Final Report OS (V2)

Team No:

6

Team Members:

Harshita Kapa	(cs19b021)
Rajendra Kumar	(cs19b034)
Saurav Gwalia	(cs19b035)
Talha Shamim	(cs19b050)

Notable Features:

- In the system:
    - User creation done in program No switching to terminals
    - Automatic file generation
    - Automatic file removal
    - Automatic permission setting up
    - Terminal control to user
    - Editing via terminal
    - Rankings of students
    - Average of students upto 1 decimal place
    - If user not in system cross on his place

- In the Code:
    - Well structured folder and files
    - Use of makefile
    - Comments in code
    - Function and abstractions used

Tools Used:

- FreeBSD
- Windows + wsl
- Ubuntu
- G++ compiler
- Vs code editor
- Github
- Git bash
- Git desktop
- Ncurses

- Xorg
- Slim
- Xfce

Dependencies:

- Ncurses
- G++ compiler

Major changes from version 1:

1) Version 1 was completely static which could accommodate only few Faculties and few Students
    - This was done to make the entries flexible and dynamic and the system without any static binding of users to it.

    - It's important because the system itself is not static and students/faculty may leave and join back again so the whole system should respond to the change making it really useful as a working system.

2) It was having no terminal navigation with no terminal bounds
    - This was done to give more enhanced experience to the students/faculty who now can navigate over the terminal and even the faculty can edit the marks then and there without looking for the file of that student anywhere in the database.

    - It’s important as the end user must be able to function in the system with less dependencies and efforts and most of the important things and it’s function should be abstracted from the end user.

3) It was having no terminal insert/update options
    - This was done to ease the work particularly for the faculty who may now can edit the marks then and there without looking for the file of that student anywhere in the database.

    - It’s important as the end user must be able to function in the system with less dependencies and efforts and most of the important things and its function should be abstracted from the end user.

4) Earlier data was using arrays but now its done using file system
    - This was done to make the system dynamic so that more users can join in later and also leave in the middle and same reasons for faculties also. And

also now we can protect the data from being read by any user in the system as they are in a separate file with specific users permissions.

- It’s important as the data must be protected from users who are not allowed to access it and privacy of the user must be respected. Also the system design provided by the problem statement included it as one of its goals.

5) All data are stored in the files and these files are dynamically created for every user and its automatically deleted when he is removed
   - This is done to make the system completely dynamic now you don't have to go to the terminal and make users separately. Just by running the program and selecting options would help to make users and delete them. Not only that, all their files with required permissions will also be done at the same time in the backend without any efforts of the person creating them as all those have been automated.

   - It’s important as this helps to fasten the process and also makes it easier to work from the same program rather than switching back and forth between terminals to enter commands. This will help to enhance the user experience and make the functioning of the system easier and smooth.

6) User interface was very simple previously but now its completely dynamic and user friendly
   - This is done to make the system more user friendly and finally more accessible to the users where it gives them the feel of using GUI and also the navigations and terminal editing options make it easy to use.

   - It’s important as this helps to fasten the process and also makes it easier to work from the same program rather than switching back and forth between terminals to enter commands. This will help to enhance the user experience and make the functioning of the system easier and smooth.

7) Code is written in well formatted file structure and directories earlier everything was in 1 file
   - This is done to make the codes more readable and easy to understand.

   - It’s important when we need to upgrade the system and it helps in maintainability and scalability of the system.

8) MakeFile was not used in the earlier version which was added which makes all the compilation on a single command with automatic linking of the library.
   - This is done to make the compilation process smooth and all the linking of libraries will be done by its own.User won't have to remember the commands and the dependencies it requires.

   - It’s important as it will make compilation easier and the user won't have to worry what all libraries he needs to link and how to run the program as it’s very beautifully abstracted in just 1 command called “make” and all the compilation will be done for him. Indirectly helping to improve the user experience.

