

**PURBANCHAL UNIVERSITY**

**GOMENDRA MULTIPLE COLLEGE**

A PROJECT REPORT ON

**STUDENT RECORD MANAGER**

IN PARTIAL FULFILLMENT OF THE

REQUIREMENT FOR THE DEGREE OF

BCA (1ST SEMESTER)

**Submitted by**

**Ankush Gautam [313625]**

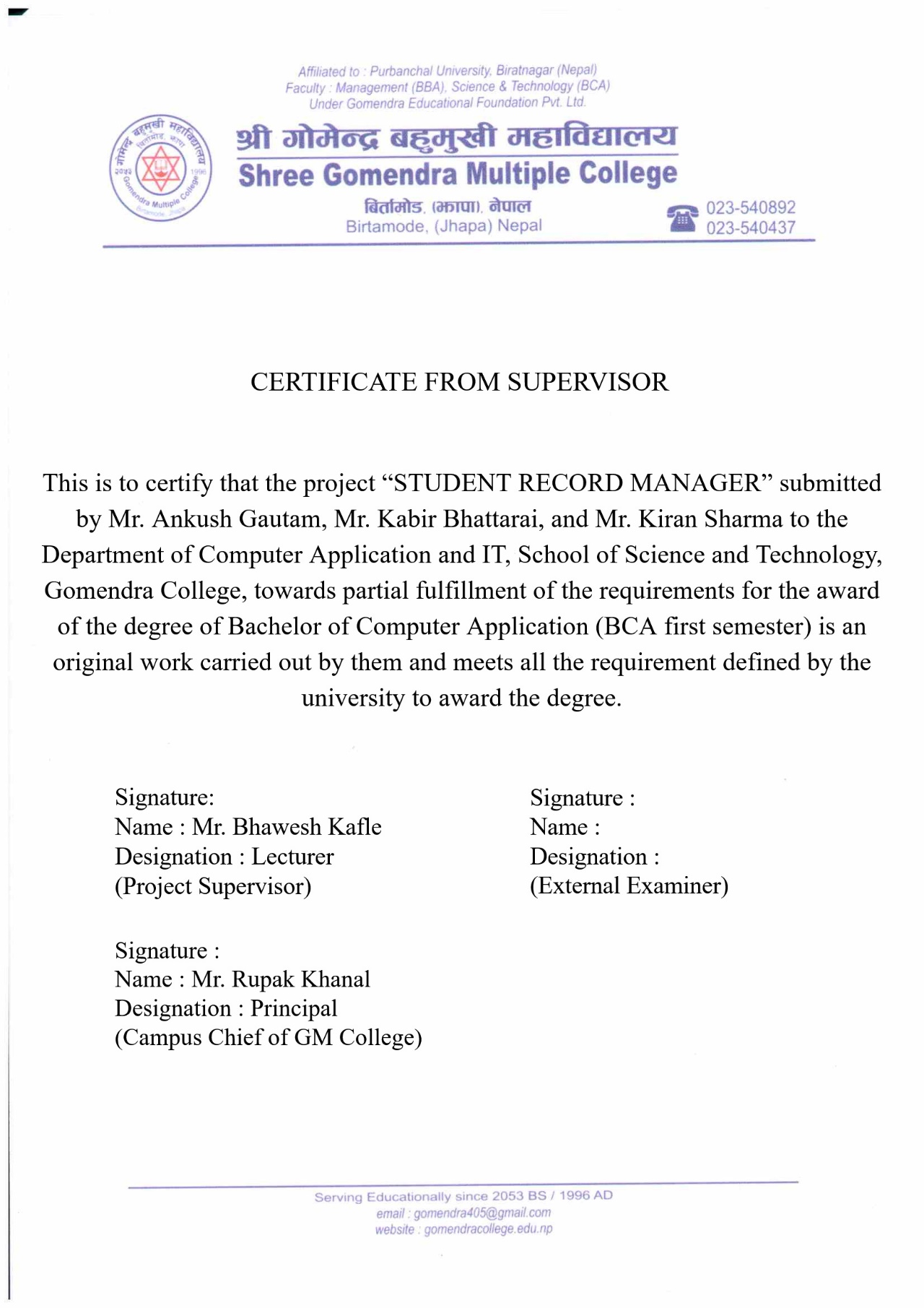
**Kabir Bhattarai [313644]**

**Kiran Sharma Panthi [313646]**

Under the Keen Supervision of

**Mr. Bhawesh Kafle**

FALGUN 2078



# ACKNOWLEDGEMENT

According to the course of BCA first semester determined by the Purbanchal University, a computer project is to be carried out for the partial requirement for Bachelor of Computer Application (BCA). Therefore, we have developed a small ‘C language’ based program, “Student Record Manager”.

First of all, we would like to express our sincere gratitude to Gomendra Multiple College for providing us the opportunity to pursue higher education in such a wonderful academic environment. We are indebted to Mr. Rupak Khanal (Campus Chief, G.M. College) who granted our project work by understanding the need of the students and encouraging us in every aspect of our academic study.

We are grateful to our respected project teacher Mr. Bhawesh Kafle for his persistent help and clear guidance throughout our academic study and completion of this project. His suggestions and guidance in every stage are one of the major reasons for the successful completion of our project. Our project would not have been accomplished in time without his proper guidance.

Finally, we are grateful to Purbanchal University for creating such a wonderful course framework that will undoubtedly assist students in advancing their careers. We are really glad to see a course where we can learn how to implement or use that knowledge in real life.

# ABSTRACT

C is a popular programming language. It was developed by Dennis Ritchie in 1972 AD. Much popular software is made using this language. Therefore, this project is a small implementation of theoretical and practical knowledge of C programming. We are glad to get an opportunity to not only learn to program but also use that learning to implement and create something that can help people. So, this is a brief abstraction of what we have made and from the knowledge, we got during our learning journey of C programming in BCA first semester.

This project is an eye-opener for us which made us realize that no matter how well you think you know something, you will not be able to make it count unless you try to implement it and use it in practice. We learned a lot during these six months of our first semester but when we started to code, we came up with different problems. So, we used to debug the program or search for solutions on the internet, we came to know why things are done this way. We were able to comprehend and see the concept of programming from a different perspective. In this way, we have made a project which has a special place in our hearts as it is our first ever programming project.

The “Student Record Manager” undertaken as a project is based on relevant technologies. The main aim of this project is to develop software for the school and educational institute. This project has been developed to carry out the processes easily and quickly, which is not possible with the current manual system. This software is developed using the C programming language in Dev C++ IDE and is meant for managing the records of students of an educational institute. We have made the software interactive and easy to use. This software improves the performance of an institute by saving the time that used to get wasted in manual recording pieces of information. This software will also help the institute to increase productivity.

# LIST OF TABLES

**Table Page**

[Member 1: Job Schedule Of **Ankush Gautam** 6](#_Toc95929474)

[Member 2: Job Schedule Of **Kabir Bhattarai** 6](#_Toc95929475)

[Member 3: Job Schedule Of **Kiran Sharma Panthi** 6](#_Toc95929476)

[Test Case 8. 1: Login Menu 25](#_Toc95929349)

[Test Case 8. 2: Grade Calculator 25](#_Toc95929350)

[Test Case 8. 3: Main Menu 26](#_Toc95929351)

[Test Case 8. 4: File Handling 26](#_Toc95929352)

[Test Case 8. 5: Sorting Records 26](#_Toc95929353)

[Table 9. 1: Result Of Testing 27](#_Toc95929363)

# LIST OF FIGURES

**Figure No. Title Page**

[Fig 6.2. 1 Flowchart 8](#_Toc95465084)

[Fig 7.4. 1 Setup Page 22](#_Toc95465278)

[Fig 7.4. 2 Main Menu 22](#_Toc95465279)

[Fig 7.4. 3 Add Record 23](#_Toc95465280)

[Fig 7.4. 4 View Record 23](#_Toc95465281)

[Fig 7.4. 5 Edit Record 24](#_Toc95465282)

[Fig 7.4. 6 Search Record 24](#_Toc95465283)

[Fig 7.4. 7 Delete Record 25](#_Toc95465284)

[Fig 7.4. 8 Grade-Sheet 25](#_Toc95465285)

# ABBREVIATIONS

PU Purbanchal University

BCA Bachelor in Computer Application

BBA Bachelor in Business Administration

IDE Integrated Development Environment

SRM Student Record Manager

GM Gomendra Multiple

MB Megabyte

EXE Executable Format

RAM Random Access Memory

GPA Grade Point Average

PC Personal Computer

**Table Of Contents**

**Title Page**

[**Acknowledgement i**](#_Toc95393006)

[**Abstract ii**](#_Toc95393007)

[**List Of Tables iii**](#_Toc95393008)

[**List Of Figures iv**](#_Toc95393009)

[**Abbreviations v**](#_Toc95393010)

[Chapter 1: Introduction 1](#_Toc95393011)

