VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi-590018

A File Structures Mini-Project Report
On

OFF MANAGEMENT SYSTEM

SUBMITTED IN PARTIAL FULFILLMENT FOR THE AWARD OF DEGREE

BACHELOR OF ENGINEERING

IN

INFORMATION SCIENCE AND ENGINEERING

SUBMITTED BY

ADITHI SATISH (1JB19IS003)

Under the Guidance of
Mrs. Sridevi G.M.
Assistant Professor
Dept. of ISE, SJBIT

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

SJB INSTITUTE OF TECHNOLOGY

BGS HEALTH AND EDUCATION CITY,
KENGERI, BENGALURU-560060, KARNATAKA, INDIA.

2021 – 2022

|| Jai Sri Gurudev ||

Sri Adichunchanagiri Shikshana Trust ®

SJB INSTITUTE OF TECHNOLOGY

BGS Health & Education City, Kengeri, Bengaluru – 560 060

Department of Information Science & Engineering

CERTIFICATE

Certified that the Mini-project work entitled **"Off Management System",** is a bonafide work carried out by **ADITHI SATISH(1JB19IS003),** students of **SJB Institute of Technology**, in partial fulfilment for 6th semester **File Structures Laboratory with mini project** in **INFORMATION SCIENCE AND ENGINEERING** of the **VISVESVARAYA TECHNOLOGICAL UNIVERSITY**, **BELAGAVI** during the academic year **2021-22.** It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of mini project prescribed for the said degree.

Mrs. SRIDEVI G M	**Dr. REKHA B**
Asst. Professor	**Professor & Head**
Dept. of ISE, SJBIT	**Dept. of ISE, SJBIT**

EXTERNAL VIVA

NAME OF THE EXAMINERS	**SIGNATURE WITH DATE**
1. ____________________	____________________
2. ____________________	____________________

ACKNOWLEDGEMENT

I would like to express my profound gratitude to His Divine Soul **Padmabhushan Sri Sri Sri Dr. Balagangadharanatha MahaSwamiji** and His Holiness **Jagadguru Sri Sri Sri Dr. Nirmalanandanatha MahaSwamiji** for providing me an opportunity to complete my academics in this esteemed institution.

I would also like to express my profound thanks to **Revered Sri Sri Dr. Prakashnath Swamiji**, **Managing Director**, SJB Institute of Technology, for his continuous support in providing amenities to carry out this project in this admired institution.

I express my gratitude to **Dr. K V Mahendra Prashanth**, **Principal**, SJB Institute of Technology, for providing me an excellent facilities and academic ambience; which has helped me in satisfactory completion of project work.

I extend my sincere thanks to **Dr. Rekha B**, **Professor & Head**, Department of Information Science and Engineering; for providing me an invaluable support throughout the period of my project work.

I wish to express my heartfelt gratitude to the **mini-project coordinator, Mrs. Sridevi G M, Asst. Prof,** Department of Information Science and Engineering for her valuable guidance, suggestions and cheerful encouragement during the entire period of my project work.

Finally, I take this opportunity to extend my earnest gratitude and respect to my parents, Teaching & Non-teaching staff of the department, and all my friends, who have directly or indirectly supported me during the period of my project work.

ADITHI SATISH (1JB19IS003)

ABSTRACT

Off Management System combines number of processes and systems to automate and easily manage employee data, leave request, track and grant leave. In many institution staffs are entitled to different types of leave, these leave are granted according to institution policy. Administrative department is mostly responsible for managing and granting leave request. To this end, most institution used conventional method of requesting, granting and managing leave. In conventional method, leave is manually requested by writing letter to head of department. The head of department minutes and forward the request to higher staff for approval. This method is time consuming, prone to error, require more paper work and difficult to manage. Hence the need for an automated leave management system that is faster, error free, less paper work and easy to manage. The system was achieved by developing an automated employee leave management system using file structures. The System is implemented using C++ which include and runs on Windows operating system. The overall functionality of the system shows that it work satisfactory and the result obtained shows that the system is error free, faster and allows staff to request for leave in a timely manner. Hence the system can be used by both academic staff and administrative department of an institution for effective and efficient management of employee.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 File Structure

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 mightalso support finding the data that matches some search criteria or reading through the data in some particular order.

An improvement in file structure design may make an application hundreds of times faster. The details of the representation of the data and the implementation of the operations determine the efficiency of the file structure for particular applications.

1.2 History

Early work with files presumed that files were on tape, since most files were. Access wassequential, and the cost of access grew in direct proportion, to the size of the file. As files grewintolerably 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. The problem was that, even with a balanced binary tree, dozens of accesseswere required to find a record in even moderate-sized files. A method was needed to keep a tree balanced when each node of thee tree was not a single record, as in a binary tree, but a fileblock containing dozens, perhaps even hundreds, of records. Hashing is a good way to get whatwe want with a single request, with files that do not change size greatly over time. Hashed indexes were used to provide fast access to files. But until recently, hashing did not work wellwith volatile, dynamic files. Extendible dynamic hashing can retrieve information with 1 or atmost 2 disk accesses, no matter how big the file became.

1.3 About the File

When we talk about a file on disk or tape, we refer to a particular collection of bytes stored there. A file, when the word is used in this sense, physically exists. A disk drive may contain hundreds, even thousands of these physical files. From the standpoint of an application program, a file is somewhat like a telephone line connection to a telephone network. The program can receive bytes through this phone line or send bytes down it, but it knows nothing about where these bytes come from or where they go. The program knows only about its end of the line. Even though there may be thousands of physical files on a disk,a single program is usually limited to the use of only about 20 files.

The application program relies on the OS to take care of the details of the telephone switching system. It could be that bytes coming down the line into the program originate from a physical file they come from the keyboard or some other input device. Similarly, bytesthe program sends down the line might end up in a file, or they could appear on the terminal screen or some other output device. Although the program doesn’t know where the bytes are coming from or where they are going, it does know which line it is using. This line is usually referred to as the logical file, to distinguish it from the physical files on the disk or tape.

1.4 Application of File Structure

Relative to other parts of a computer, disks are slow. 1 can pack thousands of megabyteson a disk that fits into a notebook computer.

The time it takes to get information from even relatively slow electronic random-accessmemory (RAM) is about 120 nanoseconds. Getting the same information from a typical disktakes 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, non-volatile capacity, is the driving force behind file structure design. Good file structure design will give us access to all the capacity without making our applications spend a lot of time waiting for thedisk.

1.5 Off Management System

This is the project about Off Management System. In this project we can easily track employee off and its details. The scope of project ‘Off management system’ is to Develop C++ based software to store the details of the leaves and maintain all information of leave related items. The primary aim of Off Management system is to improve accuracy and efficiency of tracking and keeping details of leaves entered. This project has many facilities for users. We have a console window with 7 options that is Apply leave, Delete leave, Search leave, Display leave , User Management ,Modify leave and quit program.

Firstly add users using user management option, then use apply leave, this option helps in applying different kinds of leave using Employee id. In Search leave we can search the details of the leave using employee id. Display all leaves, in this option we will get to display of all the leaves. Delete Leave, in this option we are allowed to delete any leave using its Employee id. In Modify Leave, we can modify the details of the leaves by giving the respective Employee Id. Finally the quit program this option terminates the program.

Chapter 2

SYSTEM REQUIREMENT SPECIFICATION

A computerized way of handling information about property and users details is efficient, organized and time saving, compared to a manual way of doing so. This is done through a menu driven console-based application whose requirements are mentioned in this section.

The specific requirements of Off Management System are stated as follows:

2.1 Software Requirements

Software's used

- o Operating System – Windows OS
- o Back End – Visual Studio

Technologies used

- o Front End – C++ programming
- o Controller – C++
- o Back End – C++ compiler

2.2 Hardware Requirements

Hardware Components used:

- o CPU – Intel Core i3 and Above
- o RAM – 4GB and Above
- o Peripherals – Standard PS/2 or USB Keyboard, Standard PS/2 or USB Wheel/Optical Mouse

2.3 Technology Used

C++ is a multi-paradigm programming language that supports object-oriented programming (OOP), created in 1983 at Bell Labs, C++ is an extension(superset) of C programming and the programs are written in C language can run in C++ compilers. An interpreter is a computer program that directly executes, i.e. performs, instructions written in a programming or scripting language, without requiring them previouslyto have been compiled into a machine language program. A compiler is a special program thatprocesses statements written in a particular programming language and turns them into machine language or "code" that a computer's processor uses. Typically, a programmer writeslanguage statements in a language such as Pascal or C one line at a time using an editor.

