Table of Contents

Project Objective:.....3

Project Features & Functionality:.....5

- Add New Student Records:.....5
- Display Individual Student Marks Sheet:.....5
- Update Student Records:.....6
- Delete Student Records:.....6
- View All Students' Records:.....6
- Grade Calculation:.....6
- Percentage Calculation:.....6
- File-based Data Management:.....6
- Secure Data Deletion and Updates.....6
- Error Handling:.....6
- Structured Menu System:.....6
- Standardized Marks Sheet Format:.....7
- Subject-wise Marks Input:.....7
- Dynamic Memory Handling:.....7
- Total Marks Display:.....7
- Batch Data Viewing:.....7
- GPA Calculation Placeholder:.....7
- Scalability:.....7
- Secure Input Handling:.....7
- Custom Grade Ranges:.....7
- Performance Tracking:.....7
- File Backup Support:.....8
- Transparency:.....8
- Consistent Output Design:.....8
- Data Integrity:.....8
- Error Reporting:.....8

Necessary Parts of Project:.....8

Conditional Statements:.....8

Used in:.....8

Loops:.....8

Used in:.....8

Functions:.....9

Used in: 9
Structures: 9
File Handling: 9
Used in: 9
Arrays: 9
Used in: 9
Input and Output Manipulation: 9
Used in: 9
String Handling: 9
File Operations: 10
Dynamic Input Handling: 10
Error Handling: 10
Menu-Driven Interface: 10
Mathematical Calculations: 10
Used in: 10
File Backup Logic: 10
Flowchart of Project: 10
Output Screenshots of All Screens: 12
Add New Student: 12
Display Result of Single Student: 13
Deletion of Student Record: 14
Updating of Student Record: 15
Displaying All Student: 16
Exiting the Program: 17
Screenshot of Text File used in for Data Storing: 17

Table of Figures:

Figure 1: Addition a New Student in Record 13
Figure 2: Display individual Student 14
Figure 3: Delete Record of Student 15
Figure 4: Record Updating 16
Figure 5: Record of All Student 17
Figure 6: Exiting from the Program 17
Figure 7: Text File 18

MARK SHEET MANAGEMENT SYSTEM IN C++

Project Objective:

1. The main goal of this project is to develop a comprehensive software application that facilitates the efficient management of student marks and academic records.
2. It aims to provide a seamless, user-friendly interface for educational institutions to maintain accurate and up-to-date information on student performance.
3. The system enables the recording, updating, deletion, and retrieval of student data in a secure and organized manner.
4. By centralizing all student academic data, it minimizes paperwork and streamlines administrative tasks.
5. The system allows for the creation of new student profiles, capturing essential details such as name, registration number, semester, subjects, and marks.
6. It provides functionality to display an individual student's detailed marks sheet, including total marks, percentage, and grade.
7. The project includes features to delete outdated or incorrect student records, ensuring that the database remains current and accurate.
8. The update functionality allows administrators to modify student data, ensuring that the system reflects the latest information.
9. An option to view all student records collectively promotes transparency and ease of access for stakeholders.
10. The application calculates grades based on predefined percentage criteria, standardizing performance evaluation.
11. It supports efficient storage and retrieval of data using file operations, ensuring that information is persistently saved.
12. The system enhances data integrity by automating record-keeping processes, reducing human errors.
13. It is designed to handle multiple student entries, accommodating the needs of small to medium-sized educational institutions.

14. The project aims to provide administrators with a clear overview of student performance trends for better academic planning.
15. The system ensures quick access to student records, saving time during inspections, audits, or parent-teacher meetings.
16. By implementing this system, the institution can eliminate redundancy and inconsistency in maintaining marks sheets.
17. The project provides secure mechanisms to delete or modify records, ensuring that only authorized personnel can make changes.
18. It enhances accountability by keeping a digital trail of all student records.
19. The user-friendly menu interface simplifies the navigation of options for non-technical staff.
20. It is designed to improve operational efficiency by automating routine academic tasks.
21. The software supports standard file formats, enabling easy backup and recovery of data.
22. The system calculates percentages and grades, reducing the manual workload on faculty members.
23. It aims to promote fairness and transparency in student evaluation.
24. The modular structure of the program allows for future scalability and feature enhancements.
25. With a robust backend logic, the application minimizes runtime errors and ensures reliability.
26. The software offers flexibility in handling data for different academic programs or semesters.
27. It aims to serve as a cost-effective solution for academic record management.
28. By using this system, institutions can focus more on strategic planning rather than administrative burdens.
29. The application ensures compliance with institutional policies for data management.
30. It reduces the risk of data loss associated with manual record-keeping.
31. The system encourages digital transformation in academic administration.
32. It supports analytical insights by providing accurate performance metrics for students.
33. The project is designed to be cross-compatible, functioning seamlessly on various operating systems.
34. It ensures data privacy by handling sensitive student information with care.
35. The system provides a consistent format for marks sheets, improving readability and professionalism.