[Chapter 2: Problem Statement 2](#_Toc95393012)

[Chapter 3: Objective 3](#_Toc95393013)

[Chapter 4: Background Study 4](#_Toc95393014)

[Chapter 5: Requirements 5](#_Toc95393015)

[5.1. Hardware Requirements 5](#_Toc95393016)

[5.2. Software Requirements 5](#_Toc95393017)

[5.3. Jobwise Schedule 6](#_Toc95393018)

[Chapter 6: System Design 7](#_Toc95393019)

[6.1. Algorithm 7](#_Toc95393020)

[6.2. Flowchart 8](#_Toc95393021)

[6.3. Source Code 9](#_Toc95393022)

[Chapter 7: Development 17](#_Toc95393023)

[7.1. Header Files 17](#_Toc95393024)

[7.2. Structures 17](#_Toc95393025)

[7.3. User-Defined Functions 18](#_Toc95393026)

[7.4. Screenshots Of Program 20](#_Toc95393027)

[7.5. Files Used 24](#_Toc95393028)

[Chapter 8: Testing 25](#_Toc95393029)

[Chapter 9: Project Result 27](#_Toc95393030)

[Chapter 10: Future Enhancement 28](#_Toc95393031)

[Chapter 11: Conclusion 29](#_Toc95393032)

[References 30](#_Toc95393033)

# CHAPTER 1: INTRODUCTION

**“Student Record Manager”** is a desktop application written in C Language. The main objective of this project is to manage the details of students and calculate their marks. Besides, the program can also add, record, edit, search, delete, calculate grades and sort the details of all the students as per the requirement. We used a structure to define the attributes of a student. Several members in the structure define the details of a student. For instance,

* Roll number
* Name
* Address
* Phone number
* Marks in each subject

The application begins with a setup page that asks the user to set up the program if it is the first time the Application is running. The user needs to enter the school’s name, address of the school, course name (E.g., BCA, BBA), number of subjects, and name of all the subjects. These details will only be asked Once. The user also needs to sign up and input the desired username and password for login after the setup and every time the program is opened. And, after the setup, the user will land on the program’s main menu. In the main menu, the user can choose from various options according to the need of the user.

In the main menu, there are a total of 10 options the user can choose from. We used a switch case and made a function for every option on the menu for readability and to reduce redundancy. The main menu includes adding a record, viewing a record, editing a record, searching a record, deleting records, sorting records, viewing a total number of records, grade-sheet, about the program, and exiting. The menu is inside a loop so the program won’t end unless the user hits 0 or closes the window itself.

# CHAPTER 2: PROBLEM STATEMENT

Today most schools and educational institutes use computers to calculate and store records of students. But there is very few software that understands the need of colleges and other educational institutes. We need to manually do everything from inputting all the details of students to writing formulas to calculate grades. From making a grade sheet design to viewing the overall performance of a class.

Moreover, when it comes to calculating the marks of every single student and storing it, it becomes troublesome as we need different heavy software or have to pay subscriptions to use such applications. We even have to remember formulas and should know how to operate such software. Moreover, this heavy software requires a high-end device or PC to run the software quickly and smoothly which is too expensive for a simple task. Similarly, a normal computer user can get overwhelmed by the unlimited features of that software. They cannot utilize the software if they cannot properly use the features that the program contains. And a normal user needs years of practice before he/she can learn all those formulas and functions to increase their productivity and speed.

Therefore, we decided to make a simple application that does this kind of job in a matter of seconds and is simple and easy to use. User doesn’t need to have vast knowledge to use this application nor do they have to remember any kind of formulas or code to operate this program. Basic knowledge of computers is enough to operate this program and the user just needs to input records of students and all the other calculations and sorting things will be done automatically.

# CHAPTER 3: OBJECTIVE

Due to the problems that we discussed earlier, we brainstorm what our objectives should be and we came to the conclusion that we should make something that is simple yet effective. No overwhelming features just a simple straightforward program that does what it says. Therefore, these are the objectives we wanted to achieve during our project development.

**Objectives**

* To provide a simple and easy interface
* To provide fast operations like sorting and displaying grade-sheet
* To increase productivity and efficiency
* To provide a light-weighted free of cost program
* To provide flexibility
* To save time
* To be precise and accurate
* To be dynamic so that any school or institute can use
* To provide easy access to stored data
* Security

# CHAPTER 4: BACKGROUND STUDY

As we already mentioned above that the use of software in schools and other educational institutes is increasing day by day. So, we thought of the problem regarding the bookkeeping of students’ records and details. And as we had a project to make, we decided to create a simple program for recording and calculating the marks and grades of students.

We research thoroughly what kind of feature would be useful for schools and institutions. And we tried to create a program as much as we know about this language which we learned for six months. So, we are bounded by limited knowledge as it needs years of practice to master a programming language and to create a brilliant piece of art.

But we tried to do as much as we could. Our respected and dear teacher Mr. Bhawesh Kafle had already taught us the basic concepts and logic of adding, displaying, editing, storing information in the file. So, we kind of had a structure but we needed to have some more features and functions to make our project stand out. Therefore, we build features like sorting of records according to roll number, name, or grade of students, made the program colorful and attractive, made a grade calculator, and other small features like name checker and roll checker which would validate if the roll number or name already exists in the file.

So, we had our background study of what kind of program we are doing and making logic and functions according to it. But after we gave the project a proper structure, we got more ideas and features we could add, so the process was kind of think, implement and modify and again think, implement and modify.

# CHAPTER 5: REQUIREMENTS

## 5.1. Hardware Requirements

Minimum hardware requirements for efficient operation of this project:

* Processor : Pentium and above
* Hard disk : 50 MB or above free space
* RAM : 128 MB or above

## 5.2. Software Requirements

Minimum software requirements for efficient operation of this project:

As this program is written in C language, the IDE already compiles the code and provides us with an executable file. So, we don’t need any software to run this application. We don’t even need a C-based IDE to run as long as we have the executable file (.exe file).

* Operating system : WINDOWS 98 or above
* Program Application : StudentRecordManager.exe

## 5.3. Jobwise Schedule

Member 1: Job Schedule of **Ankush Gautam**

|  |  |  |  |
| --- | --- | --- | --- |
| **JOB ID** | **JOB DESCRIPTION** | **DATE FROM** | **DATE TO** |
| A1 | Programming, Designing, Debugging | Kartik 23, 2078 | Poush 30, 2078 |
| A2 | Documentation | Poush 25, 2078 | Magh 10, 2078 |

Member 2: Job Schedule of **Kabir Bhattarai**

|  |  |  |  |
| --- | --- | --- | --- |
| **JOB ID** | **JOB DESCRIPTION** | **DATE FROM** | **DATE TO** |
| A1 | Programming | Kartik 23, 2078 | Poush 30, 2078 |
| A2 | Documentation | Poush 25, 2078 | Magh 10, 2078 |

Member 3: Job Schedule of **Kiran Sharma Panthi**

|  |  |  |  |
| --- | --- | --- | --- |
| **JOB ID** | **JOB DESCRIPTION** | **DATE FROM** | **DATE TO** |
| A1 | Programming, Debugging | Kartik 23, 2078 | Poush 30, 2078 |
| A2 | Documentation | Poush 25, 2078 | Magh 10, 2078 |

# CHAPTER 6: SYSTEM DESIGN

We have designed a system where users can add a record, view records, edit records, etc. Therefore, we need to present a variety of options to the user. So, we have used a switch case to display our main menu where the users will get various options to choose from. We designed the logic and structure of this program first with the help of an Algorithm and Flowchart which gave us a clear picture of what we wanted to make and implement.

