Table of Contents

Abstract i
1. Introduction 1
    1.1. Background 1
    1.2. Objective 1
2. Project Description 2
    2.1. Purpose and Scope 2
    2.2. Problem Description 2
    2.3. Functionality and Features 2
3. Methodology 4
    3.1. Development Process 4
    3.2. Tools and Resources 4
    3.3. Development Environment 4
4. Results and Discussion 5
    4.1. Presentation of Project Outcomes and Achievements 5
    4.2. Comparison of Actual Results with Expected Outcomes 5
    4.3. Discussion of Challenges Faced and Lessons Learned 6
5. Conclusion 7
6. Future Work 8
    6.1. Suggestion for Future Enhancements 8
    6.2. Ideas for Additional Improvements 8
7. References 9
8. Appendices 10
    8.1 Source Code 10
    8.2 Output 22

Abstract

This report is about the development of the Student Management System, a software application that aids managing student records in different schools, colleges, and other educational institutions. The application is written in C language, and is filled with features like creating, viewing, updating, deleting, and searching records. It can also sort records according to the user's need which makes it easier to extract required information. The interface is easy to use and understand which enhances the user experience. User and enter and save information like student name, registration number, faculty, semester, grade, and percentage obtained by student. The file handling feature offered by C is used to store and fetch stored data that is presented by the program. Application can sort and list out students according to different conditions. The application is designed to manage student records electronically and improve efficiency of the users so that the educational institution can enhance their effectiveness. The Student Management System's architecture, features, implementation specifics, and potential areas for future development are all outlined in this report.

1.Introduction

By automating student management in educational institutions, the Student Management System project addresses associated problems. Historically, student administration relied on inefficient and mistake-prone technologies like paper records. The goals of the development of this project are to improve organizational performance, increase efficiency, and streamline student administration processes.

1.1. Background

For educational institutions to run smoothly and effectively, academic administration relies heavily on the management of student information. Historically, spreadsheets and paper-based records were the main manual methods used for this process. The limitations of manual management, including errors, inefficiencies, and organizational problems, have become increasingly noticeable as educational establishments became bigger and more sophisticated.

The goal of the Student Management System project is to use technology and automation to change student management procedures considering these problems. The project's objective is to develop a user-friendly software program that will simplify student management procedures, boost productivity, and enhance organizational performance inside educational institutions.

1.2. Objective

This report was prepared to meet the following objectives:

- Give an overview of the Student Management System.
- Explain why automated Student Management System is necessary.
- Learn the relevance and purpose of the system.
- Describe the functions of the system.
- Talk about opportunities for expansion and improvement in the future.

2. Project Description

2.1. Purpose and Scope

The goal of the Student Management System is to completely change the way that educational establishments handle and preserve student data. The development of an advanced software application that automates many student administration tasks, including generation, display, updating, searching, and deleting is included in the project's scope. The system aims to decrease administrative tasks, boost productivity, and enhance the accuracy of student management in educational establishments.

2.2. Problem Description

Student administration in educational institutions has typically been handled manually with spreadsheets and paper-based records. These techniques, however, are prone to errors, inefficiencies, and logistical problems. Timing-consuming processes including manually entering student data, processing stored data, and retrieving processed data when needed are familiar challenges for administrators and educators. Moreover, paper-based records are more likely to be misplaced, destroyed, or lost, making student management more difficult. To address these problems and improve overall organizational effectiveness in educational institutions, the Student Management System provides a centralized and automated solution that minimizes errors and streamlines administrative procedures.

2.3. Functionality and Features

The Student Record Management System offers a comprehensive set of features and functionalities tailored to the needs of educational institutions. These include:

- **Automated Record Management:** The system automates the process of managing student records, reducing the need for manual data entry, and ensuring data accuracy.
- **Record Display:** A user-friendly interface allows users to easily view and access student records for individual students or entire courses.
- **Search Functionality:** The system includes a powerful search feature that enables users to quickly find student records based on specific criteria such as registration number.

- **Secure Deletion:** The system can securely delete student records as needed, ensuring that unnecessary files are removed without compromising data security.
- **Record Update**: The system allows users to efficiently update existing student records, ensuring that any changes in student information, such as address or course enrollment, are accurately reflected in the database. This feature maintains data integrity and keeps records up to date with minimal effort.

3. Methodology

3.1. Development Process

- **Requirements Gathering:** First, I gathered all the information related to the problem and what is required to solve the problem.
- **Design:** In the next step, I visualized a general system architecture such as the user interface and other system components. Then I sketched the design on a normal paper.
- **Implementation:** Now, I translated the design into code. This involved writing the code using C programming language in IDE called Dev-C++.
- **Testing:** After writing the code, I tested the application to check if all the components work properly. Through testing, I could make sure that the system meets all the requirements and solves the problem in an efficient manner.
- **Deployment:** Finally, after making sure that there are no bugs or errors, the system is ready to be used in the organization.

3.2. Tools and Resources

- **IDE (Integrated Development Environment):** The project used **Dev-C++** IDE to write, edit, and debug code.
- **Compiler:** It used Mingw port of GCC (GNU Compiler Collection) as its compiler.
- **Libraries:** The application used libraries such as the Standard C Library for input and output <stdio.h>. For dealing with strings, it used <string.h> library. For terminating program, it used <stdlib.h>. For user interface development, it used <windows.h> library.

3.3. Development Environment

- **Platform:** The Grade sheet Management System is designed for desktop platforms, including Windows and macOS.
- **Operating System:** For development of the application, Windows Operating System was used.

