Module Title

Fundamentals of Data Science

Assessment Weightage & Type

40% of Coursework & Regular

Year

2025

Student Name: NIRVIK K.C.

UWE ID: 25024649

Assignment Due Date: July 7, 2025

Assignment Submission Date: July 7, 2025

FINAL ASSESSMENT PROJECT

Module Details

Module Code	UFCFK1-15-0
Module Title	Fundamentals of Data Science
Module Tutors	Saurav Gautam
Year	2025 (Regular)/FebIntake
Component/Element Number	Portfolio
Weighting	40%

Dates

Submission Date	**07-July-2025**
Submission Place	Backboard
Submission Time	23:59
Submission Notes	Submit Gitlab URL

Task 1 (30: basic requirements + 10: bonus points for creativity)

Objective:

Build a **Student Profile Management System** to handle student personal information, examination data, and extracurricular activities (ECA).

Your program must use four text files:

- **users.txt**: Stores registered users' personal information, including their roles (admin/student).
- **grades.txt**: Contains marks for five subjects for students.
- **eca.txt**: Details of extracurricular activities for each student.
- **passwords.txt**: Stores usernames and their passwords (use simple storage for learning purposes)

Basic Requirements:

1. Implement a login system that validates usernames and passwords.
    - Display an error and re-prompt for incorrect credentials.
    - Differentiate between admin and student roles upon login.
2. **Admin Role**:
    - Add new users with unique IDs.
    - Update, modify, or delete student records (personal, grades, and ECA).
    - Generate useful insights from the data (e.g., average grades per subject, most active students in ECA).

3. **Student Role**:
    - Update their profile information.
    - View personal details, ECA participation, and exam grades.

4. **Ensure the program demonstrates:**
    - Object-Oriented Programming (OOP) concepts (classes, inheritance, etc.).
    - Functions for modular design.
    - File handling for reading/writing data.
    - Exception handling for error-prone sections.
5. **Creativity is rewarded for features like:**
    - Visualizing data (e.g., plotting grades).
    - Improved user interface or additional functionalities.

Task 2 (Optional Advanced Feature – 10 Bonus Points)

Implement a Performance Analytics Dashboard:

For admin users, create an analytics module that provides:

- **Grade Trends**: Display trends for students' grades using charts.
- **ECA Impact**: Correlate ECA involvement with academic performance.
- **Performance Alerts**: Identify students performing below a specific threshold and suggest interventions.

Use libraries such as **matplotlib** or **pandas** for data visualization and analysis.

Submission Requirements:

- Submit the source code.
- Include sample input/output to demonstrate the functionality.
- Submit a written report describing the project, relevant flowcharts, test output screen screenshots.
- Work contribution and reflection of project.
- Presentation slides.

Table of Content

Contents

INTRODUCTION

Smooth management of student information is fundamental in the modern learning environment. A Student Profile Management System is a program application which processes, saves, and manages student information, automating work and allowing the educational institution to focus on providing an improved learning experience.

Figure 1 : Student Management System (Cloud Assess)

Student Management System is a computer program that assists educational institutions in managing information, academic records, and extracurricular activities of students so that administrators and teachers could track the progress of students, manage admissions, and maintain contact with students and parents.

Key Features of Student Management System

Student Management System is a holistic tool for assisting learning and development of students. It separates various user roles, giving users security and ease of use. Students can edit their own personal information, including contact details and school records as the system includes a profile management. It also includes academic performance tracking, where the students' grades in different courses can be tracked and analyzed. Participation in extracurricular activities is also tracked by the system to provide students and administrators with critical feedback. They can generate reports and analysis from the system, for instance, average scores, academic achievement trends, and most active students in extracurricular activities, that can be applied in decision-making and resource management.

Applications of Student Management System

The Student Profile Management System can be utilized by schools to increase administrative efficiency, reduce paperwork, and improve student-parent communication. It aids early identification of students requiring additional assistance, enabling early intervention and resource assignment. The Student Management System also supports data-driven decision-making, enabling informed curriculum planning, resource planning, and student engagement strategies. It really enhances the learning experience by bringing academic records and profile management within easy reach.

The Student Management System is a critical component of the modern-day educational system, enabling effective management of data and the facilitation of a general quality of educational experience. It is used administratively and enables students to become masters of their learning experiences.

Use of Programming Language

The project showcases the development of a Student Profile Management System using python and its libraries.

Python is an object-oriented, high-level, and interpreted programming language that was developed by Guido van Rossum. It is very portable and cross-platform on UNIX, Mac, and Windows. Python is scalable, portable, maintainable, and simple for starters, with better code organization and provision for large programs compared to shell programming. It is robust owing to its portability and interactivity.

Variables in Python

Variables in Python are used for storing data values, defined with the assignment operator "=". Variables can be of any data type and can be assigned dynamically during runtime. Variables can be assigned new values whenever required, and their values can be printed or used in Python program operations.

Data Types in Python

Data types in Python are integer, float, boolean, string, list, tuple, and dictionary. Each of these has particular attributes and methods associated with them, whereas Python itself determines the data type according to the value given to a variable. For example, a number without decimal points is an integer data type, a number with decimal points in a float data type, whereas a value within quotation marks is a string data type.

Object Oriented Programming (OOP) in Python

Python's object-oriented style of programming focuses on classes, objects, methods, and attributes. Encapsulation, inheritance, and polymorphism are key features that facilitate organized, reusable, and maintainable code. Encapsulation hides implementation, and inheritance and polymorphism remove duplication and make code more flexible. Python's OOP simplifies programming since it facilitates modularity, reusability, and maintainability of code to enable it to be more effective and efficient.

Numbers computing in Python

Python use three types of numbers: integers, floating-point numbers, and complex numbers. It supports basic arithmetic operations, which have built-in mathematical functions in its math module. It supports various number systems like binary, octal, and hexadecimal number system. It is a good choice for scientific and numerical computing due its ability to handle large datasets, machine, and artificial intelligence.

User Defined Function

The keyword used in Python to create user-defined functions is “def” This allows program to accept parameters, return values, or modify global variables. These functions enhance code readability and reduce data duplication and help in debugging. Python also supports passing of functions as arguments and returning those functions, eventually helping in code reusability.

File Handling in Python

File handling in Python is versatile, it supports various types of file and perform operations across different platforms like Linux, Mac, and Windows. It allows for persistent data storage by performing operations like creating, writing, reading, and deleting files. Common modes include “r” (read mode), “w” (write mode), and “a” (append, adds data to the end of the file). The method writes() writes certain data to a file, while the read() method read the content in the file. For closing the files, using close() method to release system resources and using try…finally block ensures the file are enclosed even if error occurs.

Exception Handling in Python

Exception handling facility is crucial for managing errors at program runtime. It includes the try-except block, except blocks, finally block, and raise keyword. The try-except block catches and handles exceptions raised when an error occurs, while the finally block runs regardless of exception outcome. The raise keyword allows explicit raising of exceptions to indicate error conditions or trigger desired exception types. This ensures stable and error-free Python code, making software applications more reliable and user-friendly.

Python Libraries

Python libraries are modules of code that offer various functions such as input/output, data visualization, internet data retrieval, operations related in operating system, scientific calculation, and data manipulation and management.

Python libraries are sets of code in modules, including documentation, classes, and functions. As complexity of code increases, there is no need to rewrite complex code. Python libraries make these modules readily available, allowing code to be shared across different programs without invoking methods repeatedly. When a library is imported, the program linker looks for and utilizes its functionality. If a library does not exist in the Python runtime, it is installable.

Important Python Libraries

1) Matplotlib

Matplotlib library is a very crucial library in Python for data visualization that enables the creation of static, animated, and interactive plots. It possesses functions like plot(), scatter(), bar(), stem(), and step() that can be imported using the import keyword.

2)Pandas

Pandas, a very important tool in Data Sciences, is a high-performance, powerful, flexible, and easy-to-use open-source data manipulation and analysis library based on the Python programming language and NumPy. The library provides an extensive range of functions and can be utilized by installing the Pandas package.

3) NumPy

NumPy is a library for mathematical computation for large-scale numerical computations and has support for multi-dimensional and big arrays. It offers advanced functions, random number generators, linear algebra operations, and Fourier transform. NumPy supports various hardware and computing platforms and has interoperability with distributed, GPU, and sparse array libraries. Installation with the NumPy library is feasible.

4) NetworkX

The NetworkX library is a Python software package for the creation, manipulation, and study of the structure, dynamics, and functions of graphs and complex networks. It is a collection of data structures and standard graph algorithms, extended with various analysis tools for network structure.

5) TensorFlow

TensorFlow is another crucial Python library for large calculations. It offers a flexible ecosystem of tools, libraries, and community resources for researches and developers to push for machine learning operation.

6) Requests

The requests library enables the sending of HTTP/1.1 requests, adding headers, form data, multipart files, and parameters using simple dictionaries in Python, and receiving response data.

For this project, I am going to use such as libraries matplotlib and pandas for data visualization and analysis.

OBJECTIVE

- To implement a login mechanism for both students and admins inferace.
- To provide different functionalities for profile management for profile management, data analysis, and reporting.
- To utilize Object-Oriented Programming (OOP) concepts to ensure modularity and maintainability.
- To use the concept of file handling for storing and retrieving customer and product information.
- To incorporate data visualization to enhance user experience and insights.
- To implement an innovative solutions and extra features.
- To perform some basic error handling, such as checking for valid input to prevent blank or null entries and restrict invalid formats.
- To ensure the system can give fast responses even with large datasets.
- To execute a comprehensive project using the concepts gathered during the course duration.

SYSTEM DESIGN

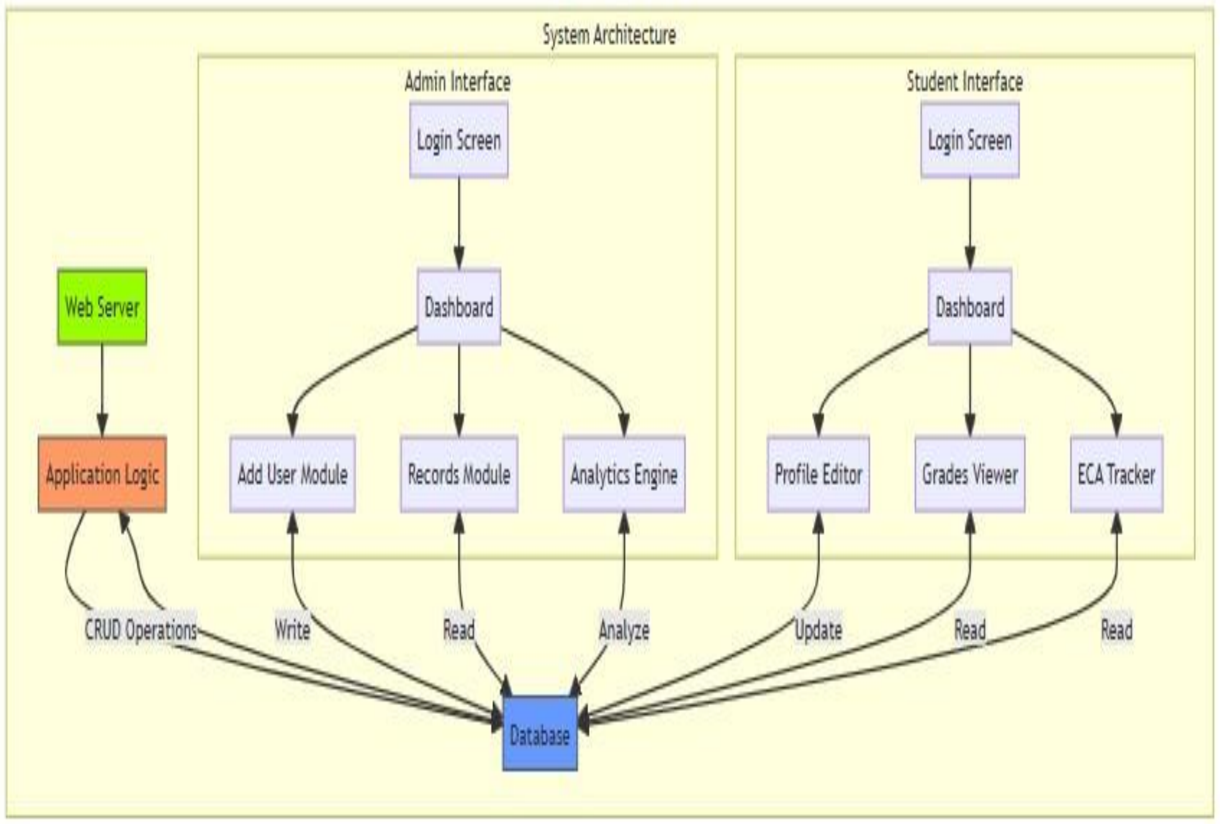


Figure 2 : System Design for the Student Profile Management System (HIX AI)