2.4 Features

C++ is object-oriented programming language and it is a very simple and easy language; it is the enhanced form of C programming language. this language has following features and here we discuss some important features of C++.

• Simple: Every C++ program can be written in simple English language so that it is very easy to understand and developed by programmer.

• Platform dependent: A language is said to be platform dependent whenever the program is executing in the same operating system where that was developed and compiled but not run and execute on another operating system. C++ is platform dependent language.

• Portability: It is the concept of carrying the instruction from one system to another system. In C++ Language. Cpp file contain source code, we can edit also this code. .exe file contain application, only we can execute this file. When we write and compile any C++ program on window operating system that program easily run on other window-based system.

• Powerful: C++ is a very powerful programming language, it has a wide verity of data types, functions, control statements, decision making statements, etc.

• Object oriented Programming language: This main advantage of C++ is; it is object-oriented programming language. It follows concept of oops like polymorphism, inheritance, encapsulation, abstraction.

• Case sensitive: C++ is a case sensitive programming language. In C++ programming 'break and BREAK' both are different. If any language treats lower case latter separately and upper-case latter separately than they can be called as case sensitive programming language [Example C, C++, java, .net are sensitive programming languages.] otherwise it is called as case insensitive programming language [Example HTML, SQL is case insensitive programming languages].

• Compiler based: C++ is a compiler-based programming language that means without compilation no C++ program can be executed. First, we need compiler to compile our program and then execute.

• Syntax based language: C++ is a strongly tight syntax-based programming language. If anylanguage, follow rules and regulation very strictly known as strongly tight syntax-based language. Example C, C++, Java, .net etc. If any language not follow rules and regulation verystrictly known as loosely tight syntax-based language. Example HTML.

• Efficient use of pointers: Pointers is a variable which hold the address of another variable, pointer directly direct access to memory address of any variable due to this performance of application is improve. In C++ language also concept of pointers are available.

Turbo C++ is a discontinued C++ compiler and integrated development environment and computer language originally from Borland. Most recently it was distributed by Embarcadero Technologies, which acquired all of Borland's compiler tools with the purchase of its Code Gear division in 2008. The original Turbo C++ product line was put on hold after 1994 and was revived in 2006 as an introductory-level IDE, essentially a striped down version of their flagship C++Builder. Turbo C++ 2006 was released on September 5, 2006 and was available in 'Explorer' and 'Professional' editions. The Explorer edition was free to download and distribute while the Professional edition was a commercial product. In October 2009 Embarcadero Technologies discontinued support of its 2006 C++ editions.

Chapter 3

SYSTEM DESIGN

The purpose of the design phase is to develop a clear understanding of what the developer wants people to gain from his/her project. As the developer works on the project, thetest for every design decision should be

"Does this feature fulfil the ultimate purpose of the project?"

A purpose statement affects the design process by explaining what the developer wantsthe project to do, rather than describing the project itself. The Design Document will verify that the current design meets all of the explicit requirements contained in the system model aswell as the implicit requirements desired by the customer.

3.1 Operations Performed on a File

Insertion:

The system is initially used to add containing Employee ID, name and salary into the file. Later the employee record can be used to apply leave according to the need.

Display:

The system can then be used to display existing employee record. The records are displayed based on the way we insert. It contains employee ID, name and salary. It is also used to display the leave applied by the employee.

Search:

The system can then be used to search the leave applied by the Employee. The user is prompted for a employee ID. If Employee ID is matched then entire details of the existing record will be displayed. If Employee ID does not exist it displays the message saying record not found.

Delete:

The system can then be used to delete existing records. The user can delete specific leave or allleaves applied. The user is prompted for a employee ID, which is needed to be deleted. The requested record, if found is cleared, a “record deleted” message is displayed, and the record of the respective employee ID will also be deleted in the file (The key of the record will be replaced by *). If absent record not found message will be displayed.

3.2 Data Flow Diagram

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-levelDFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can oftenvisually “say” things that would be hard to explain in words, and they work for both technicaland nontechnical audiences, from developer to CEO.

Data flow diagrams were popularized in the late 1970s, arising from the book Structured Design, by computing pioneers Ed Yourdon and Larry Constantine. They based it on the “data flow graph” computation models by David Martin and Gerald Estrin. The structured design concept took off in the software engineering field, and the DFD method tookoff with it. It became more popular in business circles, as it was applied to business analysis, than in academic circles.

Also contributing were two related concepts:

• Object Oriented Analysis and Design (OOAD), put forth by Yourdon and Peter Coad to

analyze and design an application or system.

• Structured Systems Analysis and Design Method (SSADM), a waterfall method to analyze and design information systems. This rigorous documentation approach contrasts with modernagile approaches such as Scrum and Dynamic Systems Development Method (DSDM.)

Three other experts contributing to this rise in DFD methodology were Tom DeMarco, Chris Gane and Trish Sarson. They teamed up in different combinations to be the main definers of the symbols and notations used for a data flow diagram. A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece. DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond. The necessary level of detail depends on the scope of what you are trying to accomplish.

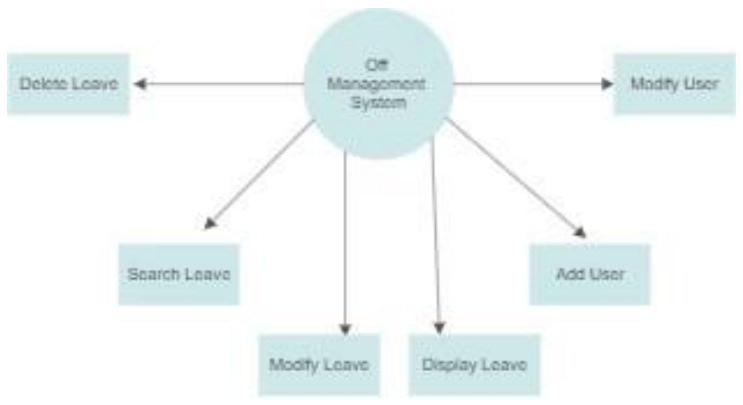


Figure 3.1 Zero level DFD

The above figure shows zero level data flow diagram of sales management system. Zero leveldata flow diagrams concentrate mainly on overview of the whole system or process being analyzed or modelled. It’s designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.

Chapter 4

IMPLEMENTATION

Implementation is the stage in the project where the theoretical design is turned into a working system and is giving confidence on the new system for the users that it will work efficiently and effectively. It involves careful planning, investigation of the current system andits constraints on implementation, design of methods to achieve the changeover, an evaluationof change over methods. Implementation is the most important phase. The most critical stage in achieving a successful new system is giving the users confidence that the new system will work and be effective. Any system developed should be secured and protected against possiblehazards.

Component testing is a method where testing of each component in an application is done separately. Component testing is also known as module, unit or program testing. It findsthe defects in the module and verifies the functioning of software. Our project is console-based,so all the implementation of our project is in console.

