



**Module Title**

**Fundamentals of Data Science**

**Assessment Weightage & Type**

**50% Individual Coursework**

**Year**

**2023-24**

**Student Name: Shrijal Pandey**

**UWE ID: 23085250**

**Assignment Due Date: April 6th, 2024**

**Assignment Submission Date: April 6<sup>th</sup>, 2024**

# 1. Introduction

In this report, we will discuss the implementation of a student management system using Python. The system allows both administrators and students to perform various tasks such as adding, modifying, deleting student records, viewing grades, and managing ECA. In this report we will be exploring the methodology used in building the system, review the technologies utilized, and reflect on the project's goals, collaboration aspects, and final conclusions.

## 2. Methodology

- ❖ **Requirement Analysis:** We carefully identified what the system needed to do (functional requirements) and how it should perform (non-functional requirements), including defining user roles, features, and how data would be managed.
- ❖ **Design:** We created a plan for how the system would be structured, including the classes it would contain, how files would be handled, and how users would interact with it.
- ❖ **Implementation:** We wrote the actual code in Python and made sure the code was organized, easy to read, and could handle errors gracefully.
- ❖ **Testing:** We thoroughly tested the system to ensure it worked as intended, was robust enough to handle different scenarios, and was secure from potential threats.
- ❖ **Maintenance:** We committed to providing ongoing support, fixing any bugs that arose, and updating the system as needed to keep it running smoothly.

### 3. Review of the Technology

The student management system was implemented using Python, and the help of its simplicity, readability, and extensive standard library. The following technologies and concepts were utilized:

- ❖ **File Handling:** Python's file handling capabilities were utilized to store and manage student records, grades, ECAs, and passwords in separate text files.
- ❖ **Object-Oriented Programming (OOP):** The system was designed using OOP principles, with classes such as Student representing entities and encapsulating related functionality.
- ❖ **Error Handling:** Python's exception handling mechanisms were used to peacefully handle errors and exceptions that may occur during program execution.
- ❖ **User Input Handling:** Input functions were utilized to interact with users, allowing them to input data for various operations such as adding, modifying and deleting student records.

### 4. Reflection

Reflecting on our project journey with the Student Management System, we can't help but feel a sense of pride and accomplishment in what we have achieved together as a team. From the initial conceptualization to the final implementation, our collaboration and dedication have been key in bringing this project to fruition.

One of the most rewarding aspects of working on this project was the opportunity to apply our Python programming skills in a real-world scenario. Developing a system capable of managing both academic and extracurricular records required us to delve deep into various Python libraries and frameworks, honing our technical abilities along the way. It was satisfying to see how our collective knowledge and expertise evolved throughout the development process.

Collaborating as a team of three presented its own set of challenges, but it also fostered a strong sense of camaraderie and mutual support. We effectively leveraged our individual strengths and expertise, with each member contributing unique insights and solutions to different aspects of the project. Regular communication and coordination were essential in ensuring that everyone stayed aligned with the project goals and timelines.

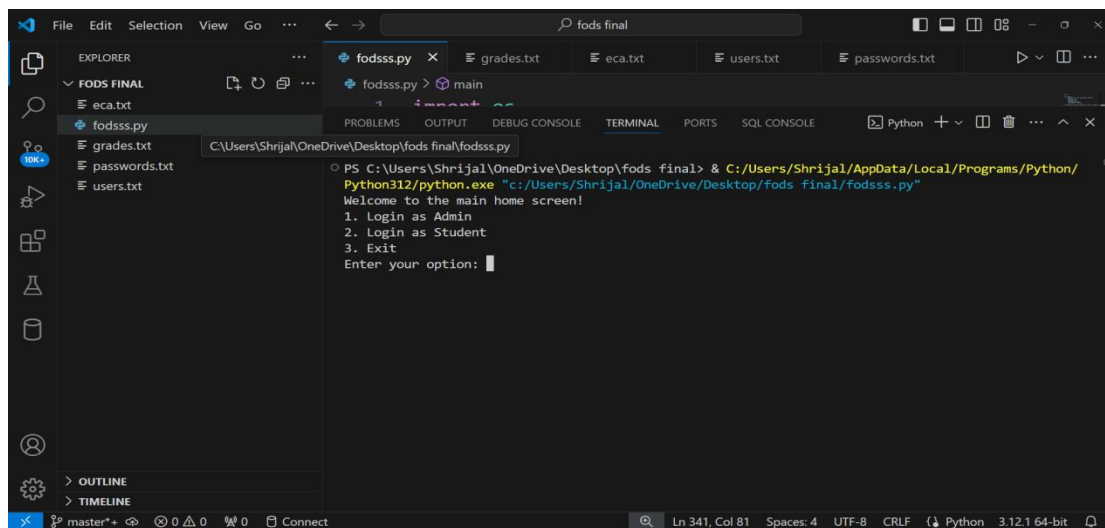
One of the highlights of our project was the iterative development approach we adopted. By breaking down the project into smaller, manageable tasks and conducting frequent reviews and iterations, we were able to make steady progress while continuously refining and improving the system. This iterative process not only helped us stay focused and organized but also allowed us to adapt to any unforeseen challenges or requirements that arose along the way.

Looking back, we believe that one of the key lessons learned from this project was the importance of effective project management and planning. From defining clear objectives and requirements to establishing a realistic timeline and allocating tasks efficiently, proper planning was crucial in keeping the project on track and ensuring its successful completion.

In conclusion, working on the Student Management System project has been a valuable learning experience that has not only enhanced our technical skills but also taught us the importance of teamwork, communication, and effective project management. We are proud of what we have accomplished together, and We're confident that the knowledge and insights gained from this project will serve us well in our future endeavors.

## Output:

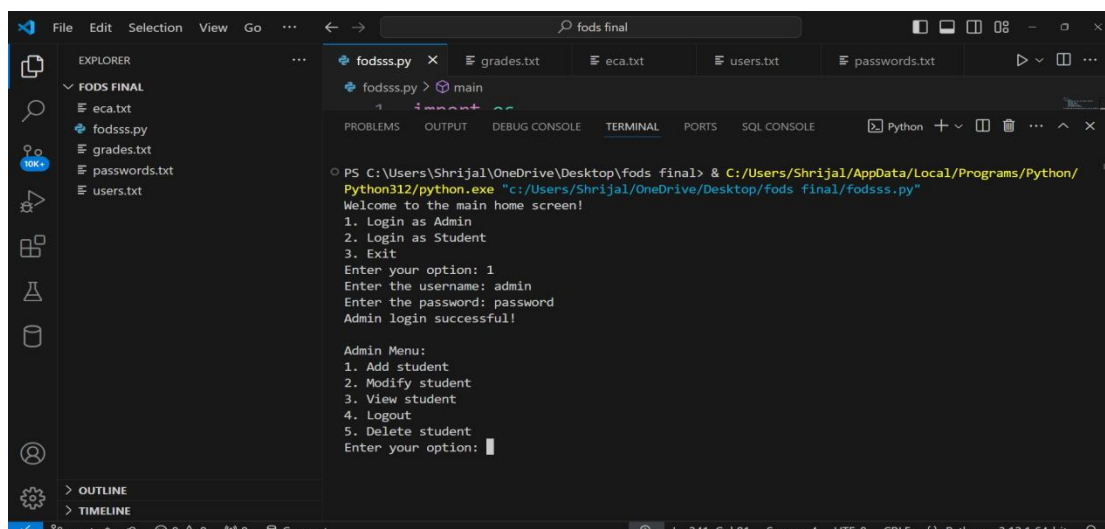
Home screen:



```
PS C:\Users\Shrijal\OneDrive\Desktop\fods final> C:/Users/Shrijal/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Shrijal/OneDrive/Desktop/fods final/fodss.py"
Welcome to the main home screen!
1. Login as Admin
2. Login as Student
3. Exit
Enter your option: 
```