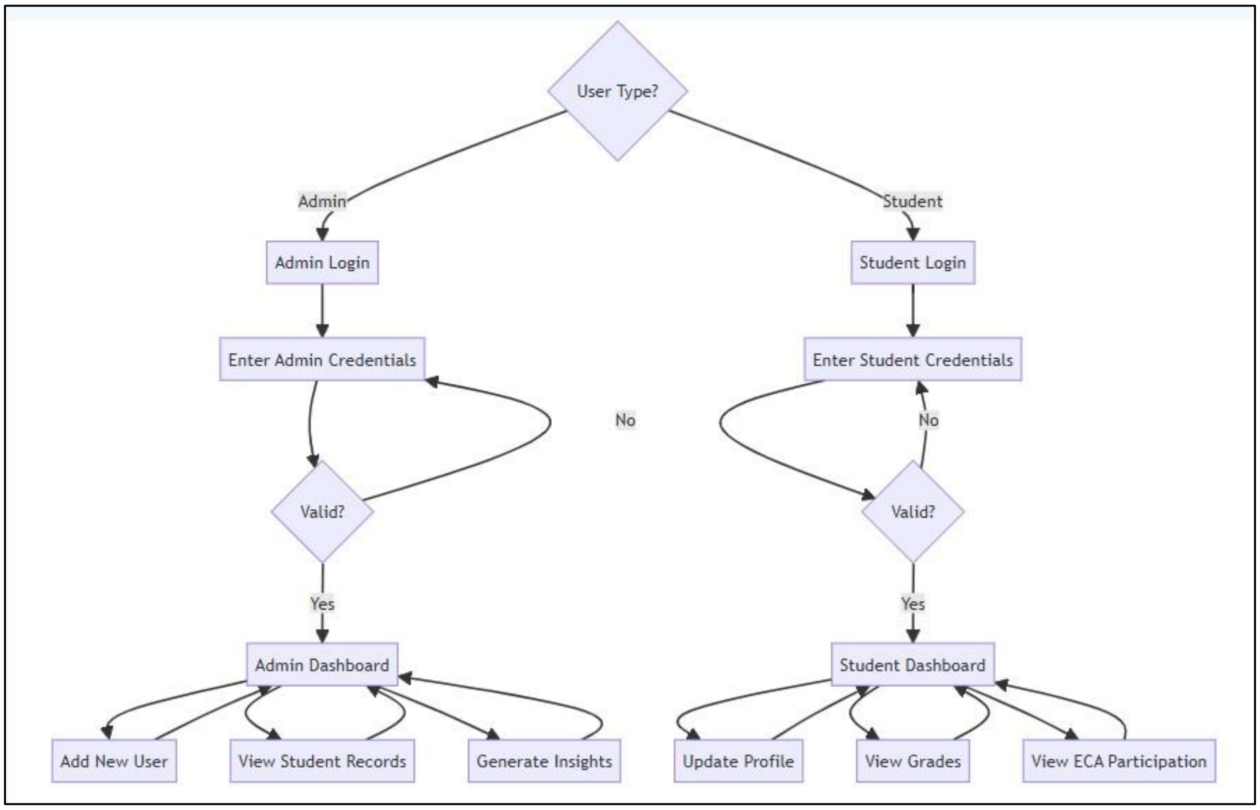


Figure 3 : Flowchart for Student Profile Management System (HIX AI)

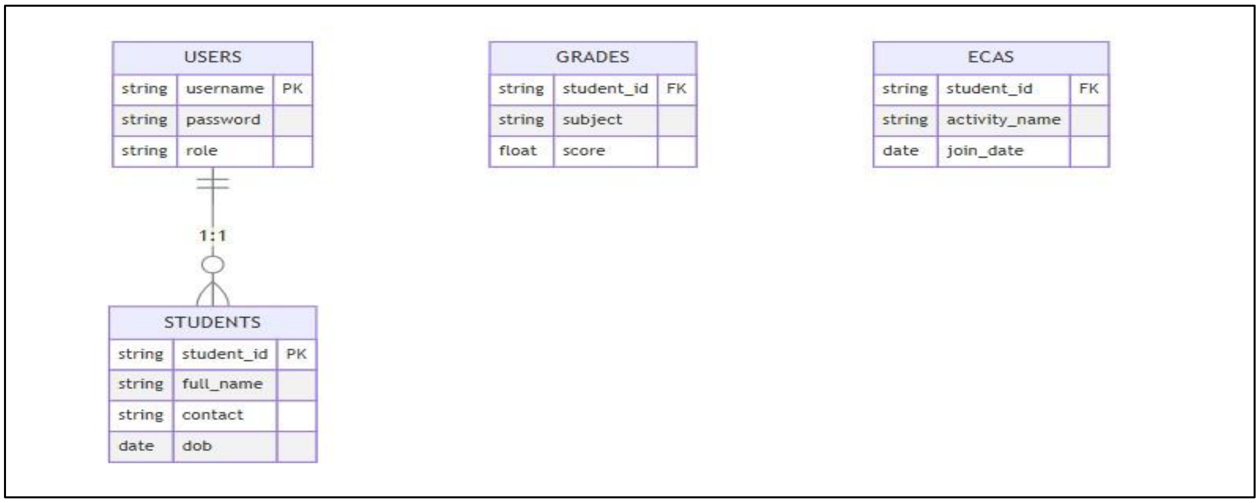


Figure 4 : Database Schema (HIX AI)

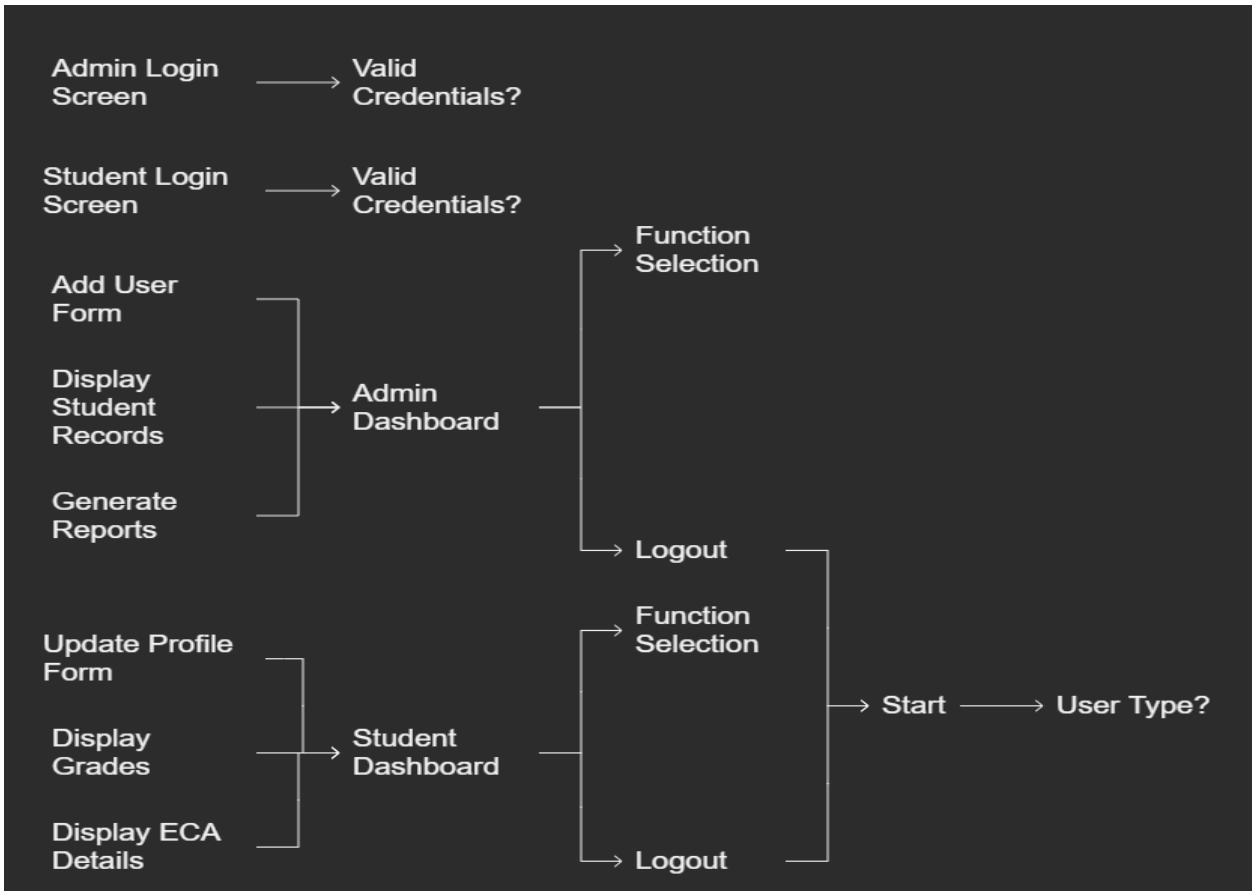
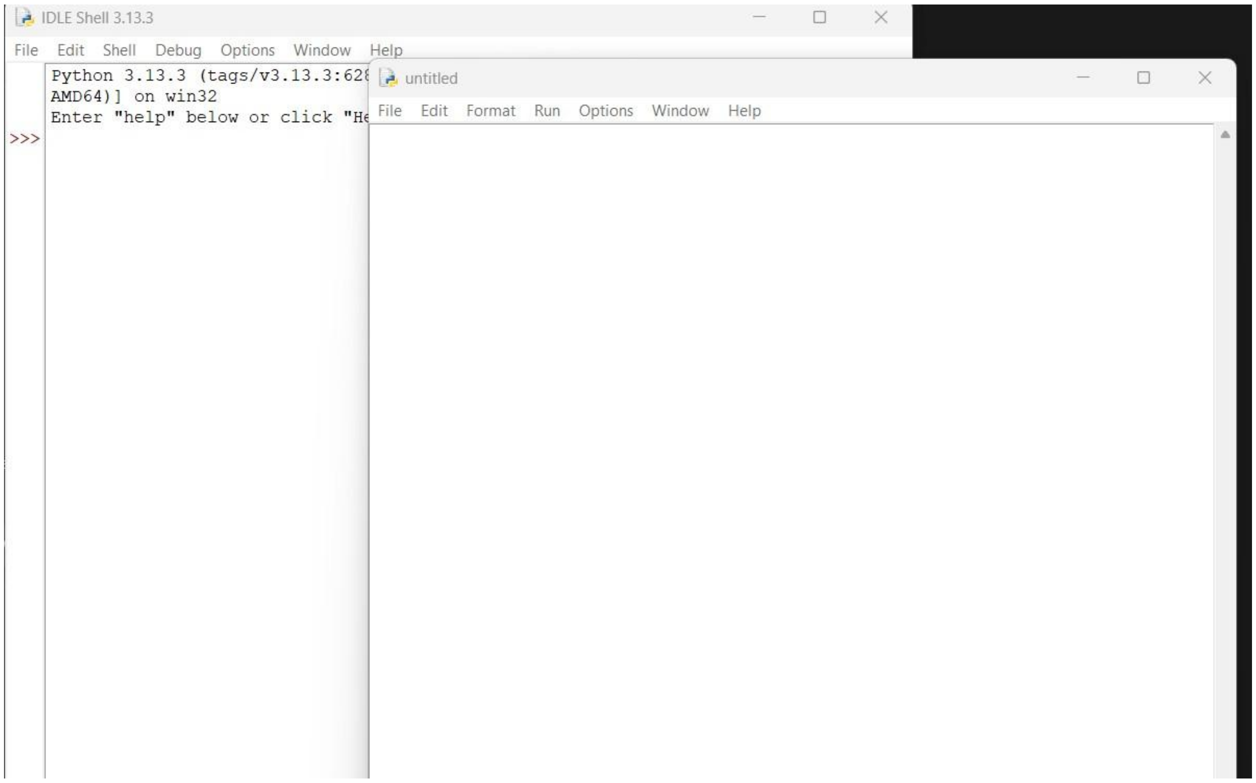


Figure 5 : Flowchart for the flow of entire program (Napkin AI)