4.1 Console Window

The console window consists of 6 options which will direct to particular

tasks.The options in main screen

• Apply Leave

• Delete Leave

• Search Leave

• Modify Leave

• Display all Leaves

• User Management

• Quit program

4.2 Code Snippets

4.2.1 Main Screen

```
while (true) {
        start:
                int ch;
                cout << s3 << "----- E M P L O Y E E  L E A V E   M A N A G E M E N T  S
Y S T E M-----" << nl << nl;
                cout << s6 << "--- M A I N  M E N U--- " << nl << nl;
                cout << s4 << "1. APPLY LEAVE" << nl;
                cout << s4 << "2. DELETE LEAVE" << nl;
                cout << s4 << "3. SEARCH LEAVE" << nl;
                cout << s4 << "4. MODIFY LEAVE" << nl;
                cout << s4 << "5. DISPLAY ALL LEAVES" << nl;
                cout << s4 << "6. USER MANAGEMENT" << nl;
                cout << s4 << "7. EXIT" << nl;
```

4.2.2 Apply Leave

```
 void modLeave()
 {
        string id, str;
        int found = 0, count = 0;
        fstream file1, file2;
        file1.open("leave.txt", ios::binary | ios::in);
        cout << s4 << "Enter the Employee ID : ";
        cin >> id;
        id.resize(8);
        if (!file1) {
                cout << s4 << "Error Occured" << nl;
                return;
        }
        int index = 0;
        while (file1) {
                file1.seekg(index, ios::beg);
                getline(file1, str, '|');
        if (str == id) {
            found = 1;
            break;
        }
        else {
            index += 100;
            str.clear();
            count--;
        }
    }
    file1.close();
    if (found == 1)
    {
    start:
        string sdate, edate;
        int ch, com = 7, beg, end, nol, ex = 0, lea, tot, days = 0, leave, nod;
        float sal, per;
        cout << s4 << "1. CASUAL LEAVE" << nl;
        cout << s4 << "2. SICK LEAVE" << nl;
        cout << s4 << "3. STUDY LEAVE" << nl;
        cout << s4 << "4. PARENTAL LEAVE" << nl << nl;
        cout << s4 << cyp;
        cin >> ch;
        cout << nl;
        if (ch == 1) { beg = 0, end = 3, tot = 20, per = .08; }
        else if (ch == 2) { beg = 1, end = 2, tot = 15, per = 0.05; }
        else if (ch == 3) { beg = 2, end = 1, tot = 10, per = 0.03; }
        else if (ch == 4) { beg = 3, end = 0, tot = 40, per = 0.15; }
        file1.open("leave.txt", ios::binary | ios::in);
        file2.open("temp1.txt", ios::binary | ios::app);
        while (count--) {
            getline(file1, str);
            file2 << str << endl;
        }
```

```
for (int i = 0; i < 2; i++) {
        getline(file1, str, '|');
        file2 << str << "|";
}
for (int i = 0; i < beg; i++) {
        getline(file1, str, '|');
        file2 << str << "|";
}
cout << s4 << "Enter the number of leaves : ";
cin >> nol;
getline(file1, str, '|');
stringstream buf1(str);
buf1 >> lea;
leave = lea;
if (lea - nol < 0)
        days = abs(lea - nol);
lea = lea - nol;
if (days != 0)
        lea = 0;
fstream buf2;
buf2.open("mod.txt", ios::binary | ios::out);
buf2 << lea << "/" << tot << "-" << days;
buf2.close();
buf2.open("mod.txt", ios::binary | ios::in);
getline(buf2, str);
buf2.close();
remove("mod.txt");
str.resize(10);
file2 << str << "|";
for (int i = 0; i < end; i++) {
        getline(file1, str, '|');
        file2 << str << "|";
}
getline(file1, str, '|');
```

```
sal = getSal(id);

if (leave - nol < 0)
        sal = sal - (float)sal * per * abs(tot - leave);

ostringstream buf4;
buf4 << sal;
str = buf4.str();;
str.resize(8);
file2 << str << "|";

cout << s4 << "Enter the Start Date(dd/mm/yyyy) : ";
cin >> sdate;
cout << s4 << "Enter the End Date(dd/mm/yyy) : ";
cin >> edate;
sdate.resize(10);
edate.resize(10);
file2 << sdate << "|";
file2 << edate << "|";
for (int i = 0; i < 3; i++)
        getline(file1, str, '|');

stringstream buf5(str);
buf5 >> nod;
nod = nod + nol;

ostringstream buf6;
buf6 << nod;
str = buf6.str();
str.resize(5);
file2 << str << "|";

while (file1) {
        getline(file1, str);
```

```
                        file2 << str;
                        if (file1)
                                file2 << endl;
                }
                file1.close();
                file2.close();
                char fn1[] = "leave.txt", fn2[] = "temp1.txt";
                recover(fn1, fn2);
                cout << s4 << "Leave updated!" << nl << nl;
        }
        else
                cout << s4 << "User not found!" << nl << nl;
}
```

4.2.3 Delete Leave

```
void deleteLeave()
{
        string id, str;
        int found = 0, count = 0;
        fstream file1, file2;
        file1.open("leave.txt", ios::binary | ios::in);
        cout << s4 << "Enter the Employee ID to delete : ";
        cin >> id;
        id.resize(8);
        if (!file1) {
                cout << s4 << "Error Occured" << nl;
                return;
        }
        int index = 0;
        while (file1) {
                file1.seekg(index, ios::beg);
                getline(file1, str, '|');
                if (str == id) {
```

```
                found = 1;
                break;
            }
            else {
                index += 100;
                str.clear();
                count = count + 1;
            }
        }
        file1.close();
        if (found == 1)
        {
            file1.open("leave.txt", ios::binary | ios::in);
            file2.open("temp1.txt", ios::binary | ios::app);
            while (count > 0) {
                getline(file1, str);
                file2 <<  str;
                count = count - 1;
                if (count > 0)
                    file2 << nl;
            }
            for (int i = 0; i < 2; i++) {
                getline(file1, str, '|');
                file2 << str << "|";
            }
            str = "20/20-0"; str.resize(10);
            file2 << str << "|";
            str = "15/15-0"; str.resize(10);
            file2 << str << "|";
            str = "10/10-0"; str.resize(10);
            file2 << str << "|";
            str = "40/40-0"; str.resize(10);
            file2 << str << "|";
            float sal = getSal(id);
```

```
                ostringstream buf;
                buf << sal;
                str = buf.str();
                str.resize(8);
                file2 << str << "|";
                str = "-"; str.resize(10);
                file2 << str << "|" << str << "|";
                str = "0"; str.resize(10);
                file2 << str << "|";
                for (int i = 0; i < 8; i++)
                        getline(file1, str, '|');
                while (file1) {
                        getline(file1, str);
                        file2 << str;
                        if (file1)
                                file2 << endl;
                }
                file1.close();
                file2.close();
                char fn1[] = "leave.txt", fn2[] = "temp1.txt";
                recover(fn1, fn2);
        }
        else
                cout << s4 << "User not found!" << nl;
}

float getSal(string id)
{
        fstream file;
        float sal;
        string str;
        file.open("salary.txt", ios::binary | ios::in);
        if (!file) {
                cout << s4 << "Error Occured" << nl;
```

```
            return 0;
        }
        while (file) {
            getline(file, str, '|');
            str.resize(8);
            if (str == id)
                break;
        }
        str.clear();
        getline(file, str, '|');
        stringstream buf(str);
        buf >> sal;
        file.close();
        return sal;
}
```

4.2.4 Search Leave

```
void searchLeave()
{
        string id, str;
        cout << s4 << "Enter the Employee ID : ";
        cin >> id;
        id.resize(8);
        fstream file;
        file.open("leave.txt", ios::binary | ios::in);
        if (!file) {
            cout << s4 << "Error Occured" << nl;
            return;
        }
        while (file) {
            getline(file, str, '|');
            str.resize(8);
            if (str == id) {
                break;
                }
        }
        cout << nl << "Employee ID\t" << "Name\t\t" << "CL(20)\t\t" << "SL(15)\t\t" <<
"STL(10)\t\t" << "PL(40)\t\t" << "Salary\t\t" << "Start Date" << sp2 << sp2 << "End Date"
<< sp2 << sp2 << "NoofLeaves" << nl << nl;
        cout << str << "\t";
        for (int i = 0; i < 9; i++) {
                getline(file, str, '|');
                cout << str << "\t";
        }
        file.close();
        cout << nl << nl;
}
```