4. Results and Discussion

4.1. Presentation of Project Outcomes and Achievements

Student management procedures in educational institutions have improved because of the development of the Student Management System, which has produced noteworthy results and successes. Important outcomes consisting of:

- **Automation of Student Management System**: Key student management tasks, like creation, display, updating, search, and deletion, are efficiently automated by the Student Management System. Administrative tasks have been made simpler by automation, which has reduced manual labor.
- **Increased Precision and Effectiveness:** The Student Management System has improved the accuracy and efficiency of student management by automating data entry and calculating processes. The technology reduces errors and conflicts while guaranteeing that student data is reliable and accurate**.**
- **User-friendly and Clean Interface:** The user-friendly architecture of the system makes it simple for administrators to collect student information. Because of its accessibility, academic planning and well-informed decision-making are encouraged, which enhances organizational effectiveness.

4.2. Comparison of Actual Results with Expected Outcomes

The actual results obtained from the SMS implementation closely match the predicted outcomes specified during the project design phase. Specifically:

- **Automation:** The SMS successfully automates student record management duties, as expected. The system efficiently stores student records based on student data, displays student records for individual students or entire faculties.
- **Efficiency and Accuracy:** The SMS has shown improved efficiency and accuracy in record processing, which meets expectations. By automating data entry and computation methods, the system reduces errors and discrepancies, resulting in dependable student data.
- **Accessibility:** The SMS improves access to student information, as planned. The user-friendly design allows users to easily retrieve student data, making information retrieval and decision-making more efficient.

4.3. Discussion of Challenges Faced and Lessons Learned

Throughout the project, a number of issues surfaced that made for excellent learning opportunities:

- **Technical Difficulties:** During the implementation stage, a few technical problems surfaced, including debugging intricate code, and improving system performance. These problems required a lot of problem solving, including consulting friends for assistance, and searching the internet for answers.
- **User feedback:** The system's ability to solve problems cannot be relied upon in the absence of appropriate input from other users. For this reason, I requested comments from a number of my acquaintances who ran their code on a computer.
- **Limited Time:** I was unable to add more features that would have improved and increased the efficiency of the system because of time constraints.

Throughout the project, a number of issues surfaced that made for excellent learning opportunities:

- **Technical Difficulties:** During the implementation stage, a few technical problems surfaced, including debugging intricate code, and improving system performance. These problems required a lot of problem solving, including consulting friends for assistance, and searching the internet for answers.
- **User feedback:** The system's ability to solve problems cannot be relied upon in the absence of appropriate input from other users. For this reason, I requested comments from a number of my acquaintances who ran their code on a computer.
- **Limited Time:** I was unable to add more features that would have improved and increased the efficiency of the system because of time constraints.

5. Conclusion

The Student Management System project, in summary, is an effective attempt to leverage technology to enhance academic administration procedures. Its implementation has created a strong basis for enhancing organizational effectiveness, advancing student administration practices, and eventually enhancing the performance of educational institutions. The system's successful completion represents a major advancement in the processes involved in academic administration. The system's potential to address common issues facing educational institutions is demonstrated by its ability to automate student management procedures, increase efficiency, and improve accuracy. The success of the Student Management System also highlights the innovative possibilities of technology in the sphere of education. Educational institutions can enhance the quality of teaching by reducing human burden, allocating resources more efficiently, and employing technology to optimize administrative procedures.

6. Future Work

6.1. Suggestion for Future Enhancements

- **Basic User Authentication:** Create a simple user authentication system in which users must input a username and password to access the record management system. This can be accomplished by using basic file handling techniques in C to store and validate user credentials.
- **Grade Average Calculation:** Add functionality to calculate each student's average grade depending on their performance in various subjects. Display the calculated average with specific topic grades to provide a complete overview of student achievement.
- **Basic Data Validation:** Implement basic data validation checks to ensure that user inputs, such as student names and subject marks, are within valid ranges. For example, ensure that marks are between 0 and 100, and student names do not contain special characters.

6.2. Ideas for Additional Improvements

In addition to future developments, there are various ideas for more features or changes that could further enhance the functioning and usability of the SMS:

- **Custom Editing Options:** Implement custom editing options for student records, such as the ability to modify only the name, address and so on or update the marks for individual subjects. Provide prompts to select the student and subject to edit, and then allow the user to enter the new marks.
- **Export to Text File:** Enable users to export grade sheet data to a text file for easy backup or sharing. Allow the user to specify the filename and location for the exported file, and then save the grade sheet data in a simple text format.
- **Subject Name Addition:** Allow users to add names of subjects to the system. Prompt the user to enter the name of the subject and any other relevant details, and then include it in the list of subjects for which grades can be recorded.

7. References

[2] “Project Description” retrieved from:

https://www.linkedin.com/pulse/what-7-disadvantages-manual-system-richard-breitmeyer/

[4] “Results and Discussion” retrieved from:

https://wpschoolpress.com/benefits-advantages-student-management-system/

8. Appendices

8.1 Source Code

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<windows.h>
#define MAX 100

struct subject {
	float marks;
};

struct marks {
	float total,percent;
	char gr[5];
};

struct student {
	int sem,subNum;
	char name[MAX],addr[MAX],regNum[MAX],faculty[MAX];
	struct subject *sub;
	struct marks m;
};

typedef struct student rec;

// Cursor Position
int *posx,*posy;

// Function prototypes
FILE* openFile(char [],char []);
rec add(FILE *);
char confirm(char [],int *,int *);
void create();
void dash();
void dashTab();
void del();
void gotoxy(int,int);
char* grade(struct subject [],int);
void printCheck(rec,int);
void prinHead();
void prinTab(rec [],int);
void prinTabFail(rec);
void prinTabPass(rec);
int passFail(rec [],char [],int);
void remo(char []);
void rena(char [],char []);
void save(char [],rec);
void search();
void sort(rec [],int,char);
void swap(rec *,rec *);
void update();
```