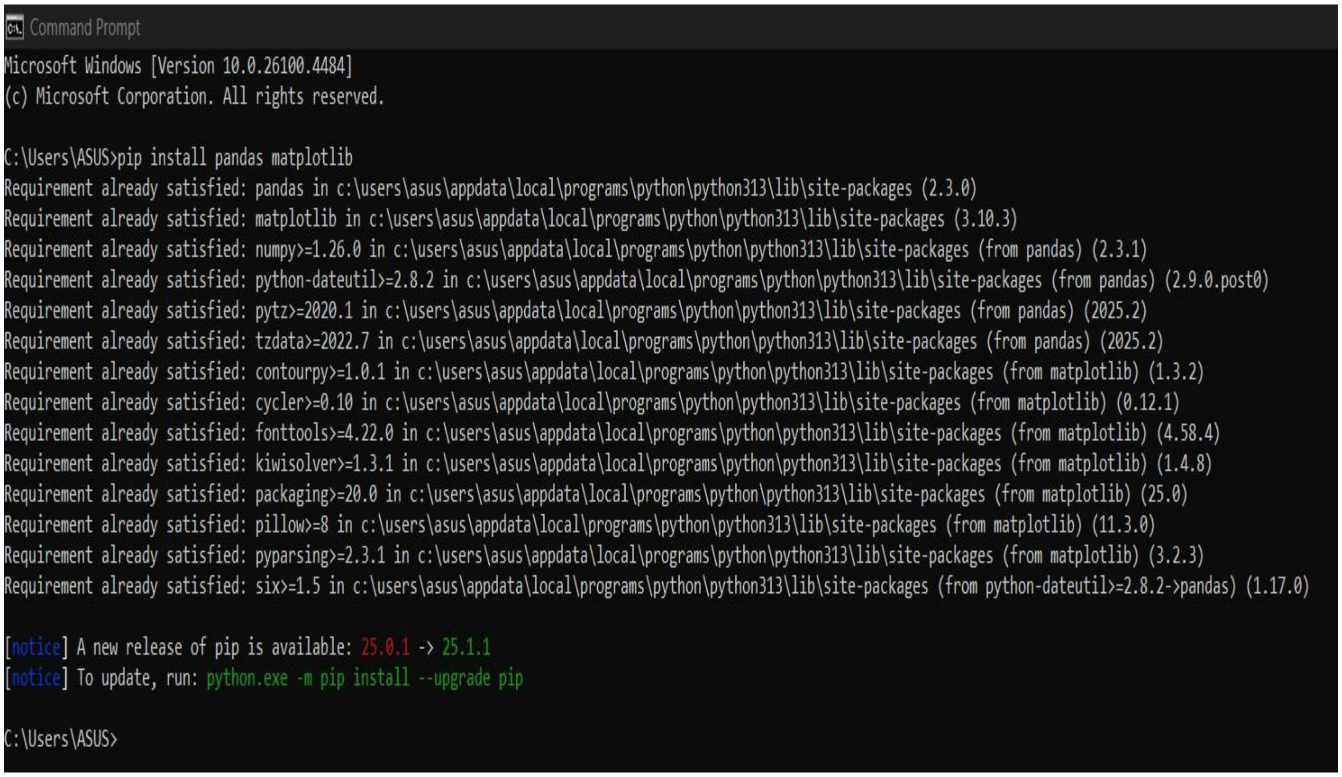
IMPLEMENTATION

Python can be executed using the command line or integrated environment (IDE) for writing and execution of code. So, I have using IDLE Python (3.13 64-bit) for wring the python code.


IDLE Shell 3.13.3
File Edit Shell Debug Options Window Help
untitled
File Edit Format Run Options Window Help

```
Python 3.13.3 (tags/v3.13.3:62
AMD64)] on win32
Enter "help" below or click "He
>>>
```

Figure 6 : IDLE Python used for writing Python program and use libraries



```
Microsoft Windows [Version 10.0.26100.4484]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS>pip install pandas matplotlib
Requirement already satisfied: pandas in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (2.3.0)
Requirement already satisfied: matplotlib in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (3.10.3)
Requirement already satisfied: numpy>=1.26.0 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2.3.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2025.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (1.3.2)
Requirement already satisfied: cycler>=0.10 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (4.58.4)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (1.4.8)
Requirement already satisfied: packaging>=20.0 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (3.2.3)
Requirement already satisfied: six>=1.5 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)

[notice] A new release of pip is available: 25.0.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\ASUS>
```

Figure 7 : Installed Python Libraries pandas and matplotlib

Class and Methods Used:

User class:

The User Class is a base class that represents a generic user in a system, containing common attributes and methods shared by administrators and students. It includes attributes like username, password, and role, and methods like login() and logout() for user authentication and session management.

Admin Class:

The Admin Class is a class that represents an administrator, inheriting from the User class. It includes methods like add_student(), modify_student(), and generate_report() for administrative tasks, allowing the admin to add new student records, update existing records, and generate performance reports.

Student Class:

The Student Class extends from User Class and describes a student with attributes studentid, grades, and ecaparticipation. With methods such as viewgrades(), updateprofile(), and manageeca() for displaying grades, fielding updates of someone’s personal details in the system, and managing extracurricular activities. These attributes would mostly relate to identifying the individual uniquely, their academic performance, and who they were engaged in with respect to the extras that they would be involved in.

DataHandler Class:

The DataHandler class handles file operations that involve user data, user grades, and user extra-curricular activities. The class contains methods like readdata(), writedata(), and updatedata() that read saved data from text files, allow saving of user information, and the ability to update saved records.

Object-Oriented Programming (OOP) Concepts Used

Encapsulation:

Each class encapsulates its data and functionality, thus controlling direct access to its internal state. For instance, the User class handles the login process internally, in which extracting information such as a password would not be possible.

Inheritance:

The Admin and Student classes descend from the User class and correlate in functionality with the User class appropriately. So, if both the Admin and Student class wanted to share the same attributes and functions as the User class, they would only have to implement their specific functionality. Therefore, the Admin and Student classes exhibit reusability while preventing redundancy.

Polymorphism:

The system allows it to define the same methods in different classes (e.g., login() in both the Admin and Student classes) and be implemented differently across classes. Thus, the interaction of different types of users with the system would be flexible.

Abstraction:

The DataHandler class abstracts the complexity of the file handler by providing a simple interface for reading from and writing to a file. So, when a user is using that class, he or she does not need to know anything about how the file operates.

File Handling Used in the Program

File Handling is an important aspect of this management system, which leverages user data, grading, and ECA all managed in text files. Student Management System supports the following 3 file actions:

Read Data - The readdata() method in the DataHandler class reads text files, and fills the corresponding objects. For example, as the program loads, we could read users.txt, which gives the ability to create User objects for all the registered users.

Write Data - The writedata() method writes the user information, grades, and ECA participation back to their respective text files, so that we are able to use that data at a later time (data persistence). As an example, if an admin added another student, the writedata() method would write an additional row in students.txt.

Update Data - The updatedata() method updates records in text files, based on updates from the admin, or students. For example, a Student could have updated their grades, or personal information.

Error Handling

The implementation of error handling is important correct input validation and operations of file.

Input Validation:

The application validates its inputs, for example, that grades are numbers and within the valid range. When a wrong grade is input by the user, the program prompts the user to input again, so that wrong data is not being stored.

File Operation Exceptions:

The program catches exceptions that occur when working with files, for example, a missing file or a read/write error. For example, if the grades. txt does not exist, instead of getting crashed the application can handle this issue and can prompt the user about this missing file.

User Friendly Error Messages:

Easy Error Messages: The software offers clear error messages, making it easy for users to see what went wrong and where to fix it. For example, if a user attempts to login with incorrect login details, the software can display warning message, like "Invalid username or password. Please try again."

User Interaction

Operations are managed via command-line console interface (or by using a GUI). The following aspects are included:

Login Screen:

A place where users enter their identification to get to the system. The system performs a check of these credentials, then it redirects the user to their corresponding dashboard (either admin or student). For instance, when an administrator logs in, they are directed to the admin dashboard where they can work on student records.

Admin Dashboard:

Administrators can insert or edit student entries, train reports and see analytics. Both of these systems feature easy to use options for each of these functions.

Student Dashboard**:**

Students can view grades, update profile and manage eca activities. The dashboard shows all these options clearly so students can interact with the system.

Feedback Mechanism:

The program gives feedback for successful operations (e.g. "Profile updated successfully") and error messages for invalid actions.

Functions Used in the Program

The system has several functions each for a specific purpose:

User Functions:

login():

Logs in users based on their credentials. This function checks the entered username and password against the stored values and grants access if it matches.

logout():

Ends the user session so sensitive information won't be accessible after the user is done using the system.

Admin Functions:

add_student():

Adds a new student record to the system. This function gets student information and calls the write_data() method to save it to the file.

modify_student():

Updates existing student information. This function gets the current data, allows admin to edit and then saves the updated information.

generate_report():

Generates report based on student performance. This function aggregates data and formats it for viewing, shows overall academic performance.

Student Functions:

view_grades():

Shows student grades. This function gets the grades from the data storage and shows it in a user-friendly format.

update_profile():

Allows students to update their profile. This function gets new data from the user and calls the update_data() method to save the changes.

manage_eca():

Manages eca participation. This function allows students to add or remove eca and updates the records.

Data Handling Functions:

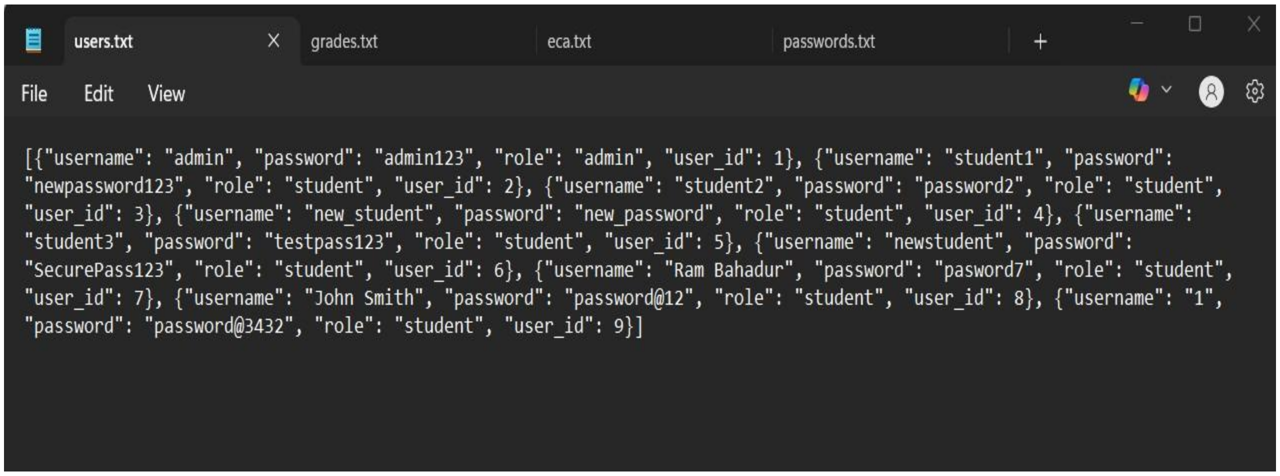
read_data(): Reads data from text files and populates objects. This function is called during the initialization of the program to load existing data.

write_data(): Writes updated data back to text files. This function is called whenever changes are made to save the latest information.

update_data(): Modifies existing records in text files. This function is used when admin or student updates their information.