## 6.1. ALGORITHM

**STEP 1: START**

**STEP 2: LOGIN ();**

1. If username and password match, go to step 3.
2. If the username and password are invalid, go to step 2 again.

**STEP 3: MENU ();**

**Case 1:** add record (); break;

**Case 2:** view record (); break;

**Case 3:** edit record (); break;

**Case 4:** search record (); break;

**Case 5:** delete record (); break;

**Case 6:** sort record (); break;

**Case 7:** no of records (); break;

**Case 8:** grade-sheet (); break;

**Case 9:** about (); break;

**Case 0:** exit (0);

**STEP 4:** Go to step 3 if Choice is not 0.

**STEP 5: STOP**

## 6.2. FLOWCHART

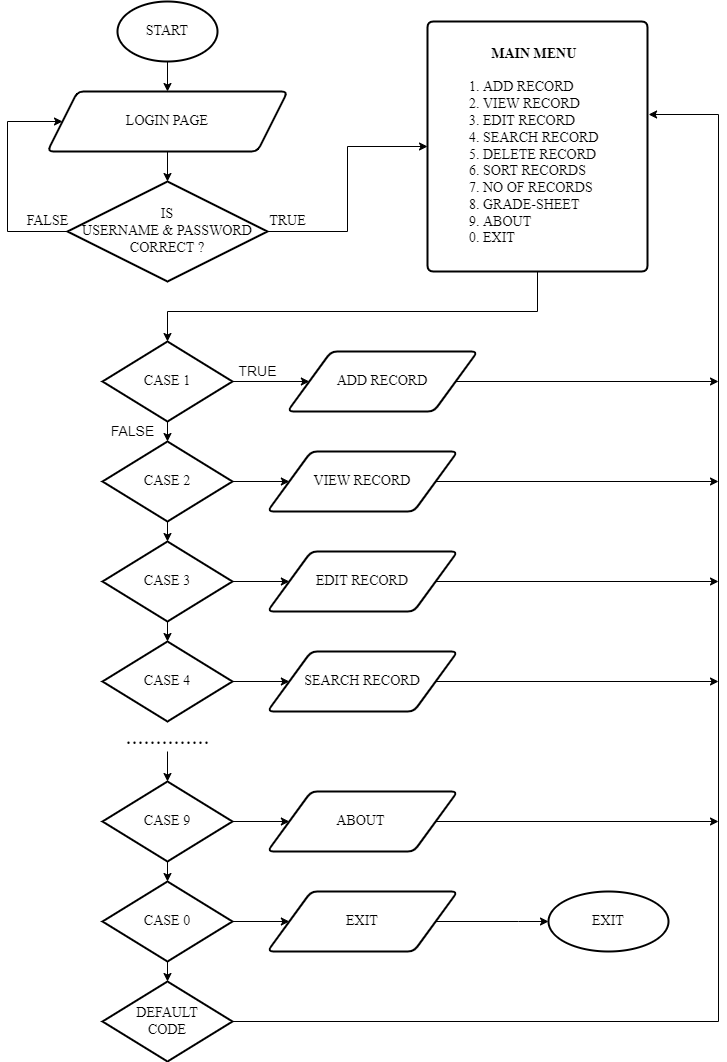
****

Fig 6.2. 1: Flowchart of the Program

## 6.3. SOURCE CODE

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

struct setups {

char schoolname[32], address[32], programme[32], subject[8][32];

int semester, subs;

struct settings {

char username[10], password[10];

} setting;

} setup;

struct student {

char name[24], address[32], phone[14], grade[2];

int roll;

float marks[8], percent, total, gpa;

char

} std;

void firstsetup();

void menu();

void add\_record();

void view\_record();

void edit\_record();

void search\_record();

void delete\_record();

void sort\_record();

int no\_of\_record();

void gradesheet();

void about();

void login();

char \*grade\_calc(float);

void inputdetails();

void menu() {

int choice;

do {

header("STUDENT RECORD MANAGER"); // title of the page

printf("\n\n\tWELCOME TO THE MAIN MENU\n\t-------------------");

printf("\n\n\t1.ADD RECORD");

printf("\n\t2.VIEW RECORD");

printf("\n\t3.EDIT RECORD");

printf("\n\t4.SEARCH RECORD");

printf("\n\t5.DELETE RECORD");

printf("\n\t6.SORT RECORDS");

printf("\n\t7.NO OF RECORDS");

printf("\n\t8.GRADE-SHEET");

printf("\n\t9.ABOUT");

printf("\n\t0.EXIT");

printf("\n\n\tEnter your choice:\n\t>> ");

scanf("%d", &choice);

switch (choice) {

case 1: add\_record(); break;

case 2: view\_record(); break;

case 3: edit\_record(); break;

case 4: search\_record(); break;

case 5: delete\_record(); break;

case 6: sort\_record(); break;

case 7: no\_of\_records(); break;

case 8: gradesheet(); break;

case 9: about(); break;

case 0: exit(0);

}

} while (choice != 0);

}

//Add record

void add\_record() {

header("ADD RECORD");

FILE \*fadd = fopen("record.dat", "ab");

int i, n = no\_of\_record();

char yesno;

printf("\n\tRecord No. %d", n + 1);

inputdetails();

fwrite(&std, sizeof(struct student), 1, fadd);

fclose(fadd);

printf("\n\tRecord successfully stored!\n\t--------------------\n");

printf("\tDo you want to add more?(Press 'y' for Yes)\n\t>> ");

scanf(" %c", &yesno);

if (yesno == 'y' || yesno == 'Y')

add\_record();

else

menu();

}

void view\_record() {

header("VIEW RECORD");

int filesize = no\_of\_record();

FILE \*fview = fopen("record.dat", "rb");

if (fview == NULL || filesize == 0) {

printf("\n\tFile doesn't exist!Please ADD RECORD first.\n\n\t");

system("pause");

menu();

}

printf("\nROLL%-30sTOTAL\tPERCENT%-12sGRADE\n","NAME", "GPA");

while (fread(&std, sizeof(std), 1, fview)) {

printf("\t %02d\t%-30s%06.2f\t\t%6.2f %%\t%-14.2f%s\n", std.roll, std.name, std.total, std.percent, std.gpa, std.grade);

}

fclose(fview);

system("pause");

}

//search-menu

void search\_record() {

header("SEARCH RECORD");

char name[24];

int j, found = 0 ,n = no\_of\_record();

FILE \*fp = fopen("record.dat", "rb");

if (fp == NULL || n == 0) {

printf("\n\tFile doesn't exist!Please ADD RECORD first.\n\n\t");

system("pause");

menu();

}

printf("\n\n\tEnter the NAME of the Student: ");

scanf(" %[^\n]", name);

while (fread(&std, sizeof(std), 1, fp)) {

if (strcmp(name, std.name) == 0) {

found = 1;

printf("\n\n\tPersonal Details\n\t----------------\n");

printf("\t%-10s %s\n\t%-10s %s\n\t%-10s %s\n", "NAME", std.name, "ADDRESS", std.address, "PHONE", std.phone);

printf("\n\n\tAcademic Details\n\t----------------\n");

printf("\t%-10s %.2f\n\t%-10s %.2f\n\t%-10s %s\n", "TOTAL", std.total, "GPA", std.gpa, "GRADE", std.grade);

printf("\n\n\t%-30s MARKS\n", "SUBJECT");

printf("\t------------------------------------\n");

for (j = 0; j < setup.subs; j++) {

printf("\t%-30s%6.2f\n", setup.subject[j],std.marks[j]);

}

}

}

if (!found) {

printf("\n\tNo Match Found!\n\t----------------");

}

printf("\n\t");

system("pause");

}

void delete\_record() {

header("DELETE RECORD");

char yesno;

int rollno, found = 0, record\_count = no\_of\_record();

FILE \*tempfile = fopen("tempfile.dat", "wb");

FILE \*fp = fopen("record.dat", "rb");

if (fp == NULL || record\_count == 0) {

printf("\n\tFile doesn't exist! Please ADD RECORD first.\n\n\t");

system("pause");

menu();

}

printf("\\tROLL%-30sTOTAL\t\tPERCENT\t%-12sGRADE\n", "NAME", "GPA");

while (fread(&std, sizeof(std), 1, fp)) {

printf("\t%02d\t%-30s%06.2f\t\t%6.2f %%\t%-14.2f%s\n", std.roll, std.name, std.total, std.percent, std.gpa, std.grade);

}

rewind(fp);

printf("\n\tEnter the ROLLNUMBER of the Student to DELETE: ");

scanf("%d", &rollno);

while (fread(&std, sizeof(std), 1, fp)) {

if (rollno == std.roll)

found = 1;

else

fwrite(&std, sizeof(std), 1, tempfile);

}

fclose(fp);

fclose(tempfile);

if (found) {

fp = fopen("record.dat", "wb");

tempfile = fopen("tempfile.dat", "rb");

while (fread(&std, sizeof(std), 1, tempfile)) {

fwrite(&std, sizeof(std), 1, fp);

}

fclose(fp);

fclose(tempfile);

printf("\n\tRecord DELETED Successfully.");

} else {

printf("\n\tNo Match Found!\n\t----------------\n");

}

printf("\tDo you want to DELETE Again?(Press 'y' for Yes)\n\t>> ");

scanf(" %c", &yesno);

if (yesno == 'y' || yesno == 'Y')

delete\_record();

else

menu();

}

//login - menu

void login() {

header("LOGIN PAGE");

char userinput[10], pass[10];

printf("\n\n\tEnter Username: ");

scanf(" %[^\n]", userinput);

printf("\tEnter Password: ");

scanf(" %[^\n]", pass);

if ((strcmp(setup.setting.username, userinput) == 0) && (strcmp(setup.setting.password, pass) == 0)) {

menu();

} else {

printf("\n\tInvalid Username or password!\n\t");

system("pause");

login();

}

}

void gradesheet() {

header("GRADE-SHEET");

int rno, j, found = 0, n = no\_of\_record();