36. The application is a stepping stone for integrating advanced features like online access or cloud storage.
37. It eliminates ambiguities in manual marks calculation and record interpretation.
38. The project's focus is to empower administrators with a dependable tool for academic management.
39. It offers educational institutions a platform to modernize their administrative workflows.
40. By managing student records efficiently, the system supports academic excellence and institutional credibility.
41. The project promotes a structured and systematic approach to data handling in education.
42. It facilitates timely decision-making by providing readily available information.
43. The application is designed with scalability in mind, allowing for expansion as institutional needs grow.
44. The system ensures compliance with academic standards and grading policies.
45. It offers robust error-handling mechanisms to address common user input issues.
46. The project's ultimate goal is to enhance the user experience for administrators, faculty, and students.
47. It serves as an example of leveraging programming skills to solve real-world problems.
48. The system provides a solid foundation for further integration with advanced technologies, such as mobile apps or web portals.
49. This project reflects a commitment to fostering innovation in the field of education.
50. It underscores the importance of leveraging technology for effective academic record management.

Project Features & Functionality:

• Add New Student Records:

Allows users to input and store essential details of new students, including their name, registration number, semester, subjects, and marks.

• Display Individual Student Marks Sheet:

Retrieve and display a detailed marks sheet for a specific student based on their registration number.

• Update Student Records:

Modify existing student information, such as semester, subjects, and marks, to ensure the database remains current.

• Delete Student Records:

Remove outdated or unnecessary student records by specifying the registration number.

• View All Students' Records:

Display a consolidated list of all student records, including registration numbers, subjects, marks, percentage, and grades.

• Grade Calculation:

Automatically compute grades based on the percentage obtained, using predefined grading criteria.

• Percentage Calculation:

Calculate the percentage of total marks obtained by a student out of 500.

• File-based Data Management:

Persistently store all student data in a text file (students.txt) to ensure records are saved between program executions.

• Secure Data Deletion and Updates:

Safely delete or update records by using a temporary file, minimizing risks of accidental data loss.

• Error Handling:

Display appropriate messages for invalid inputs, such as non-existent registration numbers or incorrect choices.

• Structured Menu System:

User-friendly menu with clearly labeled options for easy navigation.

- **Standardized Marks Sheet Format:**

Output marks sheets in a professional and readable format, complete with headers, tables, and aligned data.

- **Subject-wise Marks Input:**

Record marks for up to five subjects for each student, ensuring comprehensive evaluation.

- **Dynamic Memory Handling:**

Efficiently process multiple student records without fixed array limits, using file operations for flexibility.

- **Total Marks Display:**

Summarize the total marks obtained by a student, along with maximum marks and grades.

- **Batch Data Viewing:**

Enable administrators to view performance metrics for all students in one glance.

- **GPA Calculation Placeholder:**

Potential for integration of GPA calculation in the future based on subject credits.

- **Scalability:**

Structured program design supports future enhancements, such as additional fields or functionalities.

- **Secure Input Handling:**

Prevent data corruption by validating inputs and maintaining proper file read/write sequences.

- **Custom Grade Ranges:**

Adaptable grading system to match institutional requirements or policies.

- **Performance Tracking:**

Track student performance over semesters for academic planning.

• File Backup Support:

Allows users to create backups of student records by using simple file operations.

• Transparency:

Facilitates clear communication of student performance through accurate records.

• Consistent Output Design:

Maintain a uniform layout for marks sheets and data displays.

• Data Integrity:

Ensures accuracy and consistency of student data during updates and deletions.

• Error Reporting:

Notify users of invalid choices, file access errors, or missing data.

Necessary Parts of Project:

Conditional Statements:

Used for decision-making and control flow, such as validating user inputs and executing specific functionalities.

Used in:

- If-else statements to check whether a registration number exists.
- Switch-case for menu-driven operations.

Loops:

For repetitive operations like taking input for multiple subjects and marks or displaying all student records.

Used in:

- For loops to iterate over the 5 subjects for input and processing.
- While loops for reading records from files.

Functions:

Modular design using functions for specific tasks like adding, updating, deleting, and displaying student records.

Used in:

- Functions like addNewStudent(), displayIndividualStudent(), and calculateGrade().

Structures:

struct Student to group related attributes of a student (e.g., name, registration number, semester, subjects, and marks).

File Handling:

Read and write operations for persistent storage and retrieval of student data.

Used in:

- ifstream for reading records.
- ofstream for appending or updating records.

Arrays:

Used to store multiple subjects and marks for each student.

Used in:

- Arrays subjects[5] and marks[5] in the Student structure.

Input and Output Manipulation:

Utilizing formatted input/output for user-friendly displays.

Used in:

- setw and setprecision for alignment and precision in percentage displays.

String Handling:

String used for managing text data like names, registration numbers, and subjects.

File Operations:

Operations like creating, reading, updating, and deleting data from files (students.txt).

Dynamic Input Handling:

Using cin and getline to handle both numeric and string inputs.

Error Handling:

Simple error messages for invalid file access or incorrect user inputs.

Menu-Driven Interface:

A structured menu for navigating between features, implemented using loops and conditional statements.

Mathematical Calculations:

Computing totals, percentages, and grades.