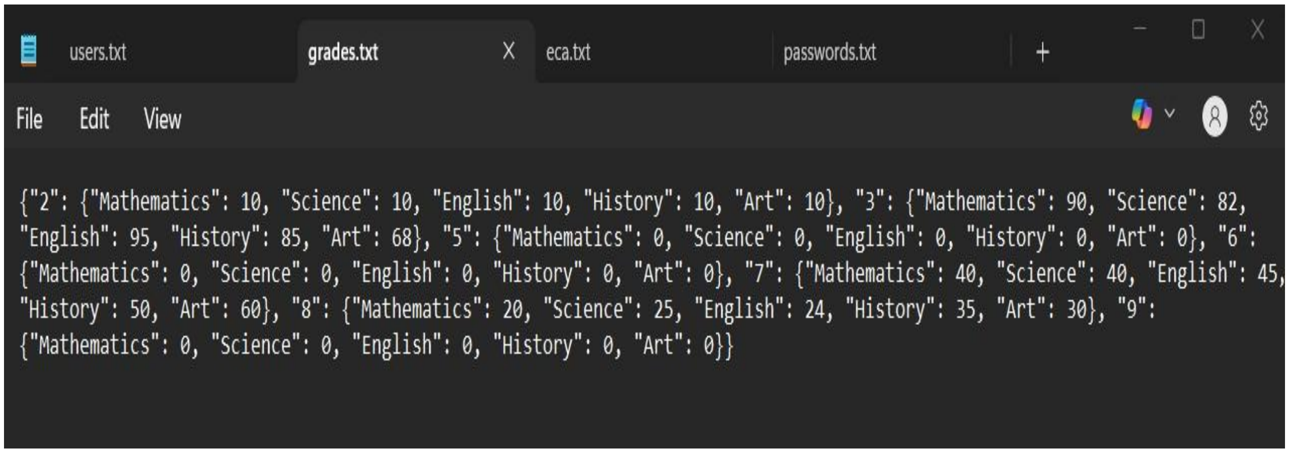
SAMPLE OUTPUT

Four text files:



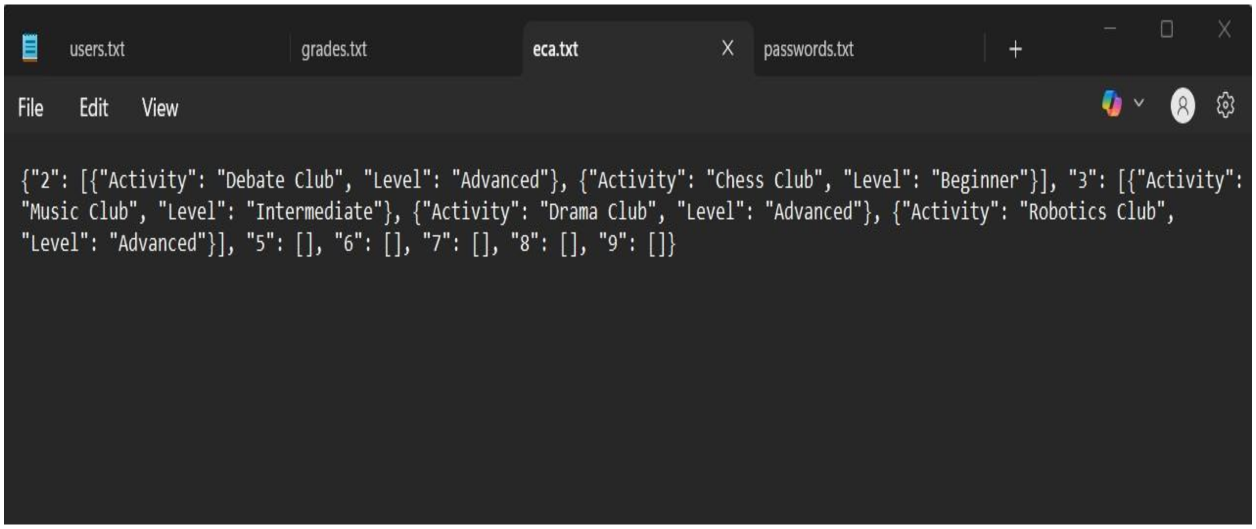
```
[{"username": "admin", "password": "admin123", "role": "admin", "user_id": 1}, {"username": "student1", "password":
"newpassword123", "role": "student", "user_id": 2}, {"username": "student2", "password": "password2", "role": "student",
"user_id": 3}, {"username": "new_student", "password": "new_password", "role": "student", "user_id": 4}, {"username":
"student3", "password": "testpass123", "role": "student", "user_id": 5}, {"username": "newstudent", "password":
"SecurePass123", "role": "student", "user_id": 6}, {"username": "Ram Bahadur", "password": "pasword7", "role": "student",
"user_id": 7}, {"username": "John Smith", "password": "password@12", "role": "student", "user_id": 8}, {"username": "1",
"password": "password@3432", "role": "student", "user_id": 9}]
```

Figure 8 : users.txt file that stores personal information (admin/student)



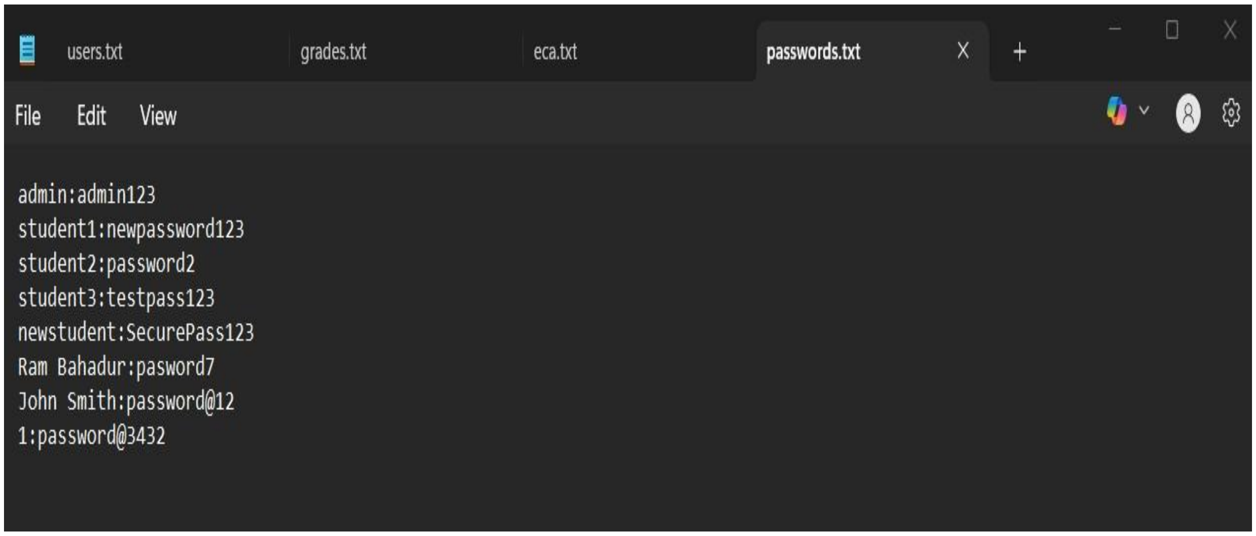
```
{"2": {"Mathematics": 10, "Science": 10, "English": 10, "History": 10, "Art": 10}, "3": {"Mathematics": 90, "Science": 82,
"English": 95, "History": 85, "Art": 68}, "5": {"Mathematics": 0, "Science": 0, "English": 0, "History": 0, "Art": 0}, "6":
{"Mathematics": 0, "Science": 0, "English": 0, "History": 0, "Art": 0}, "7": {"Mathematics": 40, "Science": 40, "English": 45,
"History": 50, "Art": 60}, "8": {"Mathematics": 20, "Science": 25, "English": 24, "History": 35, "Art": 30}, "9":
{"Mathematics": 0, "Science": 0, "English": 0, "History": 0, "Art": 0}}
```

Figure 9 : grades.txt file which contains marks for five subjects



```
{"2": [{"Activity": "Debate Club", "Level": "Advanced"}, {"Activity": "Chess Club", "Level": "Beginner"}], "3": [{"Activity":
"Music Club", "Level": "Intermediate"}, {"Activity": "Drama Club", "Level": "Advanced"}, {"Activity": "Robotics Club",
"Level": "Advanced"}], "5": [], "6": [], "7": [], "8": [], "9": []}
```

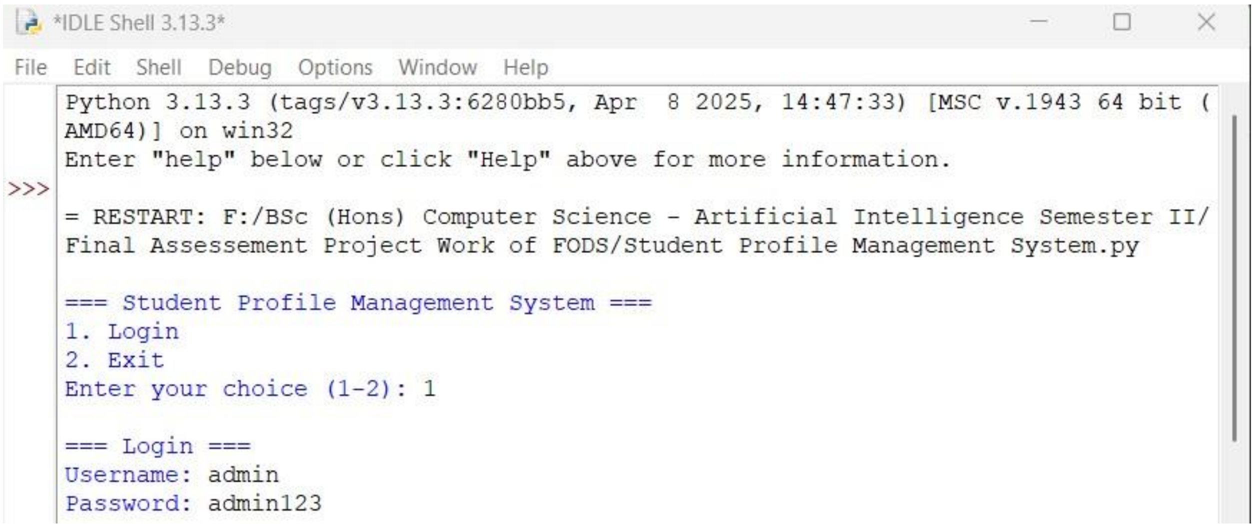
Figure 10 : eca.txt file which stores extra circular activities of each student



```
admin:admin123
student1:newpassword123
student2:password2
student3:testpass123
newstudent:SecurePass123
Ram Bahadur:pasword7
John Smith:password@12
1:password@3432
```

Figure 11 : password.txt file stores username and their passwords

Scenario 1: Log in as admin

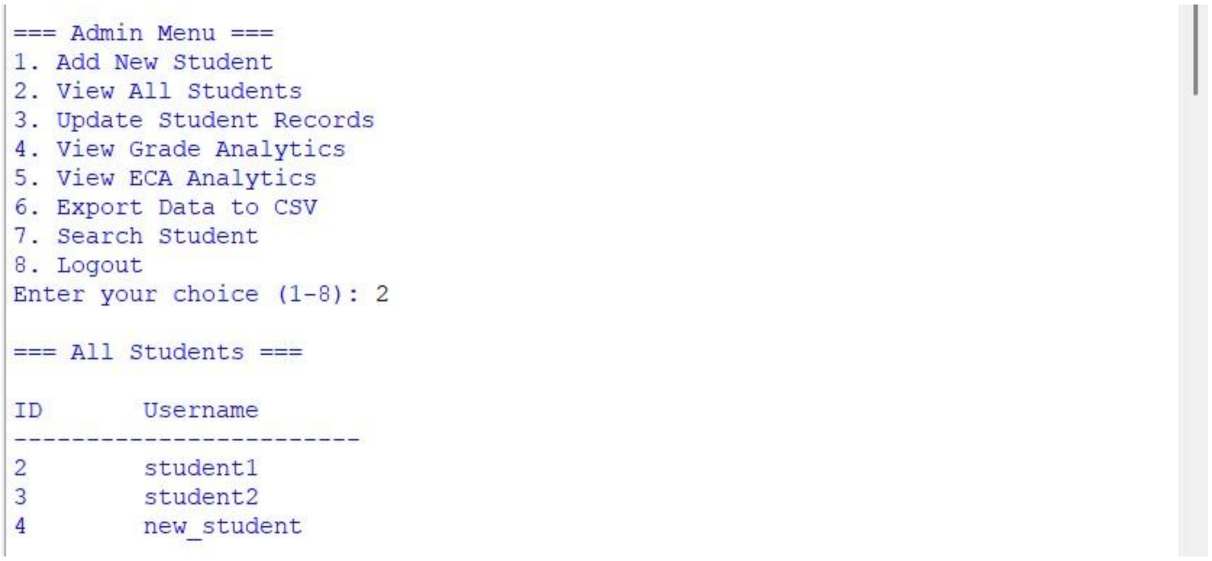


IDLE Shell 3.13.3

File Edit Shell Debug Options Window Help

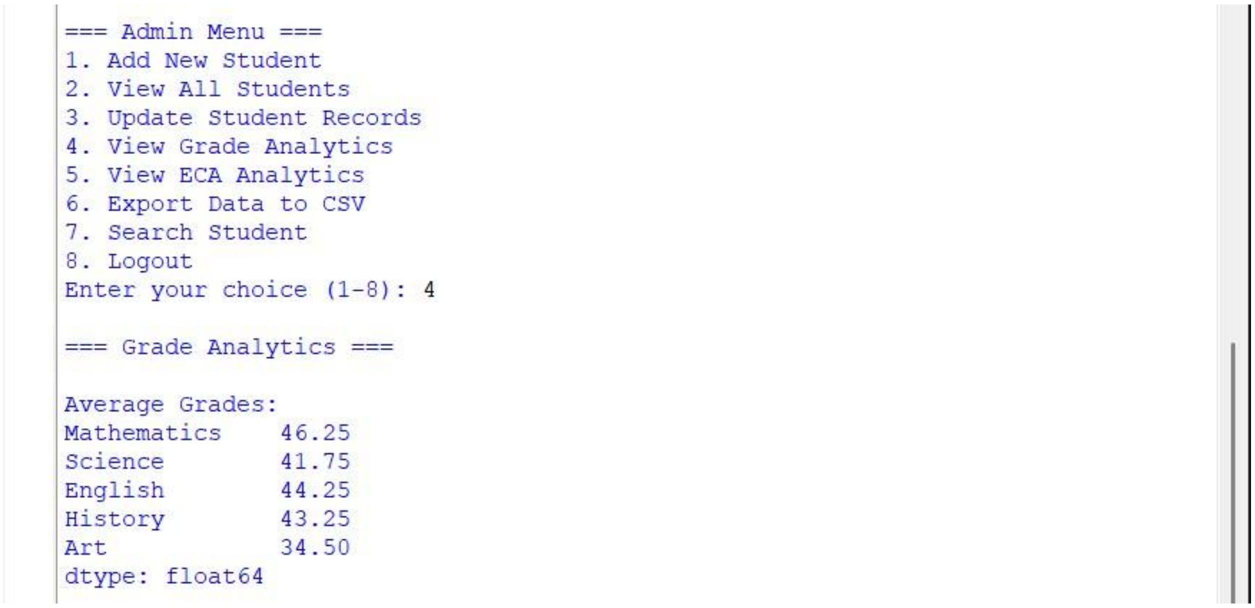
```
Python 3.13.3 (tags/v3.13.3:6280bb5, Apr  8 2025, 14:47:33) [MSC v.1943 64 bit (
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
= RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/
Final Assessement Project Work of FODS/Student Profile Management System.py

=== Student Profile Management System ===
1. Login
2. Exit
Enter your choice (1-2): 1

=== Login ===
Username: admin
Password: admin123
```