9) When a faculty clicks to edit the marks from the terminal the space becomes empty to give him a feel that he is editing that block.
   - This is done so that faculty can edit the marks of the students from the terminal only and don't have to go to the files specific to that student and change the marks, with this feature it can be done then and there instantly.

   - It’s important as this gives more flexibility to the faculties, helps to fasten the process and also makes it easier to work from the same program rather than switching back and forth between databases to change marks.This will help to enhance the user experience and make the functioning of the system easier and smooth.

10) Code is written in a professional manner with comments and small functions with no dependencies on others
    - This is done to arrange the codes in a logical manner and make the codes more readable and easy to understand.

    - It’s important when we need to upgrade the system and it helps in maintainability and scalability of the system.

11) All students and faculties are organized in groups earlier no such grouping was there

- This is done to utilize the common attribute of different users and make the code more sensible and less no of lines will thep to run the program faster and help in enhancing the user experience. The use of specific id makes it easier and simpler to broadcast all of the changes to all the users in the group.

- It’s important as it makes the program smaller and then the system becomes easier to scale and maintain.

12) Gid and Uid are used this time to identify the user avoiding directly the name from other commands
    - This is done to make use of the features available in the freeBSD which makes identifying a user and his group easier without asking directly from the users.

    - This is important as this will be taken from the freeBSD system and no wrong information can be given. Also it removes the dependency of the program from the user to provide their information.

13) Everything is done in the main program this time from adding user to deleting user earlier adding user and deleting them was terminal specific
    - This is done to ease the efforts of the users and they don't have to switch back and forth to the terminal to create or delete the users. Just by running the program all the work can be done from a single window interface.

    - It’s important as this gives more flexibility to the faculties, helps to fasten the process and also makes it easier to work from the same program rather than switching back and forth between terminals to add or remove the users.This will help to enhance the user experience and make the functioning of the system easier and smooth.

14) Automatic permissions for files are done earlier specifically permissions were to be given
    - This is done to ease the efforts of the users and they don't have to switch back and forth to the terminal to create or delete the users. Just by running the program all the work can be done from a single window interface.

- It’s important as this gives more flexibility to the faculties, helps to fasten the process and also makes it easier to work from the same program rather than switching back and forth between terminals to give permission to the users.This will help to enhance the user experience and make the functioning of the system easier and smooth.

15) Ranking of each students is being shown not present in previous version
    - This is done to give more analytical information to the students who can then better understand their performance. Also this adds this very feature to the system and makes it more broader.

    - It’s important as this feature helps enable the students to get more information.

16) Average marks upto 1 decimal digits is being shown not present in previous version
    - This is done to give more analytical information to the students who can then better understand their performance. Also this adds this very feature to the system and makes it more broader.

    - It’s important as this feature helps enable the students to get more information.

17) Abstraction is added in this version no students can see the faculty
    - This is done as part of the requirement of the problem statement provided to us and will help to improve the privacy of the faculty marking the students.

    - It’s important as this will help the system to be in accordance to the problem domain.

Learnings from Continuous Evaluation:

1) Setting up permissions
2) Terminal Navigation
3) Terminal Control
4) Using Tools like Dtrace

New Functionalities:

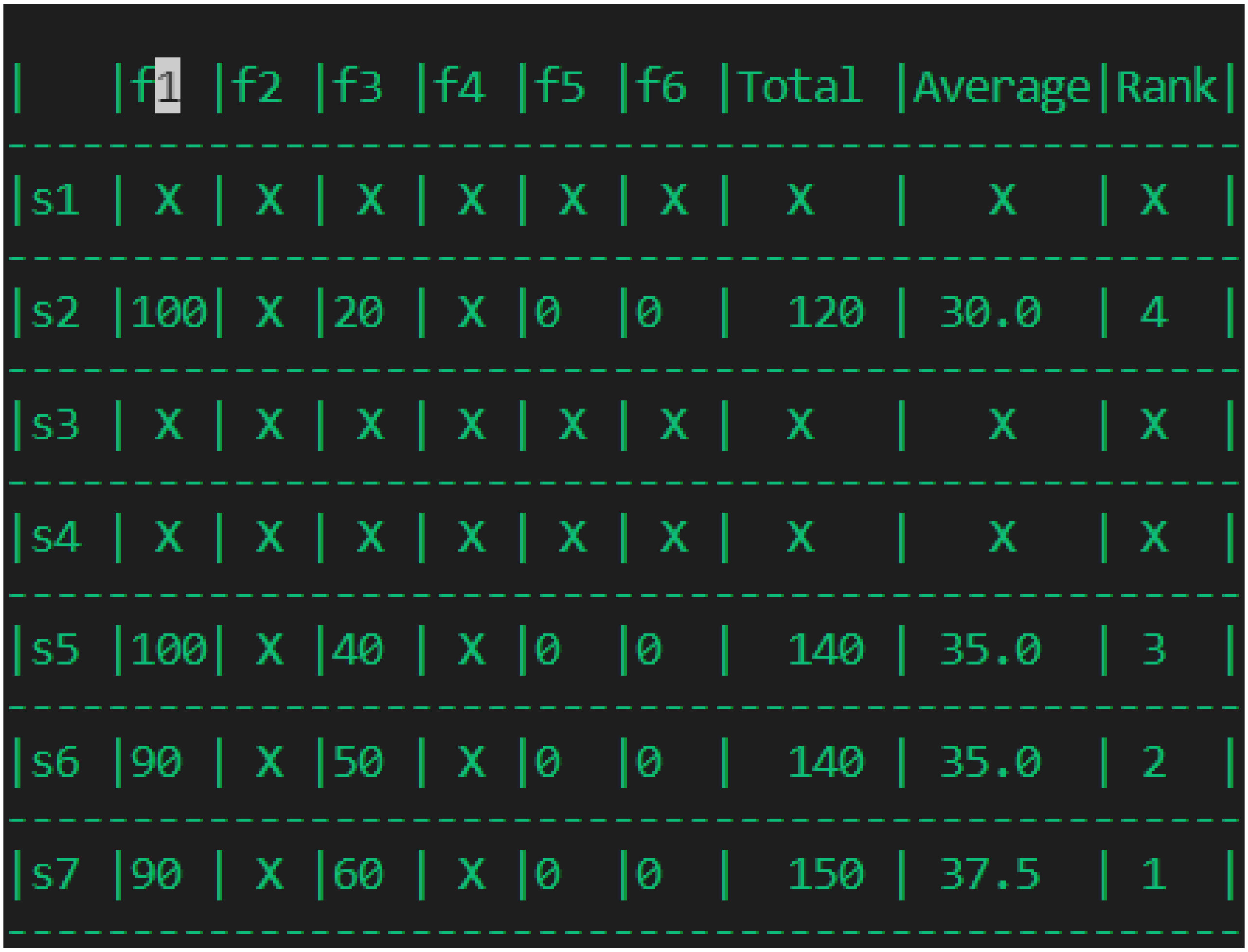
1) User creation done in program No switching to terminals
2) Automatic file generation
3) Automatic file removal
4) Automatic permission setting up
5) Terminal control to user
6) Editing via terminal

Individual roles:

- Product design - Saurav and Harshita
- Terminal control - Talha and Rajendra
- User add, delete, Permissions - Talha,Saurav
- Tools like Dtrace - Rajendra