4.2.5 Modify Leave

```
void modLeave()
{
        string id, str;
        int found = 0, count = 0;
        fstream file1, file2;
        file1.open("leave.txt", ios::binary | ios::in);
        cout << s4 << "Enter the Employee ID : ";
        cin >> id;
        id.resize(8);
        if (!file1) {
                cout << s4 << "Error Occured" << nl;
                return;
        }
        int index = 0;
        while (file1) {
                file1.seekg(index, ios::beg);
                getline(file1, str, '|');
                if (str == id) {
                        found = 1;
                break;
            }
            else {
                index += 100;
                str.clear();
                count--;
            }
    }
    file1.close();
    if (found == 1)
    {
    start:
            string sdate, edate;
            int ch, com = 7, beg, end, nol, ex = 0, lea, tot, days = 0, leave, nod;
            float sal, per;
            cout << s4 << "1. CASUAL LEAVE" << nl;
            cout << s4 << "2. SICK LEAVE" << nl;
            cout << s4 << "3. STUDY LEAVE" << nl;
            cout << s4 << "4. PARENTAL LEAVE" << nl << nl;
            cout << s4 << cyp;
            cin >> ch;
            cout << nl;
            if (ch == 1) { beg = 0, end = 3, tot = 20, per = .08; }
            else if (ch == 2) { beg = 1, end = 2, tot = 15, per = 0.05; }
            else if (ch == 3) { beg = 2, end = 1, tot = 10, per = 0.03; }
            else if (ch == 4) { beg = 3, end = 0, tot = 40, per = 0.15; }
            file1.open("leave.txt", ios::binary | ios::in);
            file2.open("temp1.txt", ios::binary | ios::app);
            while (count--) {
                getline(file1, str);
                file2 << str << endl;
            }
            for (int i = 0; i < 2; i++) {
                getline(file1, str, '|');
```

```
        file2 << str << "|";
}
for (int i = 0; i < beg; i++) {
        getline(file1, str, '|');
        file2 << str << "|";
}
cout << s4 << "Enter the number of leaves : ";
cin >> nol;
getline(file1, str, '|');
stringstream buf1(str);
buf1 >> lea;
leave = lea;
if (lea - nol < 0)
        days = abs(lea - nol);
lea = lea - nol;
if (days != 0)
        lea = 0;
fstream buf2;
buf2.open("mod.txt", ios::binary | ios::out);
buf2 << lea << "/" << tot << "-" << days;
buf2.close();
buf2.open("mod.txt", ios::binary | ios::in);
getline(buf2, str);
buf2.close();
remove("mod.txt");
str.resize(10);
file2 << str << "|";
for (int i = 0; i < end; i++) {
        getline(file1, str, '|');
        file2 << str << "|";
}
getline(file1, str, '|');
sal = getSal(id);
```

```
if (leave - nol < 0)
        sal = sal - (float)sal * per * abs(tot - leave);

ostringstream buf4;
buf4 << sal;
str = buf4.str();;
str.resize(8);
file2 << str << "|";

cout << s4 << "Enter the Start Date(dd/mm/yyyy) : ";
cin >> sdate;
cout << s4 << "Enter the End Date(dd/mm/yyy) : ";
cin >> edate;
sdate.resize(10);
edate.resize(10);
file2 << sdate << "|";
file2 << edate << "|";
for (int i = 0; i < 3; i++)
        getline(file1, str, '|');

stringstream buf5(str);
buf5 >> nod;
nod = nod + nol;

ostringstream buf6;
buf6 << nod;
str = buf6.str();
str.resize(5);
file2 << str << "|";

while (file1) {
        getline(file1, str);
        file2 << str;
        if (file1)
```

```
                        file2 << endl;
                }
                file1.close();
                file2.close();
                char fn1[] = "leave.txt", fn2[] = "temp1.txt";
                recover(fn1, fn2);
                cout << s4 << "Leave updated!" << nl << nl;
        }
        else
                cout << s4 << "User not found!" << nl << nl;
}
```

4.2.6 Display all Leaves

```
void dispLeave()
{
        string str;
        fstream file;
        file.open("leave.txt", ios::binary | ios::in);
        if (!file) {
                cout << s4 << "Error Occured" << nl;
                return;
        }
        cout << nl << "Employee ID\t" << "Name\t\t" << "CL(20)\t\t" << "SL(15)\t\t" <<
"STL(10)\t\t" << "PL(40)\t\t" << "Salary\t\t" << "Start Date" << sp2 << sp2 << "End Date" <<
sp2 << sp2 << "NoofLeaves" << nl << nl;
        while (file) {
                getline(file, str, '|');
                cout << str << "\t";
        }
        cout << nl << nl;
        cout << "Note : This table shows remaining leaves." << nl << sp2 << "CL : CASUAL
LEAVE" << nl << sp2 << "SL : SICK LEAVESEAVE" << nl << sp2 << "STL : STUDY
LEAVE" << nl << sp2 << "PL : PARENTAL LEAVE" << nl << sp2 << "Number after '-' sign
indicates extra leaves." << nl << nl;
        file.close();
}
```

4.2.7 User Manageent

```
  void addUser()
  {
        Employee e;
        cout << s4 << "Enter the ID : ";
        cin >> e.ssn;
        e.ssn.resize(8);
        cout << s4 << "Enter the Name : ";
        cin >> e.name;
        e.name.resize(8);
        cout << s4 << "Enter the Age : ";
        cin >> e.age;
        e.age.resize(8);
        cout << s4 << "Enter the Salary : ";
        cin >> e.sal;
        e.sal.resize(8);
        cout << s4 << "Enther the Experience : ";
        cin >> e.exp;
        e.exp.resize(8);
        fstream file1, file2, file3;
        file1.open("emp.txt", ios::binary | ios::app);
        file2.open("leave.txt", ios::binary | ios::app);
        file3.open("salary.txt", ios::binary | ios::app);
        if (!file1 || !file2) {
                cout << s4 << "Error Occured!" << nl;
                return;
        }
        e.casual = e.casual + "/20-0", e.sick = e.sick + "/15-0", e.study = e.study + "/10-0",
  e.parental = e.parental + "/40-0";
```

```
	e.casual.resize(10), e.sick.resize(10), e.study.resize(10), e.parental.resize(10), e.sdate.resize(10), e.edate.resize(10), e.days.resize(5);
	file1 << s4 << "|" << e.ssn << "|" << e.name << "|" << e.age << "|" << e.sal << "|" << e.exp << "|" << nl;
	file2 << e.ssn << "|" << e.name << "|" << e.casual << "|" << e.sick << "|" << e.study << "|" << e.parental << "|" << e.sal << "|" << e.sdate << "|" << e.edate << "|" << e.days << "|" << nl;
	file3 << e.ssn << "|" << e.sal << "|" << nl;
	file1.close();
	file2.close();
	file3.close();
	cout << s4 << "User added successfully!" << nl << nl << nl;
}
```

4.2.8 Quit Program

o Prompts the user to enter option 7 when he/she is in the console window.

o When user enters 7 then the program will be terminated.

Chapter 5

Testing and Debugging

5.1 Testing

The implementation phase of software development is concerned with translating design specification into source code. The preliminary goal of implementation is to write source code and internal documentation so that conformance of the code to its specifications can be easily verified, and so that debugging, testing and modifications are eased. This goal can be achieved by making the source code as clear and straight forward as possible. Simplicity,clarity and elegance are the hallmark of good programs, obscurity, cleverness, and complexityare indications of inadequate design and misdirected thinking.

Source code clarity is enhanced by structured coding techniques, by good coding style, by, appropriate supporting documents, by good internal comments, and by feature provided in modern programming languages.

The implementation team should be provided with a well-defined set of software requirement,an architectural design specification, and a detailed design description. Each team member must understand the objectives of implementation.

TERMS IN TESTING FUNDAMENTAL

- **Error**

The term error is used in two ways. It refers to the difference between the actual output of software and the correct output, in this interpretation, error is essential a measure of the difference between actual and ideal. Error is also to used to refer to human action that result in software containing a defect or fault.

- **Fault**

Fault is a condition that causes to fail in performing its required function. A fault is a basic reason for software malfunction and is synonymous with the commonly used term Bug.

o Failure

Failure is the inability of a system or component to perform a required function according to its specifications. A software failure occurs if the behavior of the software is the different from the specified behavior. Failure may be caused due to functional or performance reasons.

a. Unit Testing

The term unit testing comprises the sets of tests performed by an individual programmer prior to integration of the unit into a larger system. A program unit is usually small enough that the programmer who developed it can test it in great detail, and certainly in greater detail than will be possible when the unit is integrated into an evolving software product. In the unit testing the programs are tested separately, independent of each other. Since the check is done at the program level, it is also called program teasing.

b. Module Testing

A module and encapsulates related component. So can be tested without other system module.

c. Subsystem Testing

Subsystem testing may be independently design and implemented common problems are sub-system interface mistake in this checking we concentrate on it.