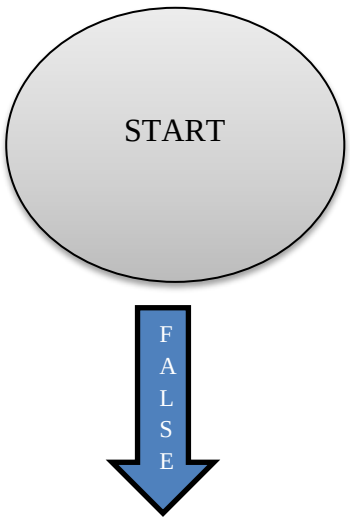
Used in:

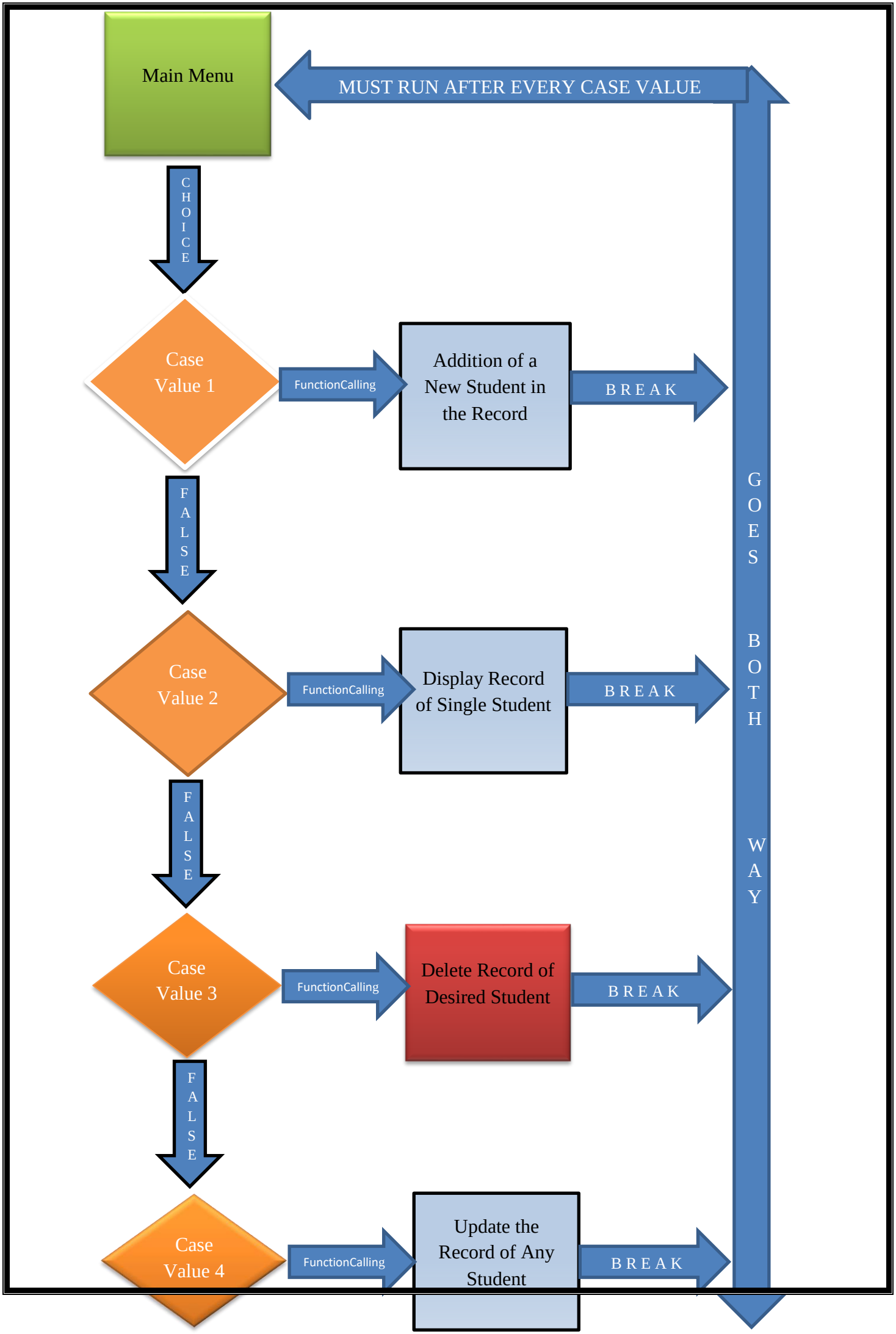
- Formula for percentage: (totalMarks / 500) * 100.