FILE \*fp = fopen("record.dat", "rb");

printf("\n\tEnter the ROLLNUMBER of the Student: ");

scanf("%d", &rno);

while (fread(&std, sizeof(std), 1, fp)) {

if (rno == std.roll) {

found = 1;

header("GRADE SHEET");

printf("\n\t%s\n\t%s ", strupr(setup.schoolname) , strupr(setup.address)););

printf("\n\n\tTHE GRADE(S) SECURED BY %s %s", strupr(std.name), strupr(std.address));

printf("\n\tROLL NO: %d\tPROGRAMME: %s\t\tSEMESTER: %d\n\n", std.roll, setup.programme, setup.semester);

printf("\tSN\t%-30sCREDIT\tMARKS\tGRADE\tGRADE-POINT\n", "SUBJECT"); for (j = 0; j < setup.subs; j++) {

printf("\t%02d\t%-30s\t4.0\t%.2f\t %s\t %.2f\n", j + 1, setup.subject[j], std.marks[j], grade\_calc(std.marks[j]), std.marks[j] / 25);

}

printf("\n\tGRADE POINT AVERAGE (GPA) : %.2f\n", std.gpa);

printf("\1.One Credit Hour equals 32 Clock Hours.\n\t2.TH : THEORY\tPR : PRACTICAL");

printf("\n\t3.\*@ : Absent\n");

printf("\n\n\tCHECKED BY:\n\t%s,%s\n", setup.schoolname, setup.address);

}

}

if (!found) {

printf("\n\tNo Match Found!\n\t----------------");

}

menu();

}

void firstsetup() {

header("SETUP");

FILE \*fsetup = fopen("setup.dat", "wb");

printf("\n\n\tPlease Take a Moment to Setup.\n");

printf("\tEnter School Name: ");

scanf(" %[^\n]s", setup.schoolname);

printf("\tEnter Address: ");

scanf(" %[^\n]s", setup.address);

printf("\tEnter Programme (For e.g: BCA, BBA): ");

scanf(" %[^\n]s", setup.programme);

printf("\tEnter semester (For e.g: 1,2,3): ");

scanf("%d", &setup.semester);

subs\_again:

printf("\tEnter No. Of Subjects: ");

scanf("%d", &setup.subs);

if (setup.subs > 8 || setup.subs <= 0) {

printf("\t(Sorry! 0 < Subject <= 8)\n");

goto subs\_again;

}

for (int i = 0; i < setup.subs; i++) {

printf("\tEnter Name of subject %d: ", i + 1);

scanf(" %[^\n]", setup.subject[i]);

}

printf("\n\n\tSIGN UP\n\t-------\n\n");

printf("\tEnter Username: ");

scanf(" %[^\n]", setup.setting.username);

printf("\tEnter Password: ");

scanf(" %[^\n]", setup.setting.password);

fwrite(&setup, sizeof(struct setups), 1, fsetup);

fclose(fsetup);

printf("\n\tSET UP SUCCESSFUL! Remember Your Details.\n\t");

menu();

}

//END OF CODE

# CHAPTER 7: DEVELOPMENT

## 7.1. Header Files

1. stdio.h
2. conio.h
3. string.h
4. stdlib.h

## 7.2. Structures

* 1. *struct student {*

*char name [24];*

*char address [32];*

*char phone [10];*

*int roll;*

*float marks [10];*

*float percent;*

*float total;*

*float gpa;*

*char grade [2];*

*} std;*

1. *struct setups {*

*char schoolname [32];*

*char address [32];*

*char programme [32];*

*int semester;*

*int subs;*

*char subject [10] [30];*

*struct settings {*

*char username [10];*

*char password [10];*

*} setting;*

*} setup;*

## 7.3. User-Defined Functions

1. Void first setup ();

In this function, the user needs to provide all the necessary details for the system to work correctly. This setup function runs when the program is run for the first time. The user needs to input details like School Name, School Address, Number of subjects, and Name of all the subjects. This function will create a file named “setup.dat”.

2. Void Login ();

In this function, the user needs to provide the username and password, he/she had entered while setting up the setup page. If the username or password matches, the user is directed to the menu. And, if the username or password is incorrect, the user has to try again.

3. Void Menu ();

This is the Center of the program. The user has a list of possible options to choose from. The user cannot enter a number other than 0 to 9. If entered invalid option, the program will be redirected to Menu again. The program will never exit unless the user entered Number ‘0’ or close the window.

4. Void Add Record ();

In this function, the program takes details of the students stores them in the file. The details include full name, roll number, phone, address, marks in all the subjects. This is the first option in the menu as it is an integral part of adding and storing records.

5. Void View Record ();

In this function, the user can view the records of students that are stored in the file. Users can view roll number, name, total marks, and GPA. This function provides an overview of the entire records of all the students that are present in the file.

6. Void Edit Record ();

In this function, the user can edit and modify the details of the student by entering the student’s roll number. The user can change any details of a student whether it is personal or academic.

7. Void Search Record ();

In this function, the user can Search and View the details of a student by entering the student’s name. The Search Menu displays the personal as well as academic details of a student including his/her marks in each subject.

8. Void Delete Record ();

In this function, the user can delete and remove the record of any students by entering the student’s Roll number. The Deleted record cannot be recovered again so the user must be careful whether the record is important or not.

9. Void Sort Record ();

In this function, the user can sort the records of students according to Name or GPA, or Roll number. Users can also view the records in the spreadsheet. This function has a sub-menu for the sorting choice.

10. Void Grade-Sheet ();

In this function, the user can view the grade sheet of any recorded student by entering the student’s Roll number. The Grade-sheet include marks of student and the overall record of a student in a proper format.