Figure 12 : Login admin username and passwords



```
=== Admin Menu ===
1. Add New Student
2. View All Students
3. Update Student Records
4. View Grade Analytics
5. View ECA Analytics
6. Export Data to CSV
7. Search Student
8. Logout
Enter your choice (1-8): 2

=== All Students ===

ID       Username
------------------------
2        student1
3        student2
4        new_student
```

Figure 13 : Admin has the option to choose from 1 to 8 and he chooses option number 2 to view the number of students with username



```
=== Admin Menu ===
1. Add New Student
2. View All Students
3. Update Student Records
4. View Grade Analytics
5. View ECA Analytics
6. Export Data to CSV
7. Search Student
8. Logout
Enter your choice (1-8): 4

=== Grade Analytics ===

Average Grades:
Mathematics    46.25
Science        41.75
English        44.25
History        43.25
Art            34.50
dtype: float64
```

Figure 14 : View grade analytics of a particular student

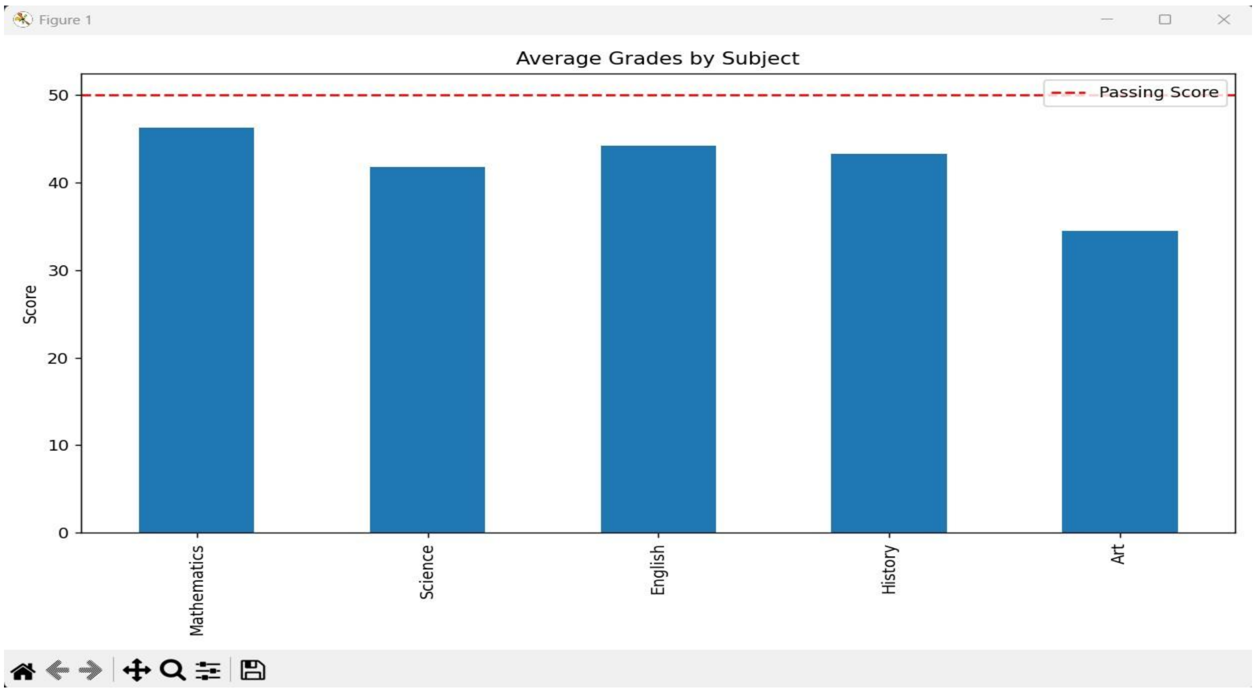
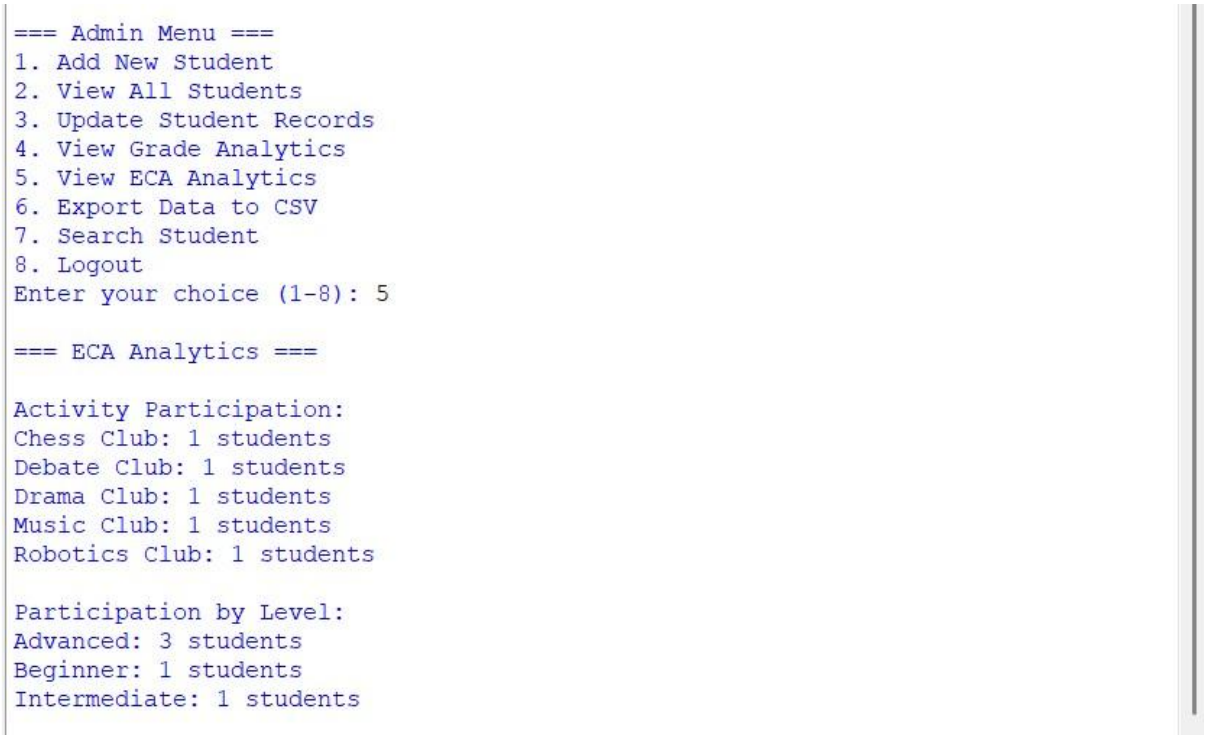


Figure 15 : Matplotlib window opens showing bar chart of averages with red line



```
=== Admin Menu ===
1. Add New Student
2. View All Students
3. Update Student Records
4. View Grade Analytics
5. View ECA Analytics
6. Export Data to CSV
7. Search Student
8. Logout
Enter your choice (1-8): 5

=== ECA Analytics ===

Activity Participation:
Chess Club: 1 students
Debate Club: 1 students
Drama Club: 1 students
Music Club: 1 students
Robotics Club: 1 students

Participation by Level:
Advanced: 3 students
Beginner: 1 students
Intermediate: 1 students
```

Figure 16 :View eca analytics of a particular student

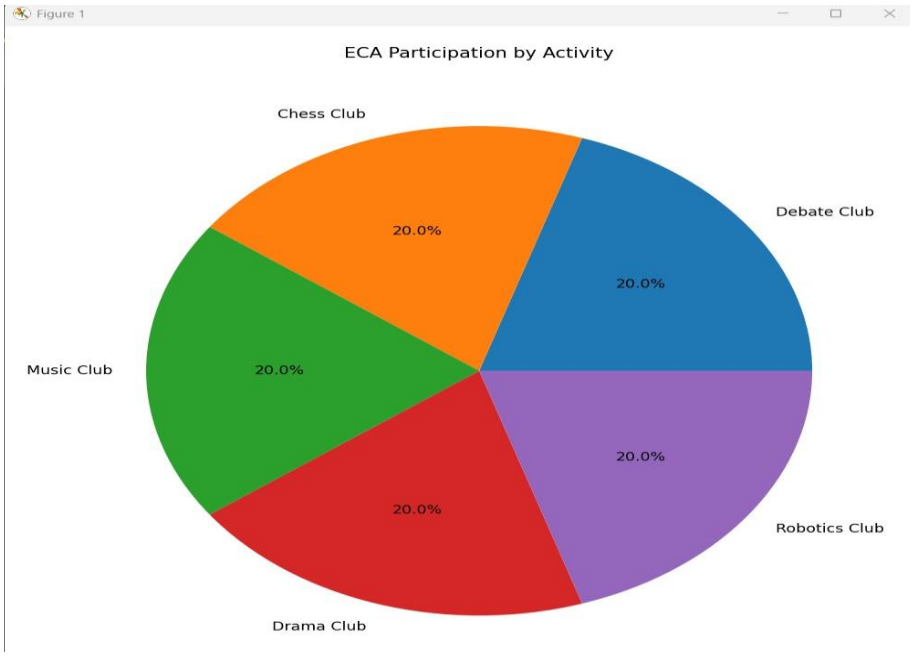
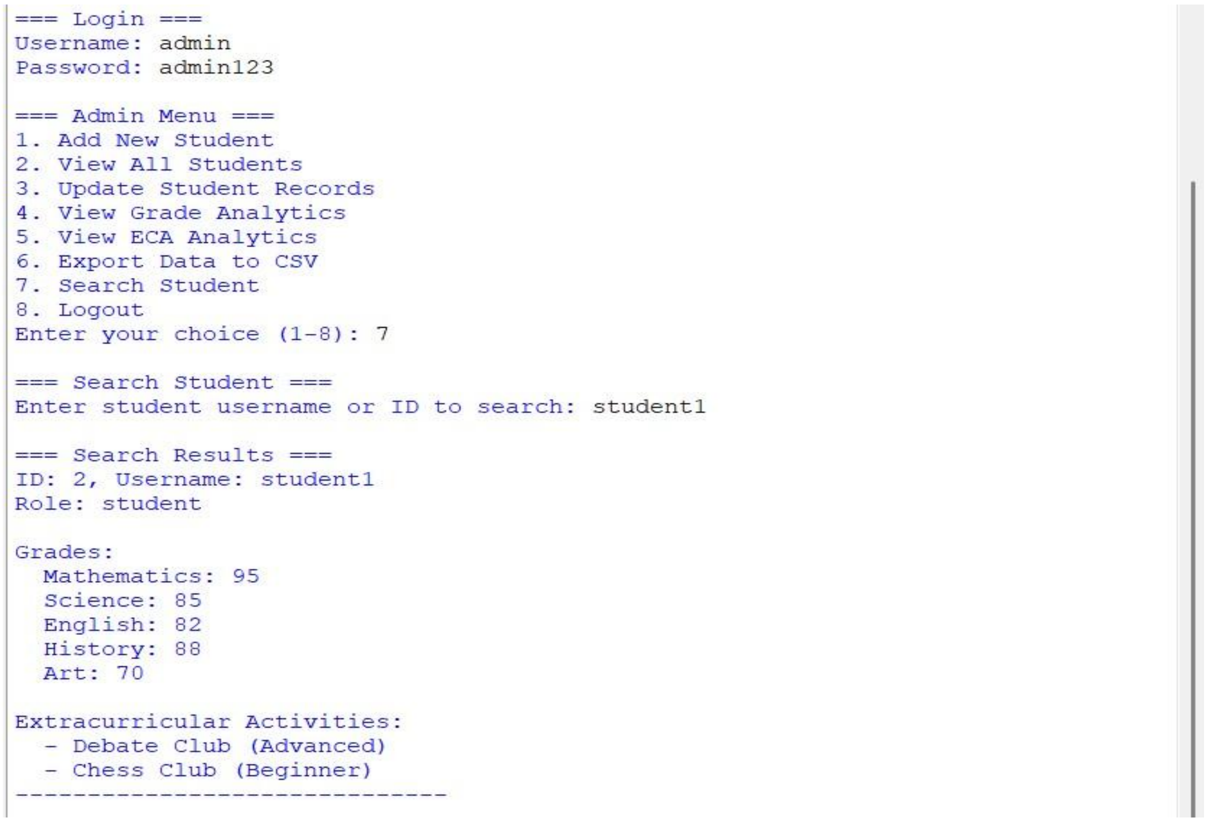
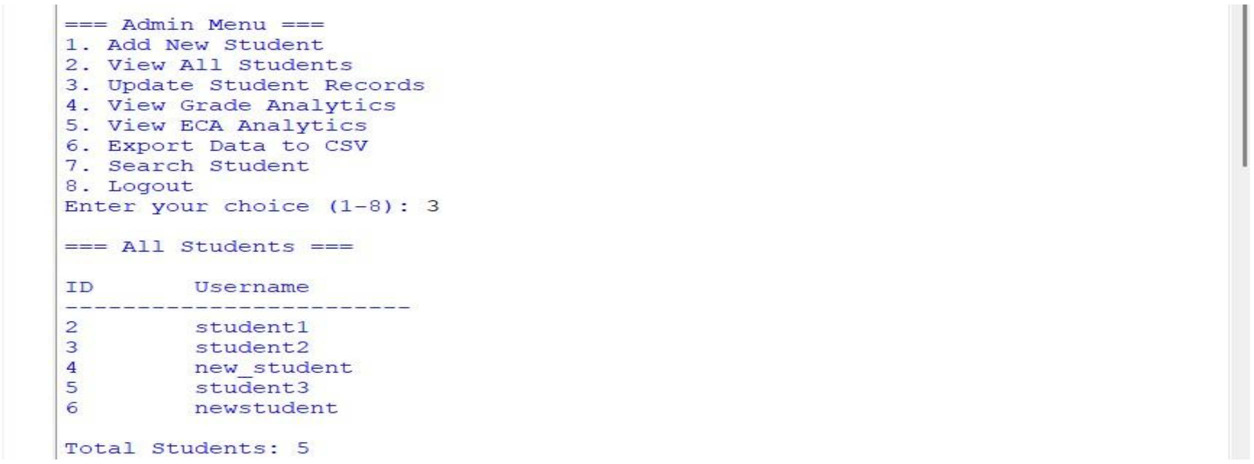


