****

**Personnel Management System**

**Sprint Implementation-2**

**Index**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Title** | **Page no.** |
| **1** | **Introduction** | **3** |
| 1.1 | Project scope | 3 |
| 1.2 | Purpose | 3 |
| 1.3 | Intended audience | 3 |
| **2** | **Design Overview** | **5** |
| 2.1 | **Data Flow Diagram** | 5 |
| 2.1.1 | Low level module interaction diagram(LLD) | 5 |
| 2.1.2 | High Level Diagram(HLD) | 5 |
| 2.2 | **Flowcharts** | 6 |
| 2.2.1 | Flowchart for Main menu | 6 |
| 2.2.2 | Flowchart of employee database | 7 |
| 2.2.3 | Flowchart to check employee with 3 projects | 7 |
| 2.3 | ER Diagram | 8 |
| 2.4 | Sequence Diagram | 9 |
| 2.5 | Use Case Diagram | 10 |
| 2.6 | Class Diagram | 11 |
| 2.7 | User interface | 11 |
| 2.8 | Pseudocode for complex functions | 12 |
| 2.9 | Validations | 14 |
| **3** | **System Architecture** | 15 |
| **3.1** | **Classes** | 15 |
| 3.1.1 | EMPLOYEE Class | 15 |
| 3.1.2 | EMPLOYEE\_PROJECT Class | 15 |
| 3.1.3 | SALARY Class | 15 |
| 3.1.4 | PROJECT Class | 16 |
| **3.2** | **Functions** | 16 |
| **3.2.1** | **Employee Details** | 16 |
| 3.2.1.1 | Add\_employee() | 16 |
| 3.2.1.2 | Edit\_employee() | 16 |
| 3.2.1.3 | View\_employee() | 16 |
| 3.2.1.4 | Delete\_employee() | 17 |
| **3.2.2** | **Project Details** | 17 |
| 3.2.2.1 | Add\_project() | 17 |
| 3.2.2.2 | Edit\_project() | 17 |
| 3.2.2.3 | View\_project() | 17 |
| 3.2.2.4 | Delete\_project() | 17 |
| 3.2.2.5 | Query() | 17 |
| 3.2.2.6 | Pay\_report() | 17 |
| 3.2.2.7 | Calc\_fi\_salary() | 17 |
| 3.2.2.8 | Emp\_project\_report() | 18 |
| 3.2.2.9 | Emp\_list\_with\_3\_project() | 18 |
| **4** | **Tools Report** | **18** |
| 4.1 | Gcov Report | 18 |
| 4.2 | Gprof Report | 26 |
| 4.3 | Valgrind Report | 30 |
| 4.4 | Splint Report | 31 |
| 4.5 | CPP unit testing Report | 32 |

**1.Introduction**

The aim of this project is to develop a Personnel Management System using C++ through a menu-driven program that allows us to add, update, delete records and project details of an employee working in an Marspro organization. The project personnel management system stores employee ID, name, department, project details and salary of the employee. Initially, it has no data. Thus, we have to add employee records choosing appropriate options in this program.

After adding the data we can perform various operations such as calculating the salary, generating various reports such as pay report, query report and employee report. This system does not allow wrong inputs as we have provided validations for various inputs such as employee name, contact number, salary, employee id, project id, department.

**1.1 Project scope**

The project aims at the development of the online personnel management system   to assist Marspro organization to keep the records of the employee along with the project they are working on and to calculate their pay stub in a more robust and efficient manner. Along with that it also provides the list of employees who are working on more than one project and able to give the details of working hours on a project for an employee.

**1.2 Purpose**

The purpose of this project is to manage the employee and provide project details as per the requirement from the organisation. With the help of this project, we can track all information related to Employee(add employee, edit employee, view employee and delete employee (if needed)), Project details on which employee is working (add project, edit project, delete project(if needed),query an employee, giving the pay stub according to the working hours of the employee on the particular project, employee should be able to view the number of hours he spent on a particular project and the application should also provide the report for the employees who are working on more than one project.

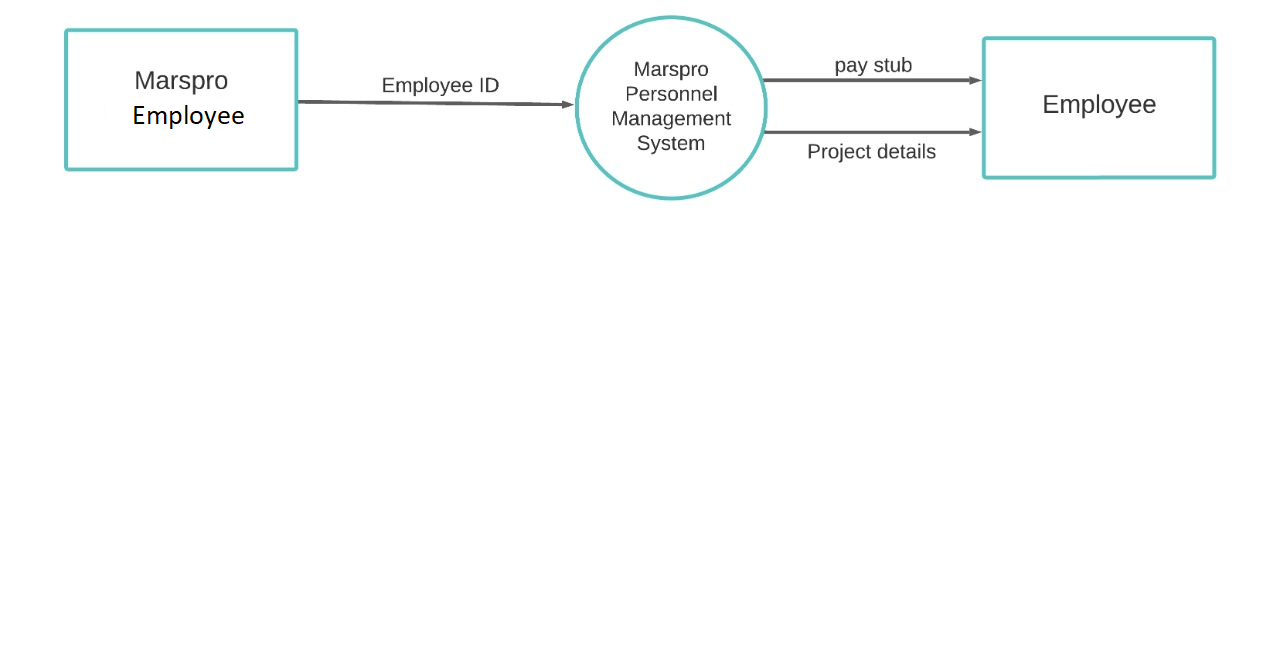
**1.3 Intended Audience**

The target audience set for this project can be identified as an Marspro organizer who is handling the employee database and automate the process of managing the details which includes maintaining records of the employees and their projects.