```
void view();
void viewFac(FILE *,char [],rec);

void main() {
	char ch,c;
	int x,y;
	posx=&x;
	posy=&y;
	do {
		system("cls");
		x=35;
		y=6;
		gotoxy(20,4);
		dash();
		gotoxy(x,++y);
		printf("|\tStudent Record Management System\t|");
		gotoxy(x,++y);
		dash();
		gotoxy(x,y+=3);
		printf("1. Add Student Record");
		gotoxy(x,++y);
		printf("2. View Student Record");
		gotoxy(x,++y);
		printf("3. Update Student Record");
		gotoxy(x,++y);
		printf("4. Delete Student Record");
		gotoxy(x,++y);
		printf("5. Search Student Record");
		gotoxy(x,++y);
		printf("6. Exit program");
		y+=2;
		c=confirm("Which operation do you want to perform? [1-6] : ",&x,&y);
		system("cls");
		switch(c) {
			case '1':
				create();
				break;
			case '2':
				view();
				break;
			case '3':
				update();
				break;
			case '4':
				del();
				break;
			case '5':
				search();
				break;
			case '6':
				gotoxy(x,4);
				dash();
				gotoxy(45,8);
				printf("Thank you for using the program.");
				gotoxy(x,10);
```

```
                                dash();
                                exit(0);
                                break;
                        default:
                                gotoxy(15,7);
                                printf("Invalid choice. Please choose from 1 - 5.");
                                *posy=10;
                }
                (*posy)++;
                ch=confirm("Do you want to continue? [Y/N] : ",posx,posy);
        }while(ch=='y' || ch=='Y');
}

void dash() {
        CONSOLE_SCREEN_BUFFER_INFO csbi;
   int width,i;
   if(!GetConsoleScreenBufferInfo(GetStdHandle(STD_OUTPUT_HANDLE),&csbi)) {
      printf("Error getting console screen buffer info.\n");
      return;
   }
   width=csbi.dwSize.X;
   putchar('\n');
   for (i=0;i<width;i++) {
      putchar('-');
   }
   printf("\n\n");
   fflush(stdout);
}

void create() {
        FILE *fp;
        int x=35,y=5;
        rec s;
        fp=openFile("record.txt","ab+");
        gotoxy(x,y);
        printf("Enter the following student details :");
        s=add(fp);
        save("record.txt",s);
        gotoxy(x,*posy);
        printf("File Successfully saved.");
        *(posy)+=2;
}

FILE* openFile(char fn[],char mode[]) {
        FILE *fp;
        fp=fopen(fn,mode);
        if(fp==NULL) {
                printf("Error opening file.");
                exit(1);
        }
        return(fp);
}

rec add(FILE *fp)
{
```

```
rec s,temp;
int i,x=35,y=7;
gotoxy(x,y);
printf("Registration No. : ");
fscanf(stdin,"%s",s.regNum);
while(fread(&temp,sizeof(rec),1,fp)) {
        while(!strcmp(temp.regNum,s.regNum)) {
                gotoxy(x=28,y+=2);
                printf("Student record with Registration No. %s already exists.",s.regNum);
                gotoxy(x=35,y+=2);
                printf("Enter registration no. again : ");
                fflush(stdin);
                fscanf(stdin,"%s",s.regNum);
                rewind(fp);
        }
}
fclose(fp);
gotoxy(x,++y);
printf("Name : ");
fflush(stdin);
fgets(s.name,sizeof(s.name),stdin);
strtok(s.name,"\n");
gotoxy(x,++y);
printf("Faculty : ");
fgets(s.faculty,sizeof(s.faculty),stdin);
strtok(s.faculty,"\n");
gotoxy(x,++y);
printf("Semester : ");
scanf("%d",&s.sem);
gotoxy(x,++y);
printf("Address : ");
fflush(stdin);
fgets(s.addr,sizeof(s.addr),stdin);
strtok(s.addr,"\n");
gotoxy(x,++y);
printf("Enter no. of subjects : ");
scanf("%d",&s.subNum);
s.m.total=0;
s.sub=(struct subject *)malloc(s.subNum*sizeof(struct subject));
if(s.sub==NULL) {
        gotoxy(x,++y);
        printf("Memory allocation failed.");
        exit(1);
}
gotoxy(x,++y);
printf("Enter marks in %d subjects:\n",s.subNum);
for(i=0;i<s.subNum;i++) {
        gotoxy(x,++y);
        printf("For subject %d : ",i+1);
        scanf("%f",&s.sub[i].marks);
        s.m.total+=s.sub[i].marks;
}
strcpy(s.m.gr,grade(s.sub,s.subNum));
if(strcmp(s.m.gr,"PASS")==0)
        s.m.percent=s.m.total/s.subNum;
```

```
	y+=3;
	*posy=y;
	*posx=x;
	return(s);
}

char* grade(struct subject s[],int n) {
	int i;
	for(i=0;i<n;i++) {
		if(s[i].marks<40)
			break;
	}
	if(i<n)
		return("FAIL");
	else
		return("PASS");
}

void remo(char fn[]) {
	if(remove(fn)!=0) {
		printf("Error deleting file.");
		exit(1);
	}
}

void rena(char old[],char newn[]) {
	if(rename(old,newn)!=0) {
		printf("Error renaming file.");
		exit(1);
	}
}

char confirm(char s[],int *x,int *y) {
	char c;
	gotoxy(*x,*y);
	printf("%s",s);
	scanf(" %c",&c);
	return(c);
}

void save(char fn[],rec s) {
	FILE *fp;
	fp=openFile(fn,"ab");
	if(fwrite(&s,sizeof(s),1,fp)!=1)
		printf("Error writing into file.");
	fclose(fp);
}

void printCheck(rec s,int y) {
	int x=15;
	dash();
	gotoxy(x,++y);
	printf("%-17s: %s","Name",s.name);
	gotoxy(x,++y);
	printf("%-17s: %s","Address",s.addr);
```