Figure 17 : Matplotlib window opens showing pie chart of averages with red line



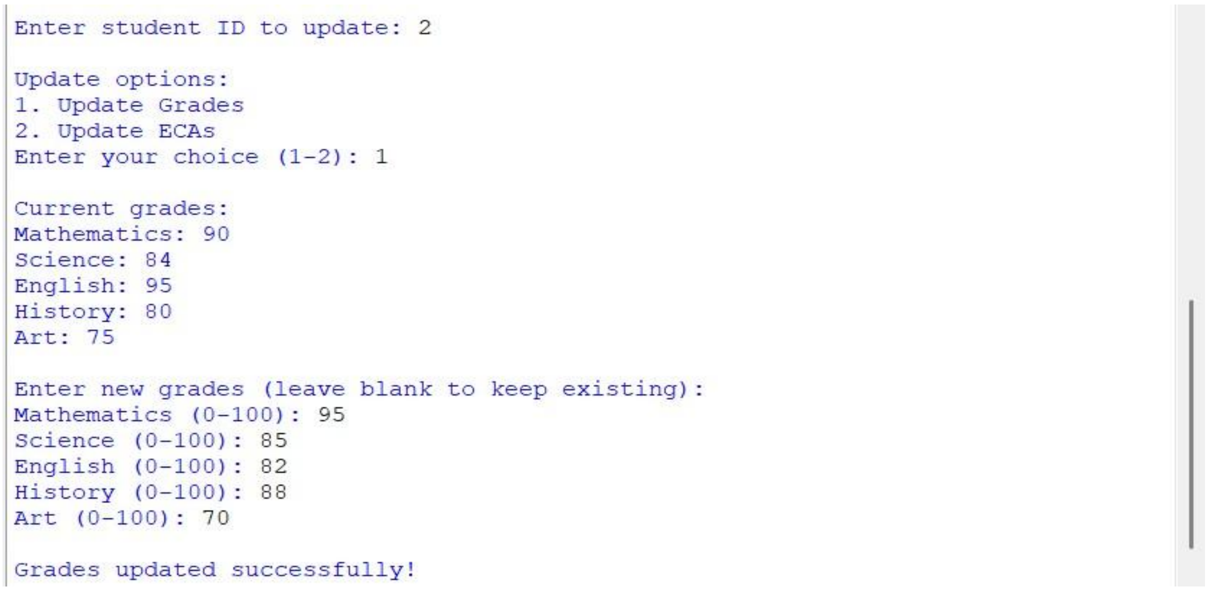
```
=== Login ===
Username: admin
Password: admin123

=== Admin Menu ===
1. Add New Student
2. View All Students
3. Update Student Records
4. View Grade Analytics
5. View ECA Analytics
6. Export Data to CSV
7. Search Student
8. Logout
Enter your choice (1-8): 7

=== Search Student ===
Enter student username or ID to search: student1

=== Search Results ===
ID: 2, Username: student1
Role: student

Grades:
  Mathematics: 95
  Science: 85
  English: 82
  History: 88
  Art: 70

Extracurricular Activities:
  - Debate Club (Advanced)
  - Chess Club (Beginner)
------------------------------
```

Figure 18 : View details of student and correlate eca activities with his/her academics



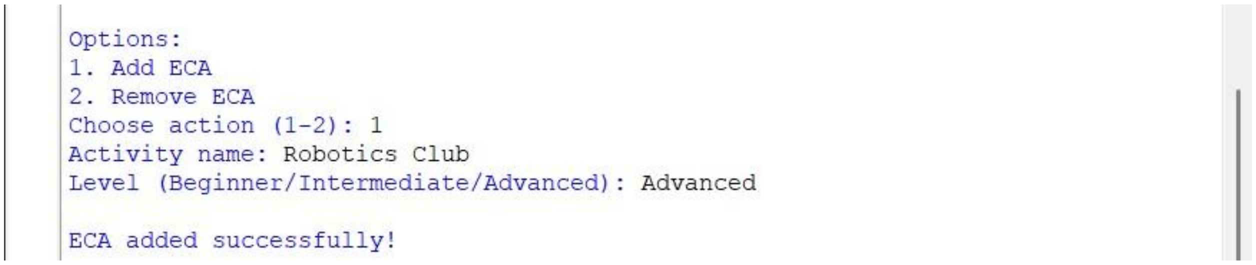
```
=== Admin Menu ===
1. Add New Student
2. View All Students
3. Update Student Records
4. View Grade Analytics
5. View ECA Analytics
6. Export Data to CSV
7. Search Student
8. Logout
Enter your choice (1-8): 3

=== All Students ===

ID        Username
------------------------
2         student1
3         student2
4         new_student
5         student3
6         newstudent

Total Students: 5
```

Figure 19 : Update student records (admin view)



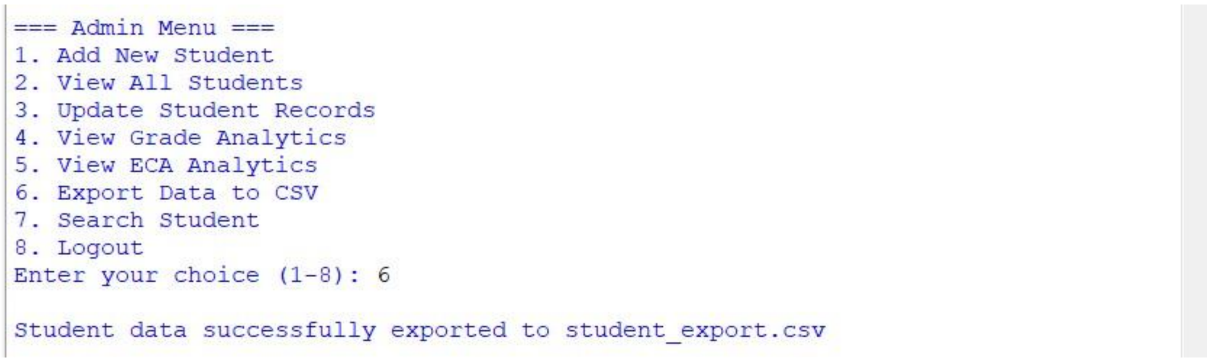
```
Enter student ID to update: 2

Update options:
1. Update Grades
2. Update ECAs
Enter your choice (1-2): 1

Current grades:
Mathematics: 90
Science: 84
English: 95
History: 80
Art: 75

Enter new grades (leave blank to keep existing):
Mathematics (0-100): 95
Science (0-100): 85
English (0-100): 82
History (0-100): 88
Art (0-100): 70

Grades updated successfully!
```

Figure 20 : Successfully update grades of the student



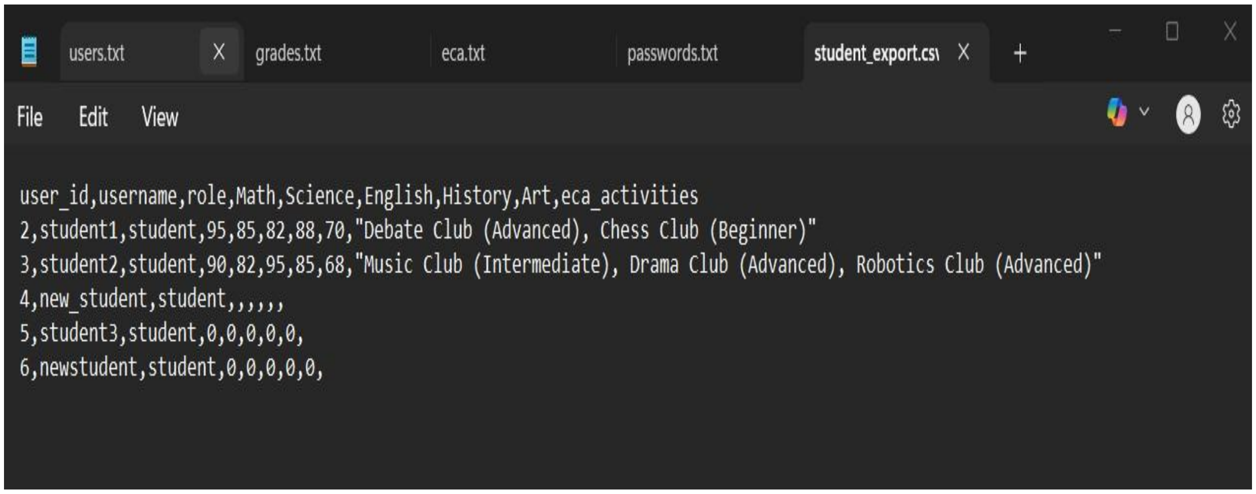
```
Options:
1. Add ECA
2. Remove ECA
Choose action (1-2): 1
Activity name: Robotics Club
Level (Beginner/Intermediate/Advanced): Advanced

ECA added successfully!
```