## 7.4. Screenshots Of Program

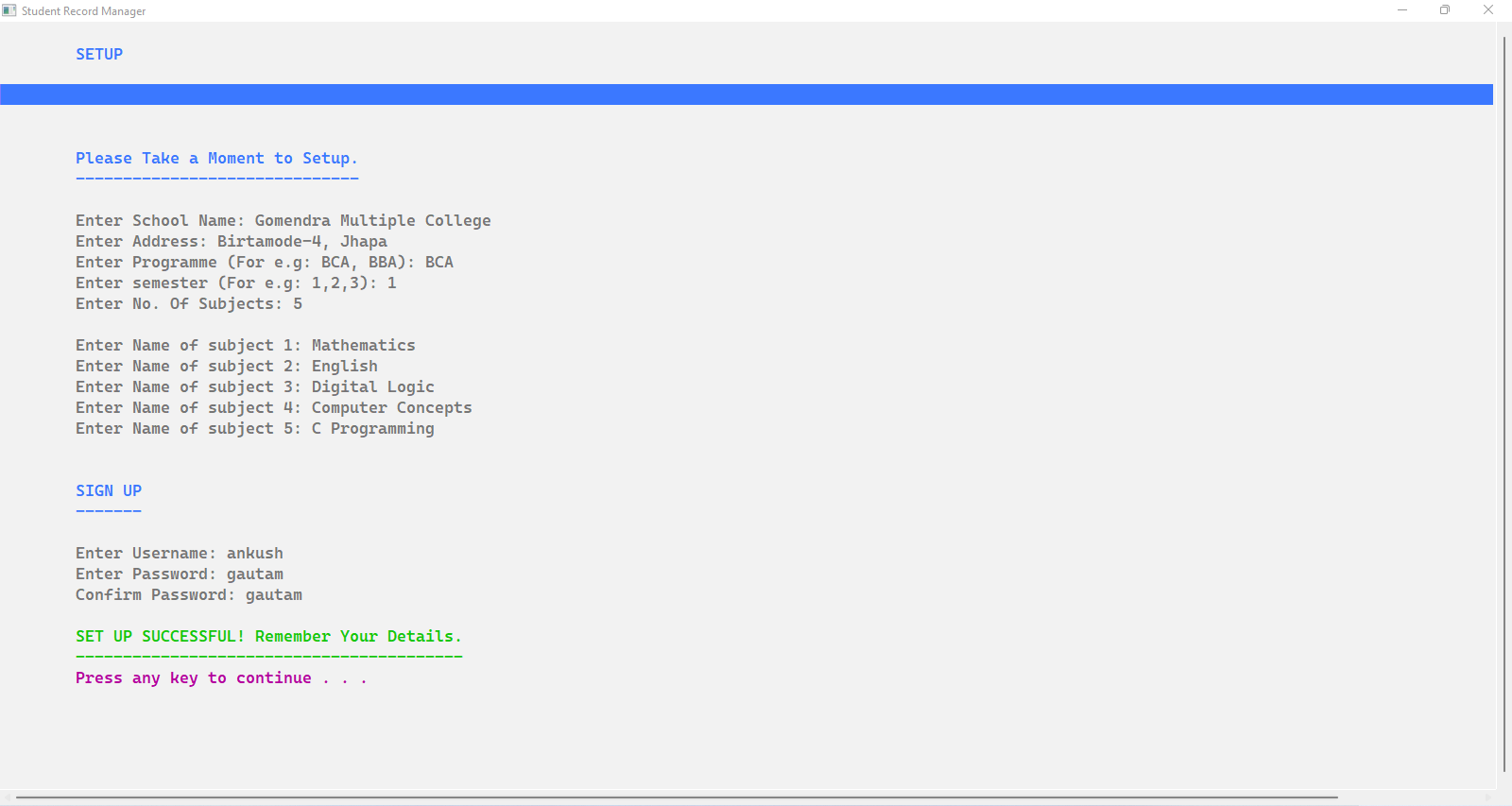
****

Fig 7.4. 1: Setup Page

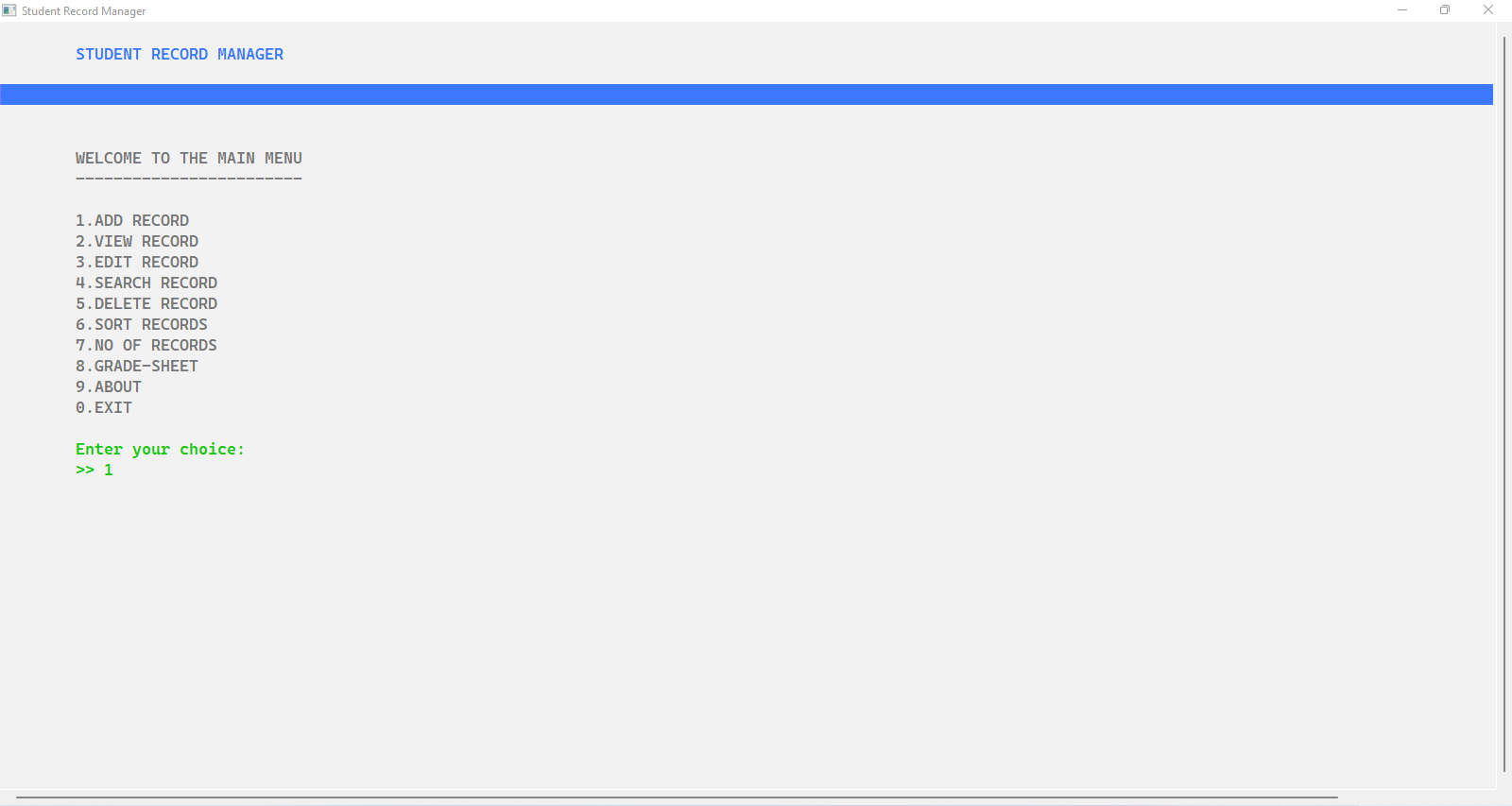
****

Fig 7.4. 2: Main Menu

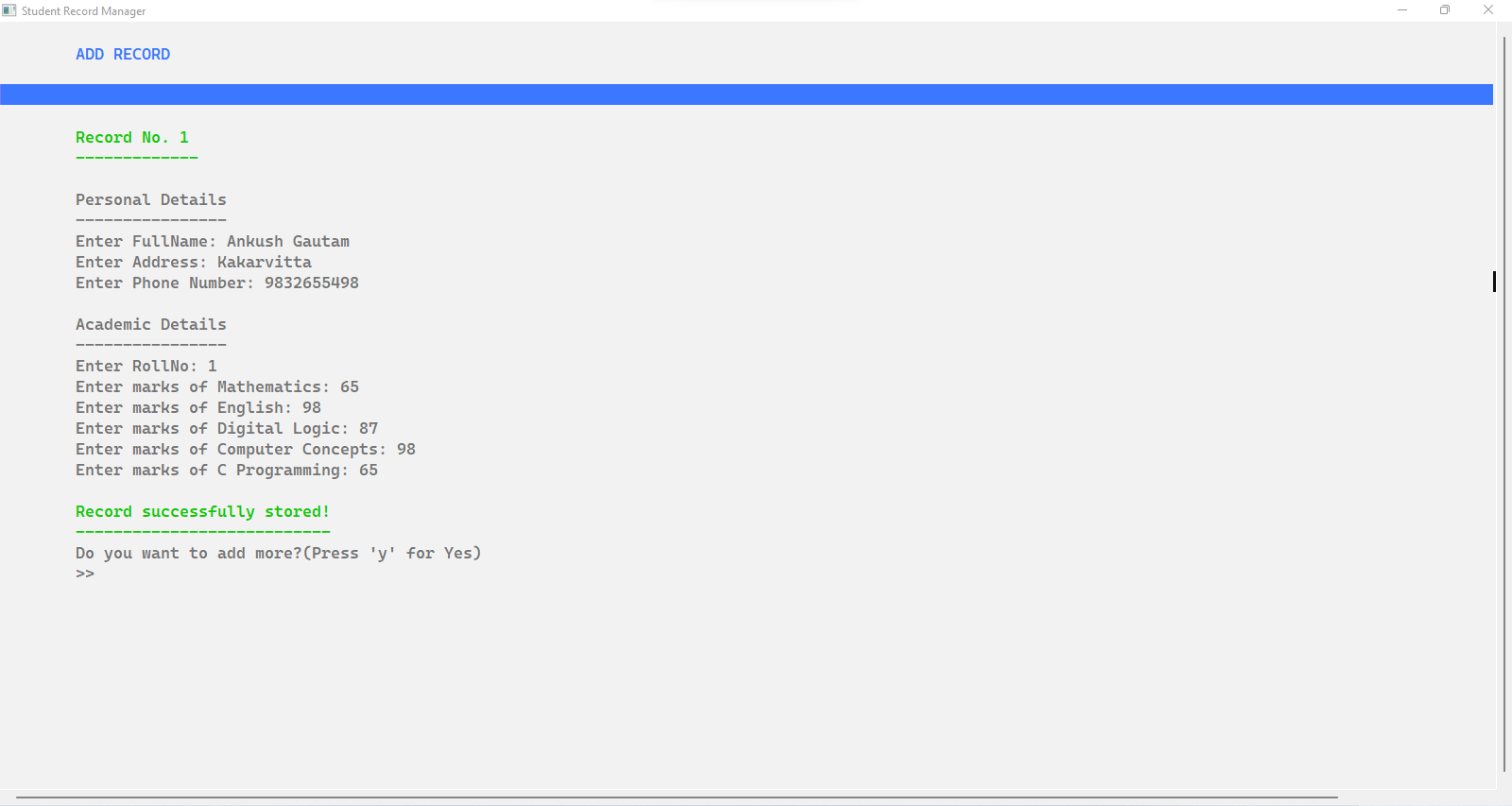
****

Fig 7.4. 3: Add Record

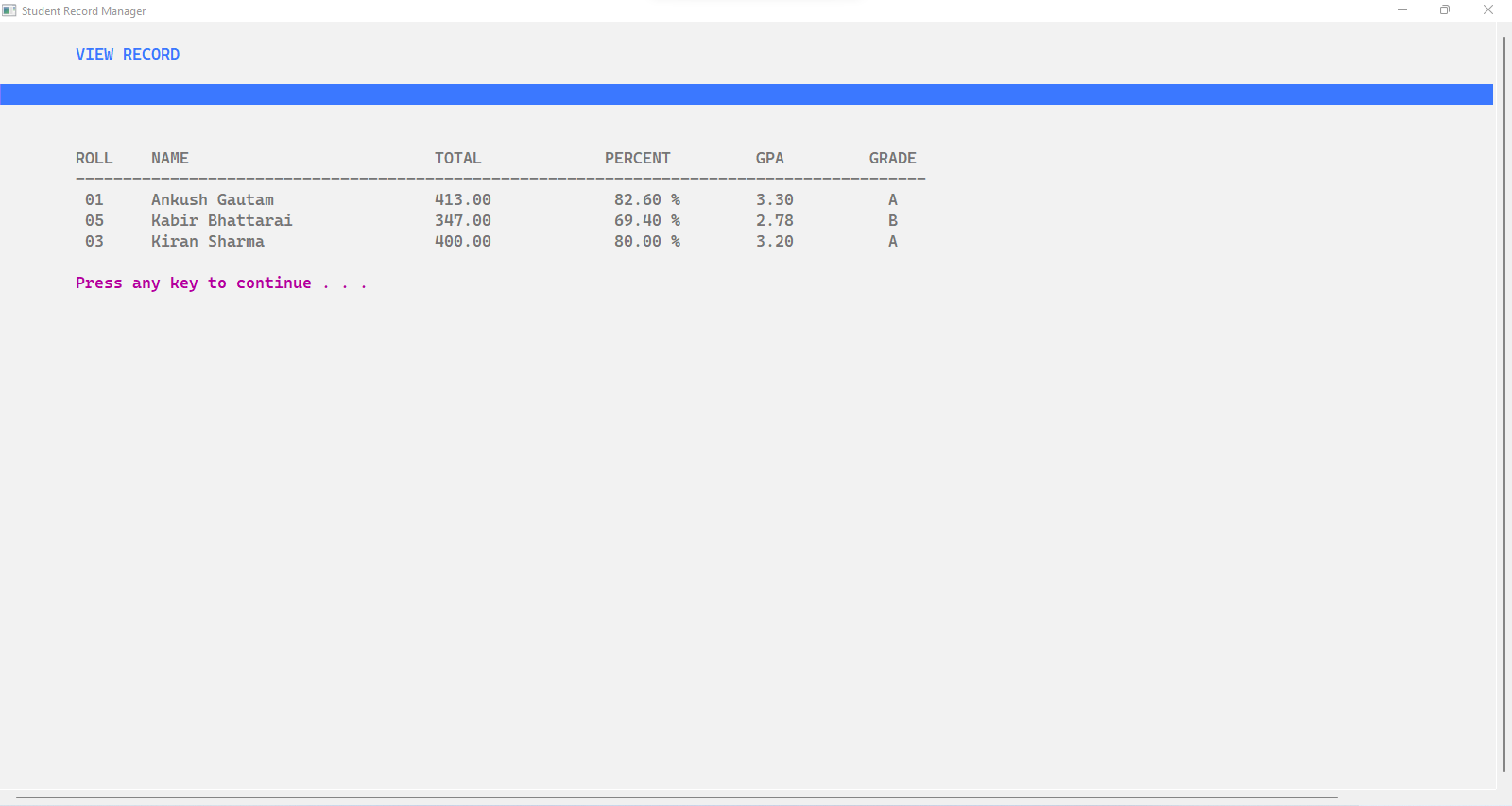
****

Fig 7.4. 4: View Record

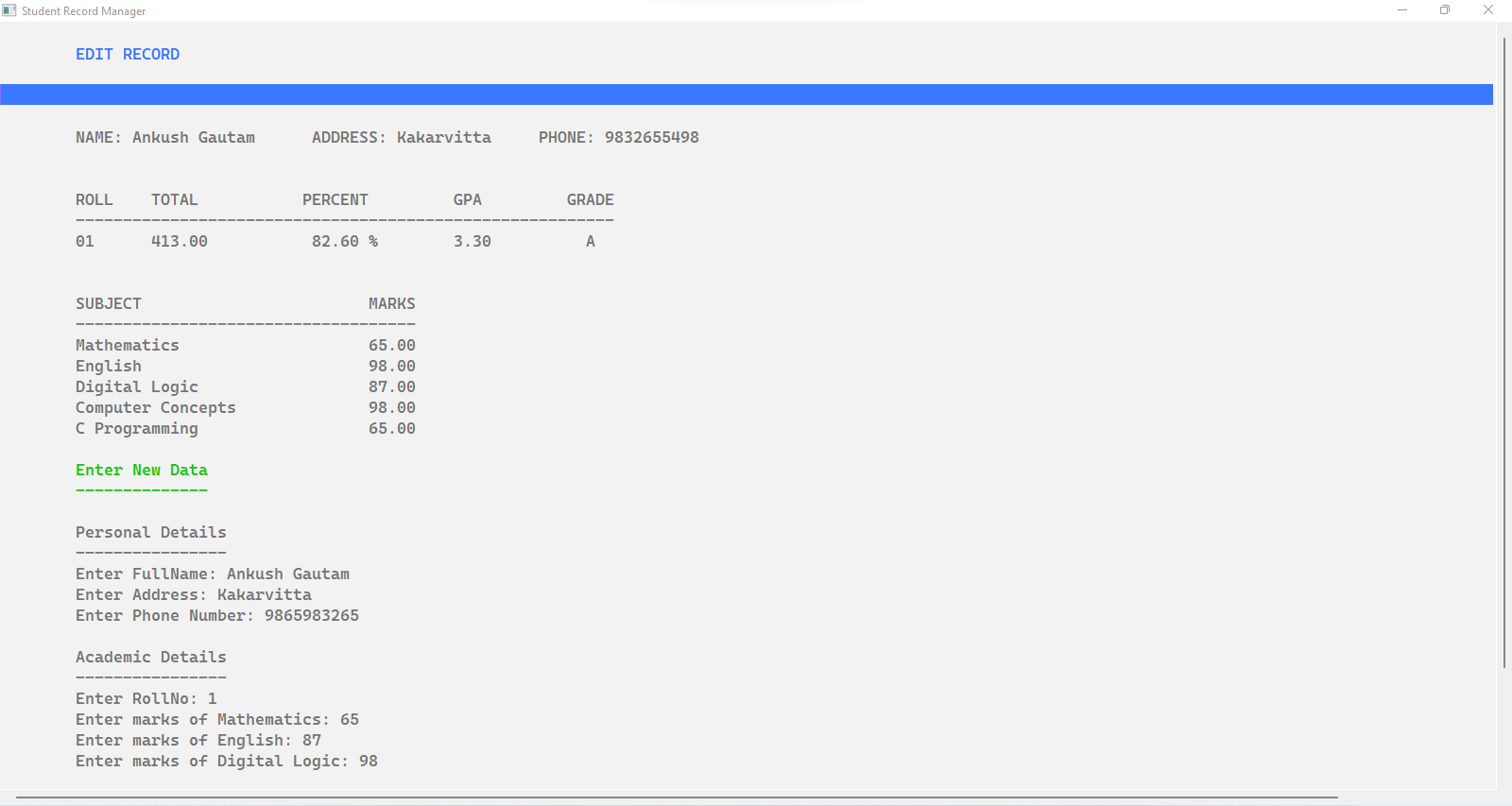
****

Fig 7.4. 5: Edit Record

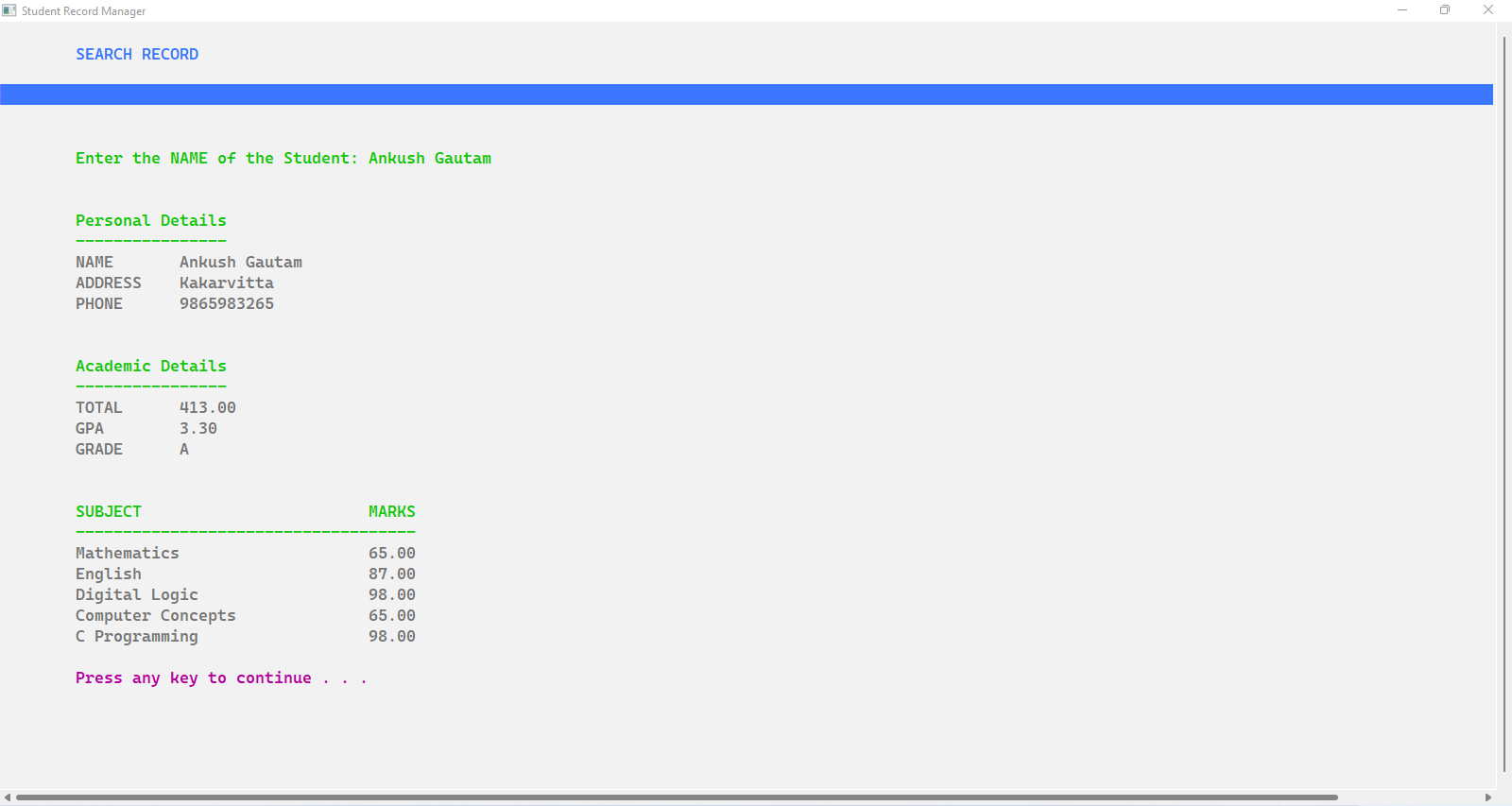
****

Fig 7.4. 6: Search Record

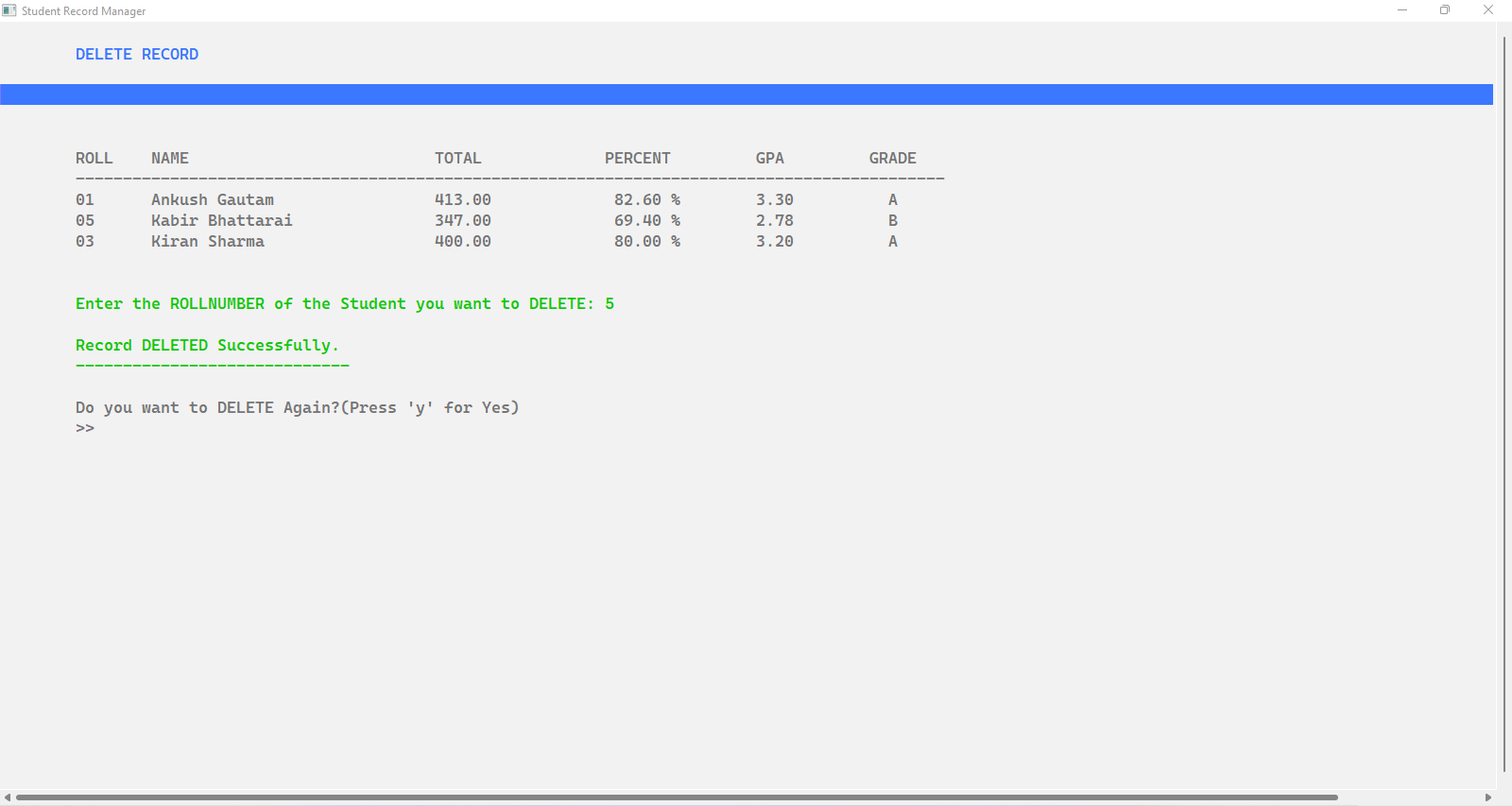
****

Fig 7.4. 7: Delete Record

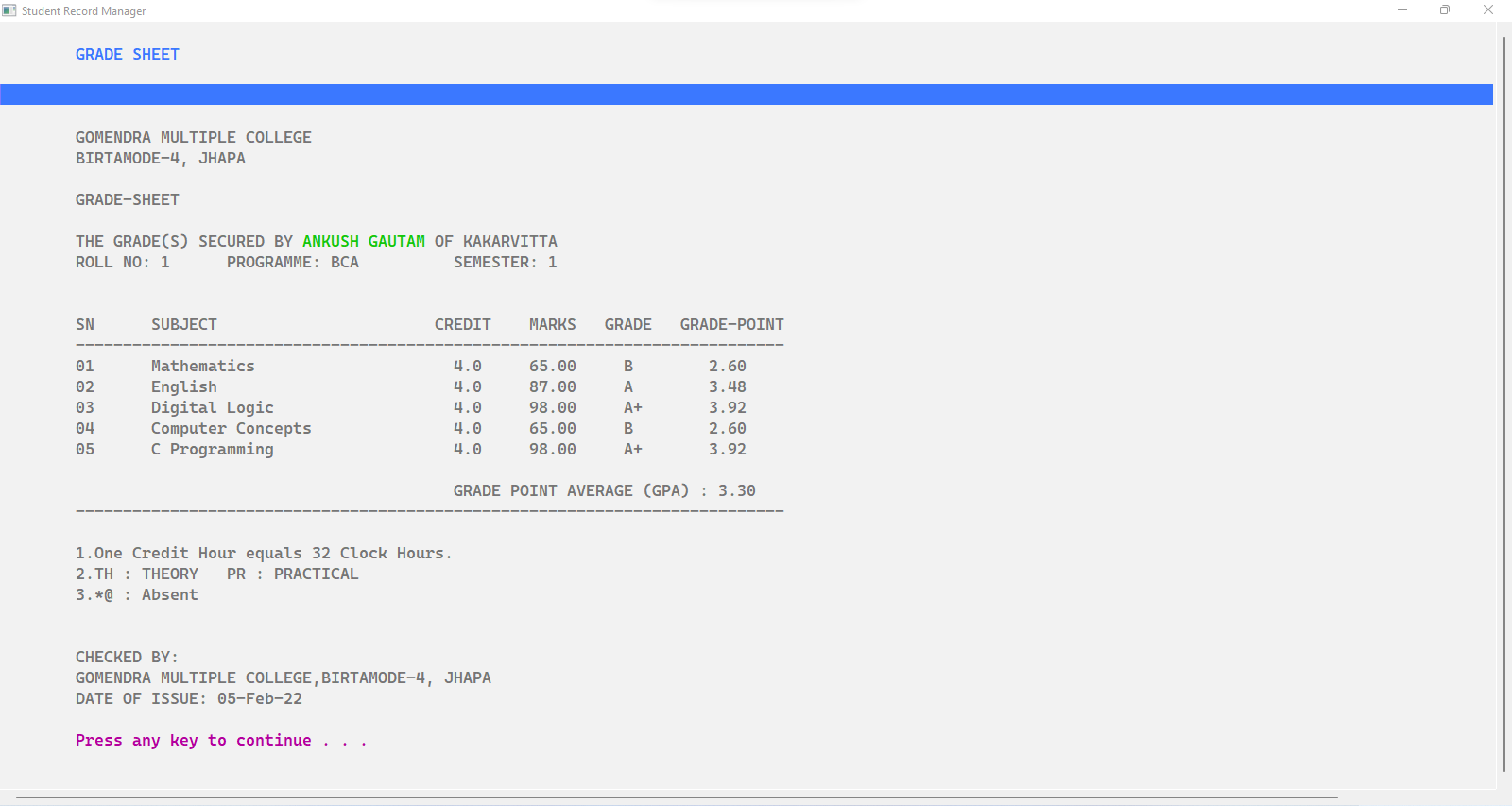
****

Fig 7.4. 8: Grade-Sheet

## 7.5. Files Used

1. **about.dat**

This file includes details about the developers and the program. This file already exists in the program as it is made directly by us.

1. **setup.dat**

This file includes the settings that the user entered while setting up the program for the first time. This includes several subjects, name of subjects, school name, school address, program and semester, sign up (username & password).

1. **record.dat**

This file is created when the user records students in the program. This file includes only the details of the students. This is the most crucial file of the program.

1. **tempfile.dat**

This file is created while deleting or editing the records of a student. This is a temporary file and is safe to be deleted.

# CHAPTER 8: TESTING

Test Case 8. 1:

**Test Case Description: LOGIN MENU**

|  |  |
| --- | --- |
| **SCENARIO NO. 1** | Entering Incorrect Username and Password. |
| **ACTION** | Enter Username: admin  Enter Password: something |
| **REQUIRED OUTPUT** | Invalid username or password! |
| **ACTUAL OUTPUT** | Invalid username or password! |
| **REMARKS** | PASS |

|  |  |
| --- | --- |
| **SCENARIO NO. 2** | Entering Correct Username and Password. |
| **ACTION** | Enter Username: admin  Enter Password: password |
| **REQUIRED OUTPUT** | Welcome to the Main Menu. |
| **ACTUAL OUTPUT** | Welcome to the Main Menu. |
| **REMARKS** | PASS |

Test Case 8. 2:

**Test Case Description: GRADE-CALCULATOR**

|  |  |
| --- | --- |
| **SCENARIO** | To check if the grade calculator works correctly |
| **ACTION** | Total marks = 439 |