```
	gotoxy(x,++y);
	printf("%-17s: %s","Registration No.",s.regNum);
	gotoxy(x,++y);
	printf("%-17s: %s","Faculty",s.faculty);
	gotoxy(x,++y);
	printf("%-17s: %d","Semester",s.sem);
	gotoxy(x,++y);
	printf("%-17s: %s","Grade",s.m.gr);
	gotoxy(x,++y);
	if(strcmp(s.m.gr,"PASS")==0)
		printf("%-17s: %.2f %%\n","Percentage",s.m.percent);
	else
		printf("%-17s: N/A\n","Percentage");
}

void sort(rec sfac[],int n,char c) {
	int i,j;
	for(i=0;i<n;i++) {
		for(j=0;j<n-i-1;j++) {
			switch(c) {
				case '2':
				case '6':
				case '7':
					if(strcmp((sfac+j)->name,(sfac+j+1)->name)>0)
						swap(&sfac[j],&sfac[j+1]);
					break;
				case '3':
					if(strcmp((sfac+j)->name,(sfac+j+1)->name)<0)
						swap(&sfac[j],&sfac[j+1]);
					break;
				case '4':
					if((sfac+j)->m.percent>(sfac+j+1)->m.percent)
						swap(&sfac[j],&sfac[j+1]);
					break;
				case '5':
					if((sfac+j)->m.percent<(sfac+j+1)->m.percent)
						swap(&sfac[j],&sfac[j+1]);
					break;
			}
		}
	}
}

void viewFac(FILE *fp,char fac[],rec s) {
	int c=0,i,j,cpass,x=35,y=6;
	char ch;
	rec sfac[MAX],spass[MAX];
	strupr(fac);
	while(fread(&s,sizeof(rec),1,fp)) {
		strupr(s.faculty);
		if(strcmp(s.faculty,fac)==0)
		{
			sfac[c]=s;
			c++;
		}
```

```
	}
	gotoxy(x,y);
	printf("1. View all Student Record");
	gotoxy(x,++y);
	printf("2. Ascending by Name");
	gotoxy(x,++y);
	printf("3. Descending by Name");
	gotoxy(x,++y);
	printf("4. Ascending by Percentage");
	gotoxy(x,++y);
	printf("5. Descending by Percentage");
	gotoxy(x,++y);
	printf("6. Passed Students");
	gotoxy(x,++y);
	printf("7. Failed Students");
	y+=2;
	ch=confirm("Enter your choice : ",&x,&y);
	system("cls");
	switch(ch) {
		case '1':
			gotoxy(15,1);
			printf("Displaying all student records of %s faculty :\n",fac);
			y=-6;
			for(i=0;i<c;i++)
			{
				printCheck(sfac[i],y+=10);
			}
			dash();
			y+=2;
			*posy=y;
			break;
		case '2':
			gotoxy(15,1);
			printf("Displaying all student records of %s faculty in ascending order by name
:\n",fac);
			goto sortName;
		case '3':
			gotoxy(15,1);
			printf("Displaying all student records of %s faculty in descending order by
name :\n",fac);
			sortName:
			sort(sfac,c,ch);
			prinTab(sfac,c);
			break;
		case '4':
			gotoxy(15,1);
			printf("Displaying all student records of %s faculty in ascending order by
percentage :\n",fac);
			cpass=passFail(sfac,"PASS",c);
			goto Sort;
		case '5':
			gotoxy(15,1);
			printf("Displaying all student records of %s faculty in descending order by
percentage :\n",fac);
			cpass=passFail(sfac,"PASS",c);
```

```
                        goto Sort;
                        break;
                case '6':
                        gotoxy(15,1);
                        printf("Displaying all student records of %s faculty who passed :\n",fac);
                        cpass=passFail(sfac,"PASS",c);
                        goto Sort;
                case '7':
                        gotoxy(15,1);
                        printf("Displaying all student records of %s faculty who failed :\n",fac);
                        cpass=passFail(sfac,"FAIL",c);
                        Sort:
                        sort(sfac,cpass,ch);
                        prinTab(sfac,cpass);
                        break;
                default:
                        gotoxy(15,7);
                        printf("Invalid choice. Please choose from 1 - 5.");
                        *posy=2;
        }
}

void gotoxy(int x,int y) {
   HANDLE hConsole=GetStdHandle(STD_OUTPUT_HANDLE);
   if(hConsole==INVALID_HANDLE_VALUE) {
      printf("Error getting console handle.");
      exit(1);
         }
   COORD cursorPos;
   cursorPos.X=x;
   cursorPos.Y=y;
   if(!SetConsoleCursorPosition(hConsole,cursorPos)) {
      printf("Error setting console cursor position.");
      exit(1);
   }
}

void swap(rec *a,rec *b) {
        rec temp;
        temp=*a;
   *a=*b;
   *b=temp;
}

int passFail(rec sfac[],char grade[],int c) {
        int i,cpass=0;
        for(i=0;i<c;i++)
        {
                if(strcmp(sfac[i].m.gr,grade)==0)
                {
                        sfac[cpass]=sfac[i];
                        cpass++;
                }
        }
        return(cpass);
```