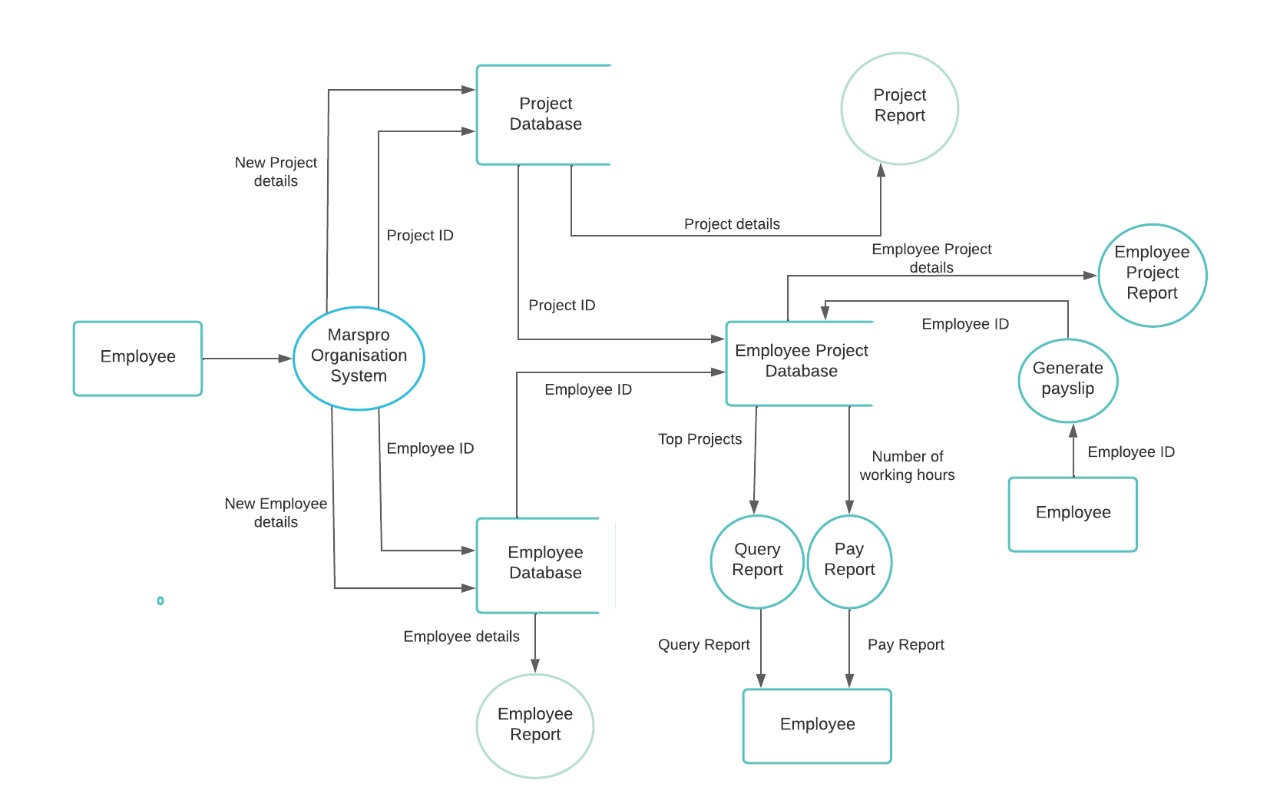
**2. Design Overview**

**2.1 Data flow Diagram**

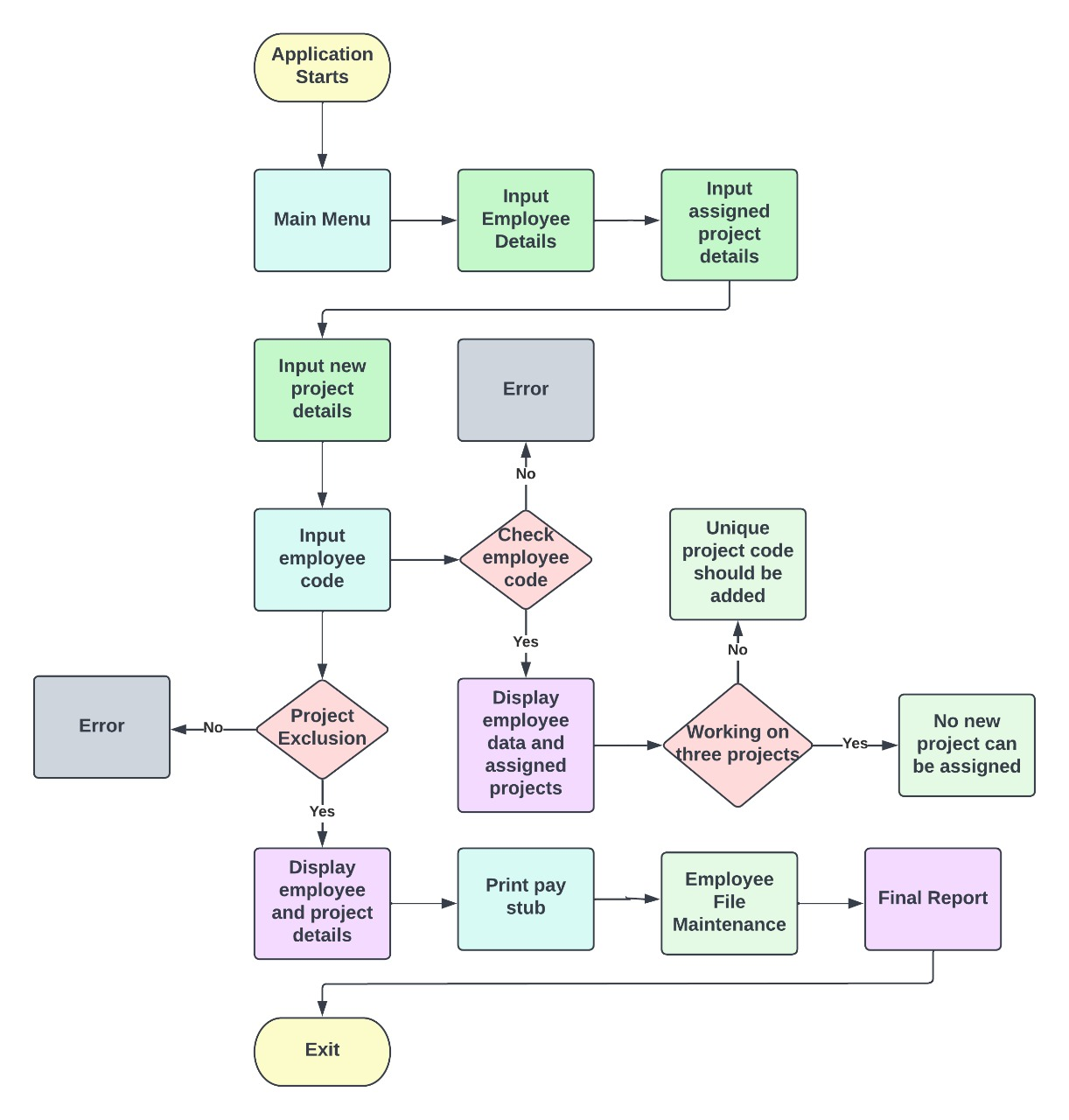
**2.1.1 Low Level Module Interaction Diagram(LLD)**