| **REQUIRED OUTPUT** | GPA = 3.51 and Grade = A |
| **ACTUAL OUTPUT** | GPA = 3.51 and Grade = A |
| **REMARKS** | PASS |

Test Case 8. 3:

**Test Case Description: FILE HANDLING**

|  |  |
| --- | --- |
| **SCENARIO** | To check if details are stored in the file correctly |
| **ACTION** | Added records of a student |
| **REQUIRED OUTPUT** | Records are stored in the file properly. |
| **ACTUAL OUTPUT** | Records are stored in the file properly. |
| **REMARKS** | PASS |

Test Case 8. 4:

**Test Case Description: MAIN MENU**

|  |  |
| --- | --- |
| **SCENARIO** | To check if the Menu displays all the options |
| **ACTION** | Viewing the main menu after logging in |
| **REQUIRED OUTPUT** | All 10 options are works fine. |
| **ACTUAL OUTPUT** | All 10 options are working fine. |
| **REMARKS** | PASS |

Test Case 8. 5:

**Test Case Description: SORTING RECORDS**

|  |  |
| --- | --- |
| **SCENARIO** | To check records are sorted correctly |
| **ACTION** | Checking sort by name, sort by GPA |
| **REQUIRED OUTPUT** | Records are sorted in a proper format |
| **ACTUAL OUTPUT** | Records are sorted in a proper format |
| **REMARKS** | PASS |

# CHAPTER 9: PROJECT RESULT

Along with all the above program testing we showed in the earlier chapter, we persistently tested every function of our project so that it doesn’t behave unexpectedly. Many unexpected behaviors arose but we tackle them. For instance, there was a problem with file handling where every student would have the same roll number but we found the solution after many hours of debugging. Every program has a bug that arises once in a while so we got to be up to date with it, to make our program less vulnerable. Some of the results of our testing are shown below:

|  |  |  |
| --- | --- | --- |
| S.N. | TEST OBJECTIVE | RESULT |
|  | To check whether the program runs or not | *Successful* |
|  | To check if the menu displays all the options or not | *Successful* |
| 3 | To check if all the menu options are correct or not | *Successful* |
| 4 | To check if the record is added in the file or not | *Successful* |
| 5 | To check if the program calculates the correct GPA | *Successful* |