Figure 21 : Successfully updated eca activities of the student



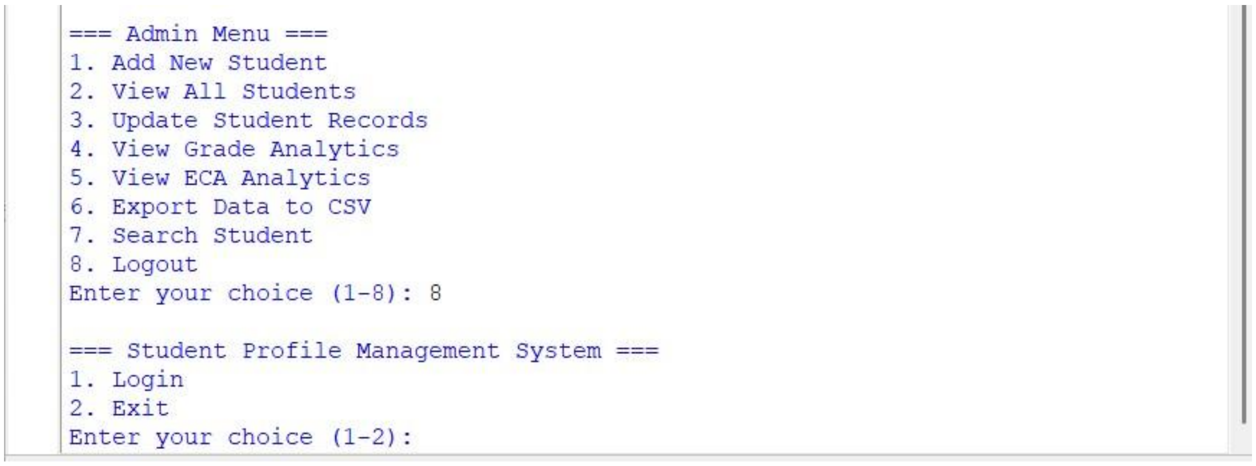
```
=== Admin Menu ===
1. Add New Student
2. View All Students
3. Update Student Records
4. View Grade Analytics
5. View ECA Analytics
6. Export Data to CSV
7. Search Student
8. Logout
Enter your choice (1-8): 6

Student data successfully exported to student_export.csv
```

Figure 22 : Student new records successfully exported to the student_export.csv file



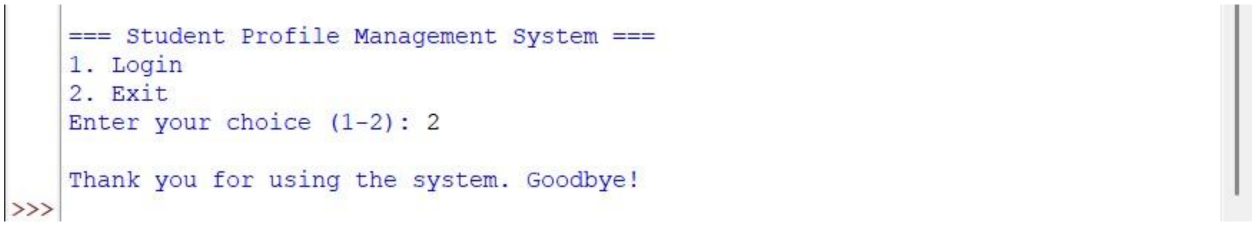
```
user_id,username,role,Math,Science,English,History,Art,eca_activities
2,student1,student,95,85,82,88,70,"Debate Club (Advanced), Chess Club (Beginner)"
3,student2,student,90,82,95,85,68,"Music Club (Intermediate), Drama Club (Advanced), Robotics Club (Advanced)"
4,new_student,student,,,,,,
5,student3,student,0,0,0,0,0,
6,newstudent,student,0,0,0,0,0,
```

Figure 23: student_csv file



```
=== Admin Menu ===
1. Add New Student
2. View All Students
3. Update Student Records
4. View Grade Analytics
5. View ECA Analytics
6. Export Data to CSV
7. Search Student
8. Logout
Enter your choice (1-8): 8

=== Student Profile Management System ===
1. Login
2. Exit
Enter your choice (1-2):
```

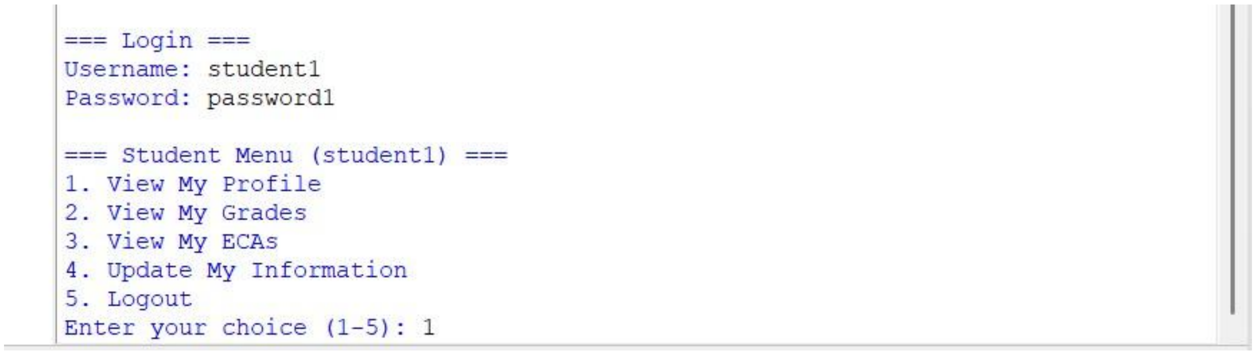
Figure 24 : Log out from admin view



```
=== Student Profile Management System ===
1. Login
2. Exit
Enter your choice (1-2): 2

Thank you for using the system. Goodbye!
>>>
```

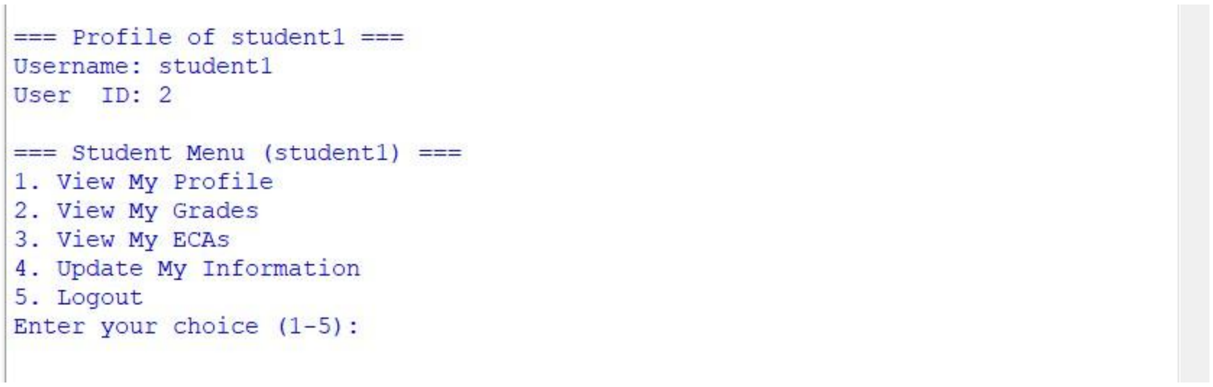
Figure 25 : Exit main menu option as admin

Scenario 2: Log in as student



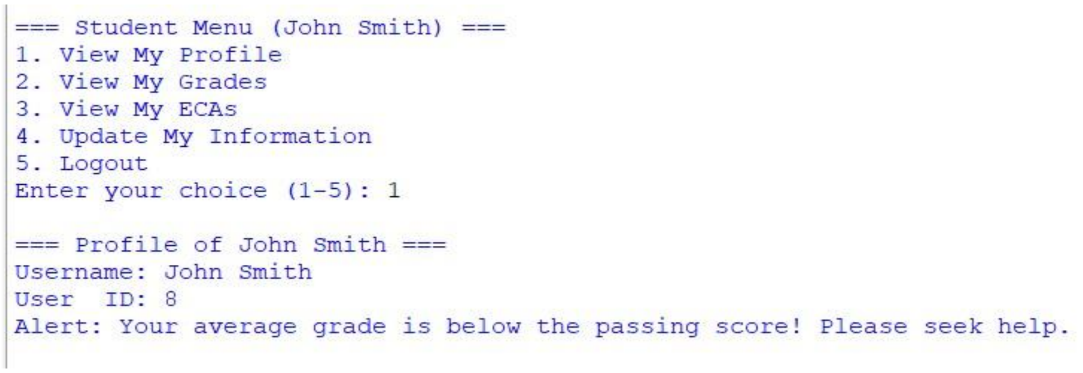
```
=== Login ===
Username: student1
Password: password1

=== Student Menu (student1) ===
1. View My Profile
2. View My Grades
3. View My ECAs
4. Update My Information
5. Logout
Enter your choice (1-5): 1
```

Figure 26 : Students need to enter username and password



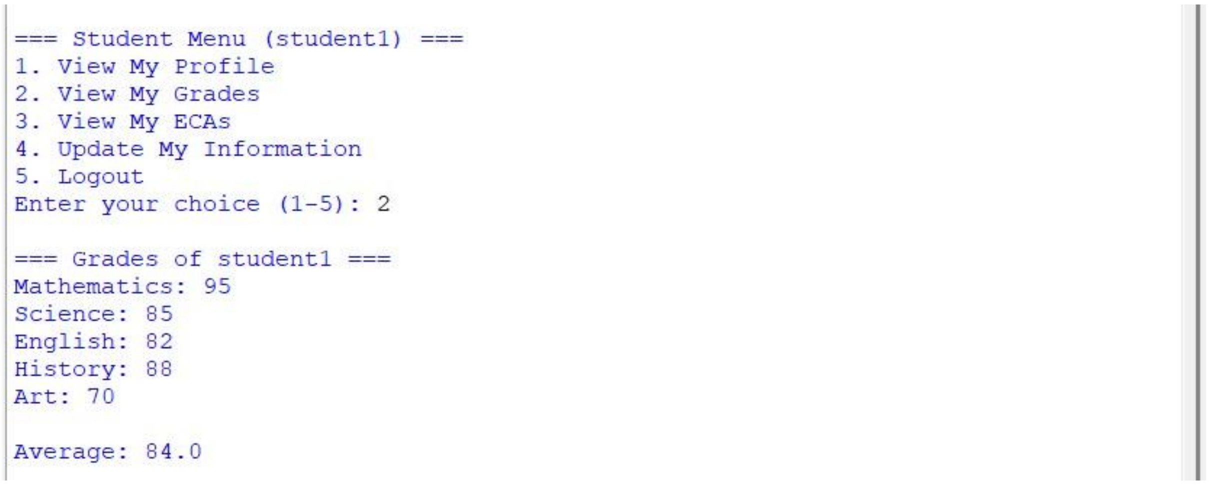
```
=== Profile of student1 ===
Username: student1
User  ID: 2

=== Student Menu (student1) ===
1. View My Profile
2. View My Grades
3. View My ECAs
4. Update My Information
5. Logout
Enter your choice (1-5):
```

Figure 27 : Student can view their profile



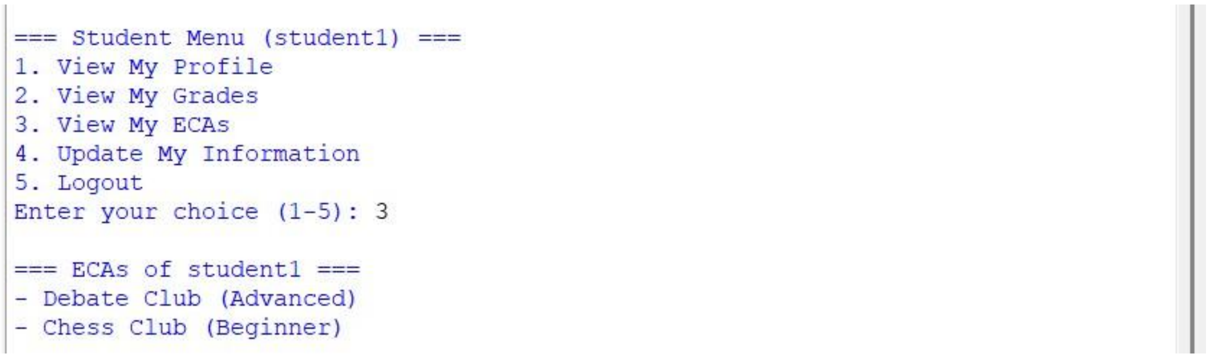
```
=== Student Menu (John Smith) ===
1. View My Profile
2. View My Grades
3. View My ECAs
4. Update My Information
5. Logout
Enter your choice (1-5): 1

=== Profile of John Smith ===
Username: John Smith
User  ID: 8
Alert: Your average grade is below the passing score! Please seek help.
```

Figure 28 : Identify students performing below a specific threshold and suggest interventions