There are four categories of tests that a programmer will typically perform on a program unit.

1) Functional test

2) Performance test

3) Stress test

4) Structure test

1) Functional Test

Functional test cases involve exercising the code with Nominal input values for which expected results are known; as well as boundary values (minimum values, maximum values and values on and just outside the functional boundaries) and special values.

2) Performance Test

Performance testing determines the amount of execution time spent in various parts of the unit, program throughput, response time, and device utilization by the program unit. A certain amount of avoid expending too much effort on fine-tuning of a program unit that contributes little to the over all performance of the entire system. Performance testing is most productive at the subsystem and system levels.

3) Stress Test

Stress test are those designed to intentionally break the unit. A great deal can be learned about the strengths and limitations of a program by examining the manner in which a program unit breaks.

4) Structure Test

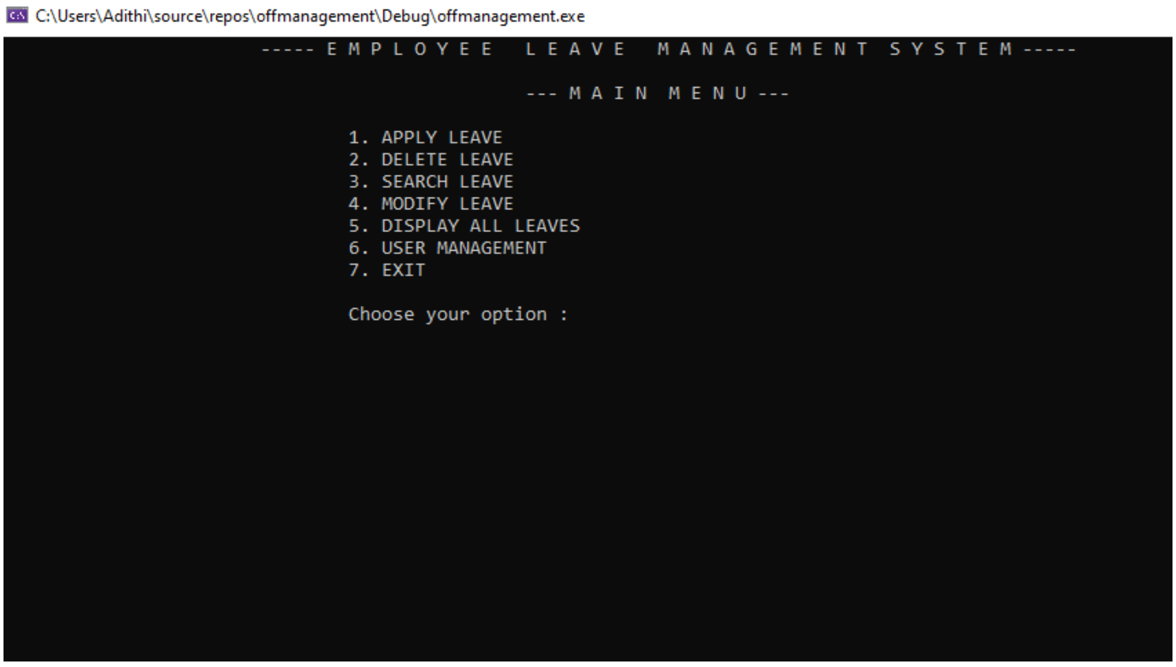
Structure tests are concerned with exercising the internal logic of a program and traversing particular execution paths. Some authors refer collectively to functional performance and stress testing as “black box” testing. While structure testing is referred to as “white box” or “glass box” testing. The major activities in structural testing are deciding which path to exercise, deriving test date to exercise those paths and measuring the test coverage achieved when the test cases are exercised.

5.2 Debugging

Defect testing is intended to find areas where the program does not confirm to its specifications. Tests are designed to reveal the presence of defect in the system. When defect have been found in the program. There must be discovered and removed. This is called “Debugging”.

Chapter 6

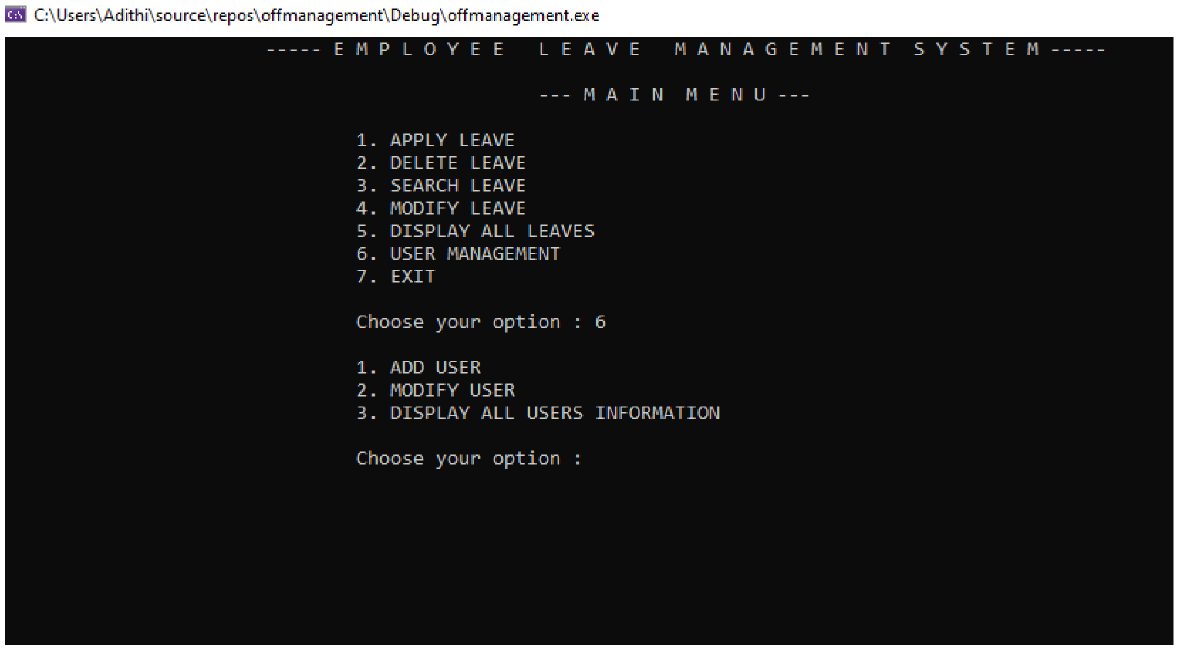
SNAPSHOTS



```
C:\Users\Adithi\source\repos\offmanagement\Debug\offmanagement.exe
                        ----- E M P L O Y E E   L E A V E   M A N A G E M E N T  S Y S T E M -----

                                        --- M A I N  M E N U ---

                                1. APPLY LEAVE
                                2. DELETE LEAVE
                                3. SEARCH LEAVE
                                4. MODIFY LEAVE
                                5. DISPLAY ALL LEAVES
                                6. USER MANAGEMENT
                                7. EXIT

                                Choose your option :
```