| 6 | To check if the ‘View Record’ displays details or not | *Successful* |
| 7 | To check if the program returns to Menu or not | *Successful* |
| 8 | To check if the exit option works or not | *Successful* |
| 9 | To check if the login/authentication works or not | *Successful* |
| 10. | To check if the sorting of records is correct or not | *Successful* |
| 11. | To check if colors are displayed correctly or not | *Successful* |

Table 9. 1: Result of Testing

# CHAPTER 10: FUTURE ENHANCEMENT

No program is free of bugs and no program is 100% completed. There is always room for improvement and enhancement. Similarly, this project also has some bugs and glitches that we can fix in the future. Some of the problems may be poor memory management like we could have optimized the code more so it can use every last bit of memory.

There is some room for improvement in memory management. We can also make a dynamic allocation of memory so we can optimize the program by allocating the memory required in the run time. For example, we have initialized a subject array with 8 subjects but the user can choose only 5 subjects, so the remaining 3 spaces are just wasted. We could optimize and work on 2-dimensional dynamic memory allocation.

Moreover, we could also add different new features and functions like we can implement sorting of records by highest marks of students in a particular subject or we can implement features like multiple users can access the program at once. We can also implement multiple classes features like this program can only store records of one class or grade but we could enhance the program by adding a feature that can store information of students of different or more than one class. There will always be some ideas we can implement.

But we are bound by our limited knowledge and time. So, as we grow and learn more about programming and how a program is made, we can implement or make more projects with more features. Any ideas or feedback is much appreciated.

# CHAPTER 11: CONCLUSION

As this is our first ever programming project, we are very happy to see how far we have come. We are glad that we came close enough to what we have imagined before starting this project. With the learning of six months, we have implemented a decent program, so we are optimistic that with persistent learning in the world of programming we can build something that can be useful to people.

