***FINAL PROJECT REPORT***

**Fundamentals of Data Science**



***Submitted by:***

Shreya Khadgi [Student ID: 25024631]

Swechhya Tamrakar [Student ID: 25024660]

Bsc. Computing

Kathmandu, Nepal

7 July, 2025

**Introduction:**

The Student Management System is a Python-based application designed to manage user accounts within an educational institution. The system allows administrators to register new users, modify user records, delete users, and manage various aspects of students account, such as viewing grades and extra-curricular activity details.

**Project Components:**

The project consists of several key components:

1. User Class: The User class serves as the base class for both administrators and students. It includes attributes such ad user ID, username, and role.
2. Admin Class: The Admin class inherits from the User class and provides functionality specific to administrators, such as managing user records and displaying the admin dashboard.
3. Student Class: Similar to Admin class, the Student class inherits from the User class but provides functionality tailored to students, such as updating profiles and viewing grades.
4. User Manager Class: The User Manager class handles user authentication, loading
5. Main Script (main.py): This script serves as the entry point for the application. It initializes the User Manager, loads user data, prompts users for login credentials, and redirects users to the appropriate dashboard based on their role.

**Flowchart**

**Display student dashboard with their information, grades and ECA activities**

**Function to add new account**

**Function to update and delete student’s grades and ECA.**

**Retrieve data from user.txt,eca.txt and grades.txt**

**Display admin dashboard with their data, student data manipulation and add account functions**

**Get student information based on their ID**

**Retrieve data from user.txt, eca.txt and grades txt**

**Check role from user.txt**

**Change the default password and login**

**Invalid password and re-prompt**

**Check User Id and Password from.txt**

**Enter User ID**

**And Password**

START

END

**Source code**

**1.Admin.py**

from student import Student

from user import User

class Admin(User):

    def \_\_init\_\_(self, user\_id, username):

        super().\_\_init\_\_(user\_id, username, "admin")

    def show\_admin\_dashboard(self):

        print("Admin Dashboard")

        print("1. Register a new user")

        print("2. Modify user record")

        print("3. Delete user record")

        option = input("Enter your choice: ")

        if option == '1':

            username = input("Enter new user's username: ")

            password = input("Enter new user's password: ")

            role = input("Enter new user's role (admin/student): ")

            self.register\_user(username, password, role)

        elif option == '2':

            user\_id = input("Enter user ID to modify: ")

            new\_username = input("Enter new username: ")

            self.modify\_user(user\_id, new\_username)

        elif option == '3':

            user\_id = input("Enter user ID to delete: ")

            self.delete\_user(user\_id);

        else:

            print("Invalid option.")

    def register\_user(self, username, password, role):

        with open('users.txt', 'r') as file:

            num\_users = sum(1 for line in file)

        user\_id = num\_users + 1

        if role == 'admin':

            user = Admin(user\_id, username)

        elif role == 'student':

            user = Student(user\_id, username)

        if user:

            try:

                with open('passwords.txt', 'a') as file:

                    file.write(password + '\n')

                with open('users.txt', 'a') as file:

                    file.write(f"{user\_id},{username},{role}\n")

                print("User registered successfully.")

            except FileNotFoundError:

                print("Error: Files not found.")

        else:

            print("Invalid role.")

    def modify\_user(self, user\_id, new\_username):

        try:

            # Read the contents of the file

            with open('users.txt', 'r') as file:

                lines = file.readlines()

            # Find the line corresponding to the user ID and modify the username

            found = False

            for i, line in enumerate(lines):

                parts = line.strip().split(',')

                if parts[0] == str(user\_id):

                    lines[i] = f"{user\_id},{new\_username},{parts[2]}\n"  # Replace old username with new\_username

                    found = True

                    break

            if not found:

                print("User not found.")

                return

            # Write the modified content back to the file

            with open('users.txt', 'w') as file:

                file.writelines(lines)

            print(f"Username for user ID {user\_id} modified successfully.")

        except FileNotFoundError:

            print("Error: Users file not found.")

    def delete\_user(self, user\_id):

        try:

            # Read the contents of the file

            with open('users.txt', 'r') as file:

                lines = file.readlines()

            # Find the line corresponding to the user ID and remove it

            found = False

            for i, line in enumerate(lines):

                parts = line.strip().split(',')

                if parts[0] == str(user\_id):

                    del lines[i]

                    found = True

                    break

            if not found:

                print("User not found.")