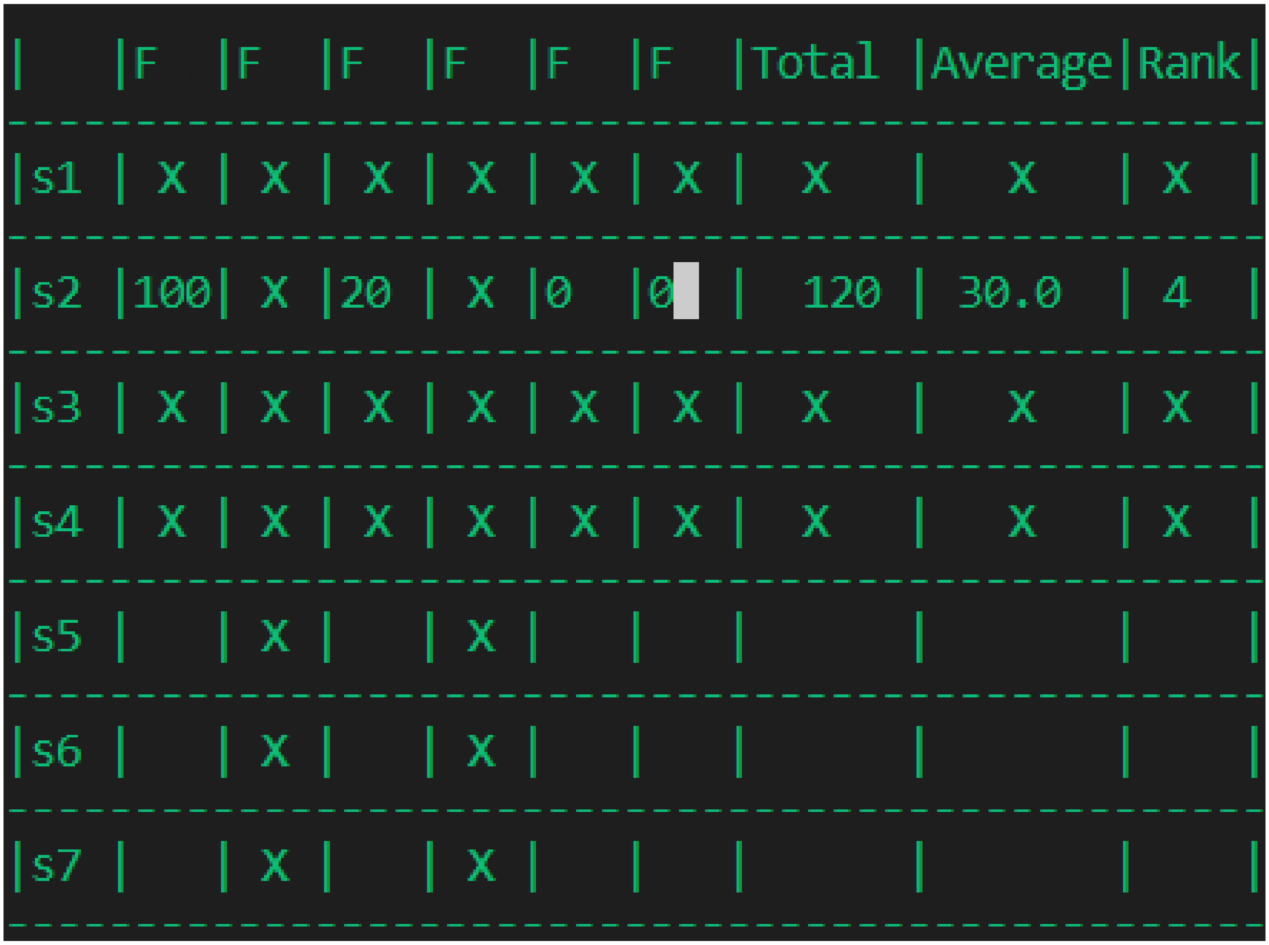
Screenshots:

1) Logged in as Instructor:
    - He can see everything
    - He can navigate anywhere in the cells.



