

# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum – 590018, Karnataka.



**A Mini Project Report**

**On**

**“STUDENT MANAGEMENT SYSTEM”**

*Submitted in partial fulfillment of the requirement for the V semester course of*

**BACHELOR OF ENGINEERING**

**In**

**INFORMATION SCIENCE AND ENGINEERING**

*Submitted by:*

**HARSHITH S (1AP18IS008)**

*Under the guidance of:*

**Ms. Sushmitha Suresh**

**Asst. Professor**

**Dept. of IS & E**



**DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING**

**APS COLLEGE of ENGINEERING**

**Anantha Gnana Gangothri,**

**26<sup>th</sup> k.m, NH-209, Kanakapura Road, Bangalore-560082.**

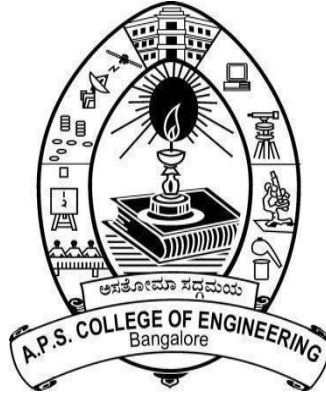
**2020-2021**

# APS COLLEGE of ENGINEERING

Anantha Gnana Gangothri,

26th k.m, NH-209, Kanakapura Road, Bangalore-560082.

**DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING**



## CERTIFICATE

Certified that the Project Work entitled ***“STUDENT MANAGEMENT SYSTEM”*** has been punctually carried out at ***APS College of Engineering, Bangalore*** by ***HARSHITH S (1AP18IS008)***, bonafide student/s of ***Sixth Semester, B.E.*** in partial fulfillment for the award of degree in ***Bachelor of Engineering in Information Science & Engineering*** affiliated to ***Visvesvaraya Technological University, Belgaum*** during academic year ***2020-2021***. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in departmental library. The project report has been approved as it satisfies the academic requirements in respect of project work for the said degree.

**Ms. Sushmitha Suresh**

Asst. Professor,  
Dept. of IS & E,  
APSCE.

Name of the external

1.

2.

**Prof. Nandeewar S B**

Assoc. Professor &  
Head Dept. of IS & E,  
APSCE.

Signature and Date

## **DECLARATION**

We hereby declare that the project work entitled “**Student Management System**” done at **APS COLLEGE OF ENGINEERING, Bangalore**, submitted to **Visvesvaraya Technological University**, in partial fulfillment of requirements for the degree of **Bachelor of Engineering in Information Science & Engineering** is a record of original work done by us and no part of it has been submitted for any degree or diploma of any institution previously.

**Place:**

**Date:**

**HARSHITH S (1AP18IS008)**

## ACKNOWLEDGEMENT

I take immense pleasure in thanking **Dr. Jagadeesh H S**, Principle, APSCE for having permitted us to carry out our Mini project “**Student Management System**”. I wish to express our deep sense of gratitude to **Prof. Nandeeswar S B**, Assoc. Professor, Head of Department of Information Science & Engineering, for his able guidance and useful suggestions, which helped us in completing this project.

I also express our deep sense of gratitude to our guides **Ms. Sushmitha Suresh**, Asst. Professor, Dept of ISE for their timely assistance which helped us to complete our project.

Words are inadequate in offering our thanks to all the teaching and non-teaching staff of Dept of ISE for their encouragement and co-operation in carrying out our project.

Finally, yet importantly, we would like to express our heartfelt thanks to our beloved parents for their blessing, our friends for their help and wishes for successful completion of this project.

### STUDENT NAME (USN)

AISHWARYA R	(1AP18IS002)
HARSHITH S	(1AP18IS008)
NAYAN V KUSTAGI	(1AP18IS017)

## **ABSTRACT**

File Structures is the Organization of Data in Secondary Storage Device in such a way that it minimizes the access time of the information and the storage space. A File Structure is a combination of representations for data in files and of operations for accessing the data. A File Structure allows applications to read, write and modify data. It might also support finding the data that matches some search criteria or reading through the data in some particular order. The type of data we can store within the file system is Char, Int, Double, Float, and Boolean.

The Student Management System is a File structure management system. It provides a user-friendly, interactive Menu Driven Interface (MDI) based on local file systems. All data is stored in files on disk. The application uses file handles to access the files. This project signifies the need for efficient management of student records in a file. The proposed system enables an administrator to keep track of all the students of the college, along with their details and maintain the students record efficiently.

In this system, we have a fixed number of fields, each with a maximum length, that combine to make a data record and variable length record. Fixing the number of fields in a record does not imply that the size of fields in the record is fixed. The records are used as containers to hold a mix of fixed and variable-length fields within a record.

## TABLE OF CONTENTS

	Declaration	III
	Acknowledgement	IV
	Abstract	V
	Table of Contents	VI
	List of Figures	VIII
<b>Chapter 1</b>	<b>INTRODUCTION</b>	
❖ 1.1	File Structures	9
❖ 1.1.1	Introduction to FS	9
❖ 1.1.2	History of FS	9
❖ 1.1.3	About the File	10
❖ 1.1.4	Application of FS	10
❖ 1.1.5	Advantages of FS	11
❖ 1.1.6	Disadvantages of FS	12
❖ 1.2	Project	13
❖ 1.2.1	Overview of Project	13
❖ 1.2.2	Features of Project	13
❖ 1.2.3	Objective and Scope of Project	13
❖ 1.3	C++	14
<b>Chapter 2</b>	<b>REQUIREMENTS SPECIFICATION</b>	
❖ 2.1	Overall Description	15
❖ 2.2	Specification Requirements	15
❖ 2.2.1	Software Requirements	15
❖ 2.2.2	Hardware Requirements	15

<b>Chapter 3</b>	<b>SYSTEM ANALYSIS</b>	
❖ 3.1	Analysis of Application	16
❖ 3.2	Structure used to Store Fields and Records	16
❖ 3.3	Operations Performed on a File	20
<b>Chapter 4</b>	<b>SYSTEM DESIGN</b>	
❖ 4.1	Design of Fields and Records	21
❖ 4.2	User Interface	21
<b>Chapter 5</b>	<b>IMPLEMENTATION</b>	
❖ 5.1	Introduction to Software used	24
❖ 5.1.1	Code::Blocks	24
❖ 4.2	Source Code	25
❖ 4.3	Problem Description	28
❖ 4.4	Result	28
<b>Chapter 6</b>	<b>TESTING</b>	
❖ 6.1	Unit Testing	29
❖ 6.2	Integration Testing	29
❖ 6.3	System Testing	29
<b>Chapter 7</b>	<b>SNAPSHOTS</b>	
❖ 7.1	Login Page	30
❖ 7.2	Menu Page	30
❖ 7.3	Insertion Page	31
❖ 7.4	Display Page	31
❖ 7.5	Modification Page	32
❖ 7.6	Searching Page	32
❖ 7.7	Deletion Page	33
❖ 7.8	Student Record File	33

Conclusion	34
Future Enhancement	34
Bibliography	35

## LIST OF FIGURES

❖ 4.2	User Interface	22
-------	----------------	----



## **CHAPTER 1**

### **INTRODUCTION**

#### **1.1 FILE STRUCTURES**

##### **1.1.1 INTRODUCTION TO FS**

File Structure is a combination of representations of data in files and operations for accessing the data. It allows applications to read, write and modify data. It supports finding the data that matches some search criteria or reading through the data in some particular order.