                return

            # Write the modified content back to the file

            with open('users.txt', 'w') as file:

                file.writelines(lines)

            print(f"User with ID {user\_id} deleted successfully.")

        except FileNotFoundError:

            print("Error: Users file not found.")

    def count\_users():

        # Read the contents of the file

        with open("users.txt", 'r') as file:

            lines = file.readlines()

        # Count the number of users

        num\_users = len(lines)

        return num\_users

**2.Eca.txt**

1, Outdoor Game: Football

2, Music: Piano

3, Outdoor Game: Football

4, Outdoor Game: Basketball

5, Indoor Game: Chess

6, Music: Guitar

7, Indoor Game: Table Tennis

8, Outdoor Game: Basketball

9, Indoor Game: Pool

10, Music: Violin

11, Indoor Game: Pool

12, Indoor Game: Table Tennis

13, Music: Cello

14, Outdoor Game: Basketball

15, Indoor Game: Pool

16, Outdoor Game: Swimming

17, Outdoor Game: Golf

18, Outdoor Game: Surfing

19, Outdoor Game: Badminton

20, Outdoor Game: Archery

21, Outdoor Game: Cricket

22, Outdoor Game: Skiing

23, Indoor Game: Table tennis

24, Indoor Game: Karate

25, Indoor Game: Chairball

26, Indoor Game: Darts

27, Indoor Game: Gymnastics

28, Indoor Game: Hockey

29, Indoor Game: Bowling

30, Indoor Game: Boxing

31, Music: Guitar

32, Outdoor Game: Cricket

33, Outdoor Game: Football

34, Indoor Game: Puzzles

35, Indoor Game: Card Game

36, Music: Violin

37, Outdoor Game: Basketball

38, Outdoor Game: Golf

39, Indoor Game: Chess

40, Indoor Game: Scrabble

41, Music: Drum

42, Outdoor Game: Volleyball

43, Outdoor Game: Cycling

44, Indoor Game: Dominos

45, Indoor Game: Majhong

46, Music: Flute

47, Outdoor Game: Baseball

48, Outdoor Game: Badminton

49, Indoor Game: Ping Pong

50, Indoor Game: Karoke

**3.Generate\_sample\_data .py**

import csv

# Read data from users.csv

users\_data = []

with open('users.csv', newline='') as csvfile:

    reader = csv.reader(csvfile)

    for row in reader:

        users\_data.append(tuple(row))

# Read data from passwords.csv

passwords\_data = []

with open('passwords.csv', newline='') as csvfile:

    reader = csv.reader(csvfile)

    for row in reader:

        passwords\_data.append(row[0])

# Write data to users.txt

with open('users.txt', 'w') as file:

    for user\_data in users\_data:

        file.write(','.join(user\_data) + '\n')

# Write data to passwords.txt

with open('passwords.txt', 'w') as file:

    for password in passwords\_data:

        file.write(password + '\n')

print("Data from CSV files has been written to text files successfully.")

**4.Grade.txt**

1,Math: A, Science: B, English: A, History: C, Geography: B

2,Math: B, Science: A, English: B, History: B, Geography: A

3,Math: A, Science: B, English: A, History: C, Geography: B

4,Math: B, Science: A, English: B, History: B, Geography: A

5,Math: A, Science: B, English: A, History: C, Geography: B

6,Math: B, Science: A, English: B, History: B, Geography: A

7,Math: A, Science: B, English: A, History: C, Geography: B

8,Math: B, Science: A, English: B, History: B, Geography: A

9,Math: A, Science: B, English: A, History: C, Geography: B

10,Math: B, Science: A, English: B, History: B, Geography: A

11,Math: A, Science: B, English: A, History: C, Geography: B

12,Math: B, Science: A, English: B, History: B, Geography: A

13,Math: A, Science: B, English: A, History: C, Geography: B

14,Math: B, Science: A, English: B, History: B, Geography: A

15,Math: A, Science: B, English: A, History: C, Geography: B

16,Math: B, Science: A, English: B, History: B, Geography: A

17,Math: A, Science: B, English: A, History: C, Geography: B

**5.main.py**

# main.py

from user\_manager import UserManager

# from user import User

from admin import Admin

from student import Student

def main():

    user\_manager = UserManager()

    user\_manager.load\_data()

    while True:

        username = input("Enter your username: ")

        password = input("Enter your password: ")

        try:

            user = user\_manager.login(username, password)

            if user:

                if isinstance(user, Admin):

                    user.show\_admin\_dashboard()

                elif isinstance(user, Student):

                    user.show\_student\_dashboard()

                else:

                    print("Invalid user type.")