```
}

void dashTab() {
	int i;
	for(i=0;i<103;i++)
		printf("-");
	printf("\n");
}

void prinHead() {
	printf("| %-20s | %-15s | %-16s | %-7s | %-8s | %-5s | %-10s |\n","Student Name","Address","Registration No.","Faculty","Semester","Grade","Percentage");
}

void prinTabPass(rec s) {
	printf("| %-20s | %-15s | %-16s | %-7s | %8d | %-5s | %8.2f %% |\n",s.name,s.addr,s.regNum,s.faculty,s.sem,s.m.gr,s.m.percent);
}

void prinTabFail(rec s) {
	printf("| %-20s | %-15s | %-16s | %-7s | %8d | %-5s | %10s |\n",s.name,s.addr,s.regNum,s.faculty,s.sem,s.m.gr,"N/A");
}

void prinTab(rec s[],int n) {
	int i,y=6,x=8;
	gotoxy(x,3);
	dashTab();
	gotoxy(x,4);
	prinHead();
	gotoxy(x,5);
	dashTab();
	for(i=0;i<n;i++) {
		gotoxy(x,y);
		if(strcmp(s[i].m.gr,"PASS")==0)
			prinTabPass(s[i]);
		else
			prinTabFail(s[i]);
		y++;
	}
	gotoxy(x,y);
	dashTab();
	y-=7;
	*posy=y;
}

void view() {
	FILE *fp;
	rec s;
	int x=35,y=6;
	char c,fac[5];
	fp=openFile("record.txt","rb");
	gotoxy(x,++y);
	printf("1. View all Student Record");
	gotoxy(x,++y);
```

```
		printf("2. View Student Record according to Faculty");
		fflush(stdin);
		y+=2;
		c=confirm("Enter your choice [1-2]: ",&x,&y);
		system("cls");
		switch(c) {
			case '1':
				y=1;
				gotoxy(15,y);
				printf("Displaying all student records :\n");
				y=-6;
				while(fread(&s,sizeof(rec),1,fp)) {
					printCheck(s,y+=10);
				}
				y+=10;
				dash();
				*posy=y;
				break;
			case '2':
				gotoxy(35,4);
				printf("Enter Faculty : ");
		fflush(stdin);
		fgets(fac,sizeof(fac),stdin);
		strtok(fac,"\n");
		viewFac(fp,fac,s);
		*(posy)+=8;
				break;
			default:
				gotoxy(15,7);
				printf("Invalid choice. Please choose from 1 & 2.");
				*posy=10;
		}
		*posx=25;
		fclose(fp);
}

void update()
{
		FILE *fp1,*fp2;
		rec s1;
		char reg[100];
		int flag=0,x=25,y=4;
		fp1=openFile("record.txt","rb");
		fp2=openFile("temp.txt","wb");
		gotoxy(x,y);
		printf("Enter Registration No. of student record to be updated : ");
		fflush(stdin);
		fscanf(stdin,"%s",reg);
		while(fread(&s1,sizeof(rec),1,fp1)) {
			if(strcmp(strupr(reg),strupr(s1.regNum))==0) {
				flag=1;
				gotoxy(x,++y);
				printf("Enter new details for %s :\n\n",s1.name);
				s1=add(fp2);
			}
```

```
                save("temp.txt",s1);
        }
        if(flag==0) {
                gotoxy(x,y+=3);
                printf("Record not found.");
        }
        else {
                gotoxy(x,y+=15);
                printf("File successfully saved.");
        }
        fclose(fp2);
        fclose(fp1);
        remo("record.txt");
        rena("temp.txt","record.txt");
        y+=2;
        *posy=y;
        *posx=x;
}

void del()
{
        FILE *fp1,*fp2;
        rec s1;
        char reg[100];
        int flag=0,x=25,y=9;
        char ch;
        fp1=openFile("record.txt","rb");
        fp2=openFile("temp.txt","wb");
        gotoxy(x,6);
        printf("Enter Registration No. of student record to be deleted : ");
        fflush(stdin);
        fscanf(stdin,"%s",reg);
        while(fread(&s1,sizeof(rec),1,fp1)) {
                if(strcmp(strupr(reg),strupr(s1.regNum))==0) {
                        flag=1;
                        printCheck(s1,y);
                        dash();
                        y=20;
                        ch=confirm("Do you want to delete this record ? [Y/N] : ",&x,&y);
                        if(ch=='y' || ch=='Y') {
                                flag=2;
                                continue;
                        }
                }
                save("temp.txt",s1);
        }
        if(flag==0) {
                gotoxy(x,y);
                printf("Record not found.");
        }
        else if(flag==2) {
                gotoxy(x,y+=3);
                printf("Record successfully deleted.");
        }
        else if(flag==1) {
```

```
                gotoxy(x,y+=3);
                printf("Record not deleted.");
        }
        fclose(fp2);
        fclose(fp1);
        remo("record.txt");
        rena("temp.txt","record.txt");
        y+=2;
        *posy=y;
        *posx=x;
}