The reason for looking at files in this manner is to help correlate the function of the file with the permissions assigned to the directories which hold them. The way in which the operating system and its users interact with a given file determines the directory in which it is placed, whether that directory is mounted with read-only or read/write permissions, and the level of access each user has to that file. The top level of this organization is crucial. Access to the underlying directories can be restricted or security problems could manifest themselves if, from the top level down, it does not adhere to a rigid structure.

##### **1.1.2 HISTORY OF FS**

Early work with files presumed that files were on tape, since most files were. Access was sequential and the cost of access grew in direct proportion to the size of the file. As files grew intolerably large for unaided sequential access and as storage devices such as hard disks became available, indexes were added to files.

The indexes made it possible to keep a list of keys and pointers in a smaller file that could be searched more quickly. With key and pointer, the user had direct access to the large, primary file. But simple indexes had some of the same sequential flaws as the data file, and as the indexes grew, they too became difficult to manage, especially for dynamic files in which the set of keys changes.

In the early 1960's, the idea of applying tree structures emerged. But trees can grow very unevenly as records are added and deleted, resulting in long searches requiring many disk accesses to find a record. In 1963, researchers developed an elegant, self-adjusting binary tree structure, called AVL tree, for data in memory.

### 1.1.3 ABOUT THE FILE

There is one important distinction that in file structures and that is the difference between the logical and physical organization of the data. On the whole a file structure will specify the logical structure of the data that is the relationships that will exist between data items independently of the way in which these relationships may actually be realized within any computer. It is this logical aspect that we will concentrate on.

The physical organization is much more concerned with optimizing the use of the storage medium when a particular logical structure is stored on, or in it. Typically for every unit of physical store there will be a number of units of the logical structure (probably records) to be stored in it. A file system consists of two or three layers. Sometimes the layers are explicitly separated, and sometimes the functions are combined.

The logical file system is responsible for interaction with the user application. It provides the application program interface (API) for file operations- open, close, read etc., and passes the requested operation to the layer below it for processing. The logical file system manages open file table entries and per-process file descriptors. This layer provides file access, directory operations, and security and protection.

The second optional layer is the virtual file system. This interface allows support for multiple concurrent instances of physical file systems, each of which is called a file system implementation.

The third layer is the physical file system. This layer is concerned with the physical operation of the storage device (ex: disk). It processes physical blocks being read or written. It handles buffering and memory management and interacts with device drivers or with channel to drive the storage device.

### 1.1.4 APPLICATION OF FS

Relative to other parts of a computer, disks are slow. 1 can pack thousands of megabytes on a disk that fits into a notebook computer. The time it takes to get information from even relatively slow electronic random-access memory (RAM) is about 120 nanoseconds. Getting the same information from a typical disk takes 30 milliseconds. So, the disk access is a quarter of a million times longer than a memory access. Hence, disks are very slow compared to memory.

On the other hand, disks provide enormous capacity at much less cost than memory. They also keep the information stored on them when they are turned off. Tension between a disk's relatively slow access time and its enormous, nonvolatile capacity, is the driving force behind file structure design.

## 1.1.5 ADVANTAGES OF DBMS

### 1. Cost Effective

File management system is extremely cost effective since it is a digital filing system. Whatever the type of document that needs to be stored as a paper can be in the form of digital. Therefore, there is no cost involved in building rents, purchasing cabinets and physical papers.

### 2. Security

The traditional method of storing files cannot match the level of security provided by the file management system. In fact, security is one of the reasons why many organizations prefer to use file management system. The documents stored in the file management system is protected using authentication methods like username and password.

### 3. Data Sharing

Data sharing is one of the key features of a file management system. FMS allows very efficient way of sharing data with each and every person. The same data that is stored on files can be shared with multiple users simultaneously.

### 4. Data Retrieval

Using file management system means that it will be very easy to retrieve data. File management system follows a digital approach that provides access to required data within few minutes. Users don't need to search copies of documents manually here. Thus, there is very less amount of time spent for data retrieval.

### 5. Data Backup

In case of a failure, file management system provides a seamless way for backing up data. For this purpose, computers on default offer functionalities. However, if needed there can be also third-party application programs be used.

### 6. Environment Friendly

Due to the fact that file management system follows a digital system and there is no paper works involved, it can be said that this technique is more environment friendly.

By green practices, not only the cost of an organization is reduced, it can increase the overall image of a company. As a result, it can provide tax benefits and other advertising opportunities as well.

## 1.1.6 DISADVANTAGES OF DBMS

### 1. Redundancy

Redundancy is a kind of duplication that occurs if the same type of information exists in different locations. In this event, there is a possibility of memory wastage to take place resulting in higher storage costs.

### 2. Inconsistency

Due to the effect of data redundancy this often leads to data inconsistency. Which means that the same copies of data located in different places contain different values. For preventing this, there should be paper listing among different files.

### 3. Accessibility

Accessing data in file management system is not an easy process. It is not convenient as it should be. Whenever a user needs to access an information using different approaches, they must execute a special program.

### 4. Integrity

The data that is present on a file management system can get integrated. Meaning it is not correct and consistent. Most often this is caused in the presence of consistency constraints. Constraints are imposed by the programmers using programming codes. If the integrity continues, it can make the process of adding new constraints to be difficult.

### 5. Atomicity

Atomicity refers to the data that is incomplete. This often happens if the data is either completely entered or not entered at all. For an example, your system could fail in the middle of a transaction leading to data atomicity. Unlike in database management system, it is difficult to ensure atomicity in file management system.

### 6. Data duplication

Since data is stored in more than one location, there is a possibility of data duplication to take place. If file management system undergoes data duplication it will cause problems in the storage space. These duplications are difficult to correct due to the fact that they are independent to each other. Hence, it requires manual correction which can take time and effort.

## **1.2 PROJECT**

### **1.2.1 OVERVIEW OF PROJECT**

Student Management System is developed using C++. Talking about the project, it contains a login site from where a authorized user can login and then the user can maintain and manage all the student records easily. For security purpose, only the authorized user can access the system. In this project, user can perform all the main functions over the records.

### **1.2.2 FEATURES OF PROJECT**

- Login page.
- User management system.
- Manage student records.
- View reports.
- View archive.

### **1.2.3 OBJECTIVE AND SCOPE OF PROJECT**

#### **OBJECTIVE**

- To efficiently maintain all the records in one place.
- To provide record access to authorized users.
- To provide integrity and security for the stored records.
- To increase efficiency and accuracy of proposed system.
- To help teachers to maintain student records.
- To reduce paper and labor work

### SCOPE

This simple mini project for Student Record System creates an external file to store the user's data permanently to perform file handling operations. This is the features of Student Record System, which is you can add student information, search student information, update student information and can delete student information.

## 1.3 C++

C++ is a general-purpose object-oriented programming (OOP) language, developed by Bjarne Stroustrup, and is an extension of the C language. It is therefore possible to code C++ in a "C style" or "object-oriented style." In certain scenarios, it can be coded in either way and is thus an effective example of a hybrid language.

C++ is considered to be an intermediate-level language, as it encapsulates both high- and low-level language features. Initially, the language was called "C with classes" as it had all the properties of the C language with an additional concept of "classes." However, it was renamed C++ in 1983.

The main highlight of C++ is a collection of predefined classes, which are data types that can be instantiated multiple times. The language also facilitates declaration of user defined classes. Classes can further accommodate member functions to implement specific functionality. Multiple objects of a particular class can be defined to implement the functions within the class. Objects can be defined as instances created at run time. These classes can also be inherited by other new classes which take in the public and protected functionalities by default.