            else:

                print("Invalid username or password. Please try again.")

        except Exception as e:

            print("An error occurred:", e)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**6.Password.csv**

7mB%Pl|9SH

0V{MzdD

9`>|m`xL

7$RZcoB

9yD6f

1F)\*NF

49!4~<

7LkuRYtDHM

8Rx<MN

8vY/XZza{

3$="C

9BeKGQtP0K

1VeG\{}"+\

6`!iFg

2)TU1PTa+

4a2~M\_

0py!2d=

2R4b?

7{'cgG

1Z$cZ

1y2#GbsdjX

6\_el(S2VK

2lv\_Y

5%d,X

83SumeK

31!+Z9<l

6(zQ1~\_BrC

1>ayt

6zl2T85

5~40SVK\X

94E=9i{aV

7EQFa

6Yar$zpi

3u7>5i\*xv(

7jun3F

2\x+6aSotn

8"vZ)v/r~\_

1VEFB#

5k<v!9G<

4%o6u!E

6ai,4Iy4\*#

6'lgR\*

9,hC0

6=>Py

7A\Hxe

4O!Fz9@N

8TyPQ<G(a

8,vQ"8

9}V/d

1GHXApass

Itzy

**7.Password.txt**

7mB%Pl|9SH

0V{MzdD

9`>|m`xL

7$RZcoB

9yD6f

1F)\*NF

49!4~<

7LkuRYtDHM

8Rx<MN

8vY/XZza{

3$="C

9BeKGQtP0K

1VeG\{}"+\

6`!iFg

2)TU1PTa+

4a2~M\_

0py!2d=

2R4b?

7{'cgG

1Z$cZ

1y2#GbsdjX

6\_el(S2VK

2lv\_Y

5%d,X

83SumeK

31!+Z9<l

6(zQ1~\_BrC

1>ayt

6zl2T85

5~40SVK\X

94E=9i{aV

7EQFa

6Yar$zpi

3u7>5i\*xv(

7jun3F

2\x+6aSotn

8"vZ)v/r~\_

1VEFB#

5k<v!9G<

4%o6u!E

6ai,4Iy4\*#

6'lgR\*

9,hC0

6=>Py

7A\Hxe

4O!Fz9@N

8TyPQ<G(a

8,vQ"8

9}V/d

1GHXApass

Itzy  
**8.Student.py**

from user import User

class Student(User):

    def \_\_init\_\_(self, user\_id, username):

        super().\_\_init\_\_(user\_id, username, "student")

    def show\_student\_dashboard(self):

        print("Student Dashboard")

        print("1. Update profile")

        print("2. View ECA details")

        print("3. View grades")

        option = input("Enter your choice: ")

        if option == '1':

            new\_username = input("Enter new username: ")

            new\_password = input("Enter new password: ")

            self.update\_profile(new\_username, new\_password)

        elif option == '2':

            self.view\_eca\_details()

        elif option == '3':

            self.view\_grades()

        else:

            print("Invalid option.")

    def update\_profile(self, new\_username, new\_password):

        try:

            # Read the contents of the file

            with open('users.txt', 'r') as file:

                lines = file.readlines()

            # Find the line corresponding to the user ID and modify the username

            found = False

            for i, line in enumerate(lines):

                parts = line.strip().split(',')

                if parts[0] == str(self.user\_id):

                    lines[i] = f"{self.user\_id},{new\_username},{parts[2]}\n"  # Replace old username with new\_username

                    found = True

                    break

            if not found:

                print("User not found.")

                return

            # Write the modified content back to the file

            with open('users.txt', 'w') as file:

                file.writelines(lines)

            print(f"Username for user ID {self.user\_id} modified successfully.")

        except FileNotFoundError:

            print("Error: Users file not found.")

    def view\_eca\_details(self):

        try:

            with open('eca.txt', 'r') as file:

                for line in file:

                    parts = line.strip().split(',', 1)  # Split only on the first comma

                    if len(parts) == 2:

                        user\_id, eca\_str = parts

                        if user\_id.strip() == self.user\_id.strip():  # Compare stripped user IDs

                            eca\_details = {}

                            eca\_list = eca\_str.split(',')

                            for item in eca\_list:

                                activity, detail = item.split(':')

                                eca\_details[activity.strip()] = detail.strip()

                            print("ECA details:", eca\_details)

                            return

                print("ECA details not found.")

        except FileNotFoundError:

            print("Error: ECA file not found.")

    def view\_grades(self):

        try:

            with open('grades.txt', 'r') as file:

                for line in file:

                    parts = line.strip().split(',', 1)  # Split only on the first comma

                    if len(parts) == 2:

                        user\_id, eca\_str = parts

                        if user\_id.strip() == self.user\_id.strip():  # Compare stripped user IDs

                            eca\_details = {}

                            eca\_list = eca\_str.split(',')

                            for item in eca\_list:

                                activity, detail = item.split(':')

                                eca\_details[activity.strip()] = detail.strip()

                            print("ECA details:", eca\_details)

                            return

                print("ECA details not found.")

        except FileNotFoundError:

            print("Error: ECA file not found.")

**9.User\_manager.py**

# user\_manager.py

from admin import Admin

from student import Student

class UserManager:

    def \_\_init\_\_(self):

        self.users = []

    def load\_data(self):

        try:

            # Load data from users.txt

            with open('users.txt', 'r') as file:

                for line in file:

                    user\_id, username, role = line.strip().split(',')

                    if role == 'admin':

                        self.users.append(Admin(user\_id, username))

                    elif role == 'student':

                        self.users.append(Student(user\_id, username))

            # Load data from passwords.txt

            with open('passwords.txt', 'r') as file:

                passwords = file.readlines()

            # Load data from grades.txt

            with open('grades.txt', 'r') as file:

                grades\_data = file.readlines()

            # Load data from eca.txt

            with open('eca.txt', 'r') as file:

                eca\_data = file.readlines()

            # Assign grades and ECA details to respective users

            for i, user in enumerate(self.users):

                user\_id = str(i + 1)

                for grades\_entry in grades\_data:

                    if grades\_entry.startswith(user\_id):

                        grades = grades\_entry.strip().split(',')[1]

                        user.grades = grades

                for eca\_entry in eca\_data:

                    if eca\_entry.startswith(user\_id):

                        eca\_details = eca\_entry.strip().split(',')[1]

                        user.eca\_details = eca\_details

        except FileNotFoundError:

            print("Error: One or more files not found.")

    def login(self, username, password):

        for user in self.users:

            if user.username == username:

                # Assuming passwords are stored in a separate file

                try:

                    with open('passwords.txt', 'r') as file:

                        passwords = file.readlines()

                        password = password.strip()

                        if passwords[self.users.index(user)].strip() == password:

                            return user

                        else:

                            return None

                except FileNotFoundError:

                    print("Error: Password file not found.")

                    return None

        return None

**10.user.py**

# user.py

class User:

    def \_\_init\_\_(self, user\_id, username, role):