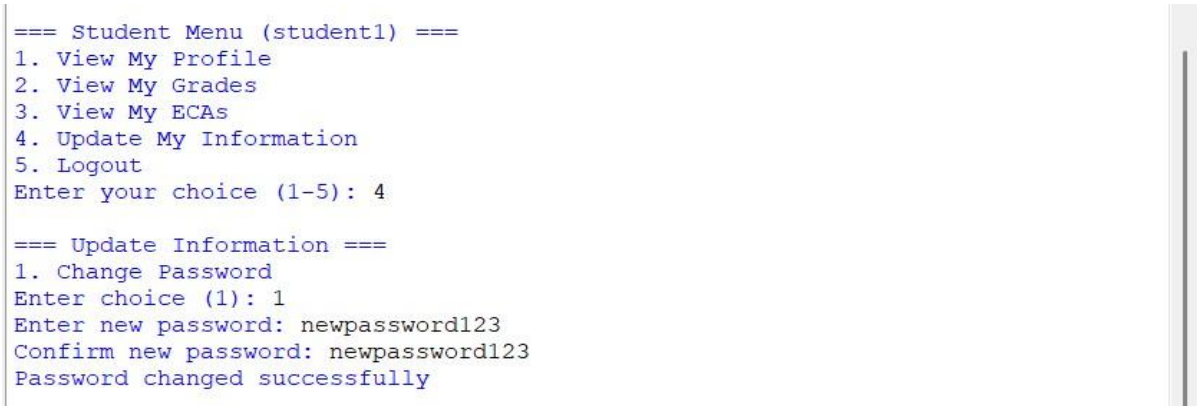
```
=== Student Menu (student1) ===
1. View My Profile
2. View My Grades
3. View My ECAs
4. Update My Information
5. Logout
Enter your choice (1-5): 2

=== Grades of student1 ===
Mathematics: 95
Science: 85
English: 82
History: 88
Art: 70

Average: 84.0
```

Figure 29 : Student can view grades of individual subject and their overall average



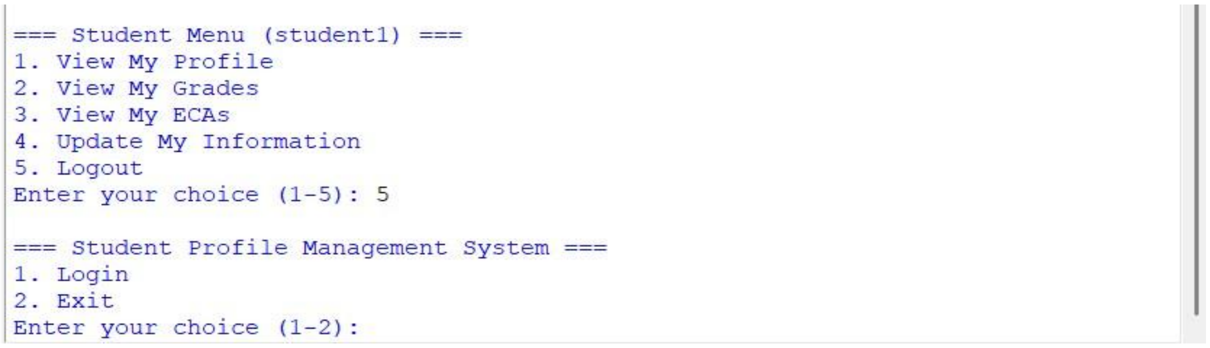
```
=== Student Menu (student1) ===
1. View My Profile
2. View My Grades
3. View My ECAs
4. Update My Information
5. Logout
Enter your choice (1-5): 3

=== ECAs of student1 ===
- Debate Club (Advanced)
- Chess Club (Beginner)
```

Figure 30 : Student can view the eca activities they are part of



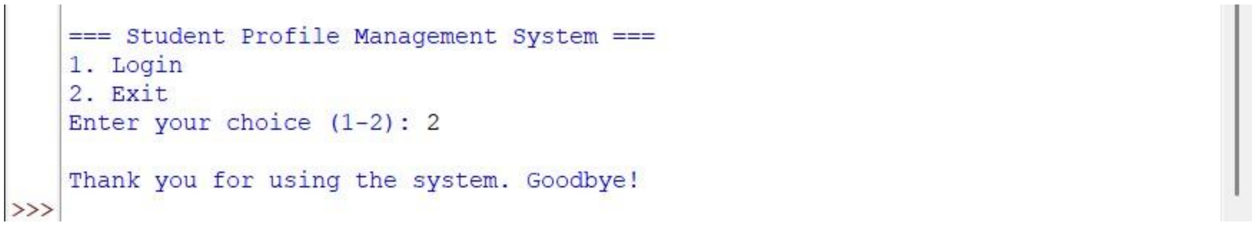
```
=== Student Menu (student1) ===
1. View My Profile
2. View My Grades
3. View My ECAs
4. Update My Information
5. Logout
Enter your choice (1-5): 4

=== Update Information ===
1. Change Password
Enter choice (1): 1
Enter new password: newpassword123
Confirm new password: newpassword123
Password changed successfully
```

Figure 31 : Student can update their passwords



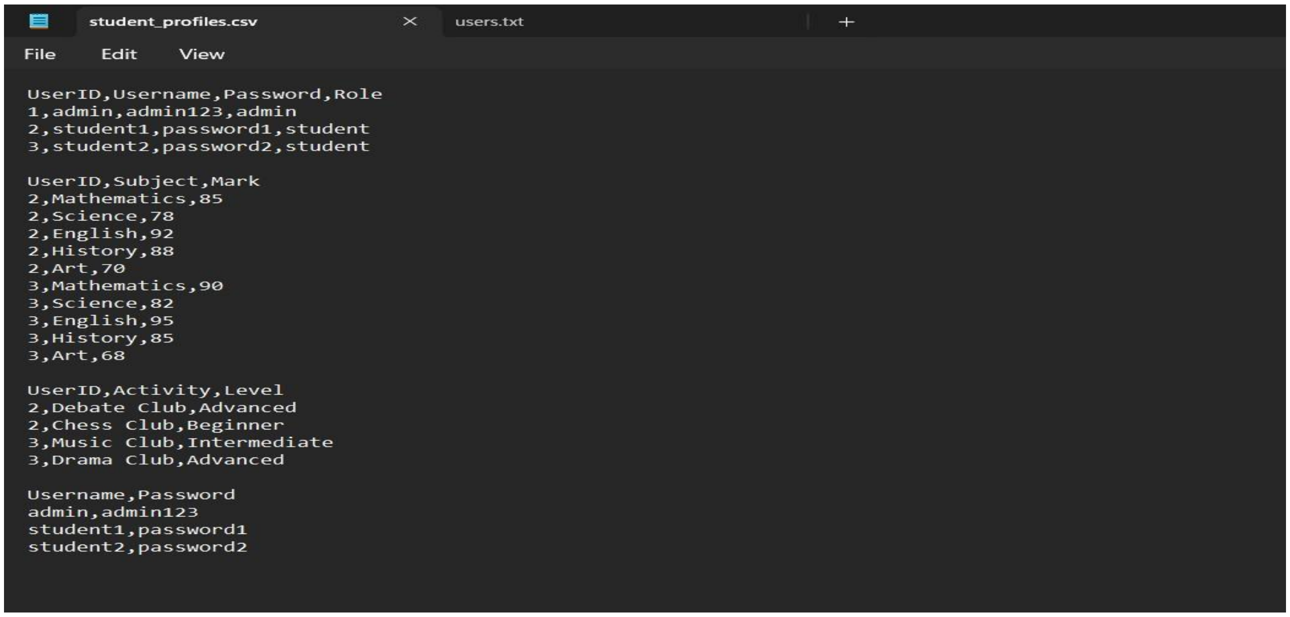
```
=== Student Menu (student1) ===
1. View My Profile
2. View My Grades
3. View My ECAs
4. Update My Information
5. Logout
Enter your choice (1-5): 5

=== Student Profile Management System ===
1. Login
2. Exit
Enter your choice (1-2):
```

Figure 32 : Log out from student view



```
=== Student Profile Management System ===
1. Login
2. Exit
Enter your choice (1-2): 2

Thank you for using the system. Goodbye!
>>>
```

Figure 33 : Exit main menu option as student



```
UserID,Username,Password,Role
1,admin,admin123,admin
2,student1,password1,student
3,student2,password2,student

UserID,Subject,Mark
2,Mathematics,85
2,Science,78
2,English,92
2,History,88
2,Art,70
3,Mathematics,90
3,Science,82
3,English,95
3,History,85
3,Art,68

UserID,Activity,Level
2,Debate Club,Advanced
2,Chess Club,Beginner
3,Music Club,Intermediate
3,Drama Club,Advanced

Username,Password
admin,admin123
student1,password1
student2,password2
```

Figure 34 : Student_profiles csv file

TESTING

a) Unit Testing

Unit testing is testing individual components or functions of a system in isolation to make sure they work. Testing each class and method separately, like the login() method in the User class, with different inputs to verify user authentication. Simple assertions are used to check expected outcomes, like updating a student's grades.

b) Integration

Integration testing is testing the functionality of multiple system components. Testing interactions between classes like Admin and DataHandler to make sure data flows correctly. This is important to ensure the overall system behaviour as it helps to find and fix any issues that may arise.

c) User Acceptance Testing (UAT)

User Acceptance Testing (UAT) is where a system is tested with real users to make sure it meets their needs and expectations. Feedback is gathered from students and administrators who perform common tasks and usability issues are documented for further improvements.

DEBUGGING

Debugging is locating and correcting errors in the software.

a) Error logging

Error logging or keeping track of errors as they occur throughout the program. With error logging, a system failure's nature, time, and context are maintained in log files. If a file operation fails, tracing the steps leading to the failure gives insights into resolving the underlying issues. That makes debugging easier.

b) Debugging tools

The processes of inspecting an application with the help of debugging tools and execution features of an IDE, like breakpoints and step execution, to determine the issues and track the flow of data. For example, one can pause execution at a breakpoint in the update_profile() method for deeper evaluation.

c) Code Review

The focus of this code review cycle was to enhance the profile of the code, prevent it from accumulating bugs, and address problems which if left unattended would become far more complex. Followed processes of review, making sure the code maintained a certain level of standard and writing quality. All flagged problems were solved before the code was closed

CHALLENGES FACED

- Difficulty in planning and system design.
- Understanding flow of the program.
- Ensuring data integrity and consistency across multiple text files.
- Creating an intuitive and user-friendly interface.
- To implement file handling with right file structure
- Validating checks for error input as part of error handling.
- Time-consuming process of choosing the starting point.
- Difficulty in exporting the files student_export.csv and student_profiles.csv.
- Difficulty in identifying classes and implementing them.

CONCLUSION

The Student Profile Management System project is an application which is effective, secure, and user-friendliness to streamline student data management. Through design and implementation, the system automates data administration, replacing manual processes with digital workflows. Th key highlights include role-based access control ensuring data security, a modular OOP design fostering maintainability, and robust error handling for seamless operation.

The implementation of the application involved creating the core workings of the application with secure file handling, UI flows that make sense, and testing to ensure stability in applied situations. Things such as admin reports and student profiles provided evidence of working systems within real-world environments. Problems involved with things like file integrity and file system optimizations were dealt with in situations like file locking, or in the case of number file system indexing. Some important takeaways were using OOP since it provided scalability, using a design that assumes that the user will not recall information or put any work into application execution on their end. Further enhancement can be made in this program like identify the current database system with a migration to increase query performance, develop a web interface for accessing information to a wide variety of users, and add features to help users visualize the data to form actionable.

In conclusion, while the Student Profile Management System project meets its original aims for usability and efficiency, it also provides an appropriate example of how organized coding can tackle a real-world problem in a complete way. This project is a base for expanding future projects and demonstrates more practices in software development for an educational institution.

REFERENCES

Matuntuta, S. (2024) What is a Student Management System? Benefits & Guide *Cloud Assess*. 20 November 2024 [online]. Available from: https://cloudassess.com/blog/what-is-sms/ [Accessed 27 June 2025].

Anon. (no date) *Scaler Topics* [online]. Available from: https://www.scaler.com/topics/python/ [Accessed 27 June 2025].

Anon. (no date) *Hix.ai* [online]. Available from: https://hix.ai/deepseek-r1?id=cmcss494b00uhj4bi6806aaw8 [Accessed 27 June 2025].

Anon. (2019) *GeeksforGeeks*. 30 October 2019 [online]. Available from: https://www.geeksforgeeks.org/python/student-management-system-in-python/ [Accessed 1 July 2025].

Anon. (no date) *Napkin.ai* [online]. Available from: https://app.napkin.ai/page/CgoiCHByb2Qtb25lEiwKBFBhZ2UaJGRhYTVlZDk1LTkwYzItNGZhMS05MTRhLWY4N2U3MzIxNDg1OA [Accessed 1 July 2025].