C++ includes several operators such as comparison, arithmetic, bit manipulation and logical operators. One of the most attractive features of C++ is that it enables the overloading of certain operators such as addition. A few of the essential concepts within the C++ programming language include polymorphism, virtual and friend functions, templates, namespaces and pointers.

## CHAPTER 2

### REQUIREMENTS SPECIFICATION

#### 2.1 OVERALL DESCRIPTION

A reliable and scalable database driven file application with security features that is easy to use and maintain is the requisite.

#### 2.2 SPECIFICATION REQUIREMENTS

The Specific Requirements for “**Student Management System**” is stated as follows:

##### 2.2.1 SOFTWARE REQUIREMENTS

- Operating system: Windows 10
- Code::Blocks

##### 2.2.2 HARDWARE REQUIREMENTS

- Processor: AMD Ryzen 5 3500U
- RAM: 8GB
- Hard disk: 1TB

## CHAPTER 3

### SYSTEM ANALYSIS

#### 3.1 ANALYSIS OF APPLICATION

The Student Record System in C++ is a console based application created using C++ programming language. This system is a simple mini project and compiled in Code::Blocks IDE using GCC compiler. In this project you can add student, search student, edit student and delete a student record.

This simple mini project for Student Record System creates an external file to store the user's data permanently to perform file handling operations. This is the features of Student Record System, which is you can add student information, search student information, update student information and can delete student information.

#### 3.2 STRUCTURE USED TO STORE FIELDS AND RECORDS

##### 1. Field Structure

A field is the smallest, logically meaningful, unit of information in a file. Field Structures: Four of the most common methods of adding structure to files to maintain the identity of fields are:

- Force the fields into a predictable length.
- Begin each field with a length indicator.
- Place a delimiter at the end of each field to separate it from the next field.
- Use a “keyword=value” expression to identify each field and its contents.



### **Method 1: Fix the Length of Fields**

Force the fields into a predictable length. This method relies on creating fields of predictable fixed size.

Disadvantages are:

- a. A lot of wasted space due to padding of fields with whitespace or empty space.
- b. Data values may not fit in the pre-determined length if the fields.

The fixed-length field approach is inappropriate for data that inherently contains a large amount of variability in the lengths of fields such as names or addresses.

### **Method 2: Begin Each Field with a Length Indicator**

We can count to the end of a field by storing the field length just ahead of the field. If the fields are not too long (less than 256 bytes), it is possible to store the length in a single byte at the start of each field. We refer to these fields as length-based.

### **Method 3: Separate the Fields with Delimiters**

This method requires that the fields be separated by a selected special character or a sequence of characters called a delimiter. Ex: If “|” is used as delimiter then a sample record would look like: Ames|Mary|123Maple|StillWater|OK|574075

This method of separating fields with a delimiter is often used. However, choosing a right delimiter is very important. In many cases white space characters are excellent delimiters because they provide a clean separation between the fields when they are listed on the console.

### **Method 4: Use a “Keyword=Value” Expression to Identify Fields**

This method requires that each field data be preceded with the field identifier/ keyword. It can also be used with the combination of delimiter to mark the field ends. Ex: last=Ames|first=Mary|address=123Maple|city=StillWater|state=ok|zip=570405

Some of the advantages are:

1. Each field provides information about itself.
2. Good format for dealing with missing fields.

The disadvantage here is that in some applications, a lot of space may be wasted on field keywords.

## **2. Record Structures**

The five most often used methods for organizing records of a file are:

- Require the records to be predictable number of bytes in length.
- Require the records to be predictable number of fields in length.
- Begin each record with a length indicator consisting of a count of the number of bytes that the record contains.
- Use a second file to keep track of the beginning byte address for each record.
- Place a delimiter at the end of each record to separate it from the next record.

### **Method 1: Make the Records a Predictable Number of Bytes (Fixed- Length-Record)**

A fixed-length record file is one in which each record contains the same number of bytes. In the field and record structure shown, we have a fixed number of fields, each with a predetermined length, that combine to make a fixed-length record. Fixing the number of bytes in a record does not imply that the size or number of fields in the record must be fixed. Fixed-length records are often used as containers to hold variable numbers of variable-length fields. It is also possible to mix fixed and variable length fields within a record.

### **Method 2: Make Records a Predictable Number of Fields**

This method specifies the number of fields in each record. Regardless of the method for storing fields, this approach allows for relatively easy means of calculating record boundaries.

### **Method 3: Begin Each Record with a Length Indicator**

We can communicate the length of records by beginning each record with a field containing an integer that indicates how many bytes there are in the rest of the record. This is commonly used to handle variable-length records.

### **Method 4: Use an Index to Keep Track of Addresses**

We can use an index to keep a byte offset for each record in the original file. The byte offset allows us to find the beginning of each successive record and compute the length of each record. We look up the position of a record in the index, then seek to the record in the data file.

### **Method 5: Place a Delimiter at the End of Each Record**

It is analogous to keeping the fields distinct. As with fields, the delimiter character must not get in the way of processing. A common choice of a record delimiter for files that contain readable text is the end-of-line character (carriage return/ new-line pair or, on Unix systems, just a new-line character: \n). Here, we use a # character as the record delimiter.

### **3.3 OPERATION PERFORMED ON A FILE**

#### **1. Insertion**

Insertion function adds new records into the file. In this process the inserted node descends to the leaf where the key fits. If the node has an empty space, insert the key/reference pair into the node. If the node is already full, split it into two nodes, distributing the keys evenly between the two nodes.

#### **2. Deletion**

Deletion is the process in which on the parent node to remove the split key that previously separated these merged nodes unless the parent is the root and we are removing the final key from the root, in which case the merged node becomes the new root.

#### **3. Update**

The reference to a deleted record is removed from index while the deleted record persists in the data file. A '\$' is placed in the first byte of the first field (ID) of the record, to help distinguish it from records that should be displayed. The requested record, if found, is marked for deletion, a "record deleted" message is displayed. If absent, a "record not found" message is displayed to the user. If present, after deletion, the system is used to insert the modified record containing, possibly, a new set of data like name, address, occupation it can be updated.

#### **4. Search**

Searching starts at the root, the tree is recursively traversed from top to bottom. At each level, the search reduces its field of view to the child pointer whose range includes the search value. A subtree's range is defined by the values, or keys, contained in its parent node. These limiting values are also known as separation values.

Binary search is typically used within nodes to find the separation values and child tree of interest. If the key matches, then the details of the record are displayed on the console else we continue to search. If the record is not found then we return -1 value and display error message on the console.

#### **5. Modify**

The system can also be used to modify existing records from all production inventory details. The user is prompted for a key, which is used as to search the records. Once the record is found we rewrite details of the corresponding product Hence modify() is delete() followed by the read(). If the record is not found then it returns -1 value and display error message on the console.

## CHAPTER 4

### SYSTEM DESIGN

#### 4.1 DESIGN OF THE FIELDS AND RECORDS

We use variable length record and fixed size fields. In variable length record, it will take the minimum space need for that field and at the end of the record there will be a delimiter placed indicating that it is the end of the record. The fields are separated using a delimiter

('|') whereas hash ('#') denotes end of a record.