****

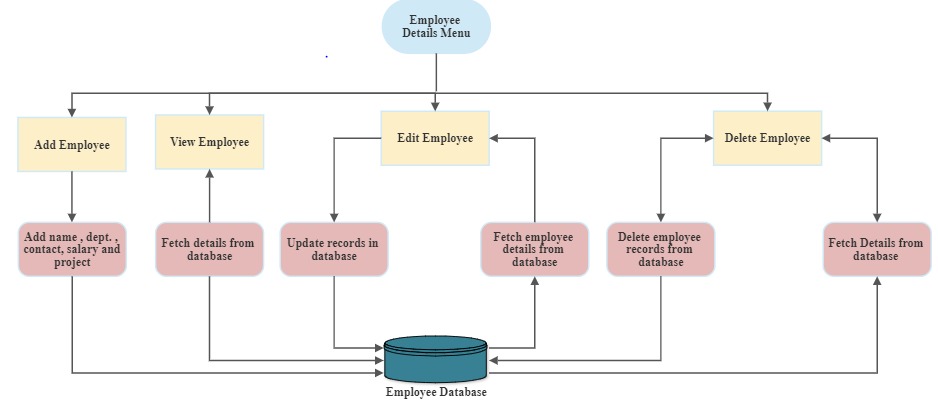
**2.1.2 High Level Diagram (HLD)**

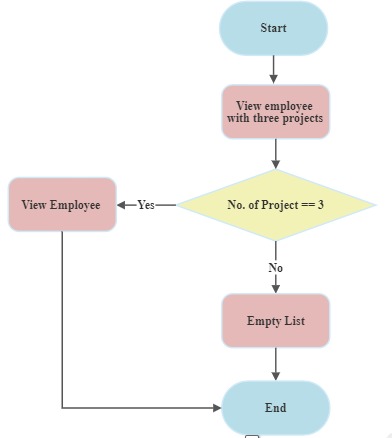


**2.2 Flowcharts**

****

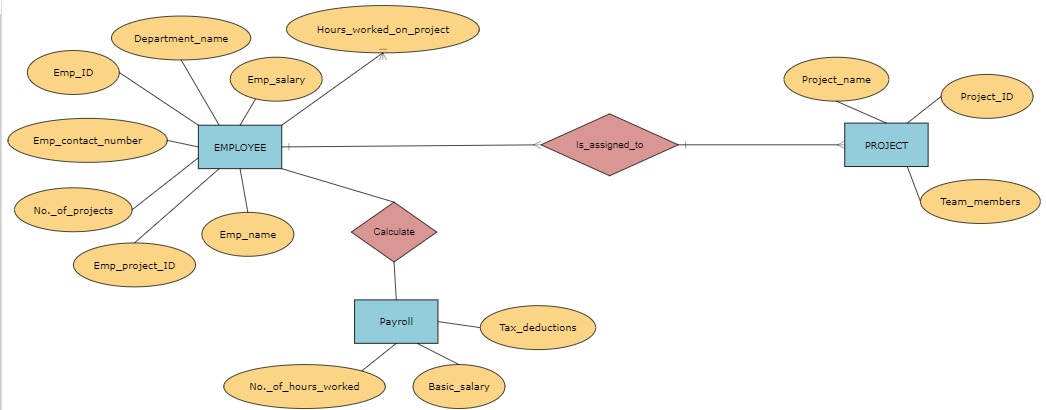
**Fig 2.2.1: Flowchart of Main menu**

**Fig 2.2.1: Flowchart of Employee Database**

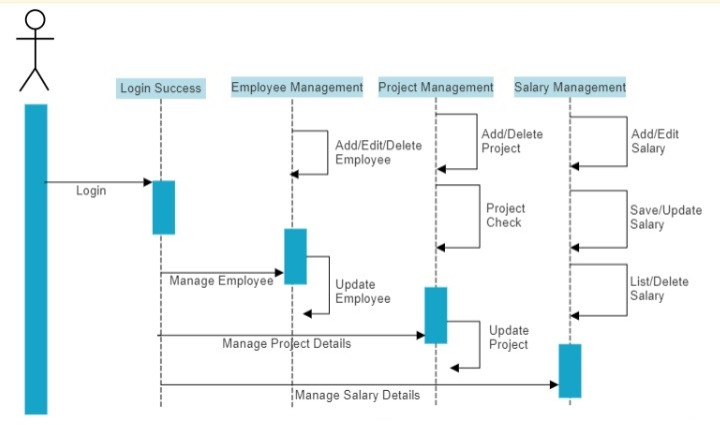


**Fig 2.2.3: Flowchart to check employee with 3 projects**

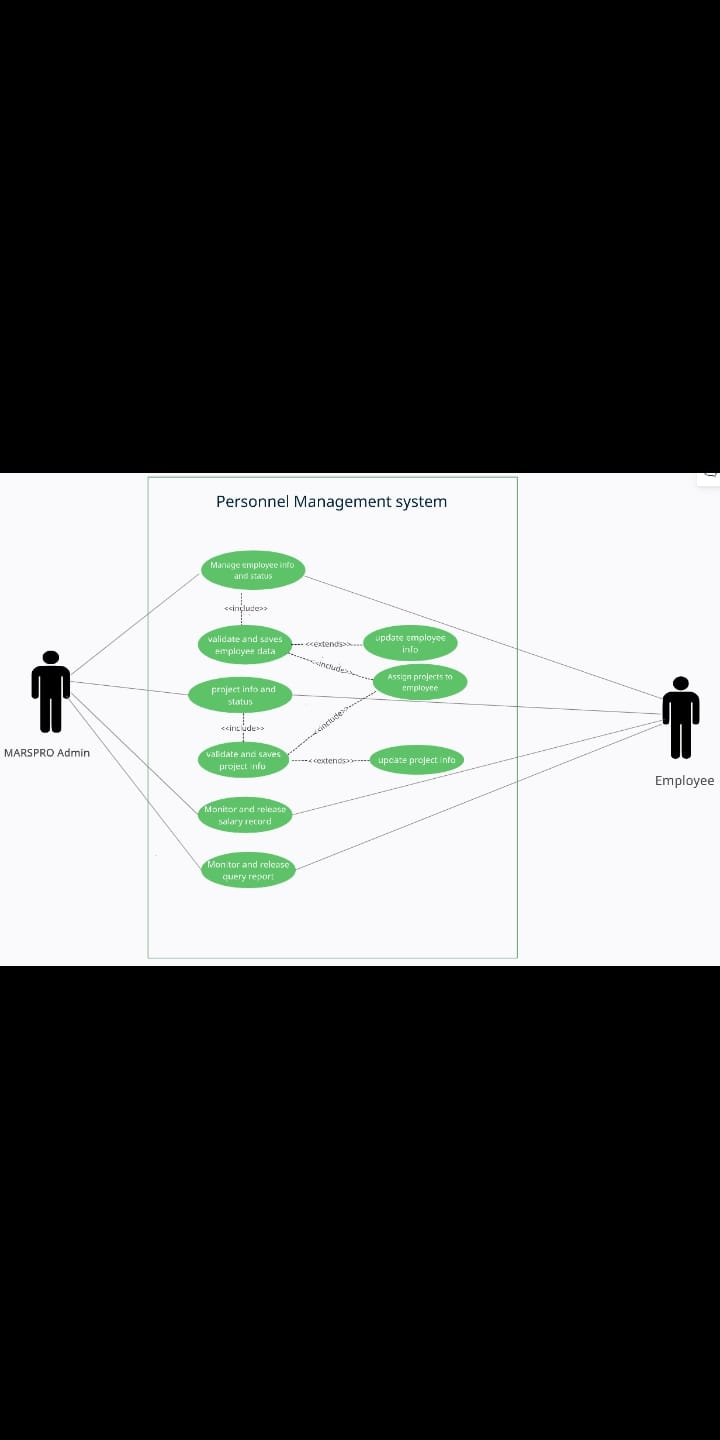
**2.3 ER Diagram**

****

**2.4 Sequence Diagram**

****

**2.5 Use Case Diagram**

****

**2.6 Class Diagram**

**Diagram

Description automatically generated**

**2.7 User Interface**

The Personnel Management System provides an option to employees by generating the reports of projects and pay stub of each employee and keeping their records.

**2.8 Pseudocode**

**1.Query:**

This function displays all the projects which have highest number of team members.

Void PROJECT:: project\_query()

{

if(pr\_list is empty)

{

Checks whether the project list Is empty or not. If It empty then the system returns Project list is empty.

}

Std::list<PROJECT>::Iterator st=pr\_list.begin();

Highest\_no will be equal to st-> team\_mem.

for(PROJECT st:pr\_list)

{

If(highest\_no is less than or equal to st.team\_mem)

{

highest\_no equal to st.team\_mem

}

}

Pr\_id, name , team\_mem are displayed on the output screen.

For(PROJECT st:pr\_list)

{

If(st.team\_mem is equal to highest\_no)

{

It will display all the projects with highest number of team members.

}

}

}

**2. Pay report**

This function is used to view pay report of all the employees working in the Marspro Organisation.

void SALARY::salary\_view()

for (std::list<SALARY>::iterator it = empsal\_list.begin(); it != empsal\_list.end(); ++it)

{

if(it->project id 1==0)

it->pr\_bonus=0;

else

{

it->pr\_bonus =( 45\*working hours per day\*working days per month)+(project 1 working hours\*10)+(project 2 working hours\*10)+( project 3 working hours\*10)

}

it->epf = (salary\*8.5)/100;

it->income\_tax = (((salaryl\*12)\*15)/100)/12;

it->fi\_salary = (sal+bonus) - it->income\_tax - it->epf;

Here it displays emp\_id, emp\_name, fi\_salary,epf,income tax of all the employees on the screen.

}

**2.9 Validations**

* As we are auto generating the employee id, project id there is no chance of duplicate id.
* For employee name, it should be in range by checking length of it and it consists of only alphabets.
* For department, it should be either IT or HR or SALES or MARKETING.
* For contact number, it should consist of exact 10 digits and no alphabets.
* For basic salary, it should be in the range of 10,000 to 200000.
* Number of projects each employee should work on is maximum three.
* Everytime when we add project for the employee, it will cross check with the project list whether that project id exists or not.

**3. System Architecture**

**3.1 Classes**

**3.1.1 EMPLOYEE Class**

* **Data Members (Protected):**
  + - * emp\_id
      * emp\_name
      * dept
      * salary
      * contact
* **Member Functions (Public):**
  + getdata()

**3.1.2 EMPLOYEE\_PROJECT Class**

* **Data Members (Protected):**
* project\_counter
* vector<int>project\_id
* vector<int>no\_hrs

**3.1.3 SALARY Class**

* **Data Members (Protected):**
  + epf
  + prof\_tax
  + income\_tax
  + fi\_salary
  + pr\_bonus
* **Member Functions**
* getdata1()
* view1()
* calc\_pr\_bonus()
* calc\_fi\_salary()
* editdata()
* deldata()
* salary\_view()
* emp\_proj\_details()
* emp\_list\_with\_3\_projects()

**3.1.4 PROJECT Class**

* **Data Members**
* proj\_id(public)
* pr\_name(private)
* team\_mem(private)
* **Member functions**
* getdata()
* view()
* edit()
* del\_project()
* project\_query()

**3.2 Functions**

**3.2.1Employee Details**

**3.2.1.1 PMS\_01 => add\_employee ()**

This feature adds the employee record to the file. It asks for employee\_id, employee\_name, department, salary, contact number whose record is to be created.

**3.2.1.2 PMS\_02 => edit\_employee ()**

This feature edits the employee record in the file. It asks for the employee\_id and asks the user to enter new employee\_name, department , salary, contact number and the list of projects done by the new employee by adding project\_id, project\_name, number of hours worked . Whose record is to be edited Upon successful editing of employee details into the employee database it will return to the main menu.

**3.2.1.3 PMS\_03 =>view\_employee ()**

This feature displays all the employee details present in the employee database. It displays employee\_id, employee\_name, department, contact number and list of projects done by the employee with project\_id, project name, number of hours worked.

**3.2.1.4 PMS\_04 => delete\_employee ()**

This feature deletes the employee details from the employee database. The user needs to provide the employee\_id to be deleted from the employee database. Upon successful deletion of employee details from the employee database it will return to the main menu.

**3.2.2 Project Details**

**3.2.2.1 PMS\_05 => add\_project()**

This feature adds the new project details into the project database. It asks for project\_id, project name and number hours to be worked on that project. Upon successful addition of a new project details into the project database then it returns to the main menu.

**3.2.2.2 PMS\_06 => edit\_project ()**

This feature edits the project's details from the project database. It asks for project\_id and asks the user to enter a new project name and number of hours to be extended to the project. Upon successful editing of project details into the project database then it returns to the main menu.

**3.2.2.3 PMS\_07 => view\_project ()**

This feature displays the project details in the project database. It displays project\_id, project name, number of hours to be worked on that project.

**3.2.2.4 PMS\_08 => delete\_project ()**

This feature deletes the project details from the project database. The user needs to provide the project\_id to be deleted from the project database. Upon successful deletion of project details from the employee database it will return to the main menu.

**3.2.2.5 PMS\_09 => query()**

This feature presents a query on top projects should show the project on which the highest number of employees are working. If the number is more than one, then all those projects should be displayed.

**3.2.2.6 PMS\_10 => pay\_report ()**

This function calls **salary\_view()** member function of SALARY class which displays the pay report for all the employees (id, name, basic salary, bonus, deduction and final salary) according to the criteria set by the Marspro organization.

**3.2.2.7 PMS\_10 => calc\_fi\_salary ()**

This feature generates the pay report for a particular employee (id, name, basic salary, bonus, deduction and final salary) according to the criteria set by the Marspro organization

**3.2.2.8 PMS\_11 => emp\_project report()**

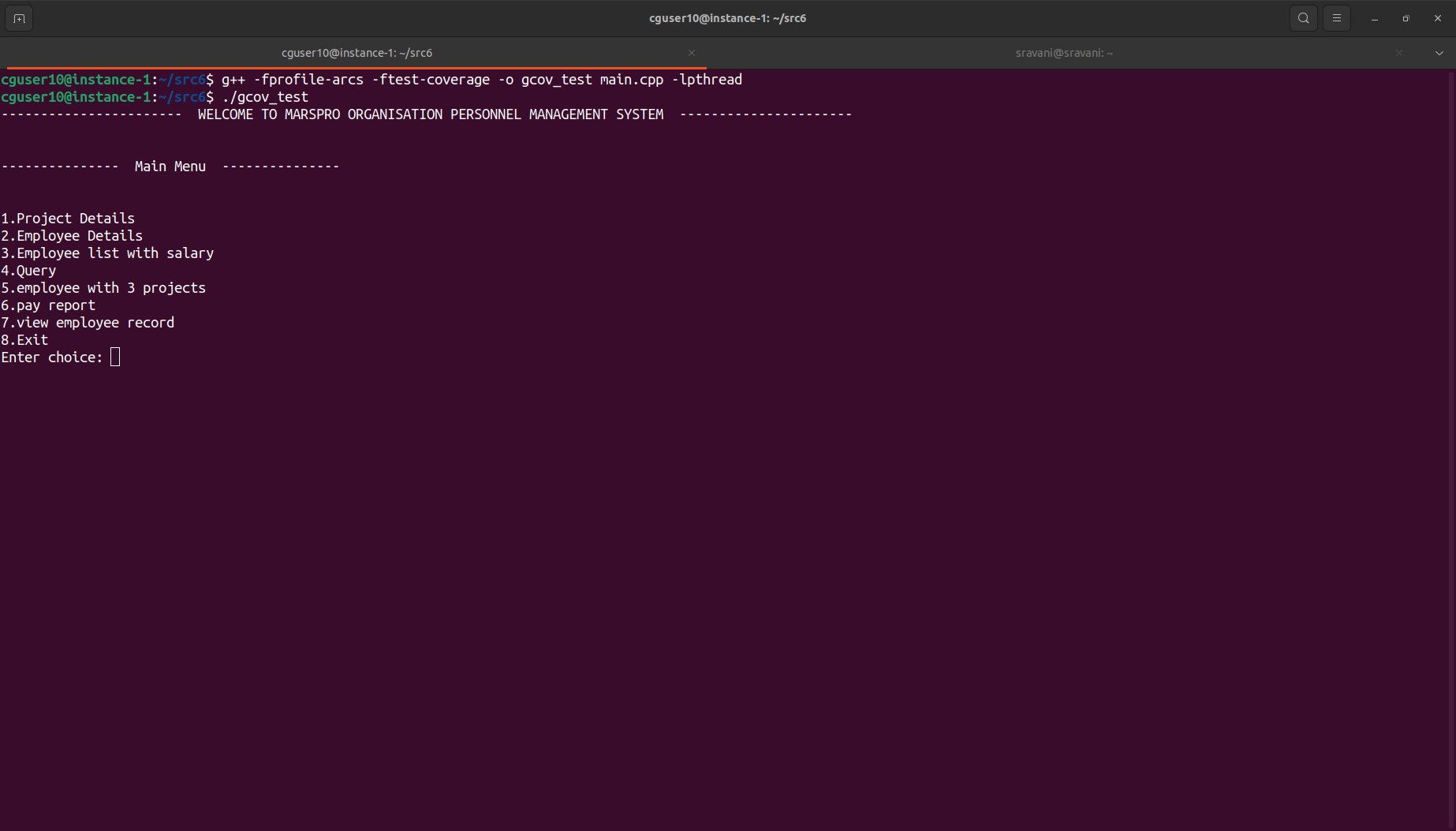
This feature allows an employee to view how many hours he has already spent on which project through a menu option. This option requires him to enter his employee\_id, which is validated, and the details are shown.