Figure 6.1 Main Menu

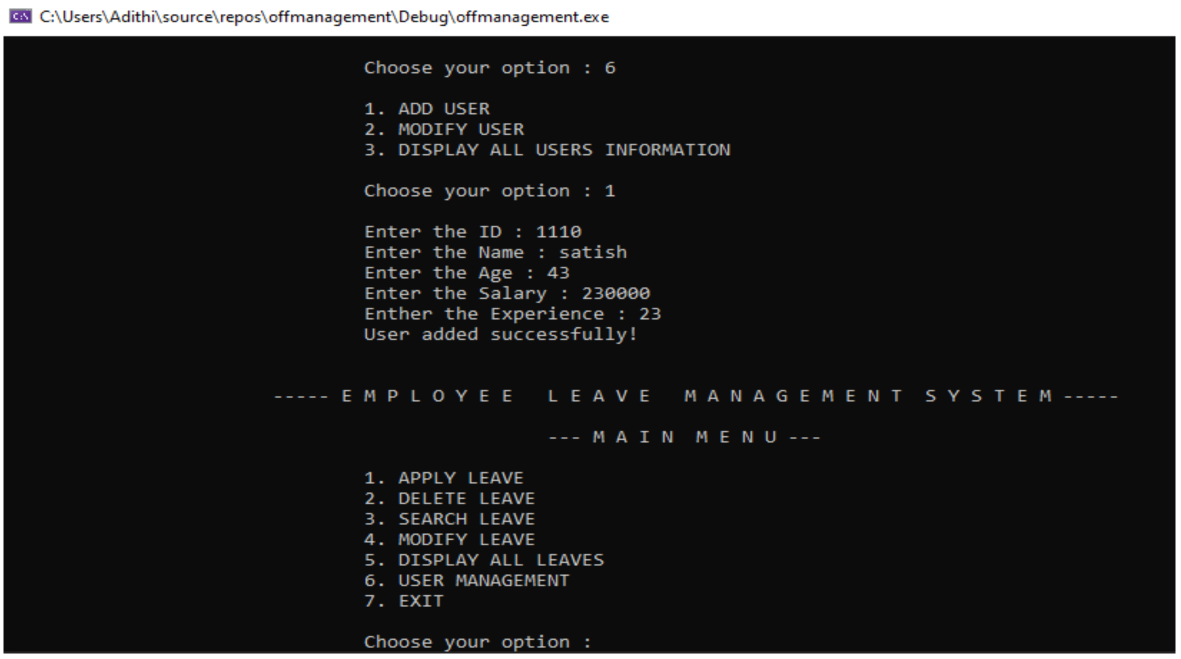
When the user runs the program, Figure 6.1 appears initially giving the user a range of options.



```
C:\Users\Adithi\source\repos\offmanagement\Debug\offmanagement.exe
                        ----- E M P L O Y E E   L E A V E   M A N A G E M E N T  S Y S T E M -----

                                        --- M A I N  M E N U ---

                                1. APPLY LEAVE
                                2. DELETE LEAVE
                                3. SEARCH LEAVE
                                4. MODIFY LEAVE
                                5. DISPLAY ALL LEAVES
                                6. USER MANAGEMENT
                                7. EXIT

                                Choose your option : 6

                                1. ADD USER
                                2. MODIFY USER
                                3. DISPLAY ALL USERS INFORMATION

                                Choose your option :
```

Figure 6.2 User Management

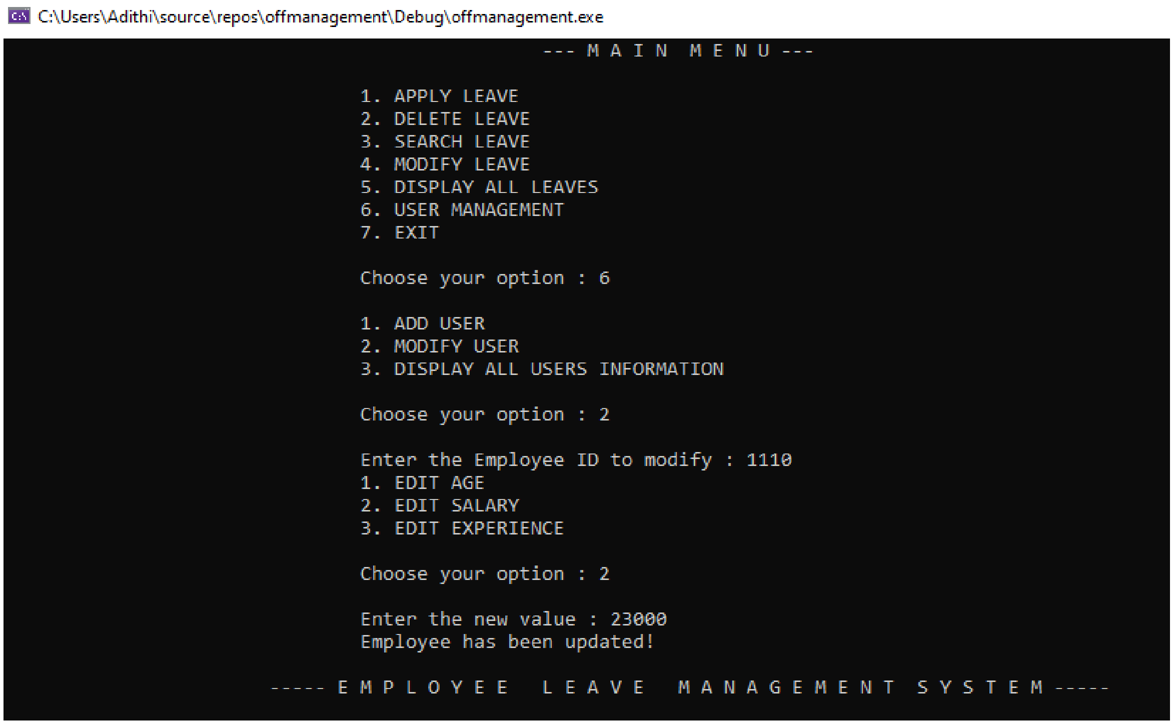
Figure 6.2 shows the list of user management ,when the user selects option 6 in main menu.



```
C:\Users\Adithi\source\repos\offmanagement\Debug\offmanagement.exe

                    Choose your option : 6

                    1. ADD USER
                    2. MODIFY USER
                    3. DISPLAY ALL USERS INFORMATION

                    Choose your option : 1

                    Enter the ID : 1110
                    Enter the Name : satish
                    Enter the Age : 43
                    Enter the Salary : 230000
                    Enther the Experience : 23
                    User added successfully!

        ----- E M P L O Y E E   L E A V E   M A N A G E M E N T  S Y S T E M -----

                            --- M A I N   M E N U ---

                    1. APPLY LEAVE
                    2. DELETE LEAVE
                    3. SEARCH LEAVE
                    4. MODIFY LEAVE
                    5. DISPLAY ALL LEAVES
                    6. USER MANAGEMENT
                    7. EXIT

                    Choose your option :
```

Figure 6.3 Add User in User Management

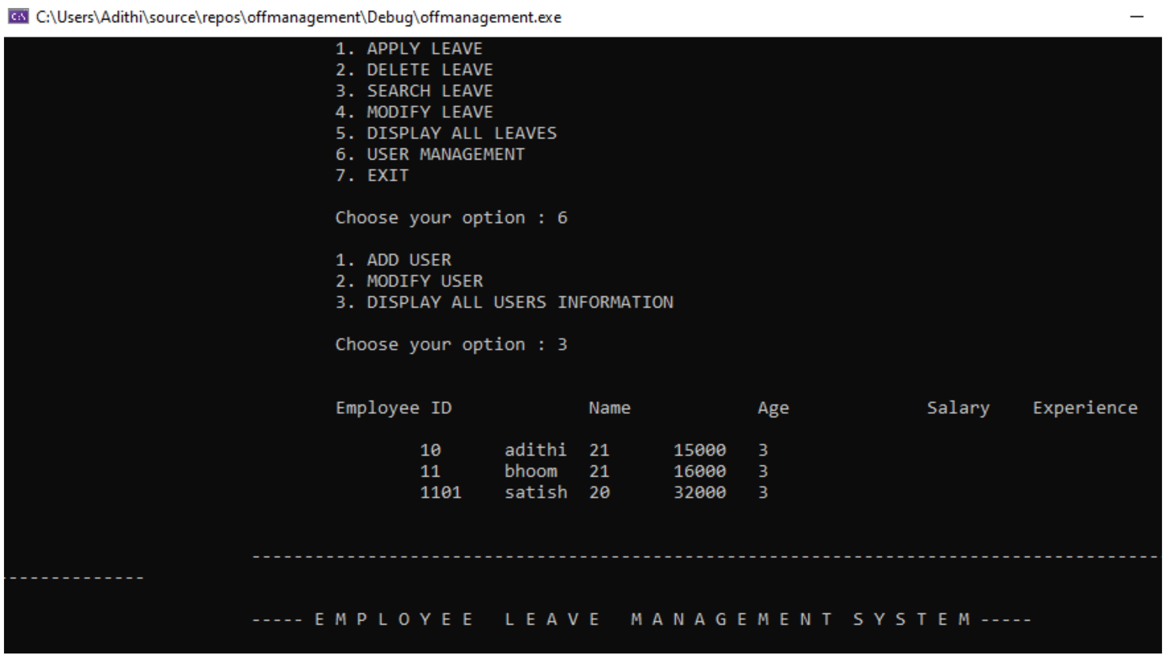
Figure 6.3 appears when a new user is to be added into the system. Details such as the user's ID ,Name, Age , Salary is taken as input .



```
C:\Users\Adithi\source\repos\offmanagement\Debug\offmanagement.exe
                            --- M A I N   M E N U ---

                    1. APPLY LEAVE
                    2. DELETE LEAVE
                    3. SEARCH LEAVE
                    4. MODIFY LEAVE
                    5. DISPLAY ALL LEAVES
                    6. USER MANAGEMENT
                    7. EXIT

                    Choose your option : 6

                    1. ADD USER
                    2. MODIFY USER
                    3. DISPLAY ALL USERS INFORMATION

                    Choose your option : 2

                    Enter the Employee ID to modify : 1110
                    1. EDIT AGE
                    2. EDIT SALARY
                    3. EDIT EXPERIENCE

                    Choose your option : 2

                    Enter the new value : 23000
                    Employee has been updated!

        ----- E M P L O Y E E   L E A V E   M A N A G E M E N T  S Y S T E M -----
```