A delimiter is a sequence of one or more characters used to specify the boundary between separate, independent regions in plain text or other data streams. In this application, there are 2 files which stores the data they are:

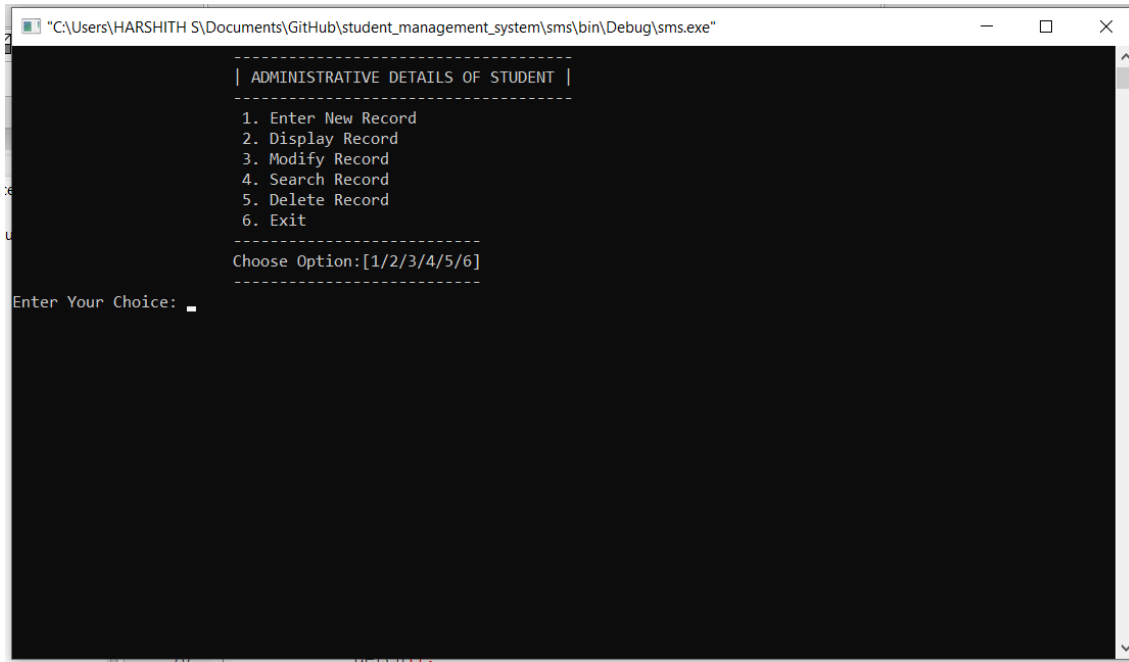
1. Student Record File
2. Login File

These files have respective attributes which identifies the uniqueness of each record.

1. **Student Record File:** It consists of the all the student details that are entered by the user. Attributes which identify the uniqueness of each record are USN and Name.
2. **Login File:** It consists of all the login details of the user. The attributes which identify the uniqueness of each record are Username, Password.

#### 4.2 USER INTERFACE

The user interface (UI), in the industrial design field of human–computer interaction, is the space where interactions between humans and machines occur. The goal of this interaction is to allow effective operation and control of the machine from the human end, whilst the machine simultaneously feeds back information that aids the operators' decision-making process.



**Fig. 4.2: User Interface**

### **1. Insertion of a Record**

If the operation desired is Insertion that is to read the data, the user is required to enter new record option as his/her choice, from the menu displayed, after which a new screen is displayed. For the student entry, the user is prompted to enter the USN followed by the Name, DOB, Gender, Mother name, Father name, Address, Email, Phone Number, Branch, Semester, Tenth Marks, PUC Marks, CGPA, Total Fee, Paid Fee and Due Fee.

The user is prompted for each input until the values for the field has been read. This means that, the user is prompted for USN for particular field is raised until the user enters exactly 10 characters each of which can be a digit or a character, else it displays invalid number message and user is prompted to read USN again.

### **2. Display of a Record**

If the operation desired is display of record, user is required to enter the USN. The records are displayed based on the USN entered by the user. If user enters valid USN, the student record will be displayed. If user enters an invalid USN, it will display error message and User will be taken back to the menu options.

### **3. Modification of a Record**

If the operation desired is Modify, the user is required to enter Modify record option as his/her choice, from the menu displayed, after which a new screen is displayed. The Modify operation is implemented as a deletion followed by an insertion. If there are no records in the file, a “RECORD NOT FOUND” message is displayed, and the user is prompted to press any key to return back to the menu screen. If there is at least a record in the file, the user is prompted for the USN, whose matching record is to be modify. USN entered is used as a key to search for a matching record. If none is found, a “Record Not Found” message is displayed and the user is then prompted to press any key to return back to the menu screen. If one is found, a “Record Deleted” message is displayed, after which a new screen is displayed.

### **4. Searching of a Record**

If the operation desired is Search, the user is required to enter Search record option as his/her choice, from the menu displayed, after which a new screen is displayed. Next, the user is prompted to enter the respective USN (key) that has to be searched. If the record is not found in the data file, a “RECORD NOT RECORD” message is displayed, and the user is prompted to press any key to return back to the menu screen. If the matching key is found, the corresponding record is to be retrieved. In each case, the user is then prompted to press any key to return back to the menu screen.

### **5. Deletion of a Record**

If the operation desired is Deletion that is to the delete the data, the user is required to enter Delete record option as his/her choice, from the menu displayed, after which a new screen is displayed. Next, the user is prompted to enter the USN, whose record from the file is to be deleted. If there are no records in the file, an error message is displayed, and the user is prompted to press any key to return back to the menu screen.

The USN entered is used as a key to search for a matching record. If USN doesn't match, “Record Not Found” message is displayed. If successful, “Record Found” message is displayed then user is prompted with alert to delete the record if successfully user enter to delete, “Record Deleted” message is displayed. In each case, the user is then prompted to press any key to return back to the menu screen.

## CHAPTER 5

### IMPLEMENTATION

#### 5.1 INTRODUCTION TO SOFTWARE USED

The software used for project is:

- Code::Blocks

##### 5.1.1 CODE::BLOCKS

Code::Blocks is a free C/C++ and Fortran IDE built to meet the most demanding needs of its users. It is designed to be very extensible and fully configurable.

Built around a plugin framework, Code::Blocks can be extended with plugins. Any kind of functionality can be added by installing/coding a plugin. For instance, event compiling and debugging functionality is provided by plugins!

There's an on-going effort to write a user manual for Code::Blocks. This is a community-driven project and contributions/criticism/suggestions are welcomed.

The initial documentation had started as an internal project of High-tech EDV-System GmbH who is now making it yet another contribution to the community. Further contributions esp. for the French version were provided by gd on.

The documentation is provided in English and French languages, in PDF, CHM and HTML formats. More formats and languages may become available.

The documentation is written in latex and can be retrieved from SVN using the address: <https://svn.code.sf.net/p/codeblocks/code/docs>.





```

cout << "\t\t\t\t\t LOGIN TO ACCESS\t\t\t\t\t|" << endl;
cout << "\t\t\t\t\t....." << endl;
cout << "\t\t\t\t\t ENTER USERNAME: ";
cin >> uname;
cout << "\t\t\t\t\t ENTER PASSWORD: ";

while (i<10)
{
    pword[i]=getch();
    c=pword[i];
    if(c==13) break;
    else printf("*");
    i++;
}
pword[i]='\0';
i=0;

if (strcmp(uname,"admin")==0 && strcmp(pword,"pass123")==0)
{
    cout << "\n\t\t\t\t\t LOGIN IS SUCCESSFUL";
    getch();
    menu();
    system("cls");
}
else
{
    cout << "\n\t\t\t\t\t LOGIN IS UNSUCCESSFUL";
    a++;
    getch();
    system("cls");
}
} while (a<=2);

if (a>2)
{
    cout << "\n\t\t\t\t\t Sorry you have entered the wrong credentials";
    getch();
}
system("cls");
}

void student::menu()
{
    menustart:
    int choice;
    char x;