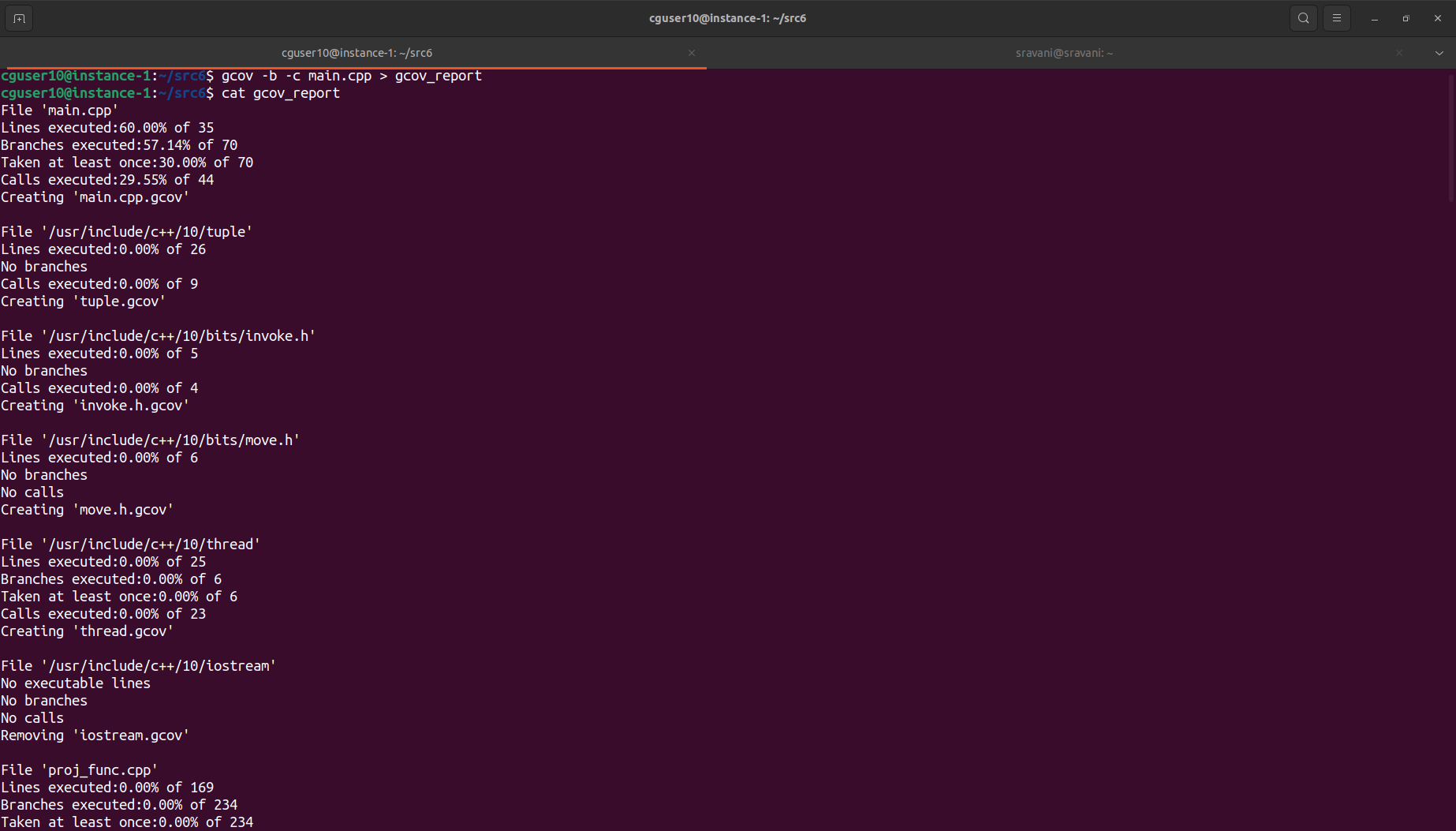
**3.2.2.9 PMS\_12 => emp\_list\_with\_3\_projects ()**

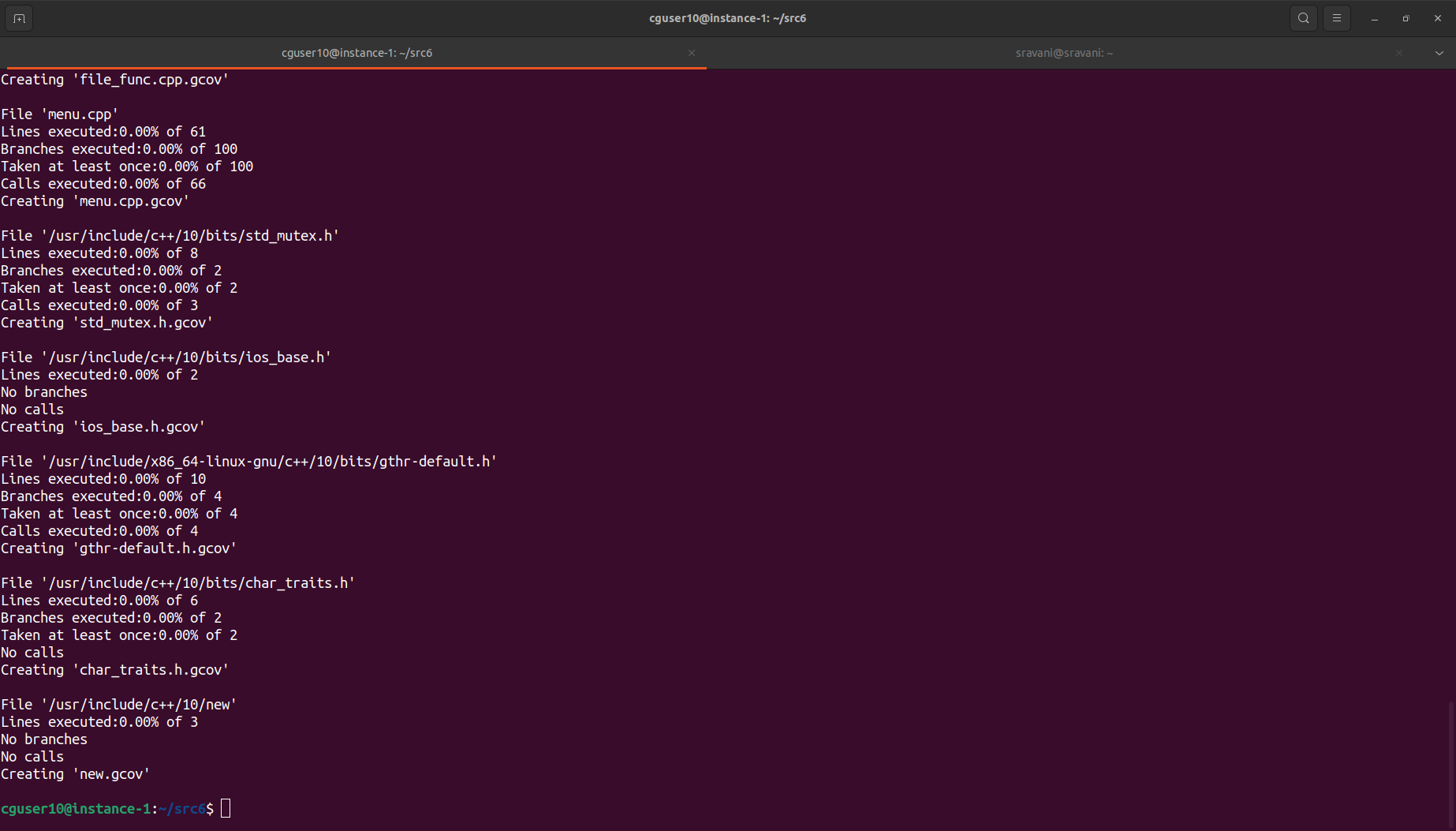
This feature shows the details of all employees who are working on 3 projects at time.