        self.user\_id = user\_id

        self.username = username

        self.role = role

**11.users.csv**

id,username,role

1,Shurlock,student

2,Jaime,admin

3,Dalston,admin

4,Bendicty,admin

5,Lind,admin

6,Wilow,admin

7,Minnnie,admin

8,Darbee,admin

9,Dillie,student

10,Massimo,admin

11,Waverly,student

12,Lexis,student

13,Joleen,student

14,Adiana,student

15,Allister,admin

16,Dianemarie,admin

17,Alexina,student

18,Janina,admin

19,Sigismondo,admin

20,Ludovico,admin

21,Gwendolen,student

22,Nigel,admin

23,Nonie,student

24,Ham,student

25,Allayne,student

26,Chrissie,admin

27,Meridith,admin

28,Judd,admin

29,Piggy,student

30,Hale,student

31,Kara,admin

32,Edgardo,student

33,Doy,admin

34,Arlina,admin

35,Derwin,student

36,Flynn,admin

37,Gilligan,student

38,Paxton,student

39,Michail,admin

40,Alic,student

41,Deina,admin

42,Hatty,student

43,Tilly,student

44,Mordecai,student

45,Lon,student

46,Elnore,admin

47,Omero,student

48,Ly,admin

49,Reena,student

50,Nancy,student

**12.user.txt**

1,Shurlock,student