```

```

system("cls");
cout << "\t\t\t....." << endl;
cout << "\t\t\t| ADMINISTRATIVE DETAILS OF STUDENT |" << endl;
cout << "\t\t\t....." << endl;
cout << "\t\t\t 1. Enter New Record" << endl;
cout << "\t\t\t 2. Display Record" << endl;
cout << "\t\t\t 3. Modify Record" << endl;
cout << "\t\t\t 4. Search Record" << endl;
cout << "\t\t\t 5. Delete Record" << endl;
cout << "\t\t\t 6. Exit" << endl;
cout << "\t\t\t -----" << endl;
cout << "\t\t\t Choose Option:[1/2/3/4/5/6]" << endl;
cout << "\t\t\t -----" << endl;
cout << "Enter Your Choice: ";
cin >> choice;
switch (choice)
{
    case 1:
        do
        {
            insert();
            cout << "\n\t\t\t Add Another Student Record (Y,N): ";
            cin >> x;
        } while (x == 'y' || x == 'Y');
        break;
    case 2:
        display();
        break;
    case 3:
        modify();
        break;
    case 4:
        search();
        break;
    case 5:
        deleted();
        break;
    case 6:
        exit(0);
    default:
        cout << "\n\t\t\t Invalid Choice, please try again";
}
getch();
goto menustart;
}

```

### **5.3 PROBLEM DESCRIPTION**

Student Management System, this system contains an admin side from where a user can manage all the student records easily. The admin plays an important role in the management of this system. In this project, the user has to perform all the main functions from the admin side.

### **5.4 RESULT**

The resulting system is able to:

- i.To add student details.
- ii.To store data in the file.
- iii.To display the student record.

## **CHAPTER 6**

# **TESTING**

### **6.1 UNIT TESTING**

Unit testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.

Unit testing is commonly automated, but may still be performed manually. The objective in unit testing is to isolate a unit and validate its correctness. A manual approach to unit testing may employ a step-by-step instructional document. The unit testing is the process of testing the part of the program to verify whether the program is working correct or not. In this part the main intention is to check the each and every input which we are inserting to our file. Here the validation concepts are used to check whether the program is taking the inputs in the correct format or not.

### **6.2 INTEGRATION TESTING**

Integration testing is also taken as integration and testing this is the major testing process where the units are combined and tested. Its main objective is to verify whether the major parts of the program is working fine or not. This testing can be done by choosing the options in the program and by giving suitable inputs it is tested.

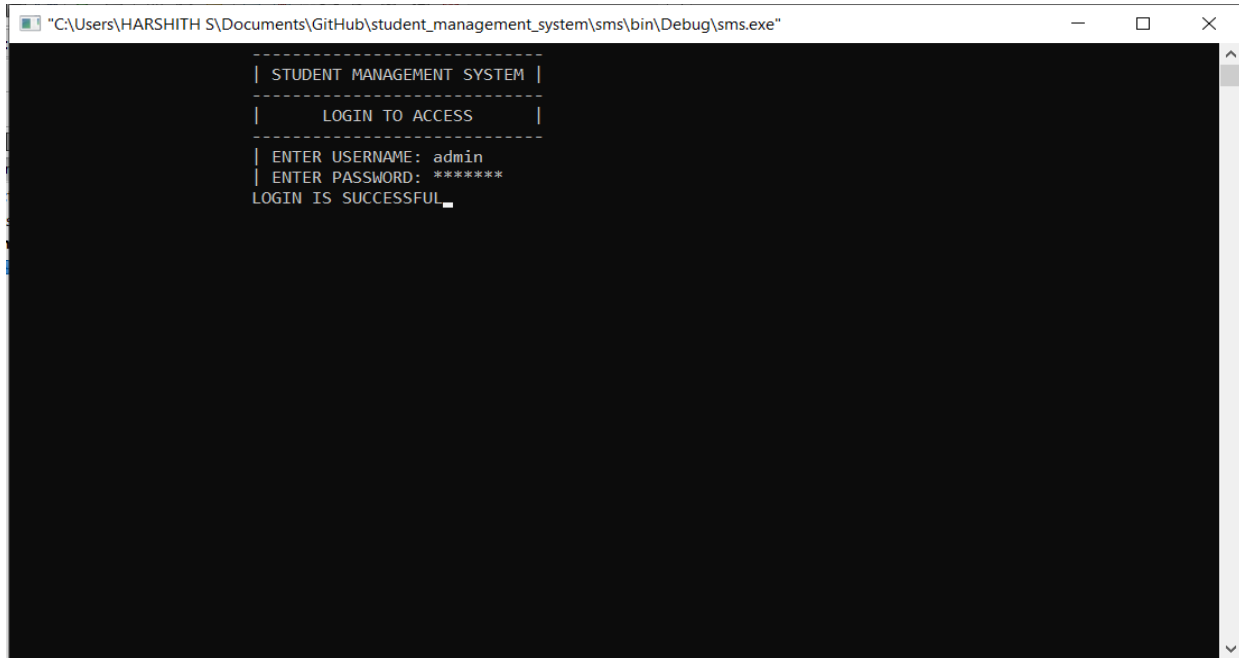
### **6.3 SYSTEM TESTING**

System testing is defined as testing of a complete and fully integrated software product. This testing falls in black-box testing wherein knowledge of the inner design of the code is not a pre-requisite and is done by the testing team. System testing is done after integration testing is complete. System testing should test functional and non-functional requirements of the software.