## FOR THE ADMIN:

When we enter 1, it asks for the admin username and password. Then the admin menu appears.



```
PS C:\Users\Shrijal\OneDrive\Desktop\fods final> C:/Users/Shrijal/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Shrijal/OneDrive/Desktop/fods final/fodss.py"
Welcome to the main home screen!
1. Login as Admin
2. Login as Student
3. Exit
Enter your option: 1
Enter the username: admin
Enter the password: password
Admin login successful!

Admin Menu:
1. Add student
2. Modify student
3. View student
4. Logout
5. Delete student
Enter your option: 
```

The details of three students have been added: id, name, ECA, marks in 5 subjects and password

This screenshot shows the first student being added to the system. The Explorer panel on the left shows the project structure with files: eca.txt, fodsss.py, grades.txt, passwords.txt, and users.txt. The main editor displays the Python script 'fodsss.py' with a 'main' function. The integrated terminal shows the program's execution: it prompts for a username (admin) and password (password), confirms a successful login, and displays an 'Admin Menu' with five options. Option 1, 'Add student', is selected. The program then prompts for the student's ID (1022), name (Sameer), ECA (Football), marks for five subjects (78, 67, 89, 90, 74), and a password (sameer123@). Finally, it confirms that the student was added successfully.

```
File Edit Selection View Go ... < -> fodsss final
EXPLORER
  FODS FINAL
    eca.txt
    fodsss.py
    grades.txt
    passwords.txt
    users.txt
  OUTLINE
  TIMELINE
  master*+ 0 0 0 0 0 Connect

fodsss.py x grades.txt eca.txt users.txt passwords.txt
fodsss.py > main
1 import os
2
3 def main():
4     login()
5     admin_menu()
6
7 def login():
8     username = input("Enter the username: ")
9     password = input("Enter the password: ")
10    if username == "admin" and password == "password":
11        print("Admin login successful!")
12    else:
13        print("Invalid username or password!")
14
15 def admin_menu():
16     print("Admin Menu:")
17     print("1. Add student")
18     print("2. Modify student")
19     print("3. View student")
20     print("4. Logout")
21     print("5. Delete student")
22     option = input("Enter your option: ")
23     if option == "1":
24         add_student()
25     elif option == "2":
26         modify_student()
27     elif option == "3":
28         view_student()
29     elif option == "4":
30         logout()
31     elif option == "5":
32         delete_student()
33
34 def add_student():
35     id = input("Enter the ID of the student: ")
36     name = input("Enter the name of the student: ")
37     eca = input("Enter the ECA done: ")
38     marks = []
39     for i in range(5):
40         mark = input(f"Enter marks for subject {i+1}: ")
41         marks.append(mark)
42     password = input("Set the password: ")
43     save_student(id, name, eca, marks, password)
44     print("Student added successfully!")
45
46 def modify_student():
47     id = input("Enter the ID of the student: ")
48     name = input("Enter the name of the student: ")
49     eca = input("Enter the ECA done: ")
50     marks = []
51     for i in range(5):
52         mark = input(f"Enter marks for subject {i+1}: ")
53         marks.append(mark)
54     password = input("Set the password: ")
55     save_student(id, name, eca, marks, password)
56     print("Student modified successfully!")
57
58 def view_student():
59     id = input("Enter the ID of the student: ")
60     student = get_student(id)
61     if student:
62         print(f"Student ID: {id}, Name: {student['name']}, ECA: {student['eca']}, Marks: {student['marks']}, Password: {student['password']}")
63     else:
64         print("Student not found!")
65
66 def logout():
67     print("Logout successful!")
68
69 def delete_student():
70     id = input("Enter the ID of the student: ")
71     student = get_student(id)
72     if student:
73         delete_student_record(id)
74         print("Student deleted successfully!")
75     else:
76         print("Student not found!")
77
78 def save_student(id, name, eca, marks, password):
79     with open("users.txt", "a") as f:
80         f.write(f"{id},{name},{eca},{marks},{password}\n")
81
82 def get_student(id):
83     with open("users.txt", "r") as f:
84         for line in f:
85             data = line.strip().split(",")
86             if data[0] == id:
87                 return {"id": data[0], "name": data[1], "eca": data[2], "marks": data[3], "password": data[4]}
88     return None
89
90 def delete_student_record(id):
91     with open("users.txt", "r") as f:
92         lines = f.readlines()
93     with open("users.txt", "w") as f:
94         for line in lines:
95             data = line.strip().split(",")
96             if data[0] != id:
97                 f.write(line)
98
99 if __name__ == "__main__":
100    main()
```

This screenshot shows the second student being added. The terminal output continues from the previous state, showing the 'Admin Menu' again. Option 1 is selected, and the program prompts for the student's ID (1023), name (Aryan), ECA (Basketball), marks for five subjects (89, 76, 69, 96, 79), and a password (aryan123@). It confirms the student was added successfully. The terminal then shows the 'Admin Menu' again, with option 1 selected, and the program prompts for the student's ID (1024), name (Swarn), ECA (cricket), marks for five subjects (78, 98, 67, 75, 77), and a password (swarn123@). It confirms the student was added successfully.

```
File Edit Selection View Go Run ... < -> fodsss final
EXPLORER
  FODS FINAL
    eca.txt
    fodsss.py
    grades.txt
    passwords.txt
    users.txt
  OUTLINE
  TIMELINE
  master*+ 0 0 0 0 0 Connect

fodsss.py x grades.txt eca.txt users.txt passwords.txt
fodsss.py > main
1 import os
2
3 def main():
4     login()
5     admin_menu()
6
7 def login():
8     username = input("Enter the username: ")
9     password = input("Enter the password: ")
10    if username == "admin" and password == "password":
11        print("Admin login successful!")
12    else:
13        print("Invalid username or password!")
14
15 def admin_menu():
16     print("Admin Menu:")
17     print("1. Add student")
18     print("2. Modify student")
19     print("3. View student")
20     print("4. Logout")
21     print("5. Delete student")
22     option = input("Enter your option: ")
23     if option == "1":
24         add_student()
25     elif option == "2":
26         modify_student()
27     elif option == "3":
28         view_student()
29     elif option == "4":
30         logout()
31     elif option == "5":
32         delete_student()
33
34 def add_student():
35     id = input("Enter the ID of the student: ")
36     name = input("Enter the name of the student: ")
37     eca = input("Enter the ECA done: ")
38     marks = []
39     for i in range(5):
40         mark = input(f"Enter marks for subject {i+1}: ")
41         marks.append(mark)
42     password = input("Set the password: ")
43     save_student(id, name, eca, marks, password)
44     print("Student added successfully!")
45
46 def modify_student():
47     id = input("Enter the ID of the student: ")
48     name = input("Enter the name of the student: ")
49     eca = input("Enter the ECA done: ")
50     marks = []
51     for i in range(5):
52         mark = input(f"Enter marks for subject {i+1}: ")
53         marks.append(mark)
54     password = input("Set the password: ")
55     save_student(id, name, eca, marks, password)
56     print("Student modified successfully!")
57
58 def view_student():
59     id = input("Enter the ID of the student: ")
60     student = get_student(id)
61     if student:
62         print(f"Student ID: {id}, Name: {student['name']}, ECA: {student['eca']}, Marks: {student['marks']}, Password: {student['password']}")
63     else:
64         print("Student not found!")
65
66 def logout():
67     print("Logout successful!")
68
69 def delete_student():
70     id = input("Enter the ID of the student: ")
71     student = get_student(id)
72     if student:
73         delete_student_record(id)
74         print("Student deleted successfully!")
75     else:
76         print("Student not found!")
77
78 def save_student(id, name, eca, marks, password):
79     with open("users.txt", "a") as f:
80         f.write(f"{id},{name},{eca},{marks},{password}\n")
81
82 def get_student(id):
83     with open("users.txt", "r") as f:
84         for line in f:
85             data = line.strip().split(",")
86             if data[0] == id:
87                 return {"id": data[0], "name": data[1], "eca": data[2], "marks": data[3], "password": data[4]}
88     return None
89
90 def delete_student_record(id):
91     with open("users.txt", "r") as f:
92         lines = f.readlines()
93     with open("users.txt", "w") as f:
94         for line in lines:
95             data = line.strip().split(",")
96             if data[0] != id:
97                 f.write(line)
98
99 if __name__ == "__main__":
100    main()
```