Figure 6.4 Modify User in User Management

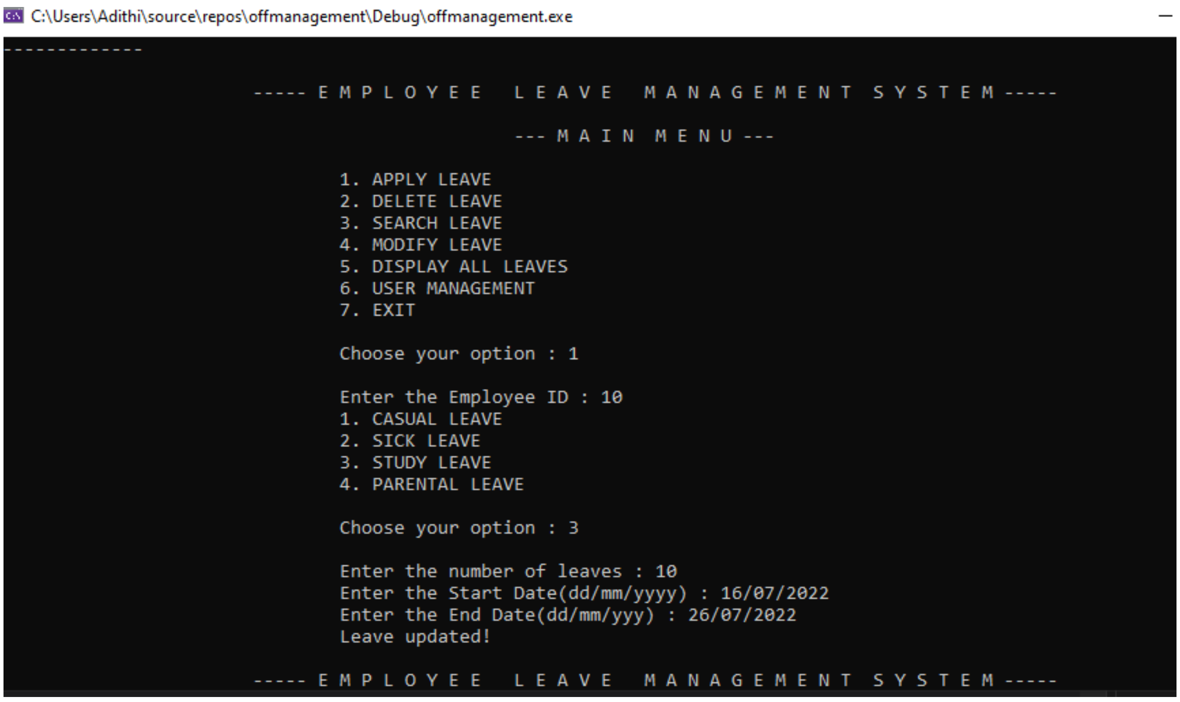
In Figure 6.4 ,details of the user can be modified in the User Management .Here, user ID is taken as the input to modify the details of the respective User.



```
C:\Users\Adithi\source\repos\offmanagement\Debug\offmanagement.exe
                              1. APPLY LEAVE
                              2. DELETE LEAVE
                              3. SEARCH LEAVE
                              4. MODIFY LEAVE
                              5. DISPLAY ALL LEAVES
                              6. USER MANAGEMENT
                              7. EXIT

                              Choose your option : 6

                              1. ADD USER
                              2. MODIFY USER
                              3. DISPLAY ALL USERS INFORMATION

                              Choose your option : 3


                              Employee ID            Name            Age            Salary    Experience

                                      10      adithi  21      15000   3
                                      11      bhoom   21      16000   3
                                      1101    satish  20      32000   3


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

                       ----- E M P L O Y E E   L E A V E   M A N A G E M E N T  S Y S T E M -----
```

Figure 6.5 Display all users information in User Management

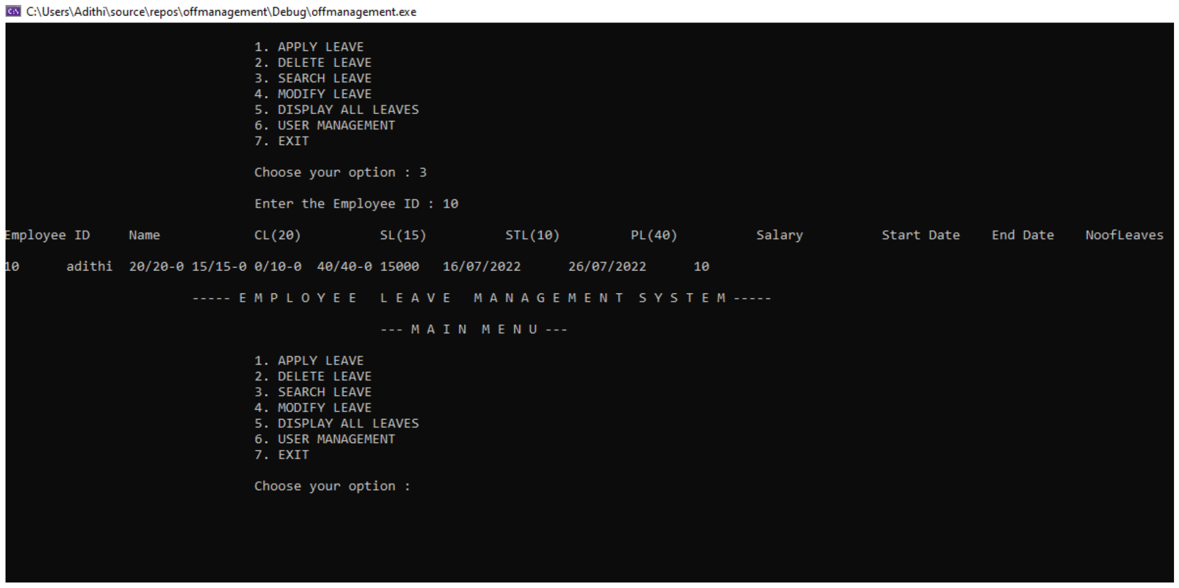
Figure 6.5 is seen on the screen when the user wishes to display all users information.



```
C:\Users\Adithi\source\repos\offmanagement\Debug\offmanagement.exe
-------------

                       ----- E M P L O Y E E   L E A V E   M A N A G E M E N T  S Y S T E M -----

                                         --- M A I N  M E N U ---

                              1. APPLY LEAVE
                              2. DELETE LEAVE
                              3. SEARCH LEAVE
                              4. MODIFY LEAVE
                              5. DISPLAY ALL LEAVES
                              6. USER MANAGEMENT
                              7. EXIT

                              Choose your option : 1

                              Enter the Employee ID : 10
                              1. CASUAL LEAVE
                              2. SICK LEAVE
                              3. STUDY LEAVE
                              4. PARENTAL LEAVE

                              Choose your option : 3

                              Enter the number of leaves : 10
                              Enter the Start Date(dd/mm/yyyy) : 16/07/2022
                              Enter the End Date(dd/mm/yyy) : 26/07/2022
                              Leave updated!

                       ----- E M P L O Y E E   L E A V E   M A N A G E M E N T  S Y S T E M -----
```