## CHAPTER 7

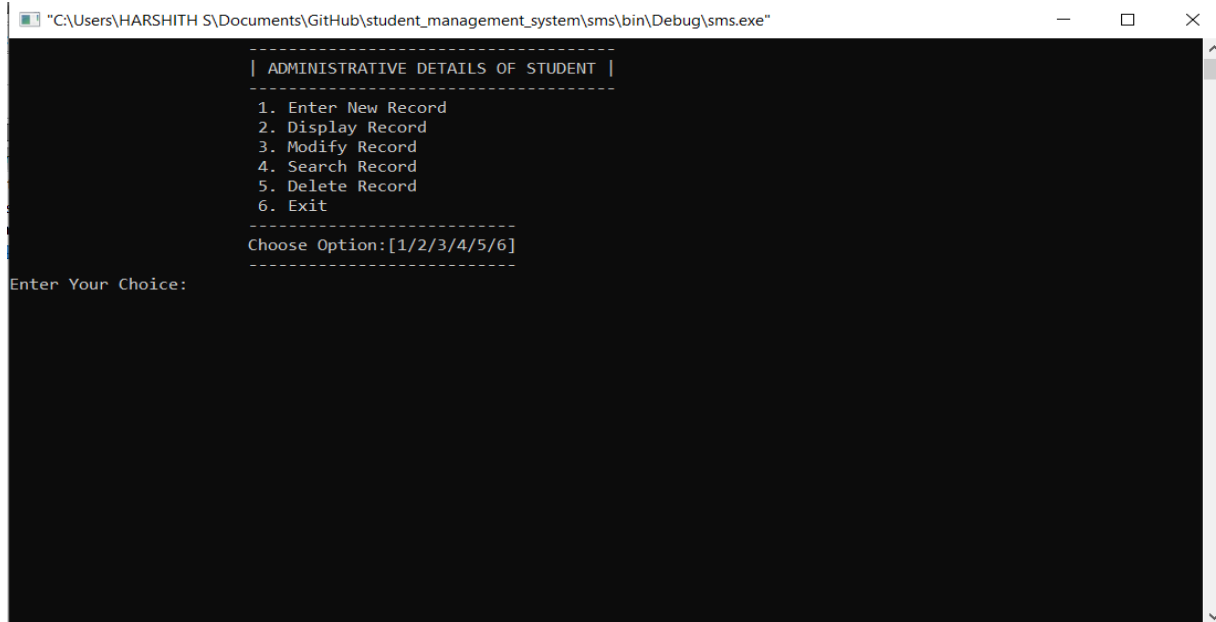
## SNAPSHOTS

### 7.1 LOGIN PAGE



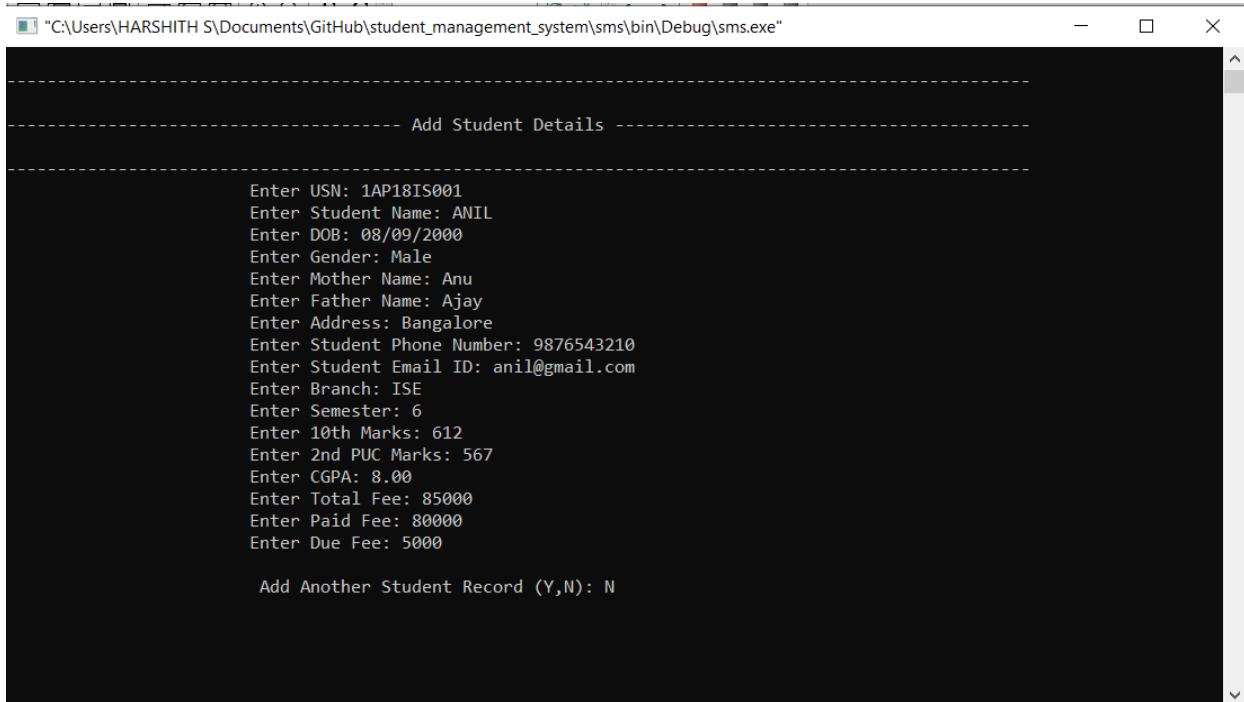
**Fig. 7.1: Login Page**

### 7.2 MENU PAGE



**Fig. 7.2: Menu Page**

## 7.3 INSERTION PAGE



```
"C:\Users\HARSHITH S\Documents\GitHub\student_management_system\sms\bin\Debug\sms.exe"

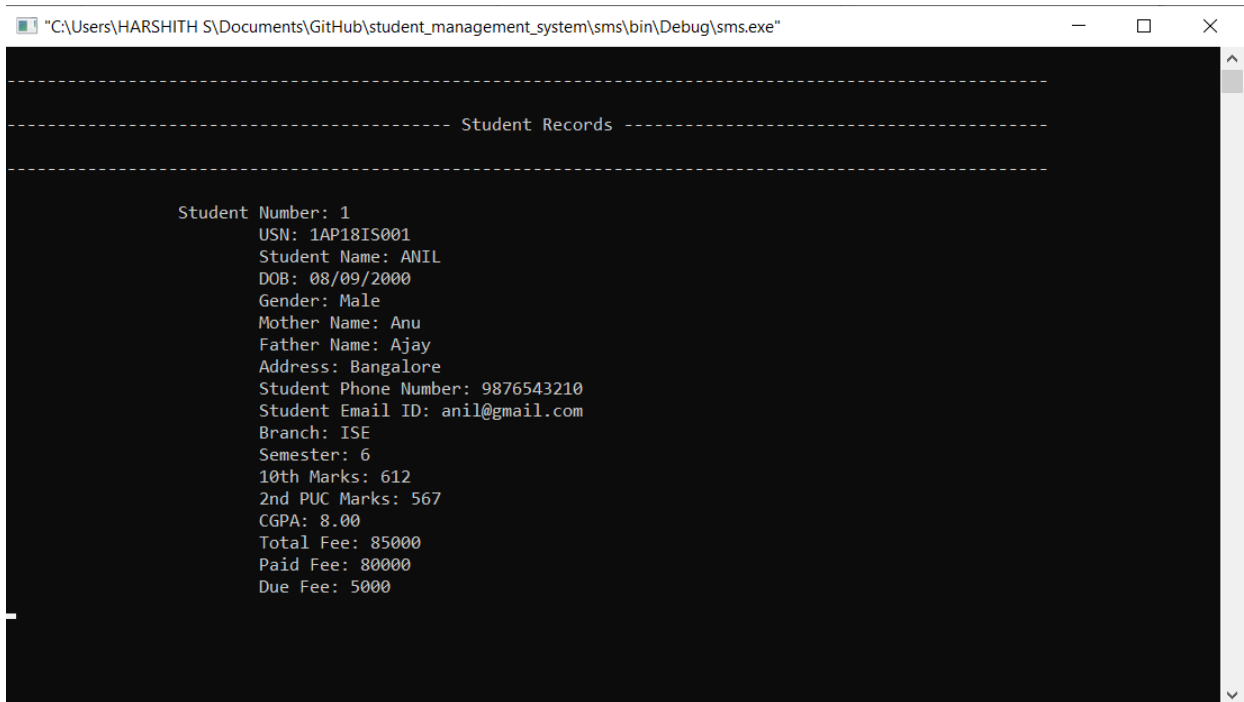
-----
----- Add Student Details -----
-----

Enter USN: 1AP18IS001
Enter Student Name: ANIL
Enter DOB: 08/09/2000
Enter Gender: Male
Enter Mother Name: Anu
Enter Father Name: Ajay
Enter Address: Bangalore
Enter Student Phone Number: 9876543210
Enter Student Email ID: anil@gmail.com
Enter Branch: ISE
Enter Semester: 6
Enter 10th Marks: 612
Enter 2nd PUC Marks: 567
Enter CGPA: 8.00
Enter Total Fee: 85000
Enter Paid Fee: 80000
Enter Due Fee: 5000

Add Another Student Record (Y,N): N
```