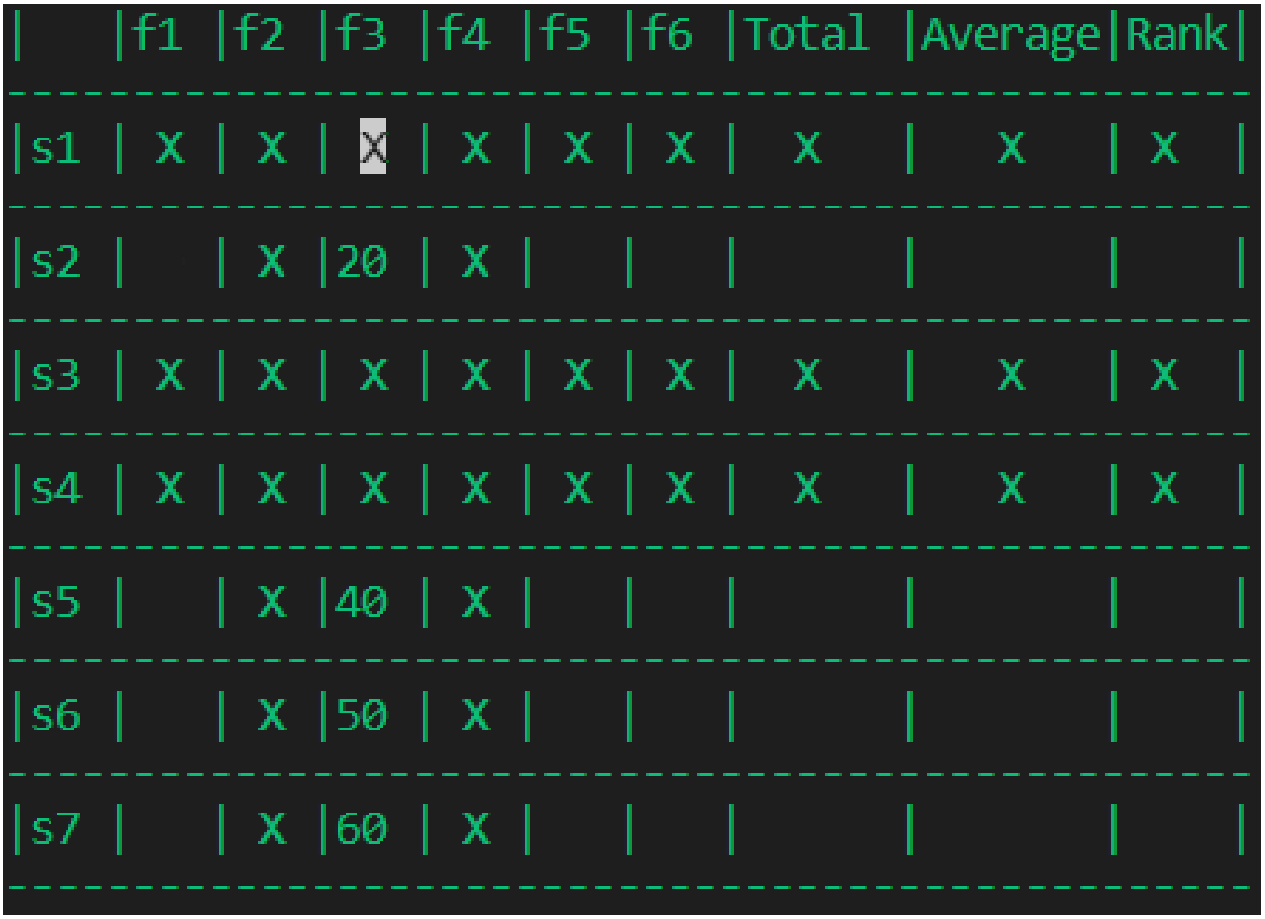
```
|   |f1 |f2 |f3 |f4 |f5 |f6 |Total |Average|Rank|
-------------------------------------------------
|s1 | X | X | X | X | X | X |  X   |   X   | X  |
-------------------------------------------------
|s2 |100| X |20 | X |0  |0  |  120 | 30.0  | 4  |
-------------------------------------------------
|s3 | X | X | X | X | X | X |  X   |   X   | X  |
-------------------------------------------------
|s4 | X | X | X | X | X | X |  X   |   X   | X  |
-------------------------------------------------
|s5 |100| X |40 | X |0  |0  |  140 | 35.0  | 3  |
-------------------------------------------------
|s6 |90 | X |50 | X |0  |0  |  140 | 35.0  | 2  |
-------------------------------------------------
|s7 |90 | X |60 | X |0  |0  |  150 | 37.5  | 1  |
-------------------------------------------------
```

2) Logged in as s2(student):
    - He can see only his row.
    - He does not have edit options.
    - He can’t navigate out of his row.
    - He can't even see the names of instructors.