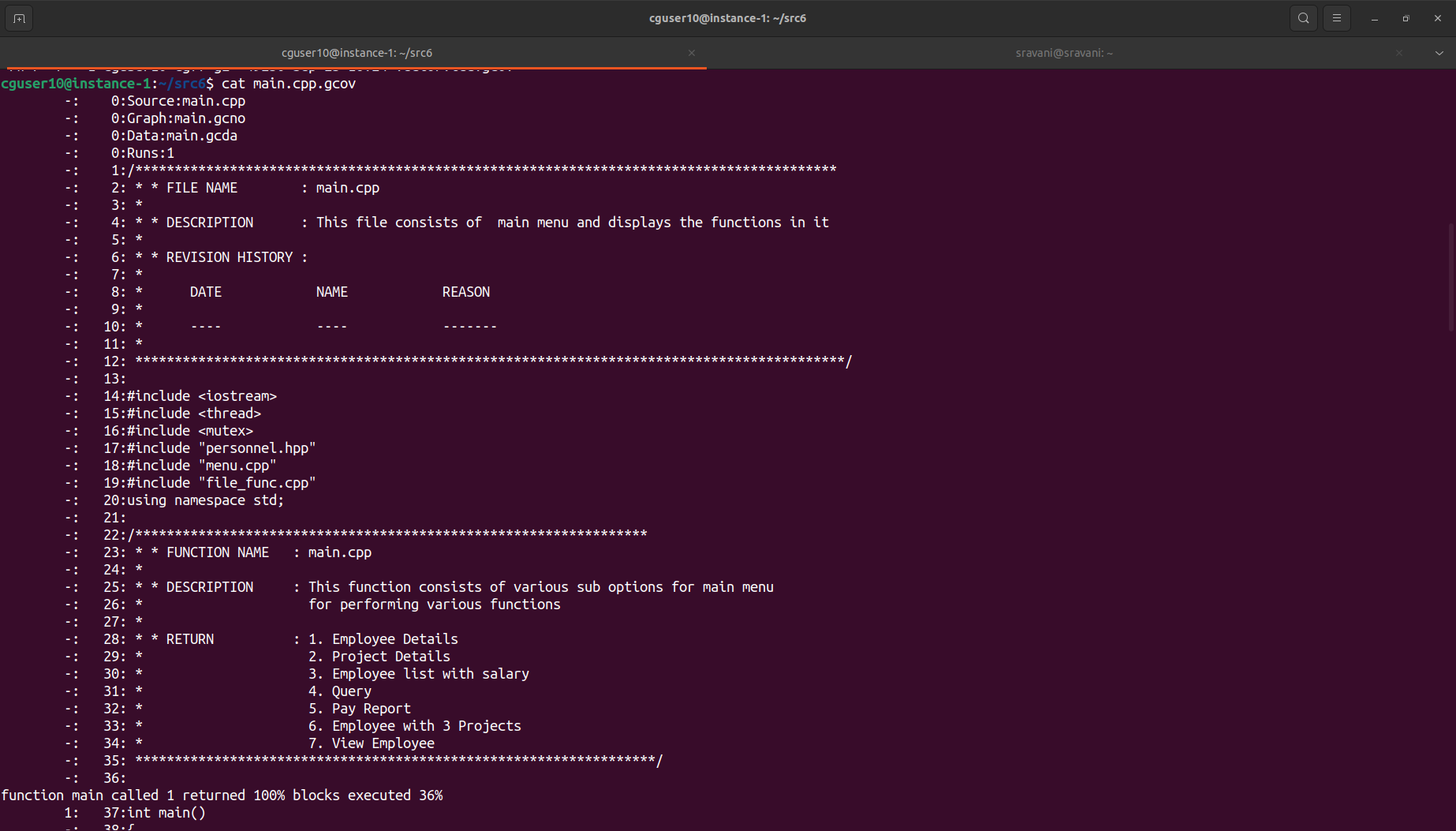
**4. Tools Report**

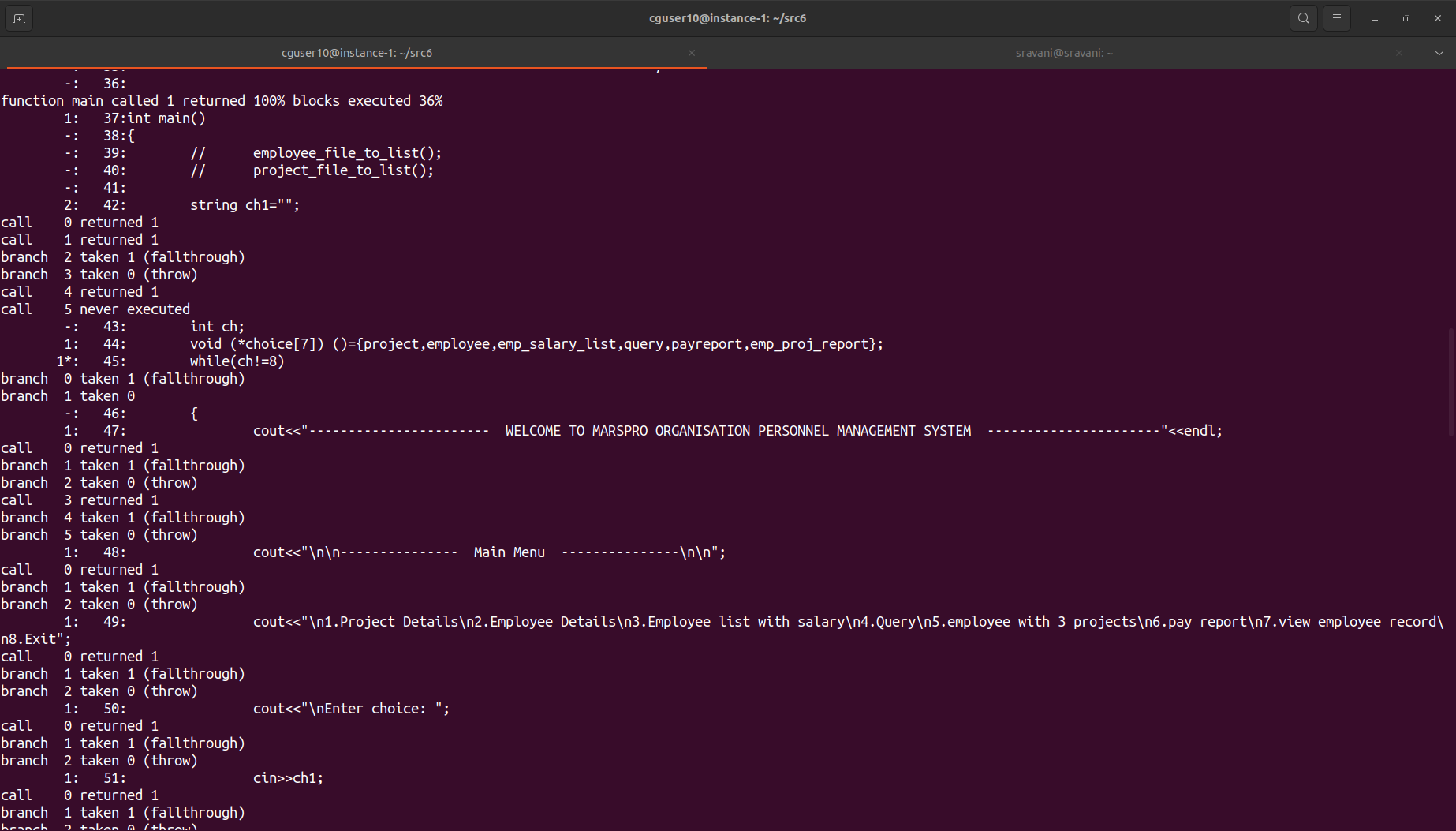
**4.1 gcov Report**

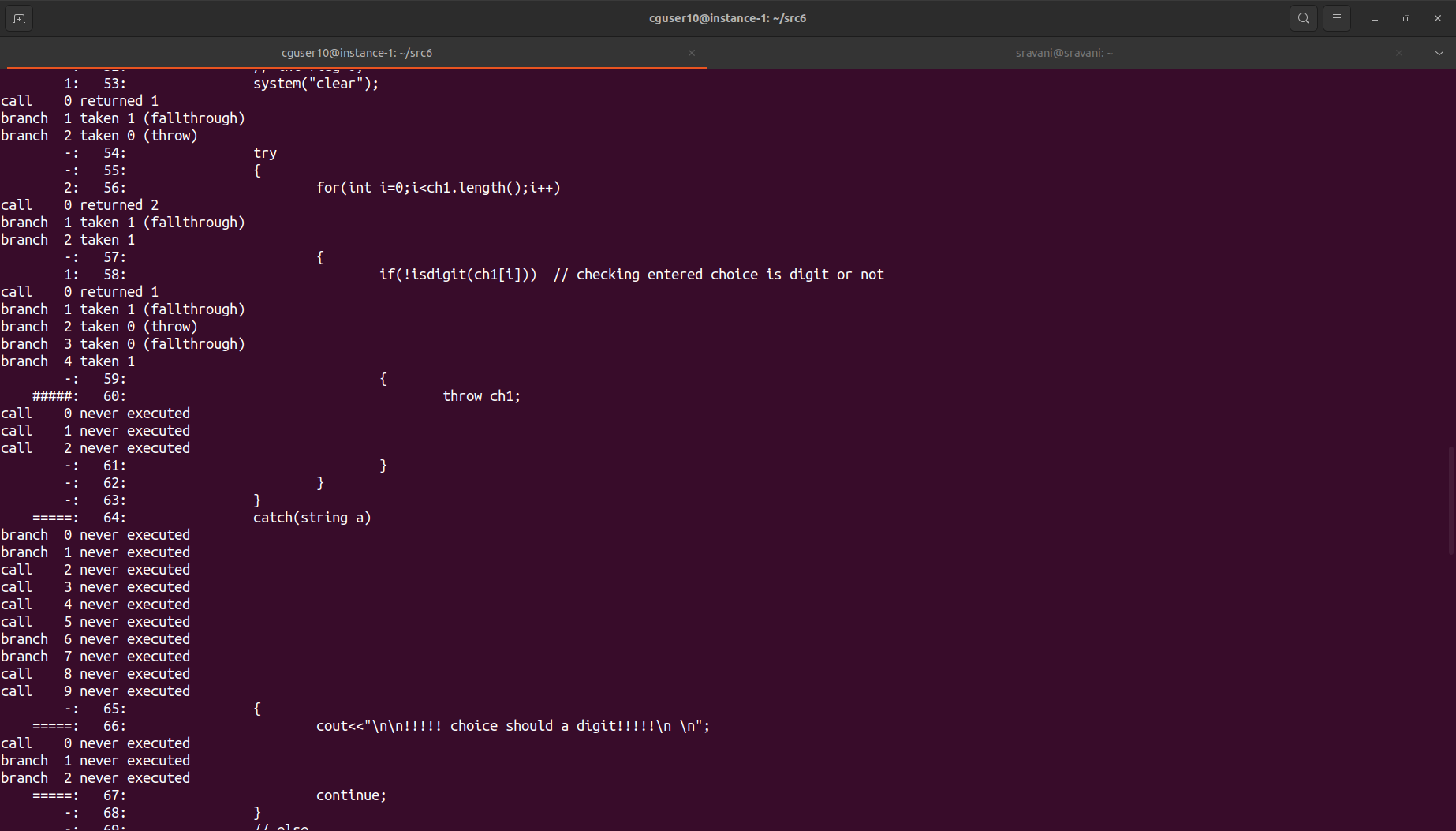