This screenshot shows the third student being added. The terminal output continues from the previous state, showing the 'Admin Menu' again. Option 1 is selected, and the program prompts for the student's ID (1024), name (Swarn), ECA (cricket), marks for five subjects (78, 98, 67, 75, 77), and a password (swarn123@). It confirms the student was added successfully. The terminal then shows the 'Admin Menu' again, with option 1 selected, and the program prompts for the student's ID (1024), name (Swarn), ECA (cricket), marks for five subjects (78, 98, 67, 75, 77), and a password (swarn123@). It confirms the student was added successfully.

```
File Edit Selection View Go Run ... < -> fodsss final
EXPLORER
  FODS FINAL
    eca.txt
    fodsss.py
    grades.txt
    passwords.txt
    users.txt
  OUTLINE
  TIMELINE
  master*+ 0 0 0 0 0 Connect

fodsss.py x grades.txt eca.txt users.txt passwords.txt
fodsss.py > main
1 import os
2
3 def main():
4     login()
5     admin_menu()
6
7 def login():
8     username = input("Enter the username: ")
9     password = input("Enter the password: ")
10    if username == "admin" and password == "password":
11        print("Admin login successful!")
12    else:
13        print("Invalid username or password!")
14
15 def admin_menu():
16     print("Admin Menu:")
17     print("1. Add student")
18     print("2. Modify student")
19     print("3. View student")
20     print("4. Logout")
21     print("5. Delete student")
22     option = input("Enter your option: ")
23     if option == "1":
24         add_student()
25     elif option == "2":
26         modify_student()
27     elif option == "3":
28         view_student()
29     elif option == "4":
30         logout()
31     elif option == "5":
32         delete_student()
33
34 def add_student():
35     id = input("Enter the ID of the student: ")
36     name = input("Enter the name of the student: ")
37     eca = input("Enter the ECA done: ")
38     marks = []
39     for i in range(5):
40         mark = input(f"Enter marks for subject {i+1}: ")
41         marks.append(mark)
42     password = input("Set the password: ")
43     save_student(id, name, eca, marks, password)
44     print("Student added successfully!")
45
46 def modify_student():
47     id = input("Enter the ID of the student: ")
48     name = input("Enter the name of the student: ")
49     eca = input("Enter the ECA done: ")
50     marks = []
51     for i in range(5):
52         mark = input(f"Enter marks for subject {i+1}: ")
53         marks.append(mark)
54     password = input("Set the password: ")
55     save_student(id, name, eca, marks, password)
56     print("Student modified successfully!")
57
58 def view_student():
59     id = input("Enter the ID of the student: ")
60     student = get_student(id)
61     if student:
62         print(f"Student ID: {id}, Name: {student['name']}, ECA: {student['eca']}, Marks: {student['marks']}, Password: {student['password']}")
63     else:
64         print("Student not found!")
65
66 def logout():
67     print("Logout successful!")
68
69 def delete_student():
70     id = input("Enter the ID of the student: ")
71     student = get_student(id)
72     if student:
73         delete_student_record(id)
74         print("Student deleted successfully!")
75     else:
76         print("Student not found!")
77
78 def save_student(id, name, eca, marks, password):
79     with open("users.txt", "a") as f:
80         f.write(f"{id},{name},{eca},{marks},{password}\n")
81
82 def get_student(id):
83     with open("users.txt", "r") as f:
84         for line in f:
85             data = line.strip().split(",")
86             if data[0] == id:
87                 return {"id": data[0], "name": data[1], "eca": data[2], "marks": data[3], "password": data[4]}
88     return None
89
90 def delete_student_record(id):
91     with open("users.txt", "r") as f:
92         lines = f.readlines()
93     with open("users.txt", "w") as f:
94         for line in lines:
95             data = line.strip().split(",")
96             if data[0] != id:
97                 f.write(line)
98
99 if __name__ == "__main__":
100    main()
```

We can see that the records have been added in the respective txt files:

The first screenshot shows the `users.txt` file with the following content:

```
1 1022,Sameer
2 1023,Aryan
3 1024,Swarn
4
```

The second screenshot shows the `grades.txt` file with the following content:

```
1 1022,Sameer,78,67,89,90,74
2 1023,Aryan,89,76,69,96,79
3 1024,Swarn,78,98,67,75,77
4
```

The third screenshot shows the `eca.txt` file with the following content:

```
1 1022,Sameer,Football
2 1023,Aryan,Basketball
3 1024,Swarn,cricket
4
```

The fourth screenshot shows the `passwords.txt` file with the following content:

```
1 1022,Sameer,sameer123@
2 1023,Aryan,aryan123@
3 1024,Swarn,swarn123@
4
```

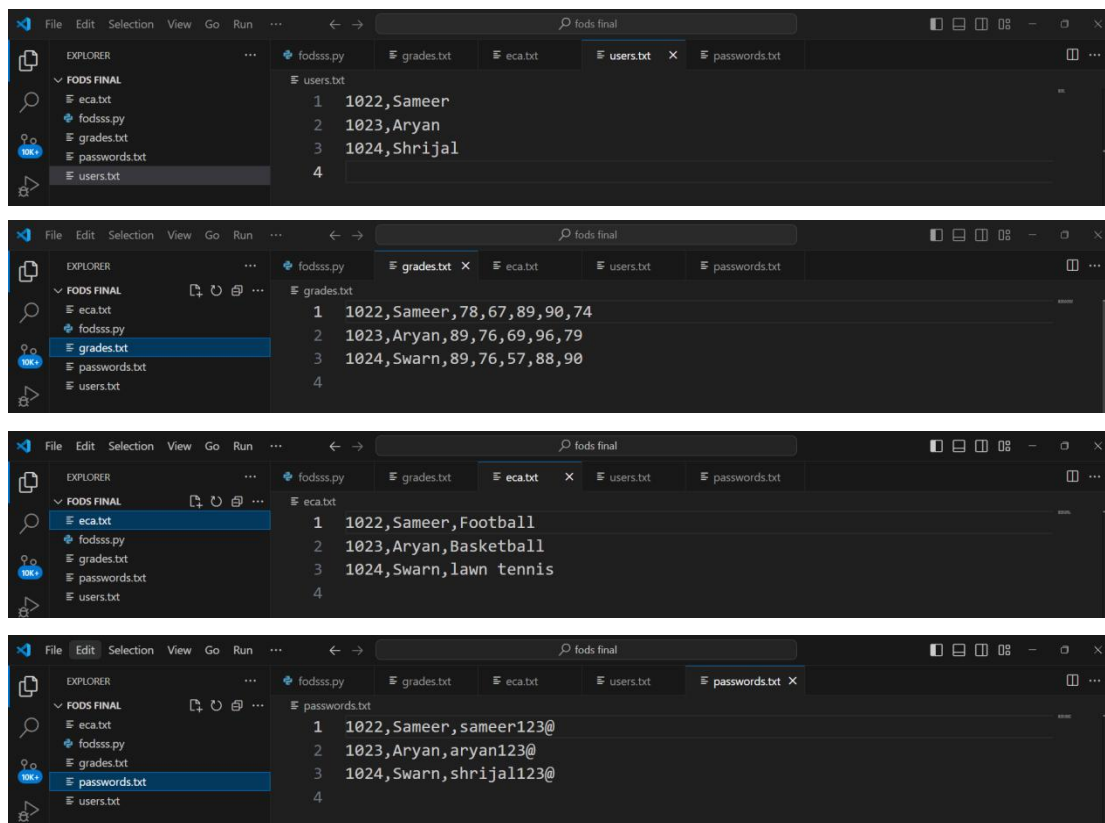
When we enter 2 for modifying students it asks for the details of the new student:

The screenshot shows the terminal output of a Python script. The user has selected option 2 to modify a student. The script prompts for the ID of the student to modify (1024), the new name (Shrijal), the new ECA (lawn tennis), and new marks for five subjects (89, 76, 57, 88, 90). The password is updated to shr1jal123@. The output confirms that the student information was modified successfully and displays the admin menu.

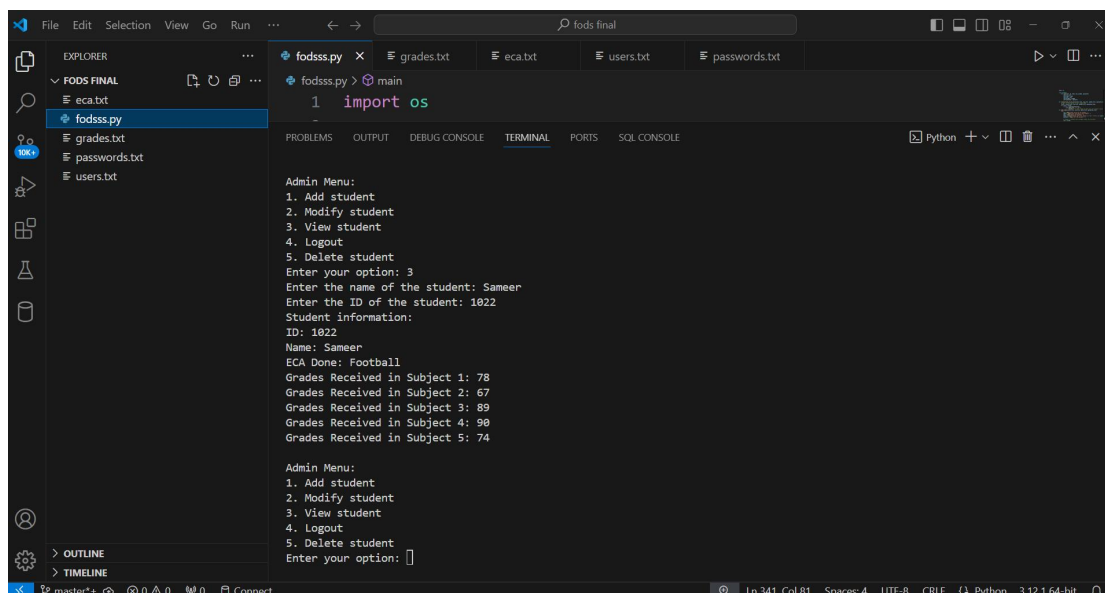
```
C:\Users\Shrijal\OneDrive\Desktop\fods final\users.txt
1. Add student
2. Modify student
3. View student
4. Logout
5. Delete student
Enter your option: 2
Enter the ID of the student to modify: 1024
Enter the new name: Shrijal
Enter the new ECA: lawn tennis
Enter new marks for subject 1: 89
Enter new marks for subject 2: 76
Enter new marks for subject 3: 57
Enter new marks for subject 4: 88
Enter new marks for subject 5: 90
Enter the new password: shr1jal123@
Student information modified successfully!

Admin Menu:
1. Add student
2. Modify student
3. View student
4. Logout
5. Delete student
Enter your option: 
```

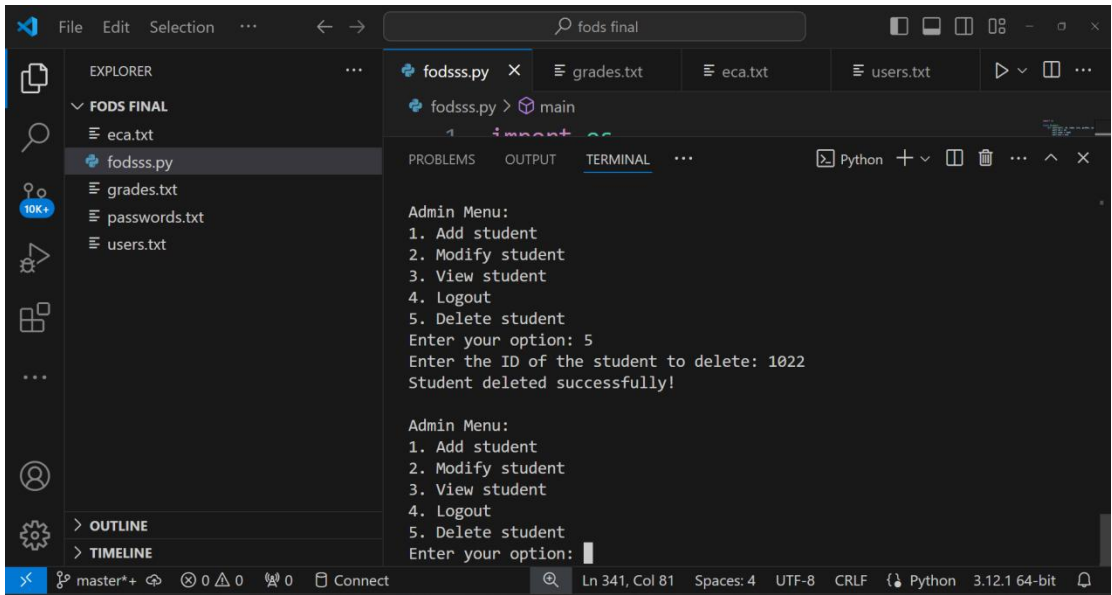
When we check the txt files, it shows with the changes:



When we view students, it shows all the records of the student



When we delete student, the data gets deleted from all the txt files:

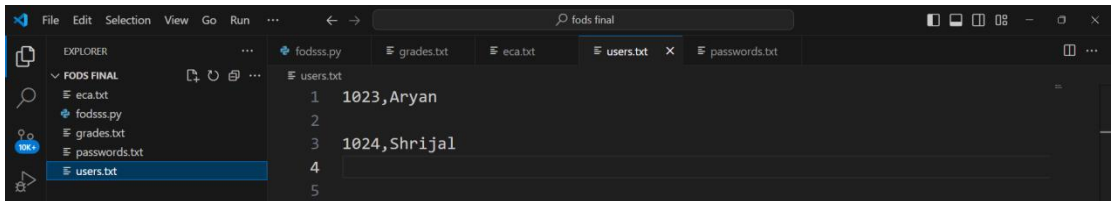


```
File Edit Selection ... fods final
EXPLORER
  FODS FINAL
    eca.txt
    fodss.py
    grades.txt
    passwords.txt
    users.txt
  OUTLINE
  TIMELINE
master*+ 0 0 0 0 Connect Ln 341, Col 81 Spaces: 4 UTF-8 CRLF Python 3.12.1 64-bit

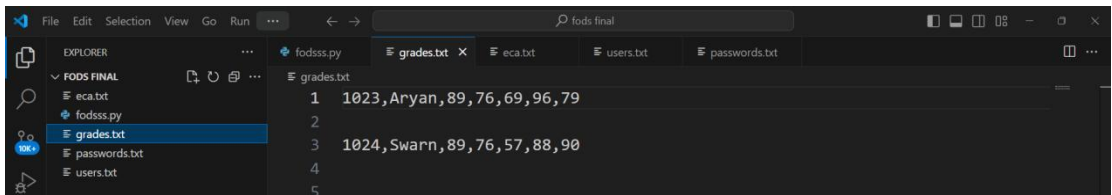
fodss.py > main
1 import

PROBLEMS OUTPUT TERMINAL
Admin Menu:
1. Add student
2. Modify student
3. View student
4. Logout
5. Delete student
Enter your option: 5
Enter the ID of the student to delete: 1022
Student deleted successfully!

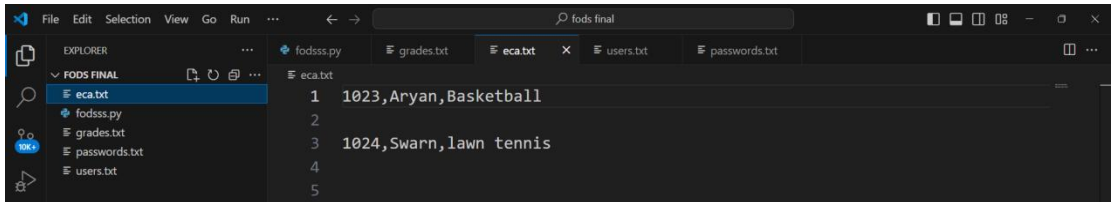
Admin Menu:
1. Add student
2. Modify student
3. View student
4. Logout
5. Delete student
Enter your option: 
```



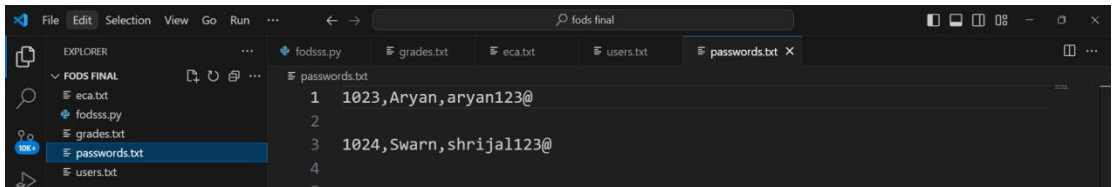
```
File Edit Selection View Go Run ... fods final
EXPLORER
  FODS FINAL
    eca.txt
    fodss.py
    grades.txt
    passwords.txt
    users.txt
users.txt
1 1023,Aryan
2
3 1024,Shrijal
4
5
```



```
File Edit Selection View Go Run ... fods final
EXPLORER
  FODS FINAL
    eca.txt
    fodss.py
    grades.txt
    passwords.txt
    users.txt
grades.txt
1 1023,Aryan,89,76,69,96,79
2
3 1024,Swarn,89,76,57,88,90
4
5
```



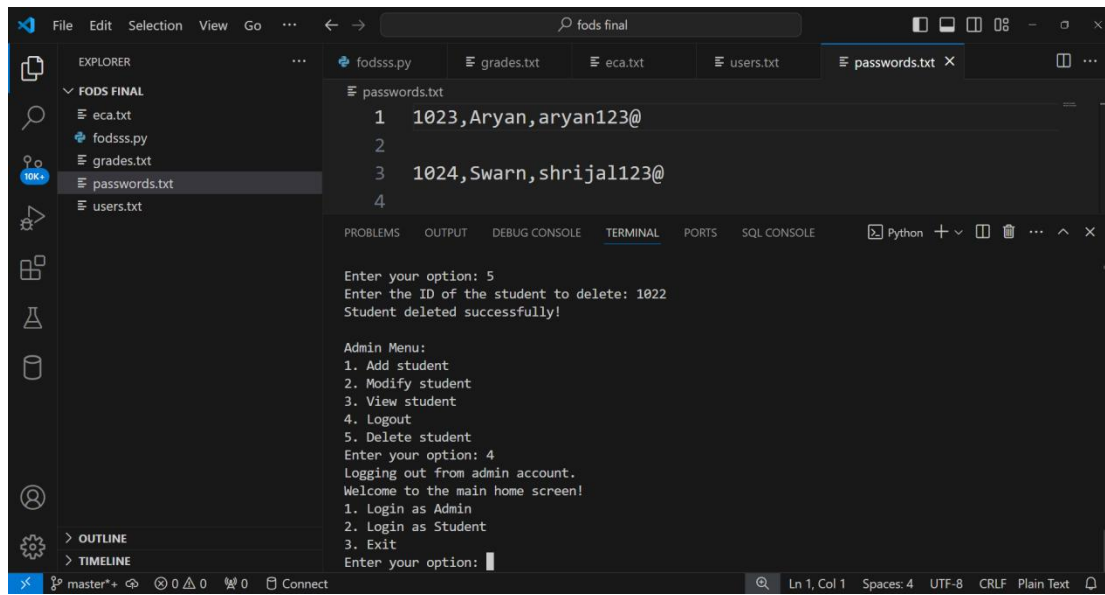
```
File Edit Selection View Go Run ... fods final
EXPLORER
  FODS FINAL
    eca.txt
    fodss.py
    grades.txt
    passwords.txt
    users.txt
eca.txt
1 1023,Aryan,Basketball
2
3 1024,Swarn,lawn tennis
4
5
```



```
File Edit Selection View Go Run ... fods final
EXPLORER
  FODS FINAL
    eca.txt
    fodss.py
    grades.txt
    passwords.txt
    users.txt
passwords.txt
1 1023,Aryan,aryan123@
2
3 1024,Swarn,shrijal123@
4
5
```



When we log out, it send us back to the home screen:



The screenshot shows a Visual Studio Code interface. The Explorer sidebar on the left shows a project named 'FODS FINAL' with files: eca.txt, fodsss.py, grades.txt, passwords.txt (selected), and users.txt. The main editor area shows the 'passwords.txt' file with the following content:

```
1 1023,Aryan,aryan123@
2
3 1024,Swarn,shrijal123@
4
```

Below the editor is a terminal window with the following output:

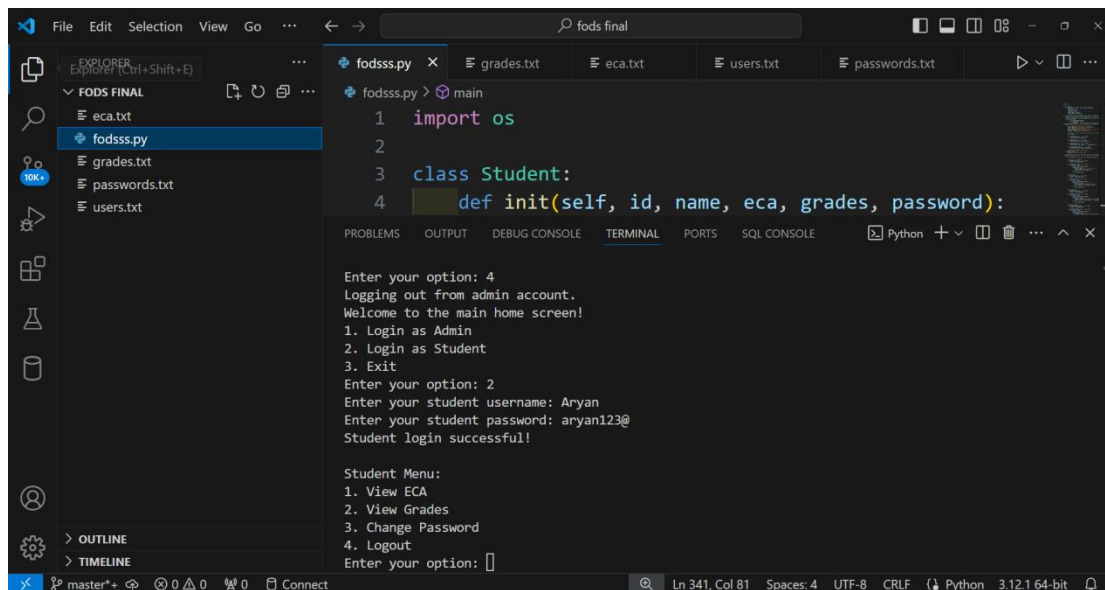
```
Enter your option: 5
Enter the ID of the student to delete: 1022
Student deleted successfully!

Admin Menu:
1. Add student
2. Modify student
3. View student
4. Logout
5. Delete student
Enter your option: 4
Logging out from admin account.
Welcome to the main home screen!
1. Login as Admin
2. Login as Student
3. Exit
Enter your option: 
```

The terminal window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (active), PORTS, and SQL CONSOLE. The status bar at the bottom shows 'master+', 'Ln 1, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', and 'Plain Text'.

## FOR STUDENT LOGIN:

When we enter 2 to login as student, it asks for username and password and finally student menu appears:

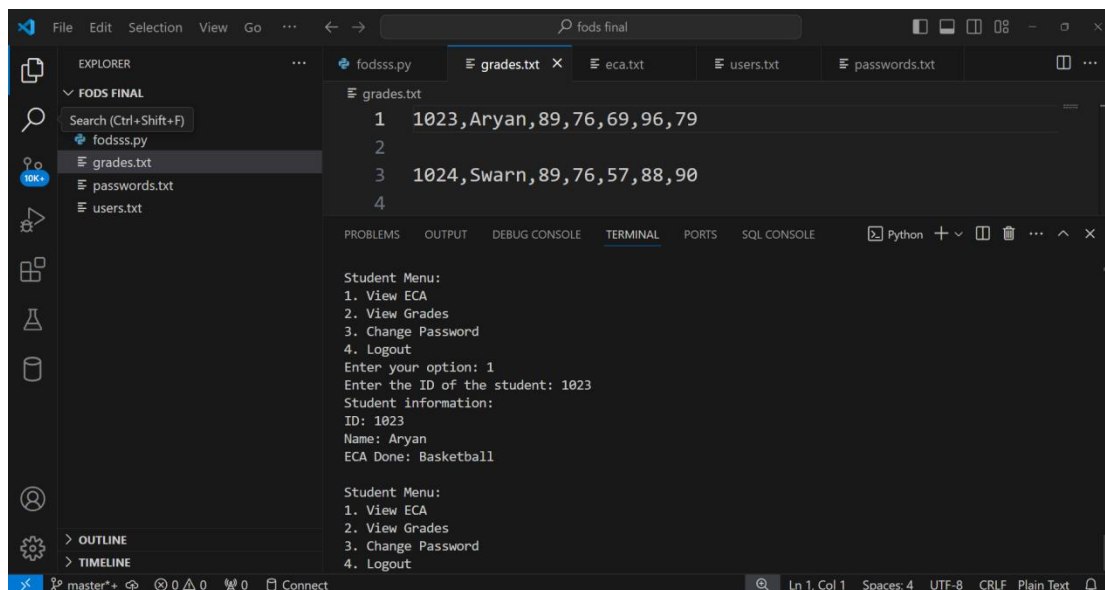


```
fodsss.py
1 import os
2
3 class Student:
4     def init(self, id, name, eca, grades, password):
```

```
Enter your option: 4
Logging out from admin account.
Welcome to the main home screen!
1. Login as Admin
2. Login as Student
3. Exit
Enter your option: 2
Enter your student username: Aryan
Enter your student password: aryan123@
Student login successful!

Student Menu:
1. View ECA
2. View Grades
3. Change Password
4. Logout
Enter your option: 
```

When we click option 1; view ECA and enter our details, it shows our ECA:

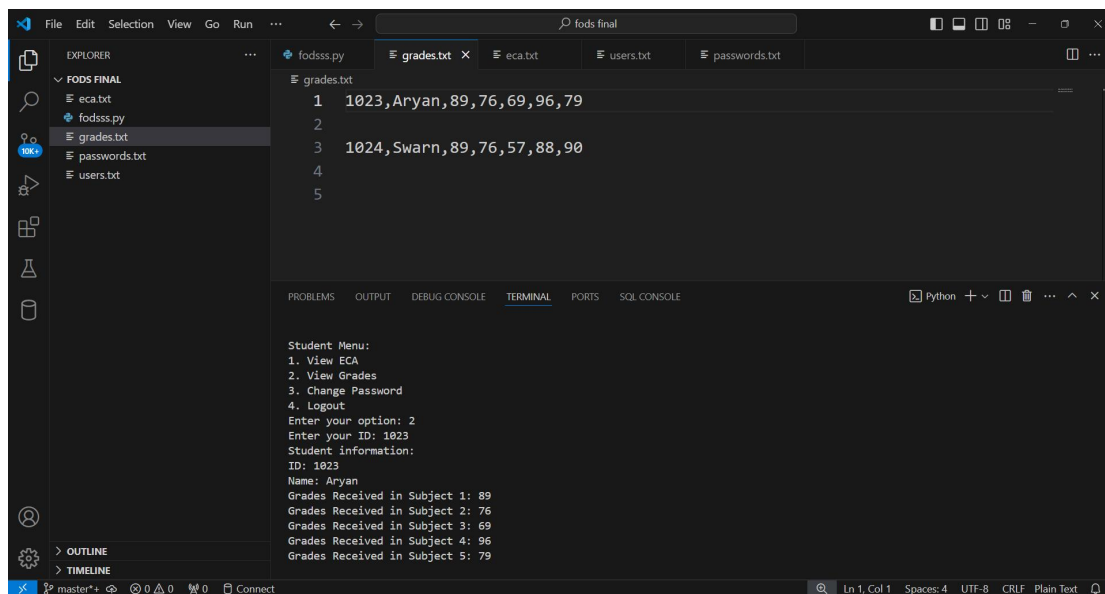


```
grades.txt
1 1023,Aryan,89,76,69,96,79
2
3 1024,Swarn,89,76,57,88,90
4
```

```
Student Menu:
1. View ECA
2. View Grades
3. Change Password
4. Logout
Enter your option: 1
Enter the ID of the student: 1023
Student information:
ID: 1023
Name: Aryan
ECA Done: Basketball

Student Menu:
1. View ECA
2. View Grades
3. Change Password
4. Logout
```

When we click option 2; view grades and enter our details, it shows our grades:

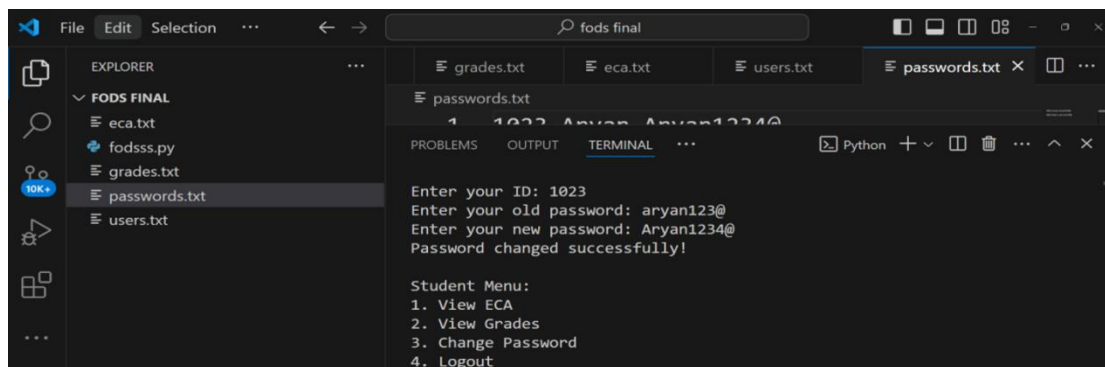


The screenshot shows the Visual Studio Code interface with the 'fods final' project open. The Explorer panel on the left shows the file structure. The 'grades.txt' file is open in the editor, displaying a list of student IDs and their grades. The terminal window at the bottom shows the output of the program, including the 'Student Menu' and the details for student ID 1023.

```
grades.txt
1 1023,Aryan,89,76,69,96,79
2
3 1024,Swarn,89,76,57,88,90
4
5

Student Menu:
1. View ECA
2. View Grades
3. Change Password
4. Logout
Enter your option: 2
Enter your ID: 1023
Student information:
ID: 1023
Name: Aryan
Grades Received in Subject 1: 89
Grades Received in Subject 2: 76
Grades Received in Subject 3: 69
Grades Received in Subject 4: 96
Grades Received in Subject 5: 79
```

When we click option 3 to change password; it asks for previous password for verification:



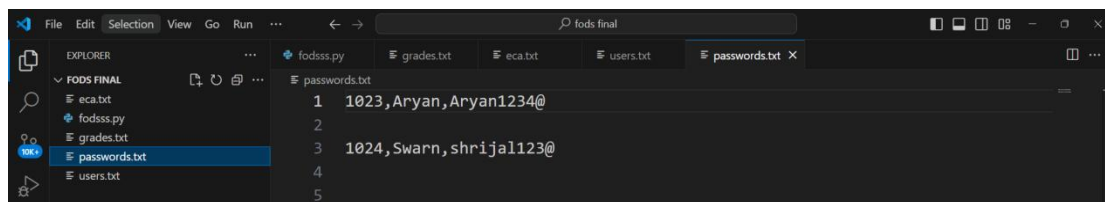
The screenshot shows the Visual Studio Code interface with the 'passwords.txt' file open in the editor. The terminal window at the bottom shows the output of the program, including the 'Student Menu' and the details for student ID 1023. The terminal output shows the user entering their ID, old password, and new password, and the program successfully changing the password.

```
passwords.txt
1 1023,Aryan,Aryan1234@
2
3 1024,Swarn,shrijal123@
4
5

Enter your ID: 1023
Enter your old password: aryan123@
Enter your new password: Aryan1234@
Password changed successfully!

Student Menu:
1. View ECA
2. View Grades
3. Change Password
4. Logout
```

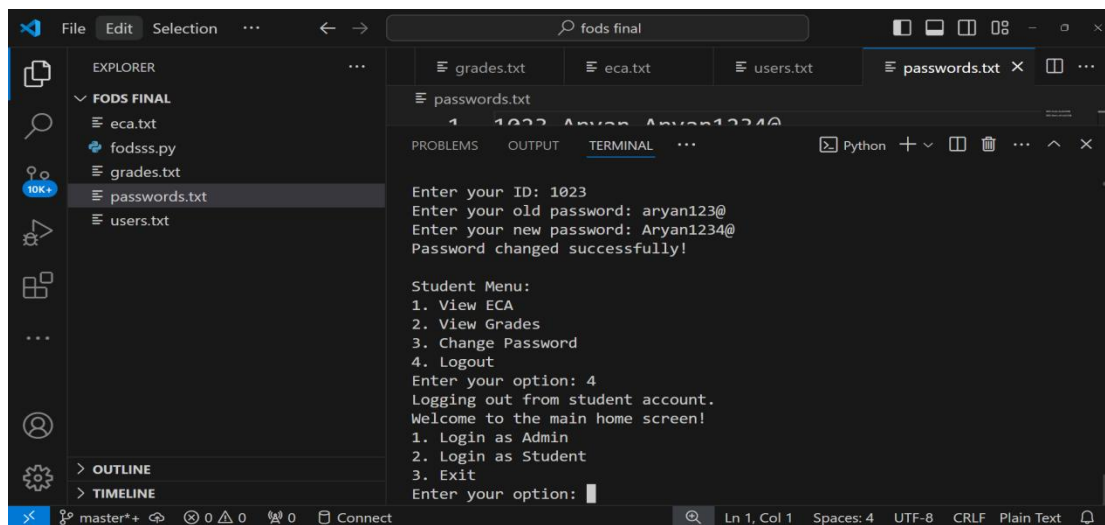
We can see the changed password here in the txt:



The screenshot shows the Visual Studio Code interface with the 'passwords.txt' file open in the editor. The file content shows the updated password for student ID 1023.

```
passwords.txt
1 1023,Aryan,Aryan1234@
2
3 1024,Swarn,shrijal123@
4
5
```

When we click on option 4 for logout, it sends us back to the home screen:

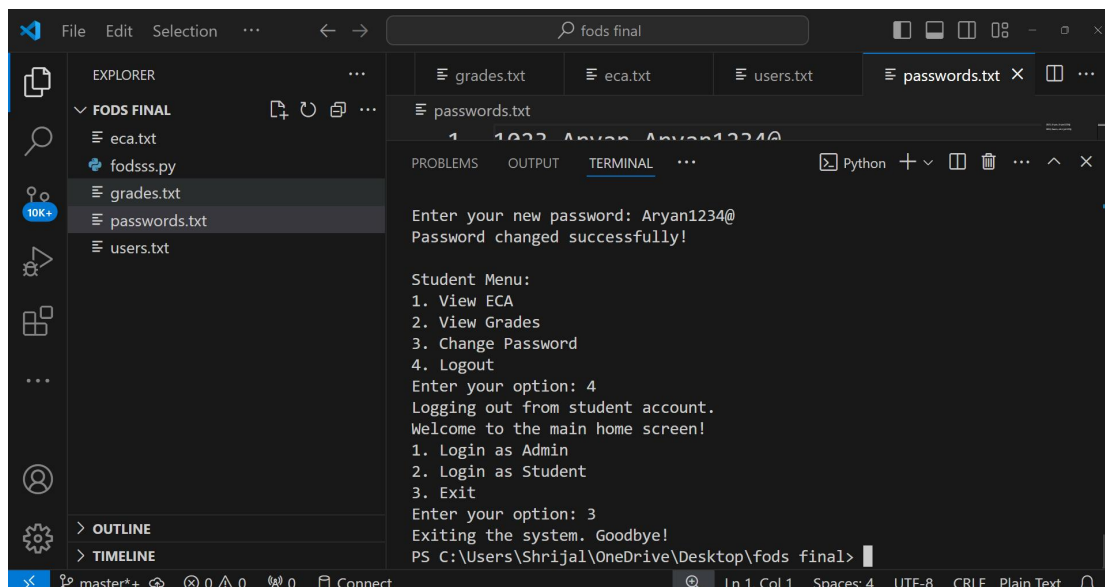


```
File Edit Selection ... fods final
EXPLORER
  FODS FINAL
    eca.txt
    fodss.py
    grades.txt
    passwords.txt
    users.txt
  OUTLINE
  TIMELINE
  master*+ 0 0 0 0 Connect Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Plain Text

passwords.txt
1 1023 Aryan Aryan1234@
PROBLEMS OUTPUT TERMINAL ... Python + - - - ^ X
Enter your ID: 1023
Enter your old password: aryan123@
Enter your new password: Aryan1234@
Password changed successfully!

Student Menu:
1. View ECA
2. View Grades
3. Change Password
4. Logout
Enter your option: 4
Logging out from student account.
Welcome to the main home screen!
1. Login as Admin
2. Login as Student
3. Exit
Enter your option: 
```

When we enter option 3 to exit, it says Exiting the system. Goodbye!