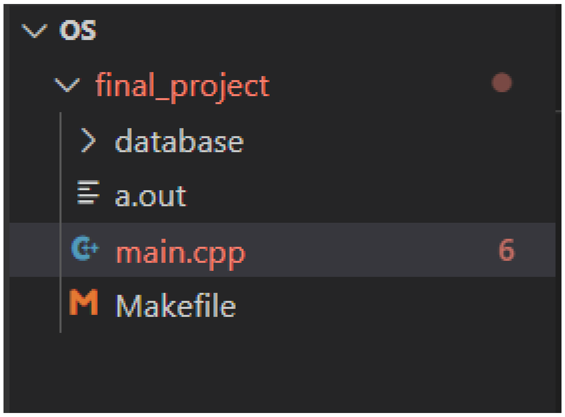
```
|   |F  |F  |F  |F  |F  |F  |Total |Average|Rank|
--------------------------------------------------
|s1 | X | X | X | X | X | X |  X   |   X   | X  |
--------------------------------------------------
|s2 |100| X |20 | X |0  |0  | 120  | 30.0  | 4  |
--------------------------------------------------
|s3 | X | X | X | X | X | X |  X   |   X   | X  |
--------------------------------------------------
|s4 | X | X | X | X | X | X |  X   |   X   | X  |
--------------------------------------------------
|s5 |   | X |   | X |   |   |      |       |    |
--------------------------------------------------
|s6 |   | X |   | X |   |   |      |       |    |
--------------------------------------------------
|s7 |   | X |   | X |   |   |      |       |    |
--------------------------------------------------
```

3) Logged in as f3 (faculty):
    - He can see only his column.
    - He has edit options.
    - He can’t navigate out of his column.

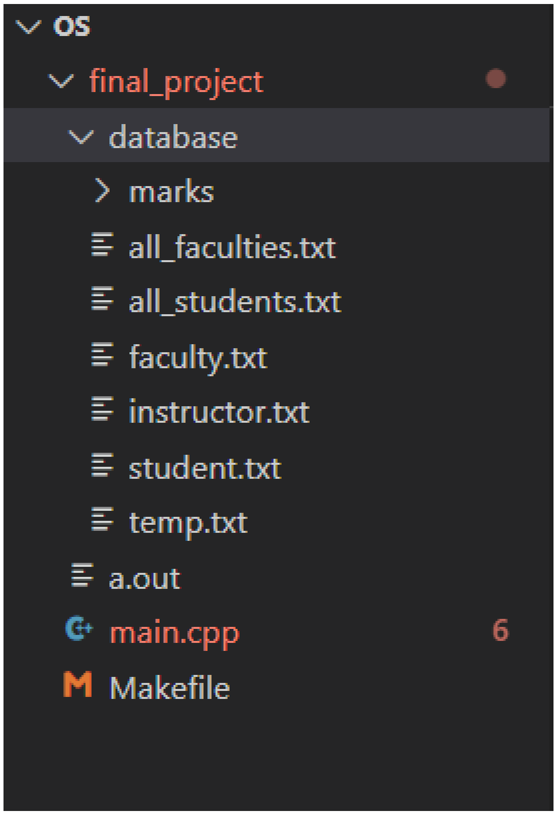


```
|   |f1 |f2 |f3 |f4 |f5 |f6 |Total |Average|Rank|
-------------------------------------------------
|s1 | X | X | X | X | X | X |  X   |   X   | X  |
-------------------------------------------------
|s2 |   | X |20 | X |   |   |      |       |    |
-------------------------------------------------
|s3 | X | X | X | X | X | X |  X   |   X   | X  |
-------------------------------------------------
|s4 | X | X | X | X | X | X |  X   |   X   | X  |
-------------------------------------------------
|s5 |   | X |40 | X |   |   |      |       |    |
-------------------------------------------------
|s6 |   | X |50 | X |   |   |      |       |    |
-------------------------------------------------
|s7 |   | X |60 | X |   |   |      |       |    |
-------------------------------------------------
```