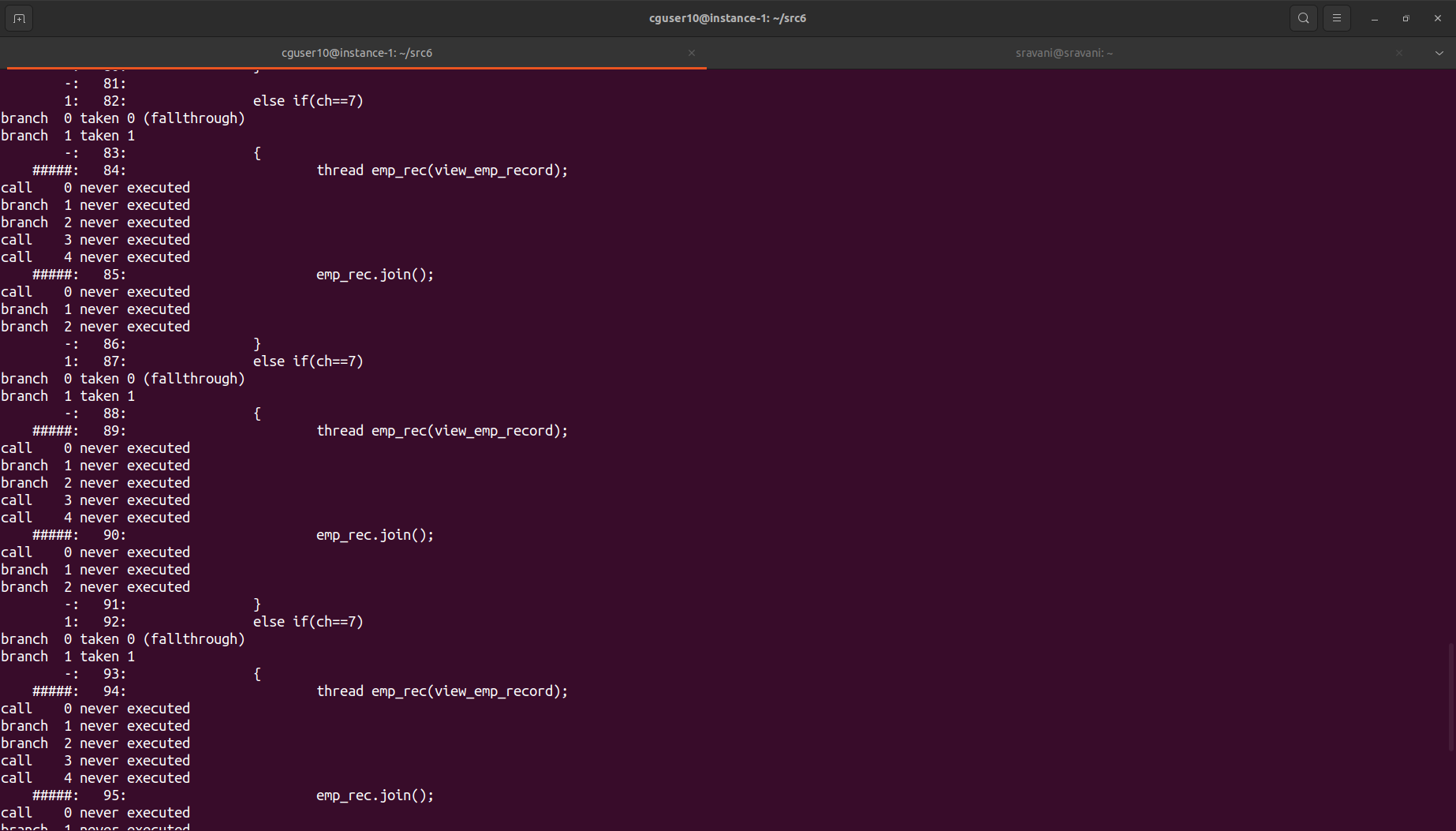


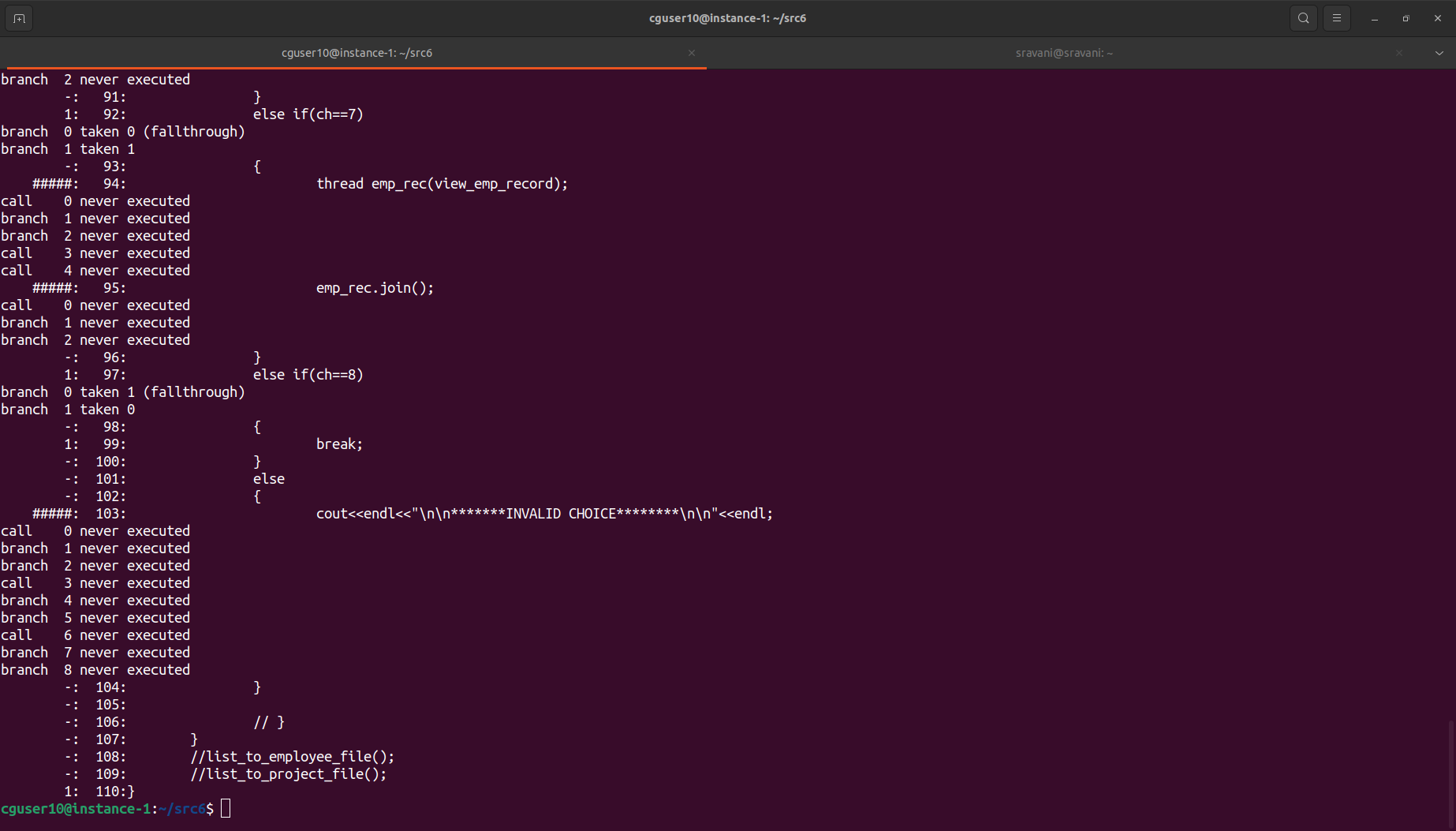




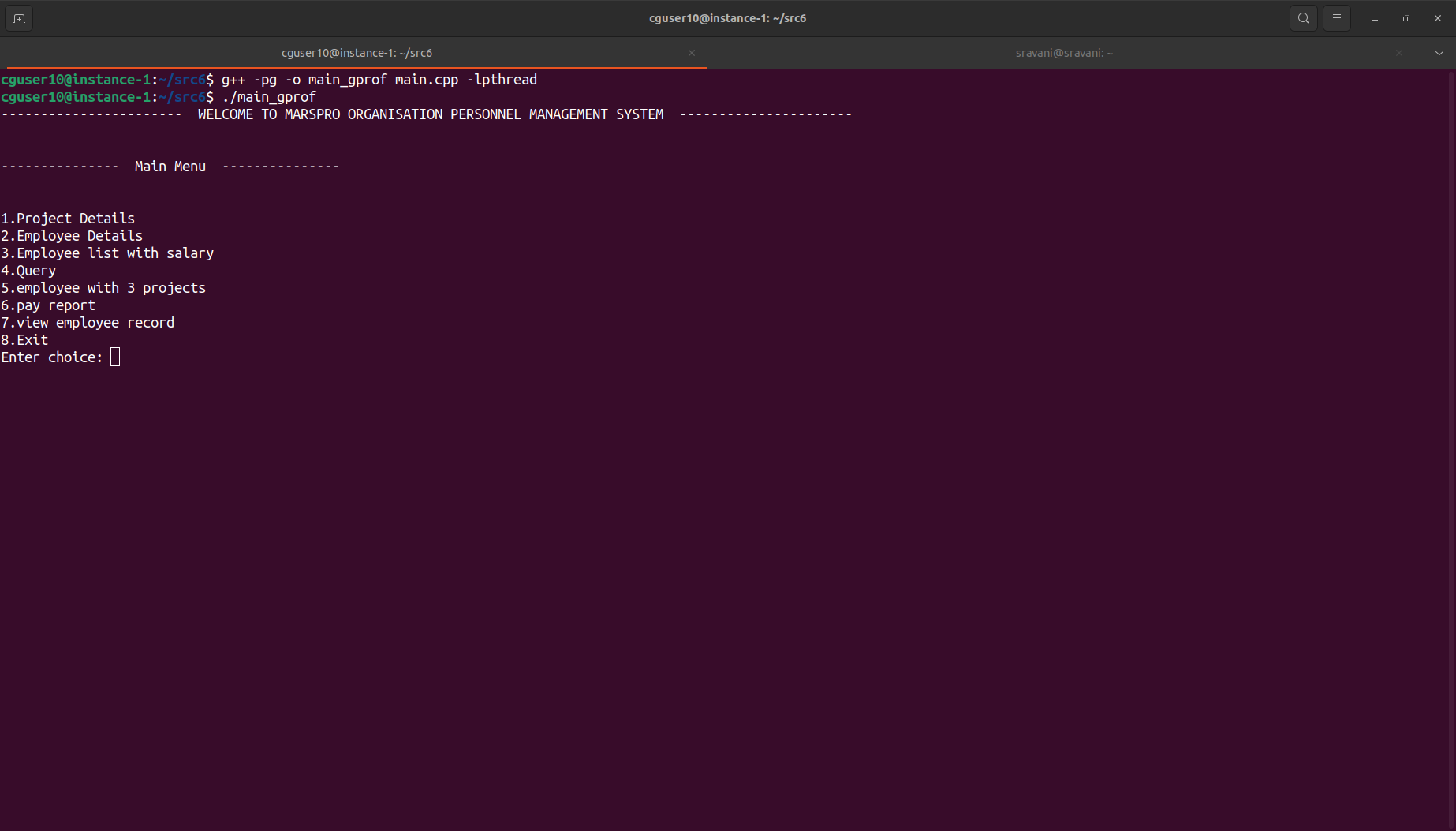