Fig. 7.3: Insertion Page

## 7.4 DISPLAY PAGE



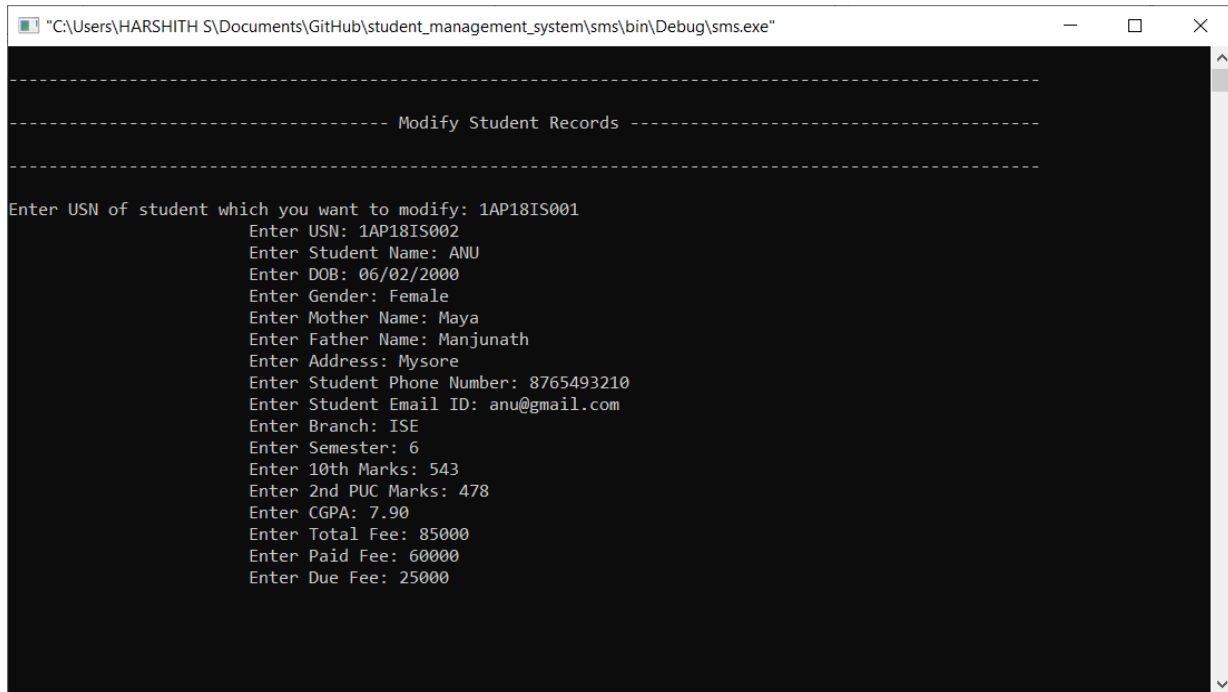
```
"C:\Users\HARSHITH S\Documents\GitHub\student_management_system\sms\bin\Debug\sms.exe"

-----
----- Student Records -----
-----

Student Number: 1
USN: 1AP18IS001
Student Name: ANIL
DOB: 08/09/2000
Gender: Male
Mother Name: Anu
Father Name: Ajay
Address: Bangalore
Student Phone Number: 9876543210
Student Email ID: anil@gmail.com
Branch: ISE
Semester: 6
10th Marks: 612
2nd PUC Marks: 567
CGPA: 8.00
Total Fee: 85000
Paid Fee: 80000
Due Fee: 5000
```

Fig. 7.4: Display Page

## 7.5 MODIFICATION PAGE



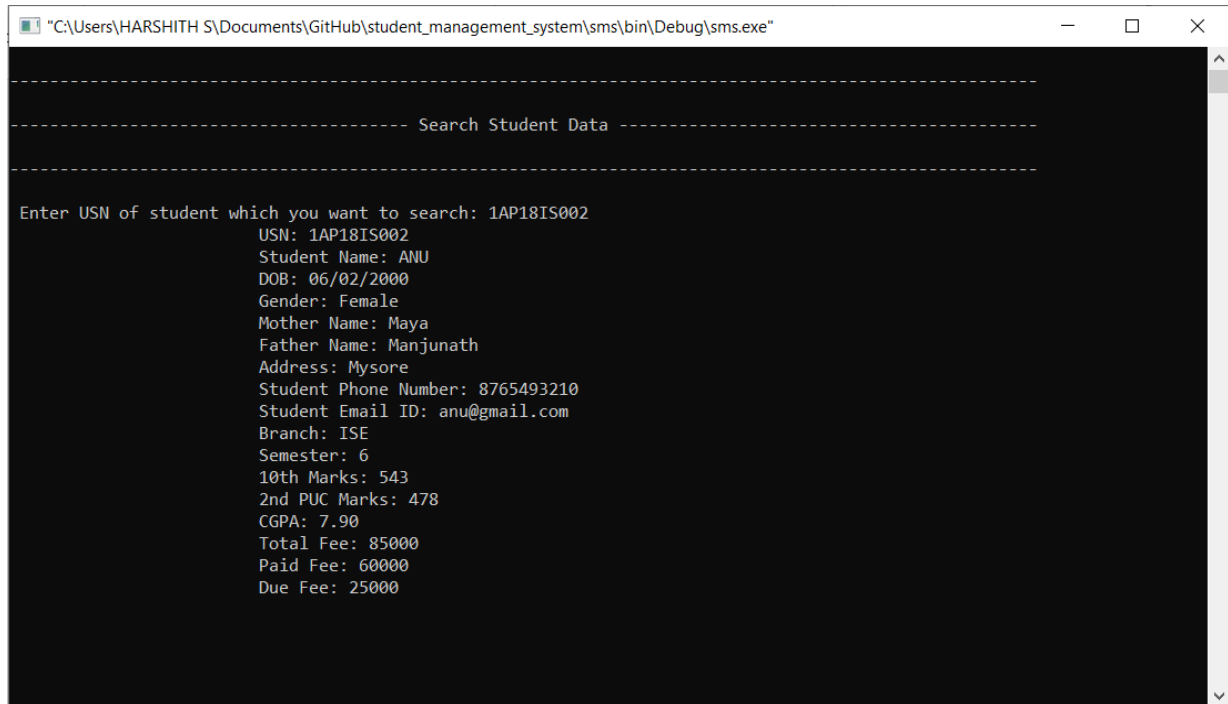
```
"C:\Users\HARSHITH S\Documents\GitHub\student_management_system\sms\bin\Debug\sms.exe"

----- Modify Student Records -----

Enter USN of student which you want to modify: 1AP18IS001
Enter USN: 1AP18IS002
Enter Student Name: ANU
Enter DOB: 06/02/2000
Enter Gender: Female
Enter Mother Name: Maya
Enter Father Name: Manjunath
Enter Address: Mysore
Enter Student Phone Number: 8765493210
Enter Student Email ID: anu@gmail.com
Enter Branch: ISE
Enter Semester: 6
Enter 10th Marks: 543
Enter 2nd PUC Marks: 478
Enter CGPA: 7.90
Enter Total Fee: 85000
Enter Paid Fee: 60000
Enter Due Fee: 25000
```

Fig. 7.5: Modification Page

## 7.6 SEARCHING PAGE



```
"C:\Users\HARSHITH S\Documents\GitHub\student_management_system\sms\bin\Debug\sms.exe"

----- Search Student Data -----

Enter USN of student which you want to search: 1AP18IS002
USN: 1AP18IS002
Student Name: ANU
DOB: 06/02/2000
Gender: Female
Mother Name: Maya
Father Name: Manjunath
Address: Mysore
Student Phone Number: 8765493210
Student Email ID: anu@gmail.com
Branch: ISE
Semester: 6
10th Marks: 543
2nd PUC Marks: 478
CGPA: 7.90
Total Fee: 85000
Paid Fee: 60000
Due Fee: 25000
```