File and folder structures:

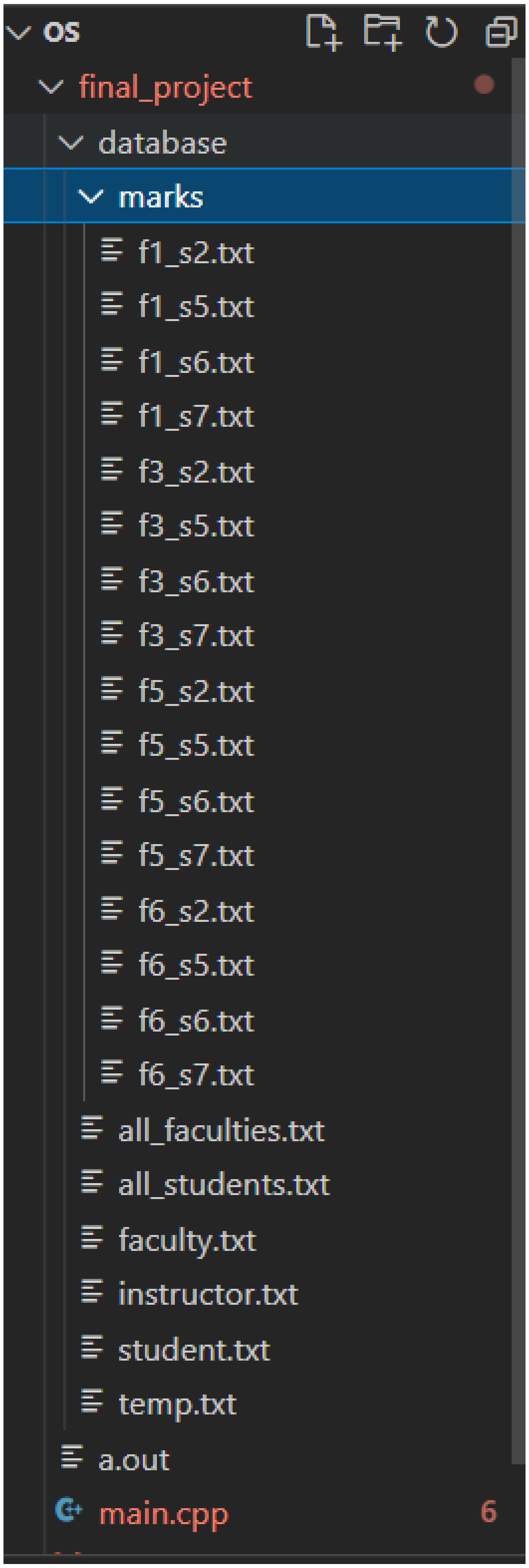


```
OS
  final_project
    database
    a.out
    main.cpp        6
    Makefile
```



```
OS
  final_project
    database
      marks
      all_faculties.txt
      all_students.txt
      faculty.txt
      instructor.txt
      student.txt
      temp.txt
    a.out
    main.cpp        6
    Makefile
```

All these files are dynamically created and deleted when a user is added or removed from the system respectively.



```
OS
  final_project
    database
      marks
        f1_s2.txt
        f1_s5.txt
        f1_s6.txt
        f1_s7.txt
        f3_s2.txt
        f3_s5.txt
        f3_s6.txt
        f3_s7.txt
        f5_s2.txt
        f5_s5.txt
        f5_s6.txt
        f5_s7.txt
        f6_s2.txt
        f6_s5.txt
        f6_s6.txt
        f6_s7.txt
      all_faculties.txt
      all_students.txt
      faculty.txt
      instructor.txt
      student.txt
      temp.txt
    a.out
    main.cpp                    6
```