2,bwebs,admin

3,David,admin

4,Bendicty,admin

5,Lind,admin

6,Wilow,admin

7,Minnnie,admin

8,Darbee,admin

9,Dillie,student

10,Massimo,admin

11,absayo,student

12,Lexis,student

13,Joleen,student

14,Adiana,student

15,Allister,admin

16,Dianemarie,admin

17,Alexina,student

18,Janina,admin

19,Sigismondo,admin

20,Ludovico,admin

21,Gwendolen,student

22,Nigel,admin

23,Nonie,student

24,heyy,student

25,Allayne,student

26,Chrissie,admin

27,Meridith,admin

28,Judd,admin

29,Piggy,student

30,Hale,student

31,Kara,admin

32,Edgardo,student

33,Doy,admin

34,Arlina,admin

35,Derwin,student

36,Flynn,admin

37,Gilligan,student

38,Paxton,student

39,Michail,admin

40,Alic,student

41,Deina,admin

42,Hatty,student

43,Tilly,student

44,Mordecai,student

45,Lon,student

46,Elnore,admin

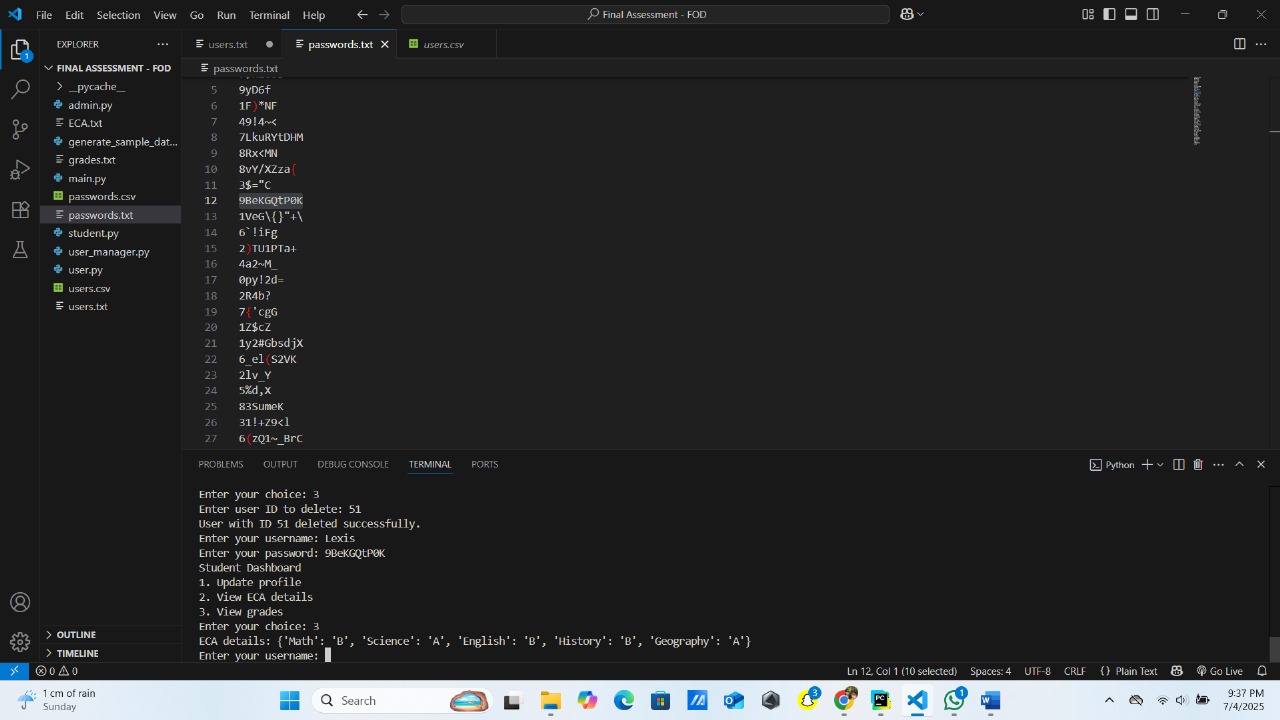
47,Omero,student

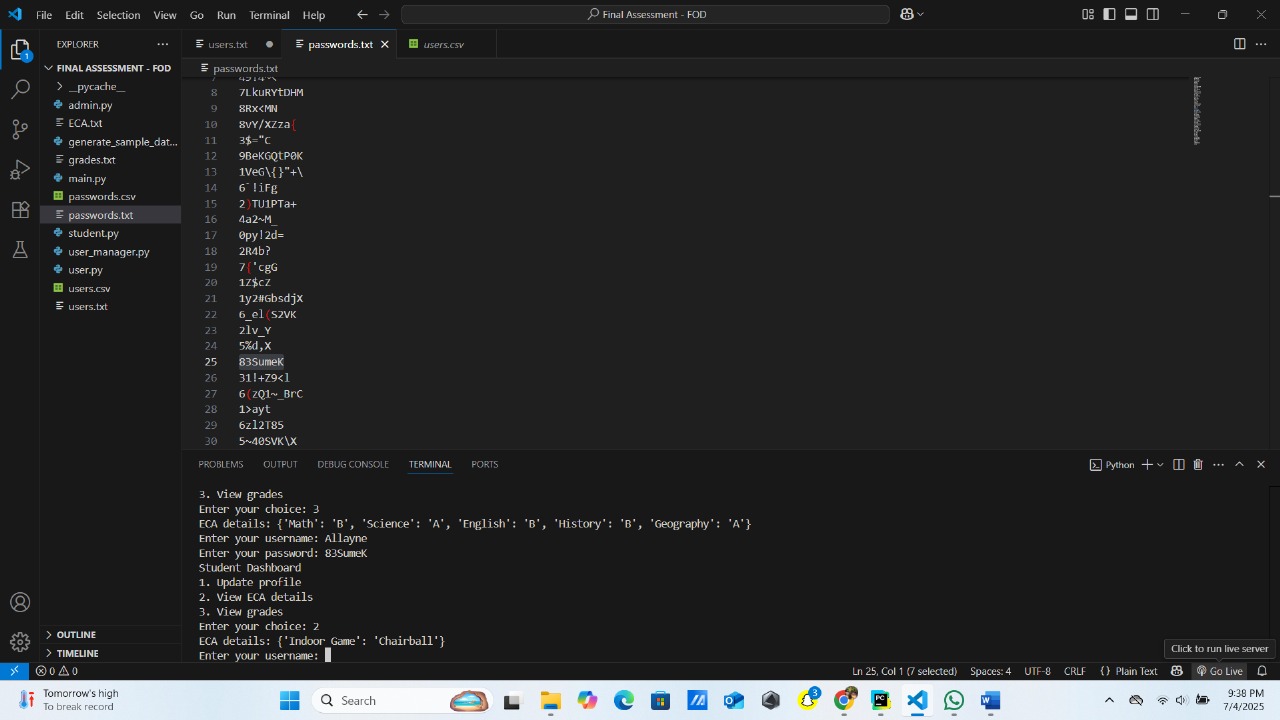
48,Ly,admin

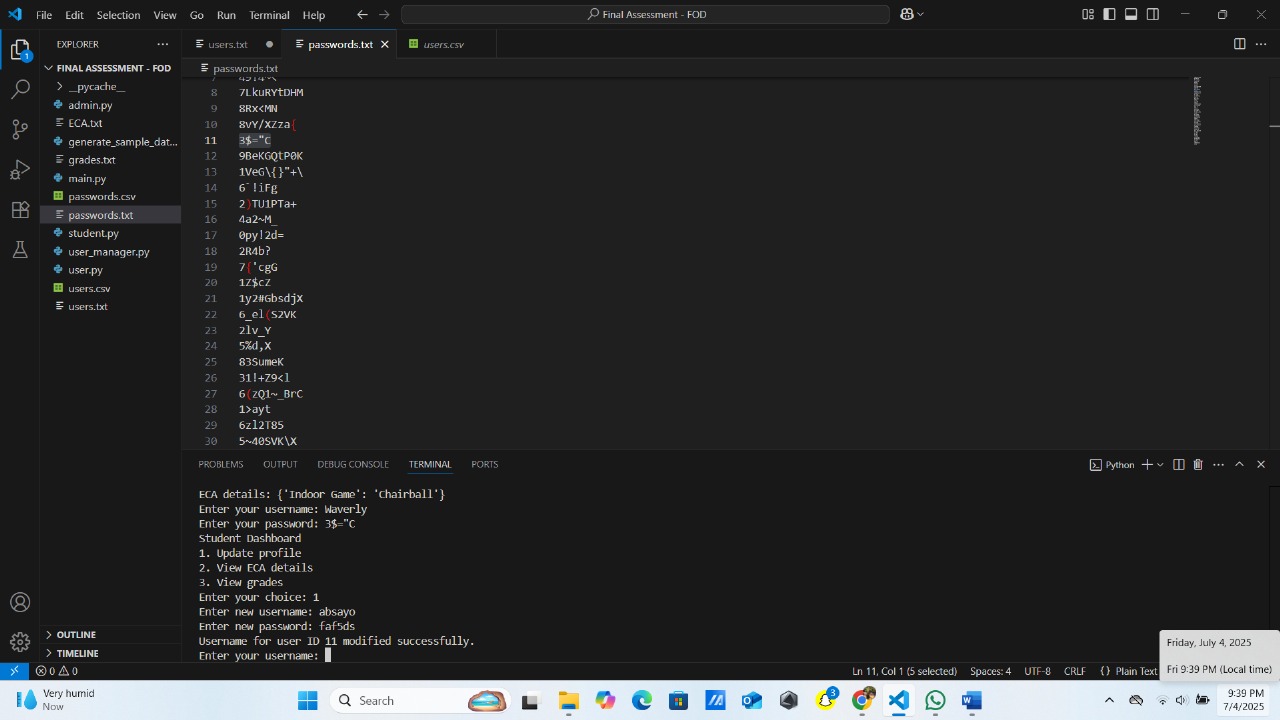
49,Reena,student

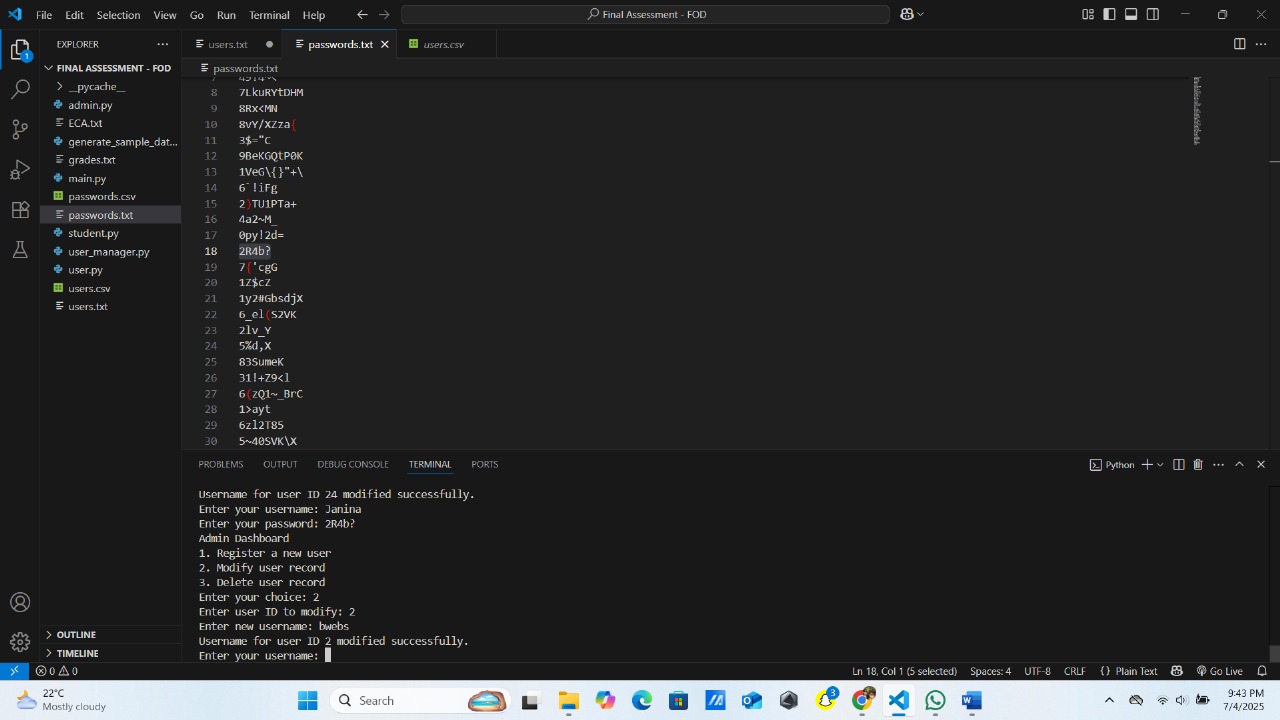
50,Nancy,student