void search()
{
        FILE *fp;
        rec s;
        char reg[100];
        int flag=0,x=25,y=9;
        fp=openFile("record.txt","rb");
        gotoxy(x,6);
        printf("Enter Registration No. of student record to be searched : ");
        fflush(stdin);
        fscanf(stdin,"%s",reg);
        while(fread(&s,sizeof(rec),1,fp)) {
                if(strcmp(strupr(reg),strupr(s.regNum))==0) {
                        flag=1;
                        printCheck(s,y);
                        dash();
                        y=19;
                        x=40;
                        break;
                }
        }
        if(flag==0) {
                gotoxy(x,y);
                printf("Record not found.");
                y+=2;
        }
        *posy=y;
        *posx=255;
        fclose(fp);
}
```

8.2 Output

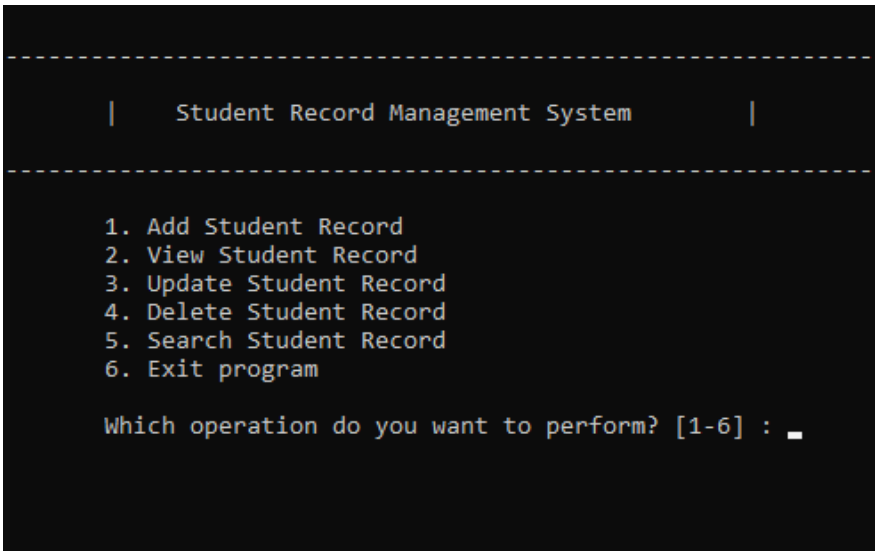

Fig 8.1: Home interface

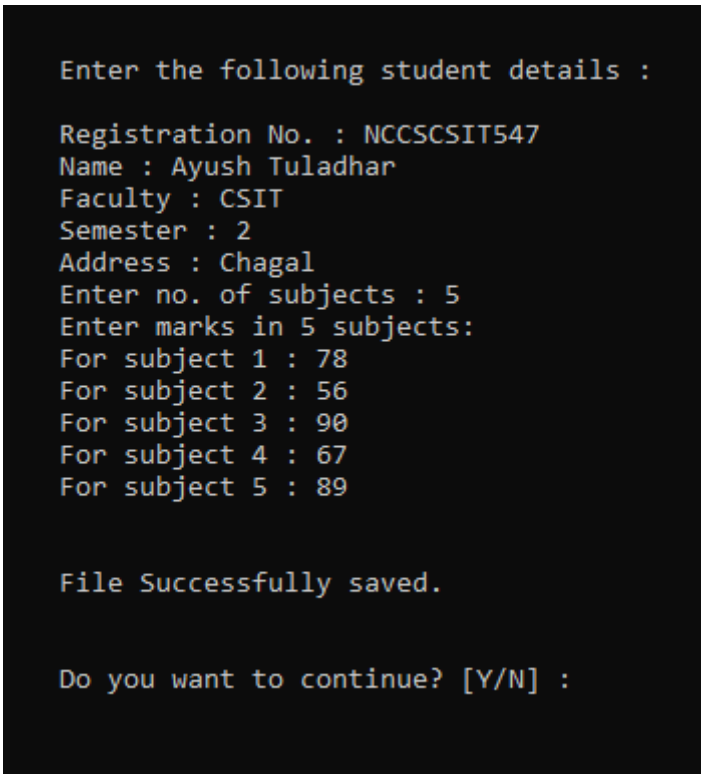

Fig 8.2: Option 1(Input details)

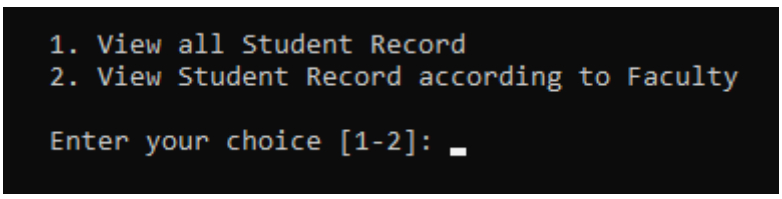

Fig 8.3: Option 2(Display option)

```
Displaying all student records :

--------------------------------------------------------------------------------------------------------------------------------

Name             : Sophia Jones
Address          : Address 2
Registration No. : REG2
Faculty          : BHM
Semester         : 2
Grade            : PASS
Percentage       : 78.75 %

--------------------------------------------------------------------------------------------------------------------------------

Name             : Olivia Brown
Address          : Address 4
Registration No. : REG4
Faculty          : CSIT
Semester         : 4
Grade            : PASS
Percentage       : 61.60 %

--------------------------------------------------------------------------------------------------------------------------------

Name             : Olivia Davis
Address          : Address 6
Registration No. : REG6
Faculty          : BBM
```

Fig 8.3.1: Option 2.1(Display all records)

```
Enter Faculty : CSIT

1. View all Student Record
2. Ascending by Name
3. Descending by Name
4. Ascending by Percentage
5. Descending by Percentage
6. Passed Students
7. Failed Students

Enter your choice :
```

Fig 8.3.2: Option 2.2(Display option based on faculty)

```
Displaying all student records of CSIT faculty :

--------------------------------------------------------------------------------------------------------------------------------

Name             : Olivia Brown
Address          : Address 4
Registration No. : REG4
Faculty          : CSIT
Semester         : 4
Grade            : PASS
Percentage       : 61.60 %

--------------------------------------------------------------------------------------------------------------------------------

Name             : William Brown
Address          : Address 9
Registration No. : REG9
Faculty          : CSIT
Semester         : 1
Grade            : PASS
Percentage       : 63.00 %