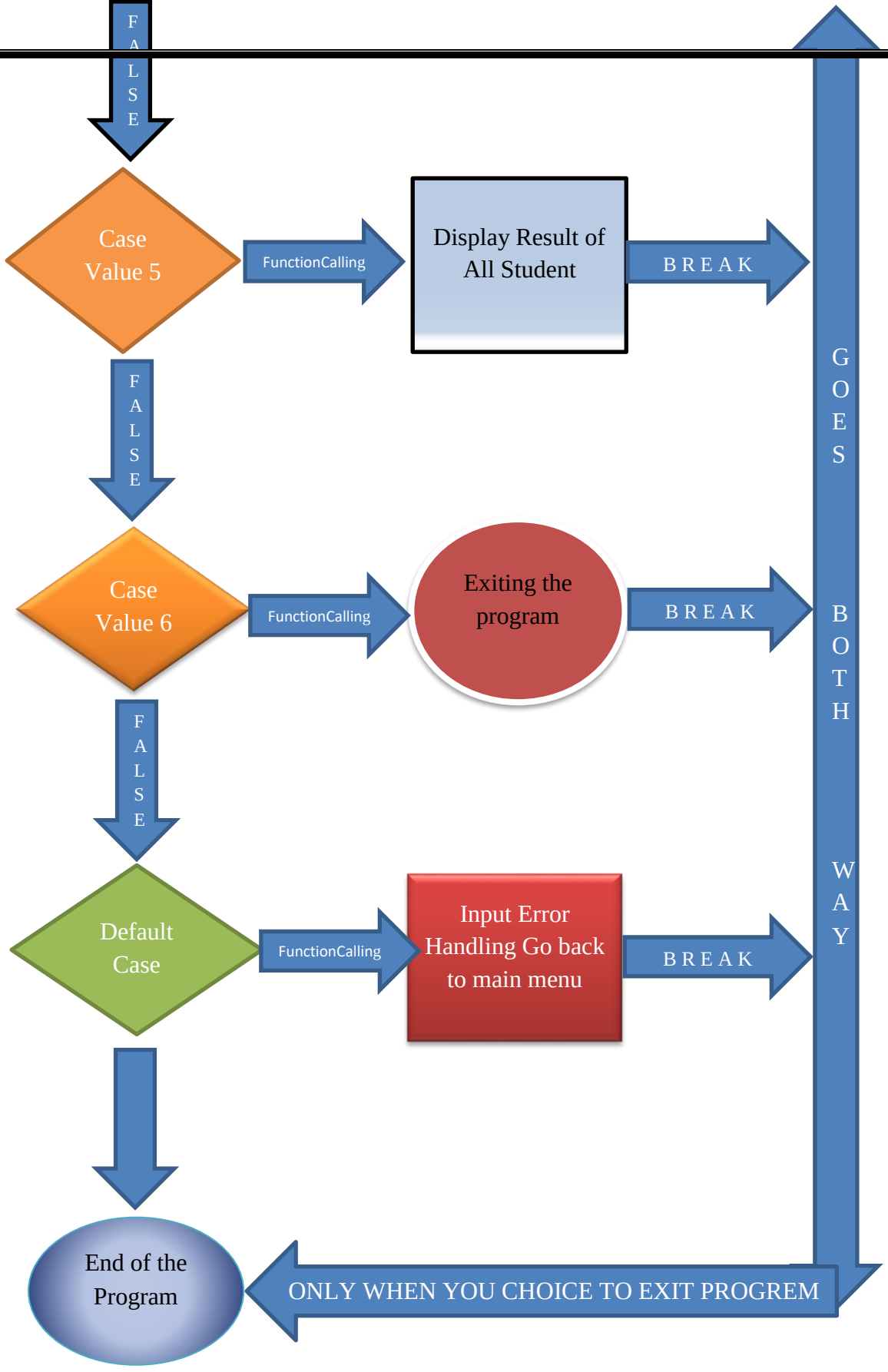
File Backup Logic:

Temporary file creation for safe updates and deletion operations (temp.txt).

Flowchart of Project:




Main Menu
MUST RUN AFTER EVERY CASE VALUE
CHOICE
Case Value 1
FunctionCalling
Addition of a New Student in the Record
BREAK
FALSE
Case Value 2
FunctionCalling
Display Record of Single Student
BREAK
FALSE
Case Value 3
FunctionCalling
Delete Record of Desired Student
BREAK
FALSE
Case Value 4
FunctionCalling
Update the Record of Any Student
BREAK
GOES BOTH WAY



Output Screenshots of All Screens:

Add New Student:

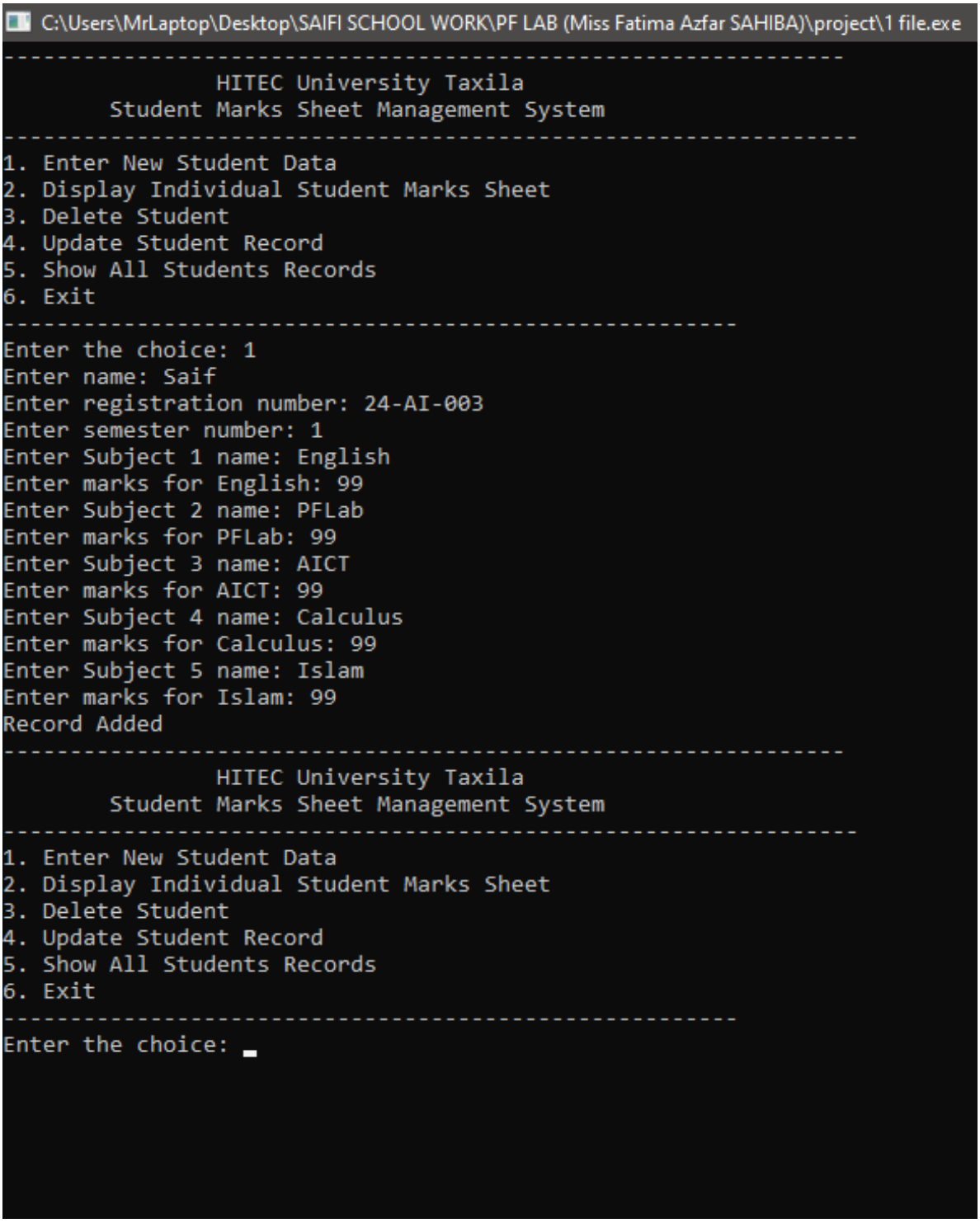


Figure 1: Addition a New Student in Record

Display Result of Single Student:

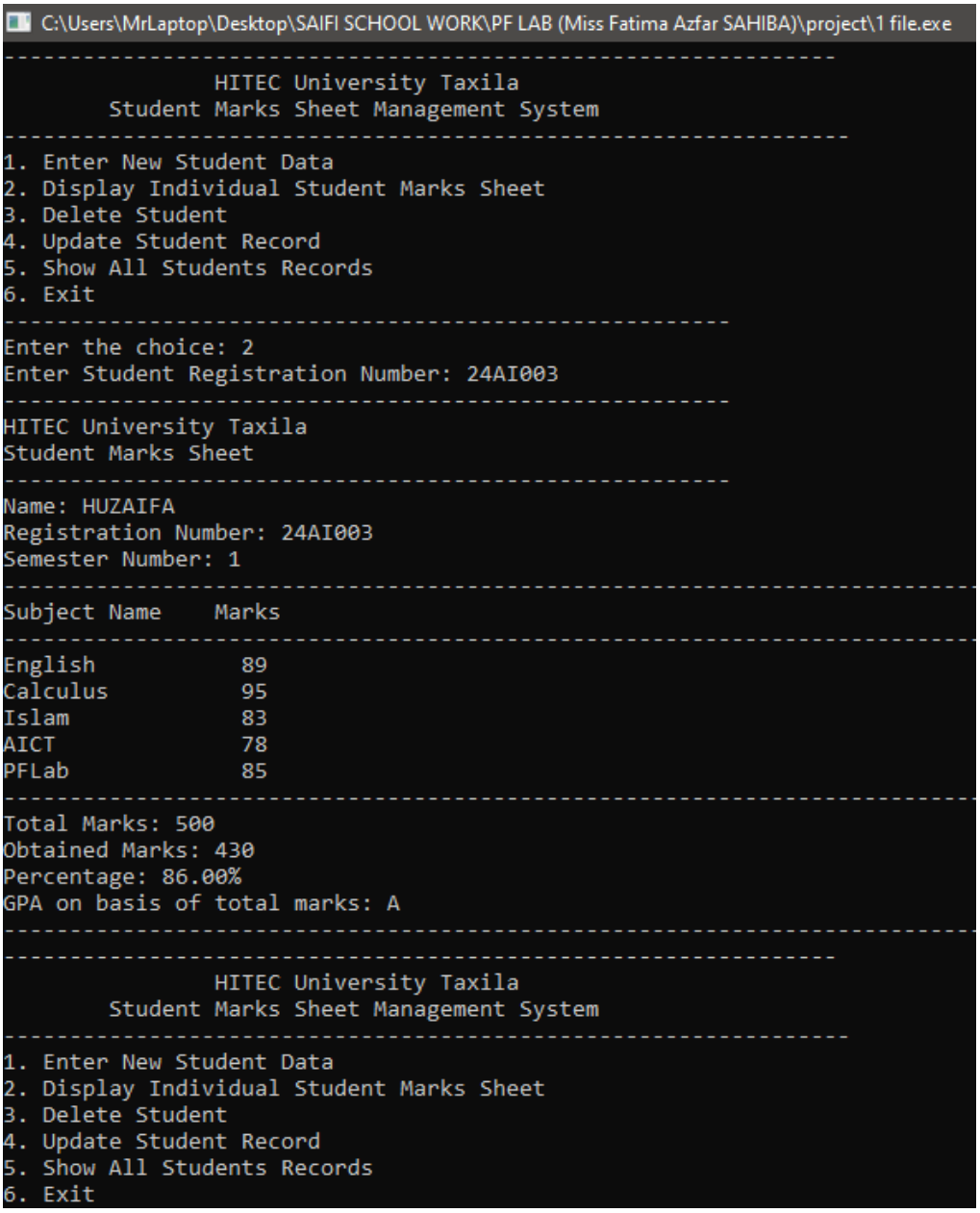


Figure 2: Display individual Student

Deletion of Student Record:

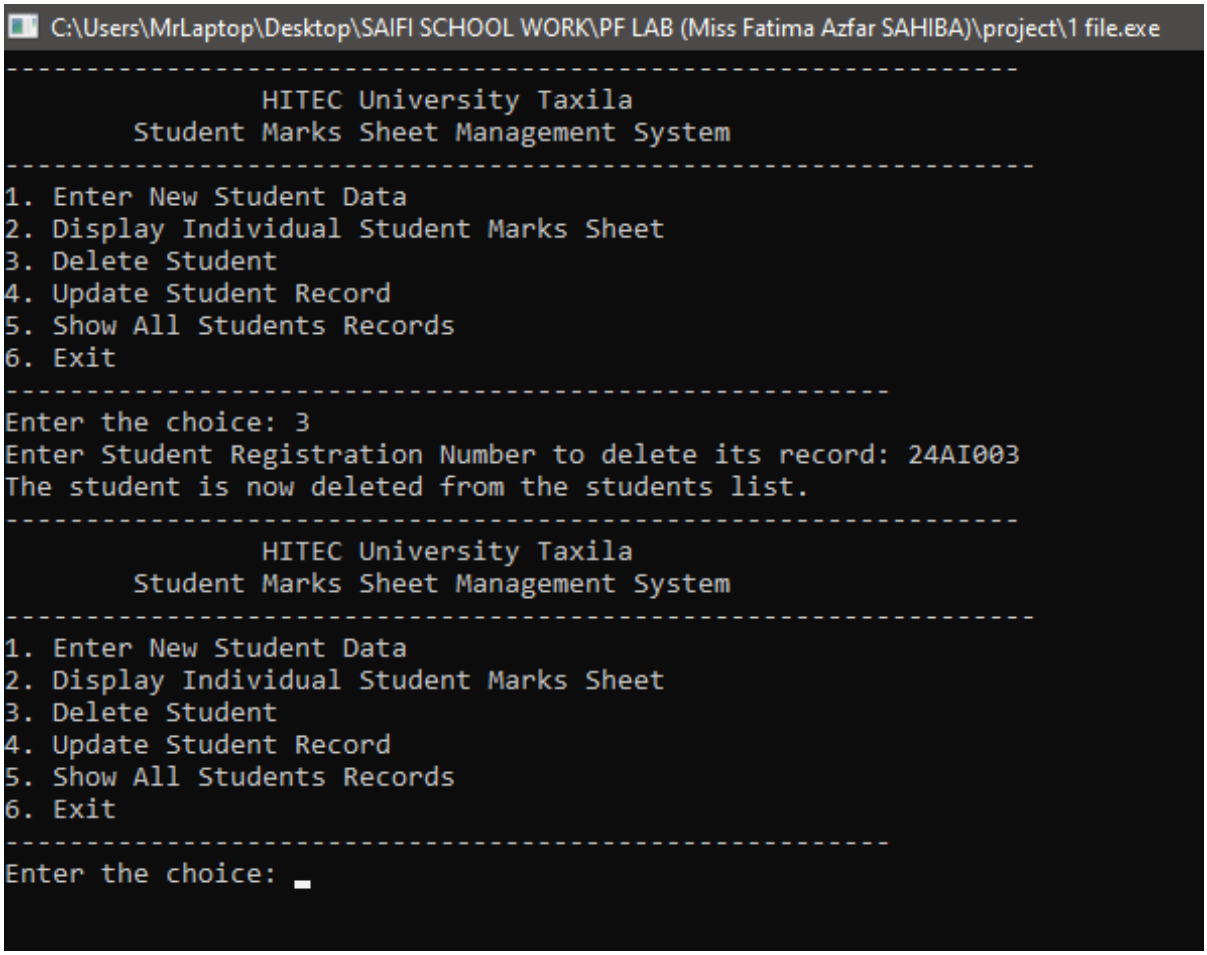


Figure 3: Delete Record of Student

Updating of Student Record:

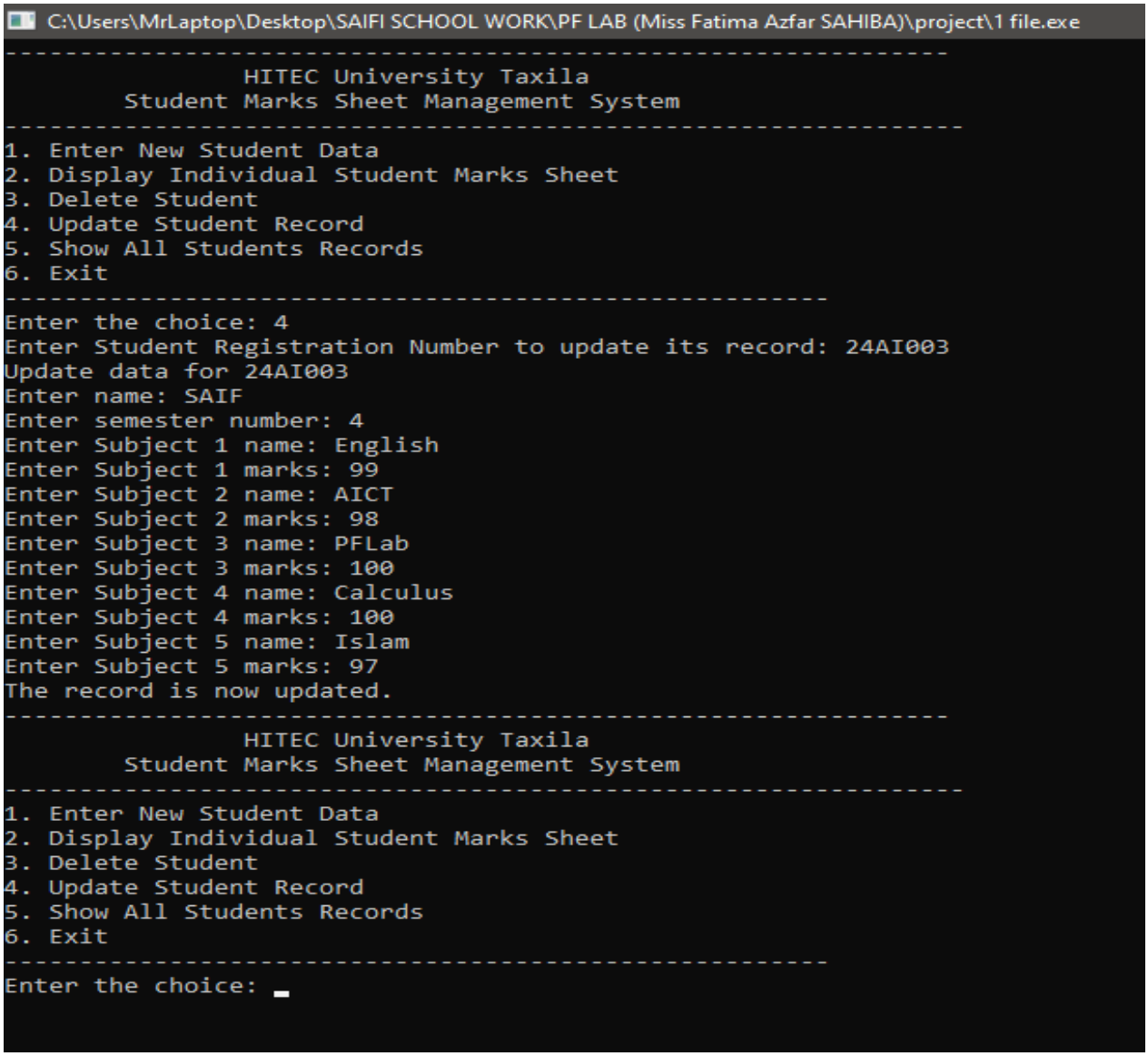


Figure 4: Record Updating

Displaying All Student:

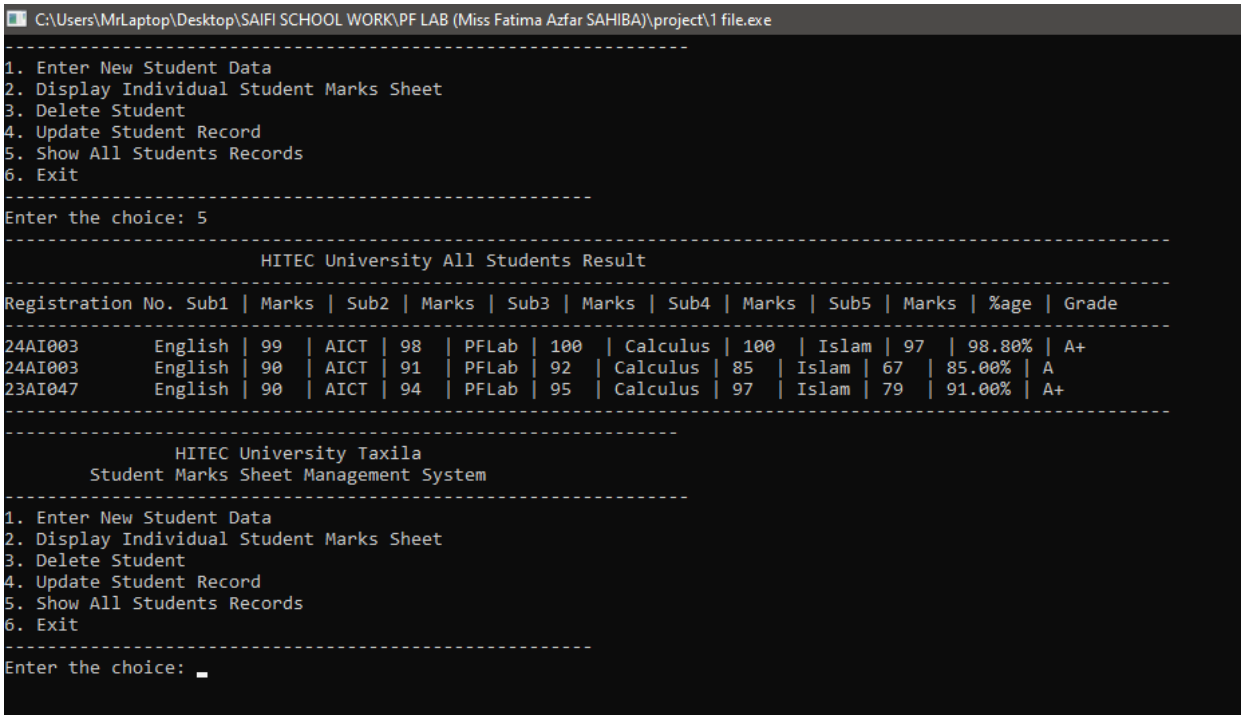


Figure 5: Record of All Student

Exiting the Program:

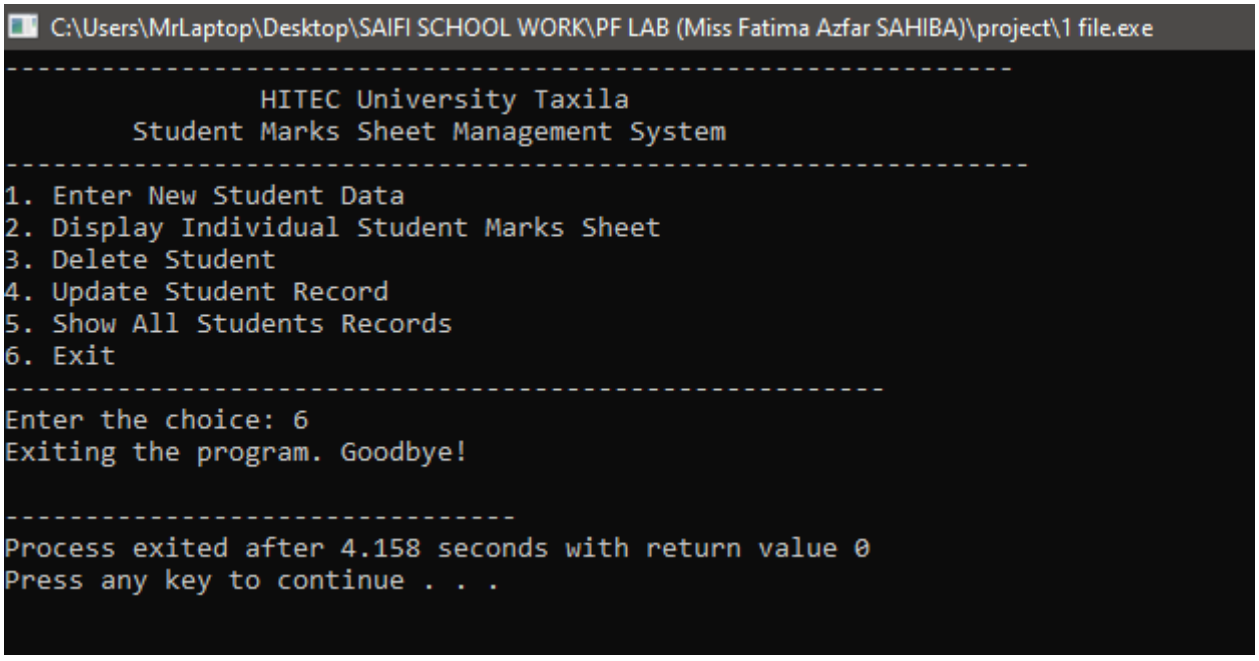


Figure 6: Exiting from the Program

Screenshot of Text File used in for Data Storing:

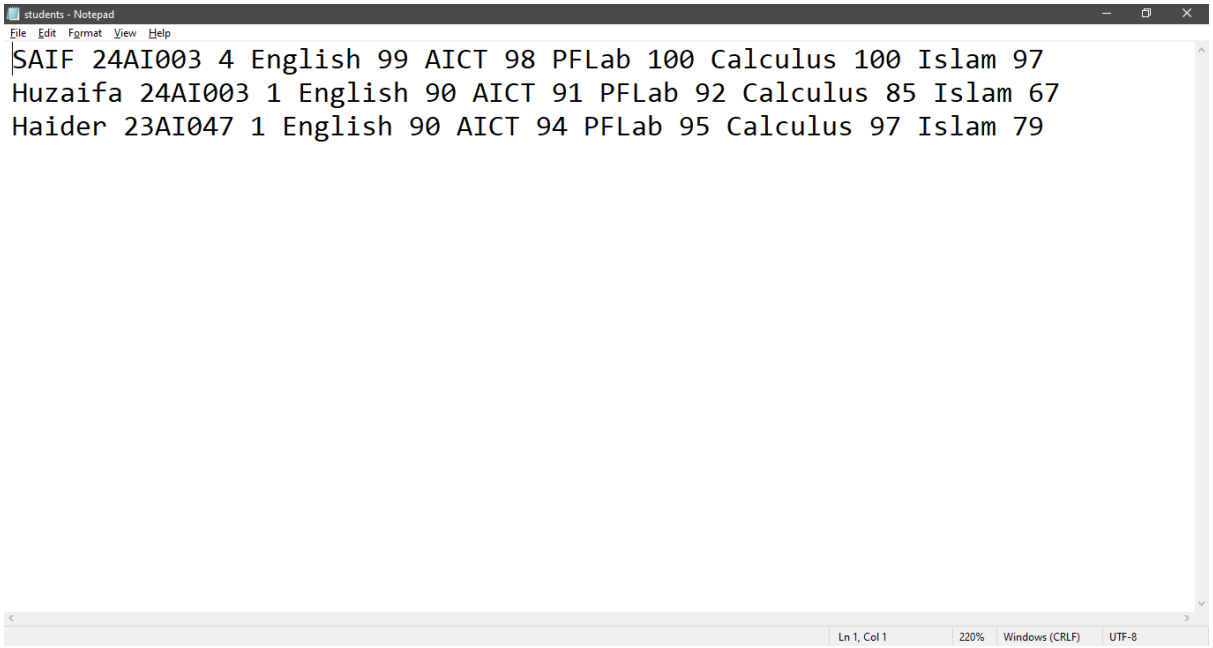


Figure 7: Text File

//