“Student Record Manager” has evolved the way we wanted and we are grateful for all the help we got. We tried to make a simple yet effective program and we think we kind of were successful. This project is a small attempt to help educational institutions to increase their productivity and provide them with a free-of-cost program that calculates and displays the result of an entire class in a matter of seconds.

As we already discussed in the above chapter about further enhancement and every program has room for further improvement so any kind of feedback is most welcomed. We are not only open to feedback but any constructive criticism is also welcomed with a happy face.

We would again like to thank our respected teacher Mr. Bhawesh Kafle for consistent support and help. We are also thankful to our classmates for being so creative and helpful that inspire us to push ourselves further and learn more every day.

Thank you everyone for your time.

# REFERENCES

**Online**

* https://www.geeksforgeeks.org/c-programming-language
* <https://www.tutorialspoint.com/cprogramming>
* <https://www.youtube.com/watch?v=ytGATjX3nqc&ab_channel=EngineerMn>
* <https://www.youtube.com/watch?v=L00f3z7WBRU&ab_channel=Brightgoal>
* https://youtu.be/1lNL31q4T5I

**Books**

1. [Kernighan, Brian](https://en.wikipedia.org/wiki/Brian_Kernighan); [Ritchie, Dennis M.](https://en.wikipedia.org/wiki/Dennis_Ritchie) (March 1988). [*The C Programming Language*](https://archive.org/details/cprogramminglang00bria) (2nd ed.). [Englewood Cliffs, NJ](https://en.wikipedia.org/wiki/Englewood_Cliffs,_NJ): [Prentice-Hall](https://en.wikipedia.org/wiki/Prentice_Hall). [ISBN](https://en.wikipedia.org/wiki/ISBN_(identifier)) [0-13-1103628](https://en.wikipedia.org/wiki/Special:BookSources/0-13-110362-8)
2. King, K.N. (2008). *C Programming: A Modern Approach* (2 ed.). W. W. Norton. [ISBN](https://en.wikipedia.org/wiki/ISBN_(identifier)) [978-0393979503](https://en.wikipedia.org/wiki/Special:BookSources/978-0393979503).