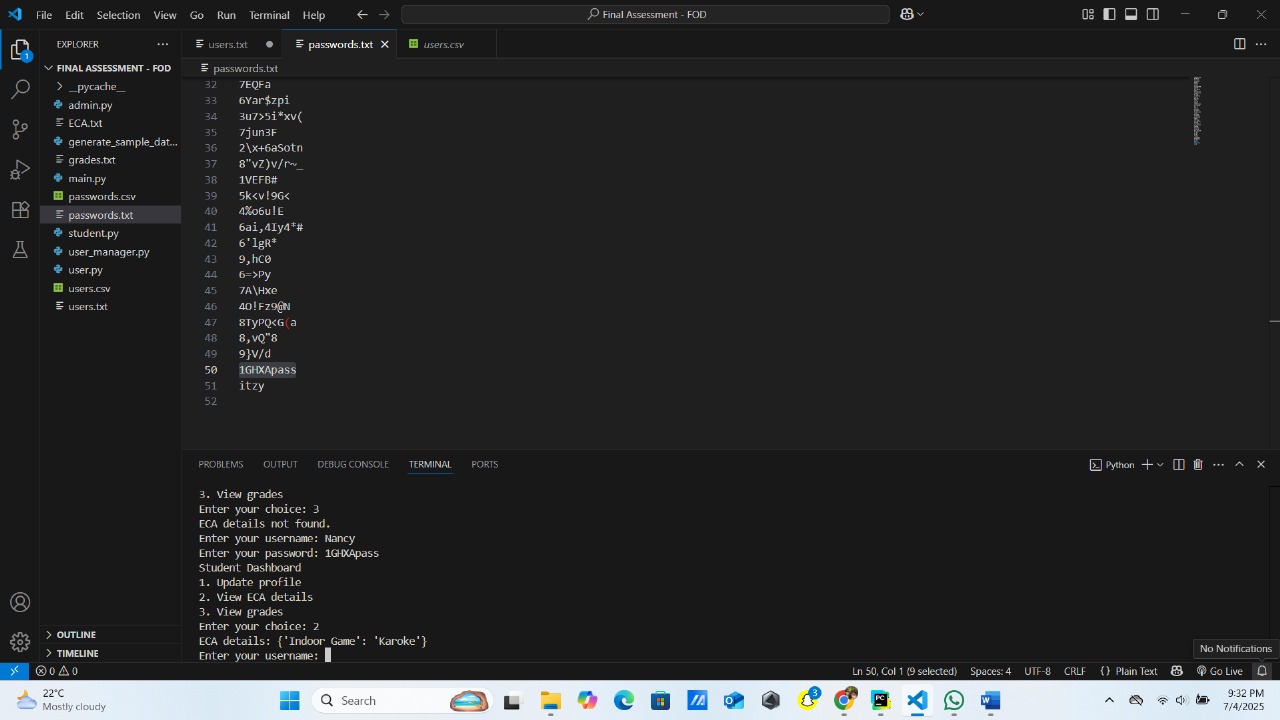
**INPUT/OUTPUTN SCREENSHOT:**

****

****

****

****

****

**Work contribution and Reflection**

1. Shreya Khadgi

Work Contribution:

* Took the lead in designing the solution/planning for the project.
* Structured files and functions in a logical and organized manner to ensure scalability and maintainability.
* Define clear responsibilities for each module or component within the system.
* Provided explaining the concepts of functions and Object-Oriented programming (OOP) to the team.

Reflection:

* Recognized the importance of proper solution design in ensuring the success of the project.
* Found that structuring files and functions early on saved time and effort during development.
* Learned valuable lessons about the importance of planning and architecture in software development projects.
* Found that explaining complex concepts like OOP in a simple and understandable way was challenging but rewarding.
* Enjoyed the opportunity to share knowledge and help team members grow their skills in programming concepts.

1. Swechhya Tamrakar

Work Contribution:

* Focused on implementing mechanism for handling files within the project, guaranteeing efficient data reading and writing process.
* Top of Form
* Developed error-handling strategies to anticipate and manage potential issues during file operations.
* Collaborated with team members to integrate error-handling mechanisms smoothly into the project’s workflow.
* Ensured that all team members understood the fundamentals of functions and OOP principles.

Reflection:

* Found that mastering file handling techniques required attention to detail and thorough testing to ensure reliability.
* Realized the importance of error handling in preventing unexpected program crashes and maintaining data integrity.
* Learned valuable problem-solving skills by addressing various file-related errors and challenges encountered during development.
* Recognized the importance of strong foundational knowledge in functions and OPP for building robust software systems.