Fig. 7.6: Searching Page



## 7.7 DELETION PAGE

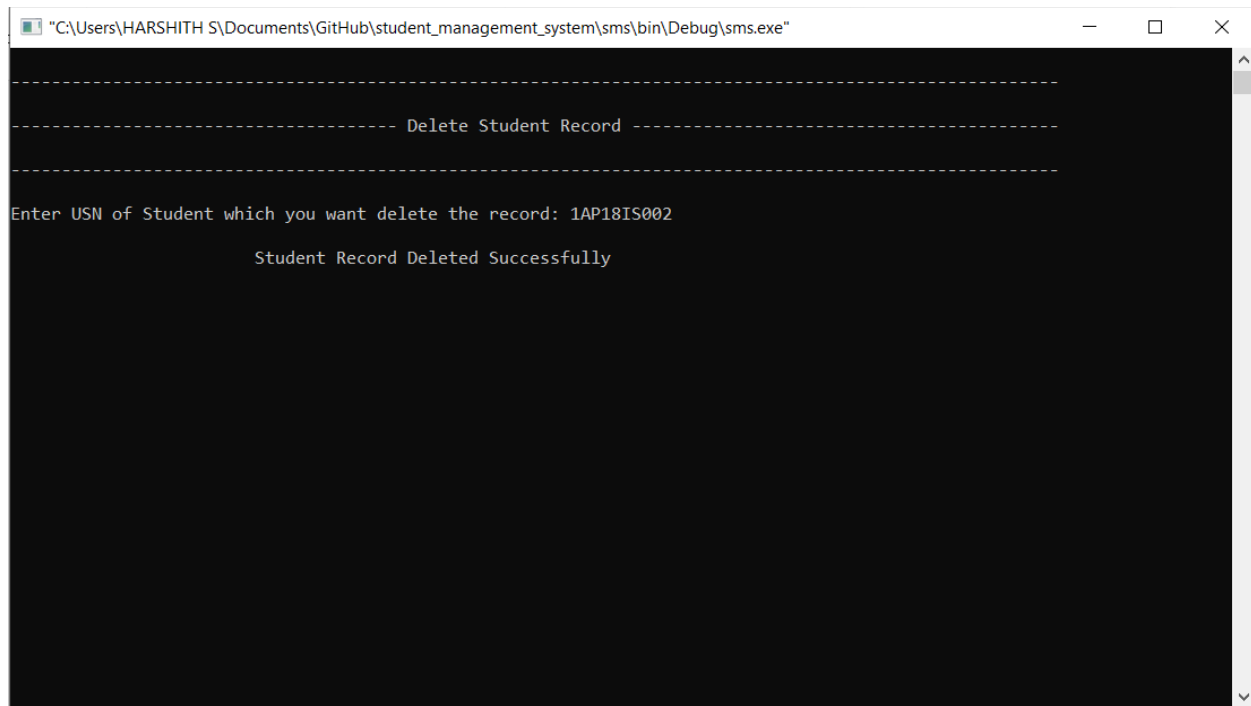


Fig. 7.7: Deletion Page

## 7.8 STUDENT RECORD FILE

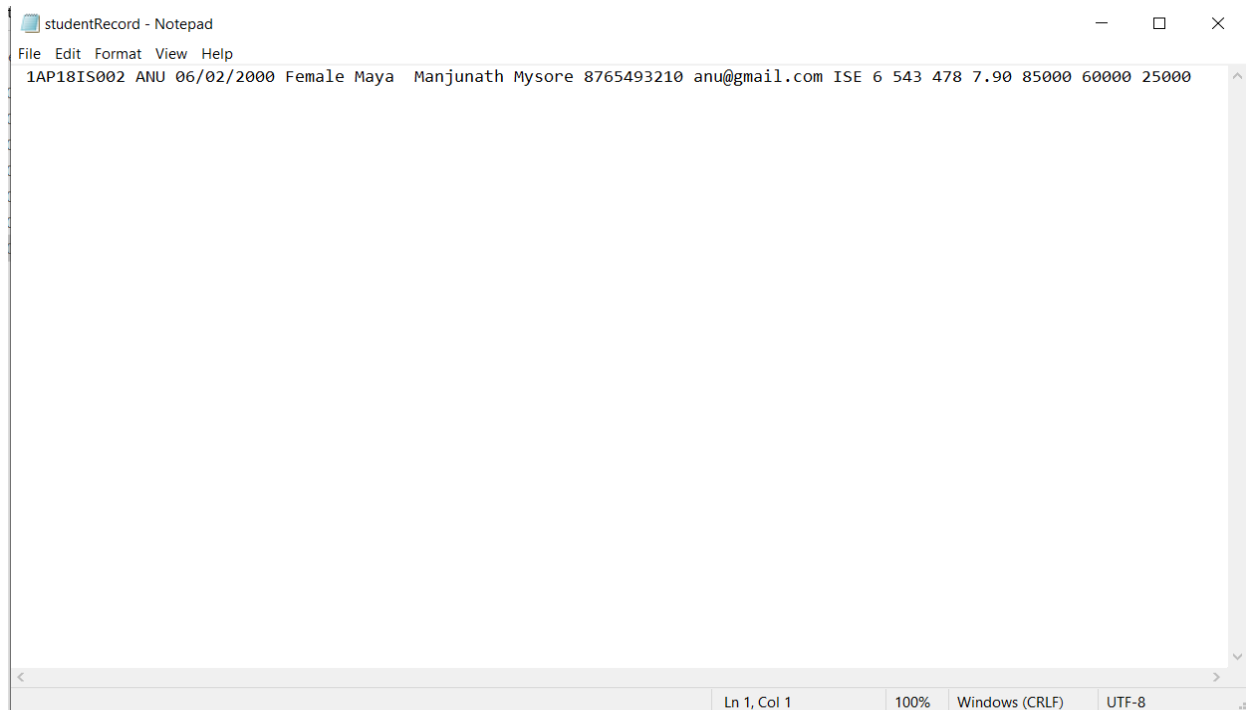


Fig. 7.8: Student Record File

## **CONCLUSION**

Generally, this system can be considered a useful system since it helps the lecturer to improve their process of preparing the student records. By providing support through the Student Management System, the usage can be increased to any faculties. If the system is successfully upload to host, to assist administrator, lecturer and student on how to use the system.

The Student Management System allows the user to store the student details. This software allows storing the details of all the data related to students. The implementation of the system will reduce data entry time and provide readily calculated reports. Since production inventory involves huge quantities of materials and products, it requires a structure which can store large data.

## **FUTURE ENHANCEMENT**

- The project can further be enhanced by using hashing technique to further reduce the number of accesses and improve the performance.
- Hashing technique allows record access in only one or very few runs with the help of a hashing function.
- Use more improved GUI for the same implementation.
- Using such advanced techniques for better indexing, we can implement complex functions.

## BIBLIOGRAPHY

1. Michael J Folk, Bill Zoellick, Greg Riccardi: File Structures – An Object-Oriented Approach with C++, 3 Edition, Pearson Education, 1998
2. K.R. Venugopal, K.G. Srinivas, P.M. Krishnaraj: File Structures Using C++, Tata McGraw-Hill, 2008.
3. Scot Robert Ladd: C++ Components and Algorithms, BPB Publications, 1993
4. Raghu Ramakrishnan and Johannes Gehrke: Database Management Systems, 3rd Edition, McGraw Hill, 2003
5. [www.geeksforgeeks.com](http://www.geeksforgeeks.com)
6. [www.codebind.com](http://www.codebind.com)
7. [www.includehelp.com](http://www.includehelp.com)

