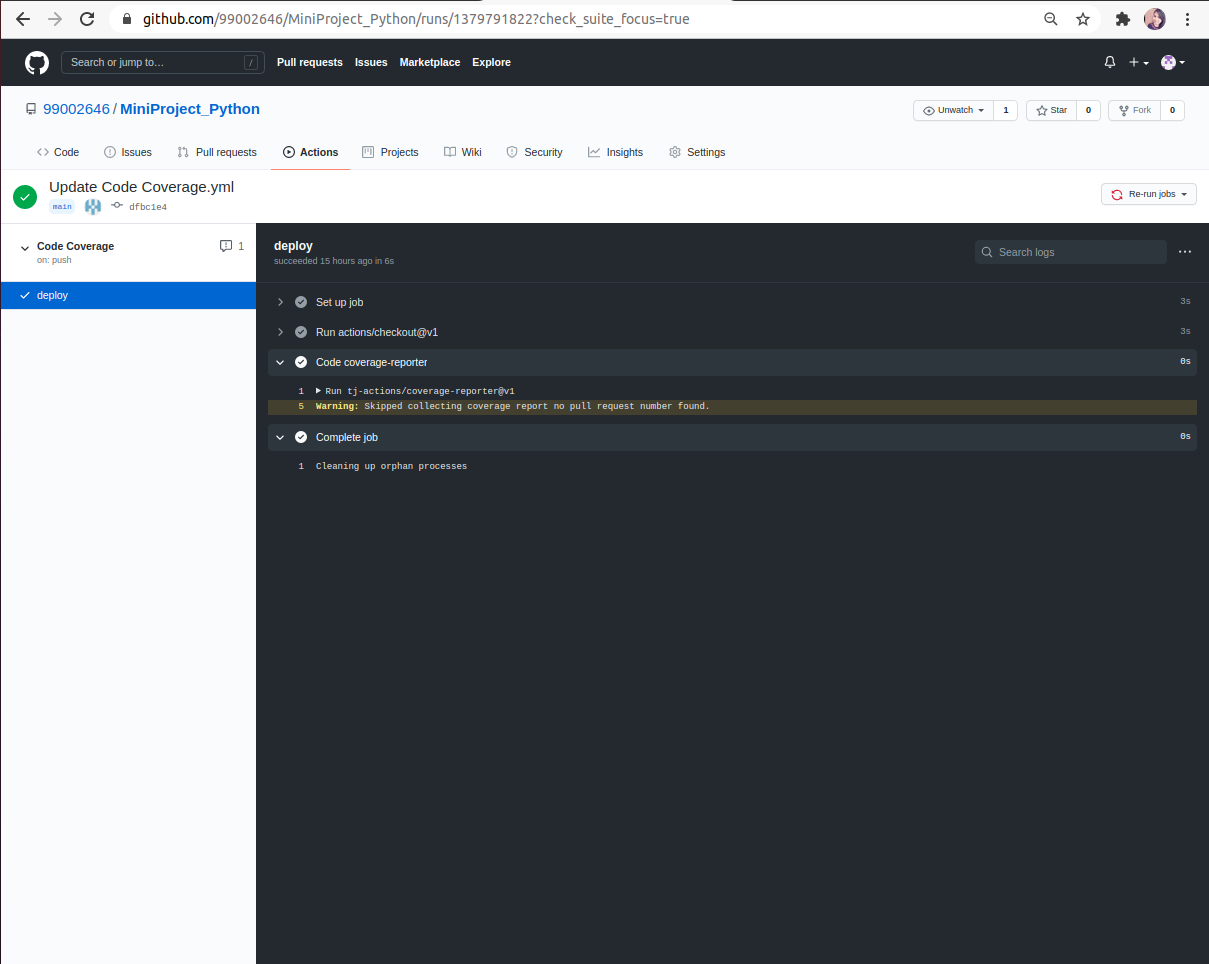


Figure 20 : Code Coverage

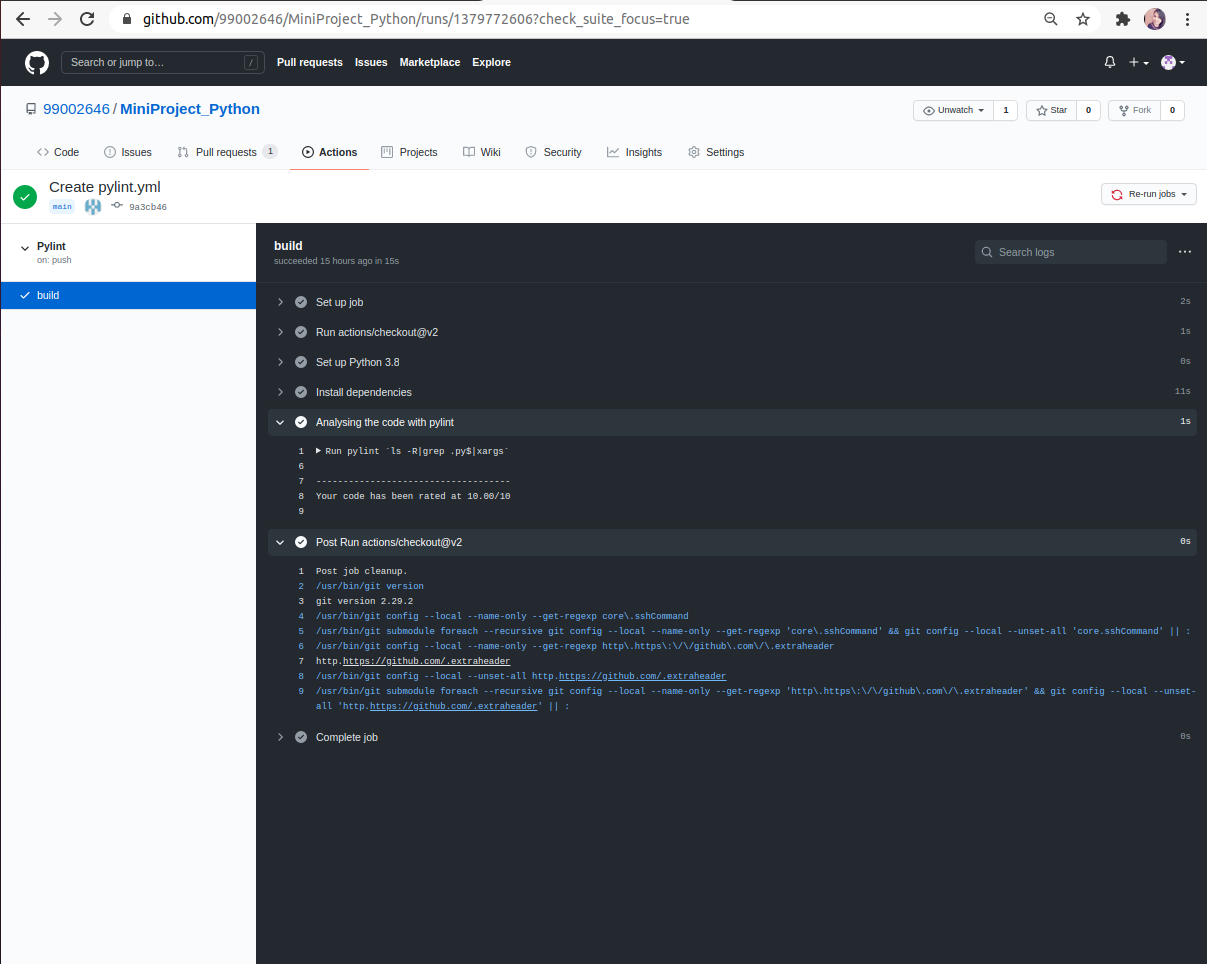


Figure 21 : Pylint

## **5.** **Individual Contribution & Highlights**

### 5.1 Challenges faced and how were they overcome

* While importing the data from the data set we have faced some of the problems with the particular row and columns.
* We have overcome that problem with trail and error.

### 5.2 Future Scope

* This project can be implemented further by adding some other features like displaying the players personal details if we give their Name or ID..

## **6. References**

[1] https://www.geeksforgeeks.org/introduction-to-linux-operating-system/

[2] https://www.w3schools.com/python/

[3] https://en.cppreference.com/w/

[4] <https://www.cplusplus.com/reference/>