--------------------------------------------------------------------------------------------------------------------------------

Name             : Emma Williams
Address          : Address 20
Registration No. : REG20
Faculty          : CSIT
```

Fig 8.3.2.1: Option 2.2.1(Display all records of a faculty)

```
      Displaying all student records of CSIT faculty in ascending order by name :

-------------------------------------------------------------------------------------------------------
| Student Name         | Address         | Registration No. | Faculty | Semester | Grade | Percentage |
-------------------------------------------------------------------------------------------------------
| Ava Williams         | Address 25      | REG25            | CSIT    |        1 | PASS  |    52.40 % |
| Ayush Tuladhar       | Chagal          | NCCSCSIT547      | CSIT    |        2 | PASS  |    76.00 % |
| Emma Williams        | Address 20      | REG20            | CSIT    |        4 | PASS  |    65.00 % |
| James Brown          | Address 27      | REG27            | CSIT    |        3 | PASS  |    53.63 % |
| Olivia Brown         | Address 4       | REG4             | CSIT    |        4 | PASS  |    61.60 % |
| William Brown        | Address 9       | REG9             | CSIT    |        1 | PASS  |    63.00 % |
-------------------------------------------------------------------------------------------------------

               Do you want to continue? [Y/N] :
```

Fig 8.3.1: Option 2.2.2(Display records of a faculty in ascending order by name)

```
      Displaying all student records of CSIT faculty in descending order by name :

-------------------------------------------------------------------------------------------------------
| Student Name         | Address         | Registration No. | Faculty | Semester | Grade | Percentage |
-------------------------------------------------------------------------------------------------------
| William Brown        | Address 9       | REG9             | CSIT    |        1 | PASS  |    63.00 % |
| Olivia Brown         | Address 4       | REG4             | CSIT    |        4 | PASS  |    61.60 % |
| James Brown          | Address 27      | REG27            | CSIT    |        3 | PASS  |    53.63 % |
| Emma Williams        | Address 20      | REG20            | CSIT    |        4 | PASS  |    65.00 % |
| Ayush Tuladhar       | Chagal          | NCCSCSIT547      | CSIT    |        2 | PASS  |    76.00 % |
| Ava Williams         | Address 25      | REG25            | CSIT    |        1 | PASS  |    52.40 % |
-------------------------------------------------------------------------------------------------------

               Do you want to continue? [Y/N] : _
```

Fig 8.3.2: Option 2.2.3(Display records of a faculty in descending order by name)

```
      Displaying all student records of BIM faculty in ascending order by percentage :

-------------------------------------------------------------------------------------------------------
| Student Name         | Address         | Registration No. | Faculty | Semester | Grade | Percentage |
-------------------------------------------------------------------------------------------------------
| Olivia Miller        | Address 30      | REG30            | BIM     |        6 | PASS  |    61.87 % |
| Sophia Brown         | Address 22      | REG22            | BIM     |        6 | PASS  |    62.50 % |
| Emma Smith           | Address 29      | REG29            | BIM     |        5 | PASS  |    64.25 % |
| Sophia Davis         | Address 17      | REG17            | BIM     |        1 | PASS  |    65.50 % |
| Ava Johnson          | Address 24      | REG24            | BIM     |        8 | PASS  |    72.29 % |
| John Jones           | Address 28      | REG28            | BIM     |        4 | PASS  |    75.38 % |
-------------------------------------------------------------------------------------------------------

               Do you want to continue? [Y/N] :
```

Fig 8.3.3: Option 2.2.4(Display records of a faculty in ascending order by percentage)

```
      Displaying all student records of BIM faculty in descending order by percentage :

-------------------------------------------------------------------------------------------------------
| Student Name         | Address         | Registration No. | Faculty | Semester | Grade | Percentage |
-------------------------------------------------------------------------------------------------------
| John Jones           | Address 28      | REG28            | BIM     |        4 | PASS  |    75.38 % |
| Ava Johnson          | Address 24      | REG24            | BIM     |        8 | PASS  |    72.29 % |
| Sophia Davis         | Address 17      | REG17            | BIM     |        1 | PASS  |    65.50 % |
| Emma Smith           | Address 29      | REG29            | BIM     |        5 | PASS  |    64.25 % |
| Sophia Brown         | Address 22      | REG22            | BIM     |        6 | PASS  |    62.50 % |
| Olivia Miller        | Address 30      | REG30            | BIM     |        6 | PASS  |    61.87 % |
-------------------------------------------------------------------------------------------------------