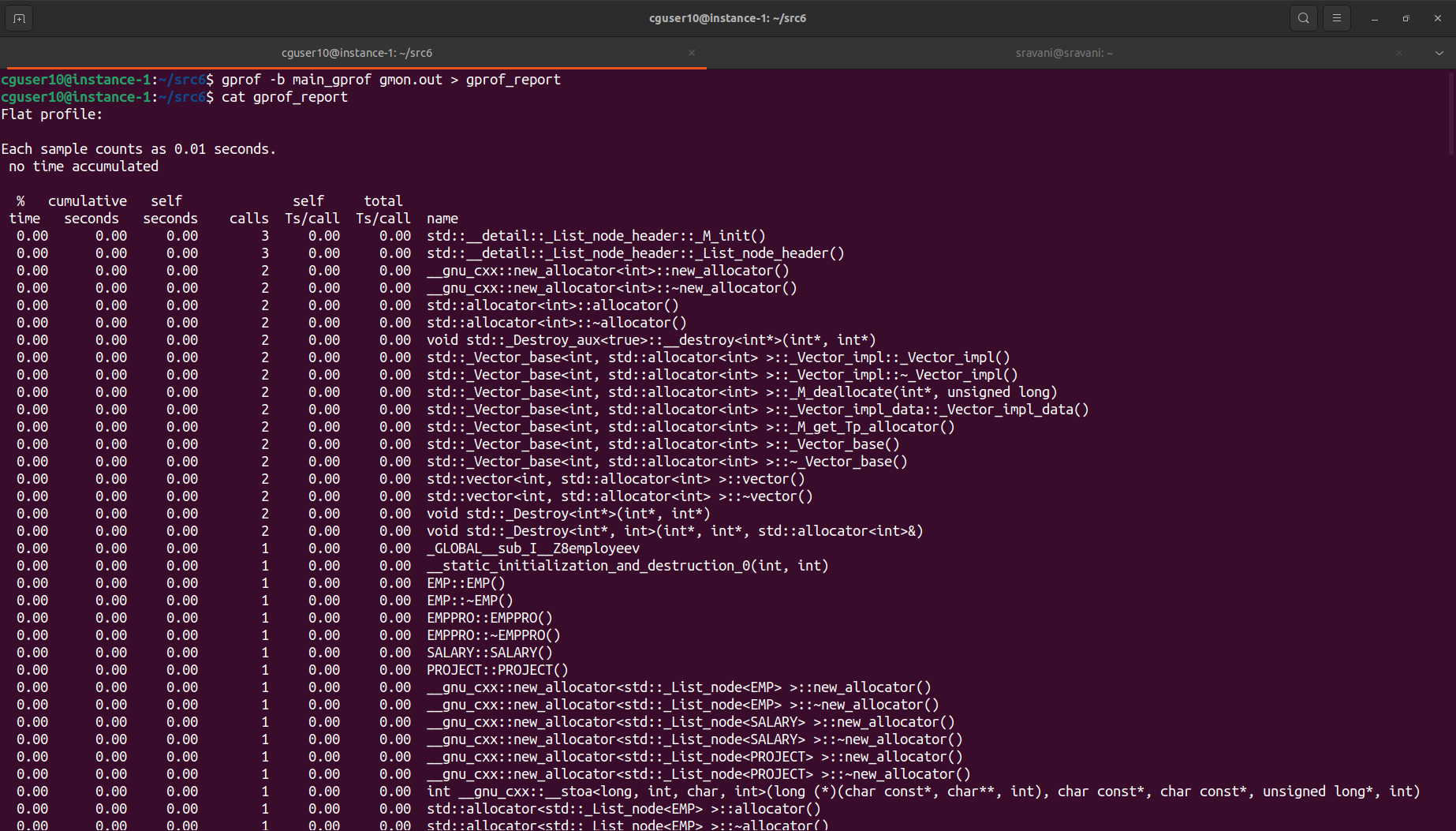


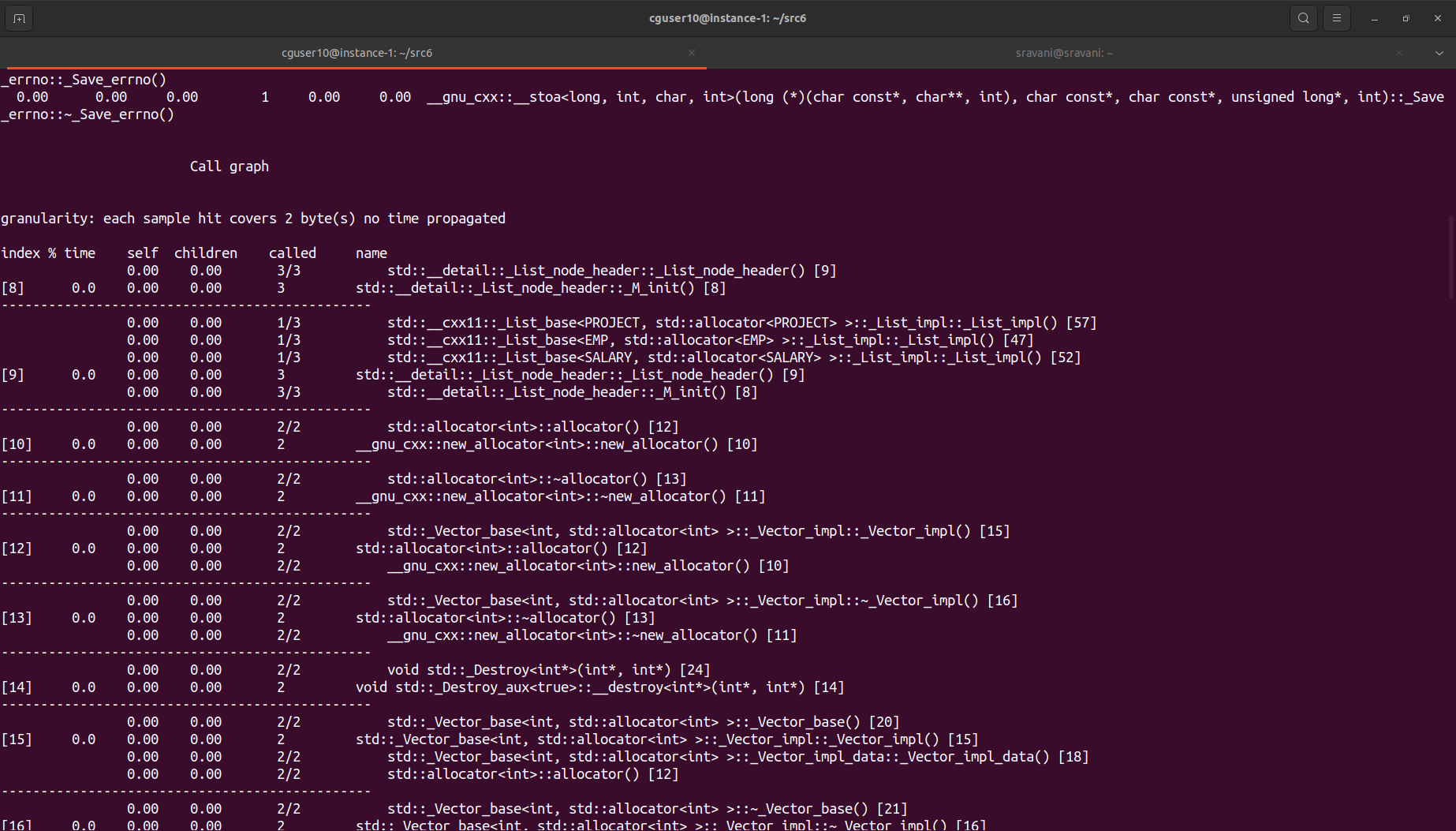


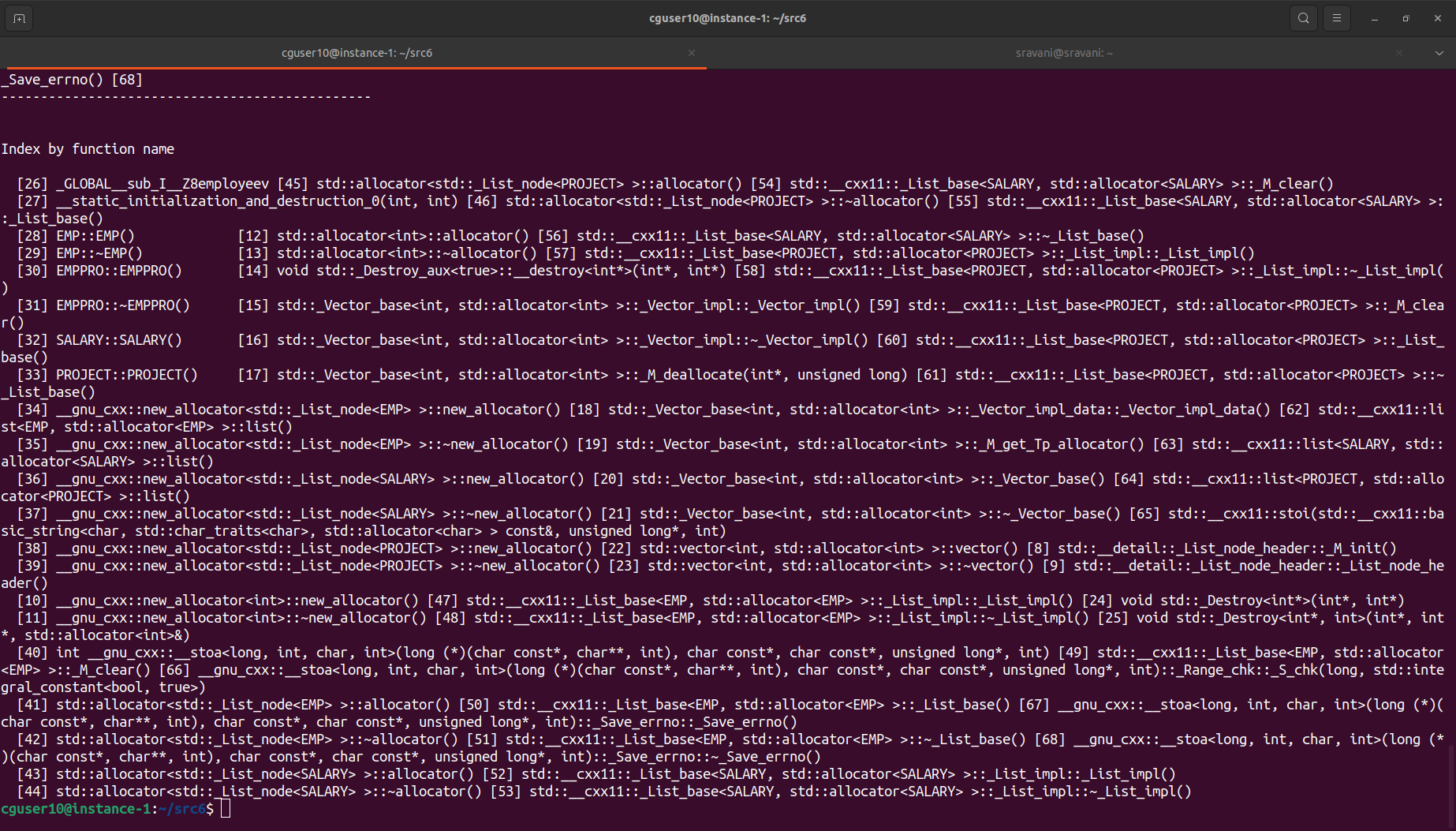


**4.2 gprof Report**