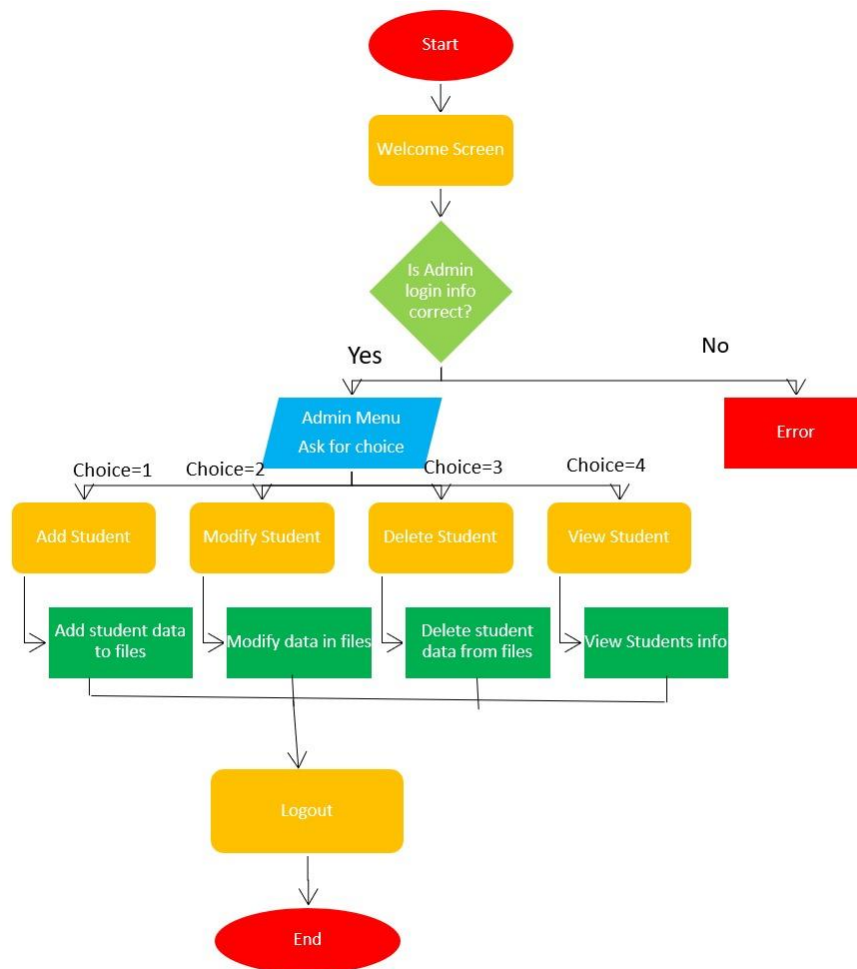
```
File Edit Selection ... fods final
EXPLORER
  FODS FINAL
    eca.txt
    fodss.py
    grades.txt
    passwords.txt
    users.txt
  OUTLINE
  TIMELINE
  master*+ 0 0 0 0 Connect Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Plain Text

passwords.txt
1 1023 Aryan Aryan1234@
PROBLEMS OUTPUT TERMINAL ... Python + - - - ^ X
Enter your new password: Aryan1234@
Password changed successfully!

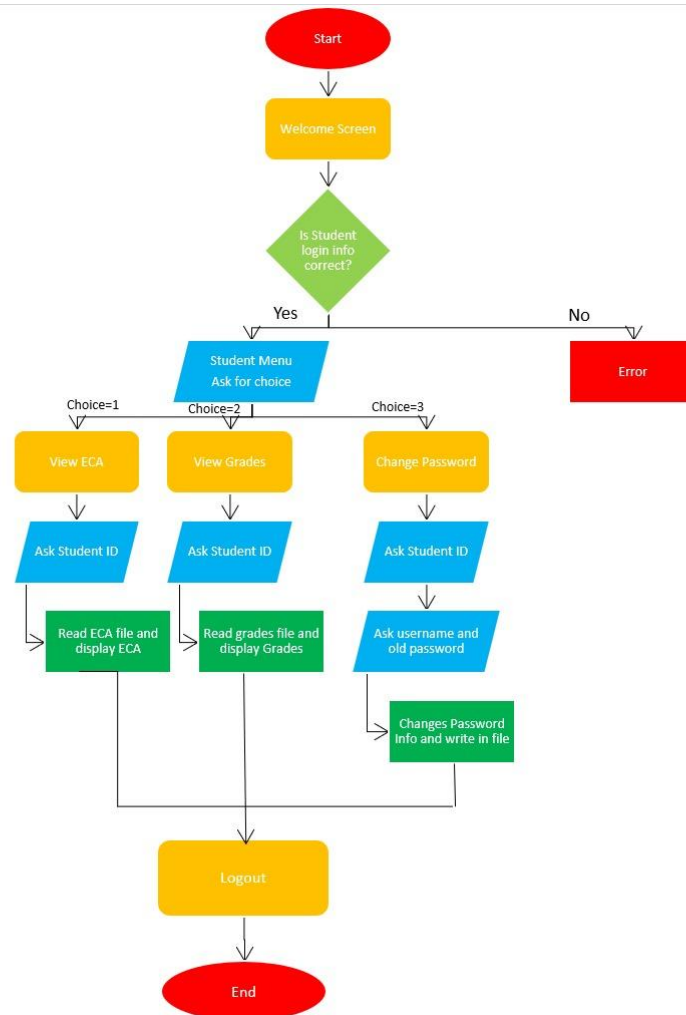
Student Menu:
1. View ECA
2. View Grades
3. Change Password
4. Logout
Enter your option: 4
Logging out from student account.
Welcome to the main home screen!
1. Login as Admin
2. Login as Student
3. Exit
Enter your option: 3
Exiting the system. Goodbye!
PS C:\Users\Shrijal\OneDrive\Desktop\fods final>
```

# FLOWCHART:

For Admin login:



For Student login:



# **LIMITATIONS:**

## **❖ Security:**

Passwords are stored as plain text, which is unsafe. They could be hashed and salted. There's no proper authentication mechanism for admin login, making it vulnerable to unauthorized access.

## **❖ Input Validation:**

User inputs aren't thoroughly checked, which can lead to problems. Converting user input to integers without proper error handling can also cause problems.

## **❖ File Handling:**

File operations lack error handling, which can result in unexpected behavior.

## **❖ Scalability:**

Managing data in files may become inefficient as the system grows. Using a database would be better for scalability and data integrity.

## **❖ Error Reporting:**

Error messages don't provide enough information to users, making it hard to understand and fix issues.

## **❖ User Experience:**

The command-line interface may not be user-friendly, especially for non-technical users. A more intuitive GUI based interface would improve usability.

## **❖ Error Handling:**

Error handling is minimal and needs improvement to provide better feedback and handle unexpected situations properly.