               Do you want to continue? [Y/N] : _
```

Fig 8.3.4: Option 2.2.5(Display records of a faculty in descending order by percentage)

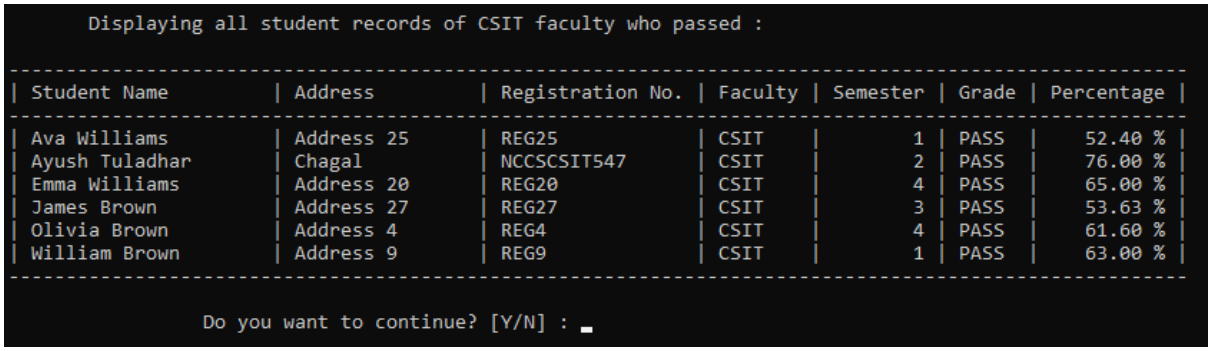

Fig 8.3.5: Option 2.2.6(Display records of a faculty who passed)

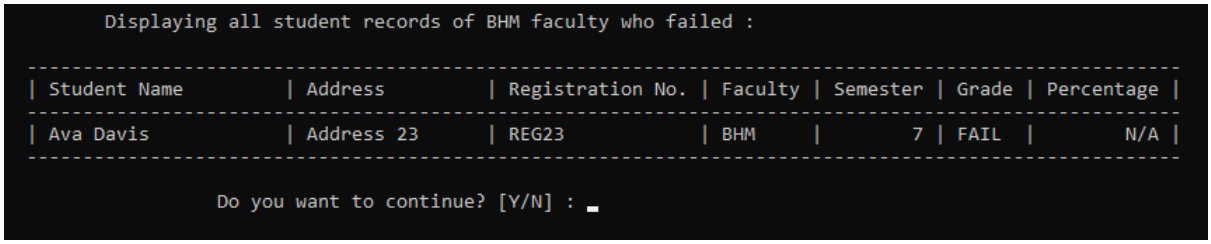

Fig 8.3.6: Option 2.2.7(Display records of a faculty who failed)

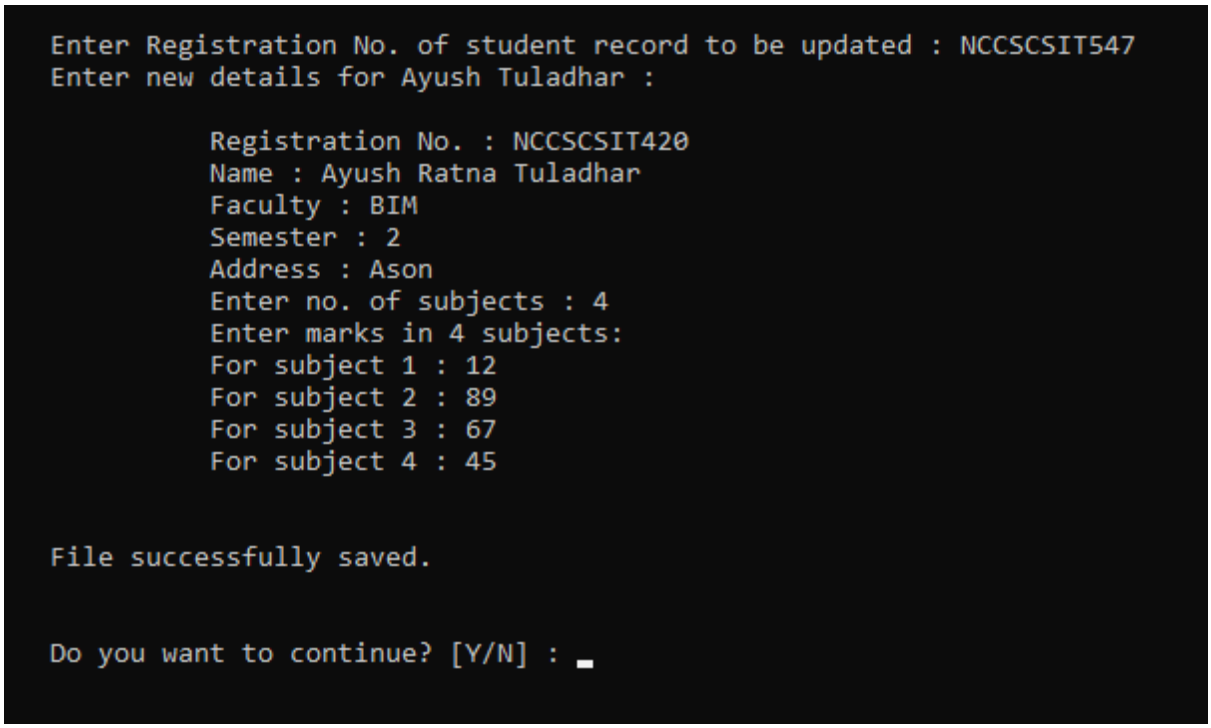

Fig 8.4: Option 3(Update record)

```
            Enter Registration No. of student record to be deleted : reg30
--------------------------------------------------------------------------------

   Name             : Olivia Miller
   Address          : Address 30
   Registration No. : REG30
   Faculty          : BIM
   Semester         : 6
   Grade            : PASS
   Percentage       : 61.87 %

--------------------------------------------------------------------------------

            Do you want to delete this record ? [Y/N] : y


            Record successfully deleted.


            Do you want to continue? [Y/N] : _
```

Fig 8.5: Option 4(Delete record)

```
           Enter Registration No. of student record to be searched : NCCSCSIT420
--------------------------------------------------------------------------------

  Name             : Ayush Ratna Tuladhar
  Address          : Ason
  Registration No. : NCCSCSIT420
  Faculty          : BIM
  Semester         : 2
  Grade            : FAIL
  Percentage       : N/A

--------------------------------------------------------------------------------

           Do you want to continue? [Y/N] : _
```

Fig 8.6: Option 5(Search record)

```
--------------------------------------------------------------------------------


                    Thank you for using the program.


--------------------------------------------------------------------------------
```

Fig 8.7: Option 6(Exit program)