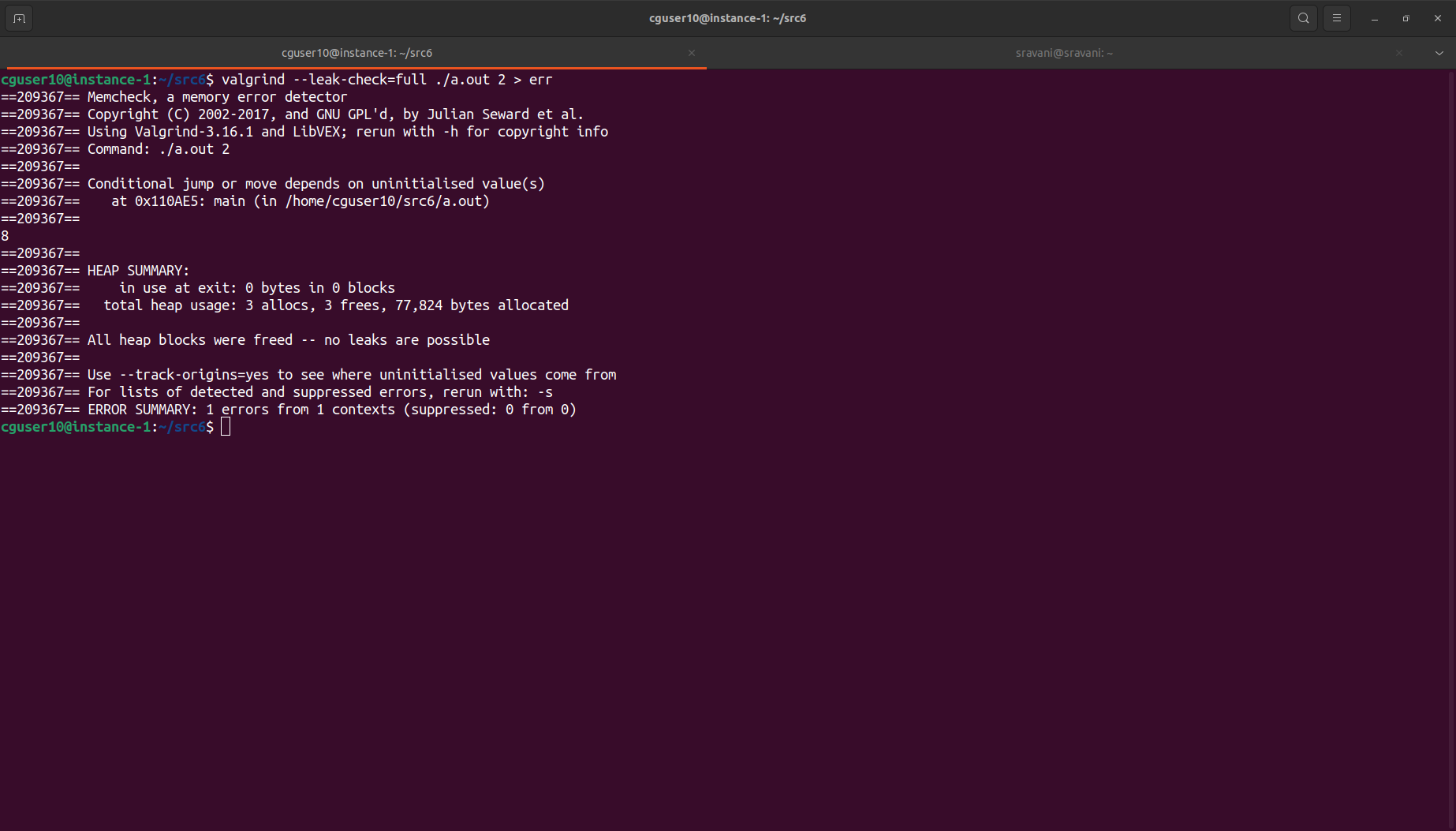


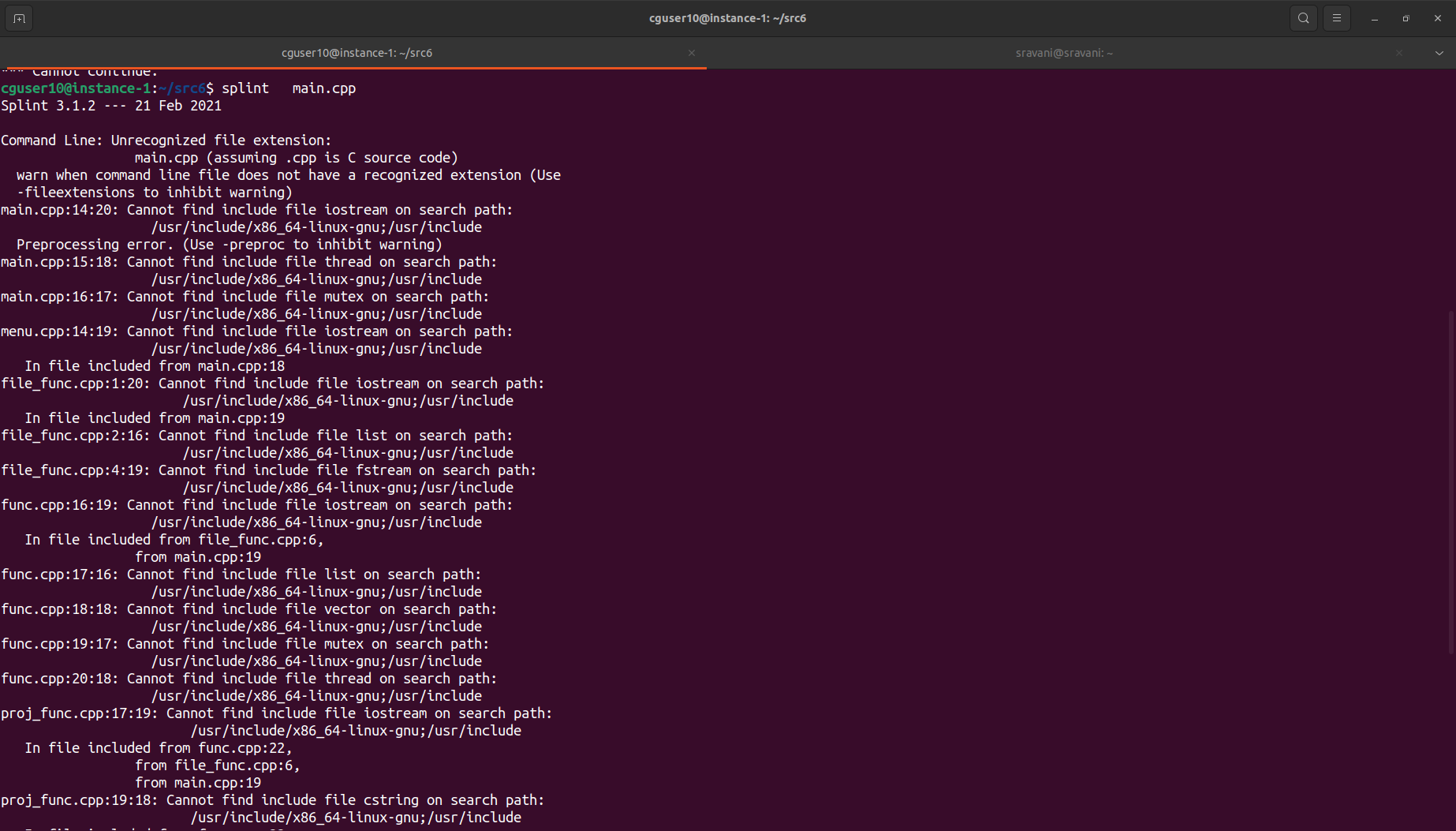






**4.3 Valgrind Report**



**4.4 Splint Report**

**4.5 CPP unit Testing Report**