Figure 6.6 Apply Leave

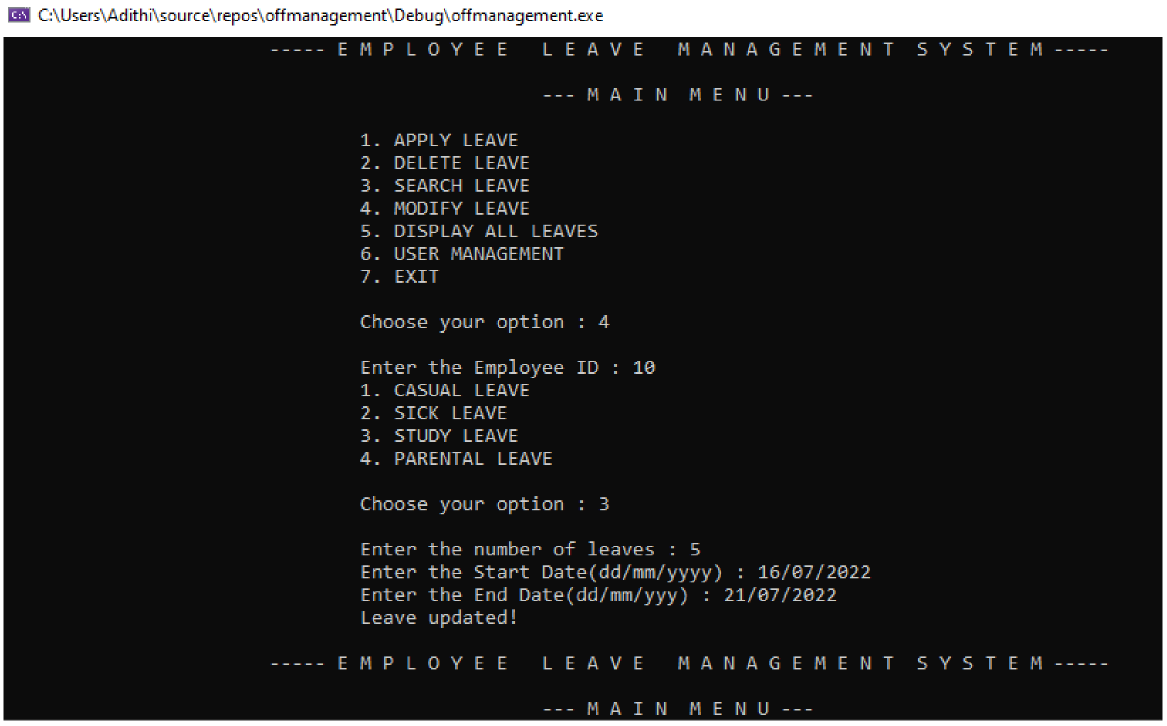
When the user wishes to Apply leave, option 1 is selected from the main menu and Figure 6.6 is seen where the user enters the ID and selects the type of leave ,mentioning the start and end date.



```
C:\Users\Adithi\source\repos\offmanagement\Debug\offmanagement.exe
                                1. APPLY LEAVE
                                2. DELETE LEAVE
                                3. SEARCH LEAVE
                                4. MODIFY LEAVE
                                5. DISPLAY ALL LEAVES
                                6. USER MANAGEMENT
                                7. EXIT

                                Choose your option : 3

                                Enter the Employee ID : 10

Employee ID     Name            CL(20)          SL(15)          STL(10)         PL(40)          Salary          Start Date      End Date        NoofLeaves

10      adithi  20/20-0 15/15-0 0/10-0  40/40-0 15000   16/07/2022      26/07/2022      10
                        ----- E M P L O Y E E   L E A V E   M A N A G E M E N T  S Y S T E M -----

                                        --- M A I N  M E N U ---

                                1. APPLY LEAVE
                                2. DELETE LEAVE
                                3. SEARCH LEAVE
                                4. MODIFY LEAVE
                                5. DISPLAY ALL LEAVES
                                6. USER MANAGEMENT
                                7. EXIT

                                Choose your option :
```

Figure 6.7 Search Leave

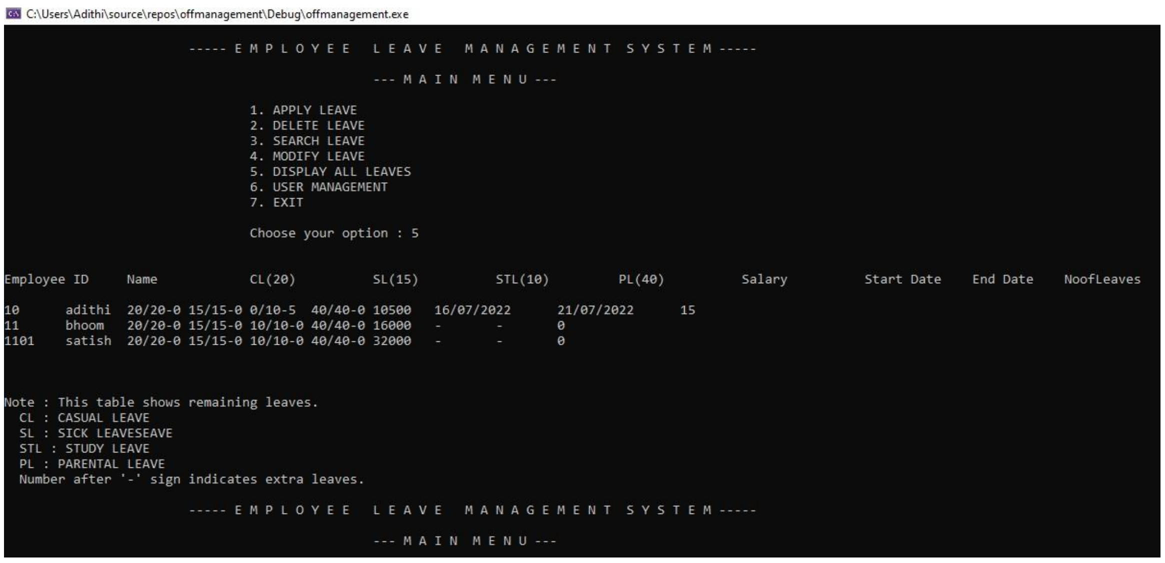
When the user wishes to search the leave, option 3 is selected from the main menu and Figure 6.7 is seen where the user enters the ID which has to be searched.



```
C:\Users\Adithi\source\repos\offmanagement\Debug\offmanagement.exe
                        ----- E M P L O Y E E   L E A V E   M A N A G E M E N T  S Y S T E M -----

                                        --- M A I N  M E N U ---

                                1. APPLY LEAVE
                                2. DELETE LEAVE
                                3. SEARCH LEAVE
                                4. MODIFY LEAVE
                                5. DISPLAY ALL LEAVES
                                6. USER MANAGEMENT
                                7. EXIT

                                Choose your option : 4

                                Enter the Employee ID : 10
                                1. CASUAL LEAVE
                                2. SICK LEAVE
                                3. STUDY LEAVE
                                4. PARENTAL LEAVE

                                Choose your option : 3

                                Enter the number of leaves : 5
                                Enter the Start Date(dd/mm/yyyy) : 16/07/2022
                                Enter the End Date(dd/mm/yyy) : 21/07/2022
                                Leave updated!

                        ----- E M P L O Y E E   L E A V E   M A N A G E M E N T  S Y S T E M -----

                                        --- M A I N  M E N U ---
```

Figure 6.8 Modify Leave

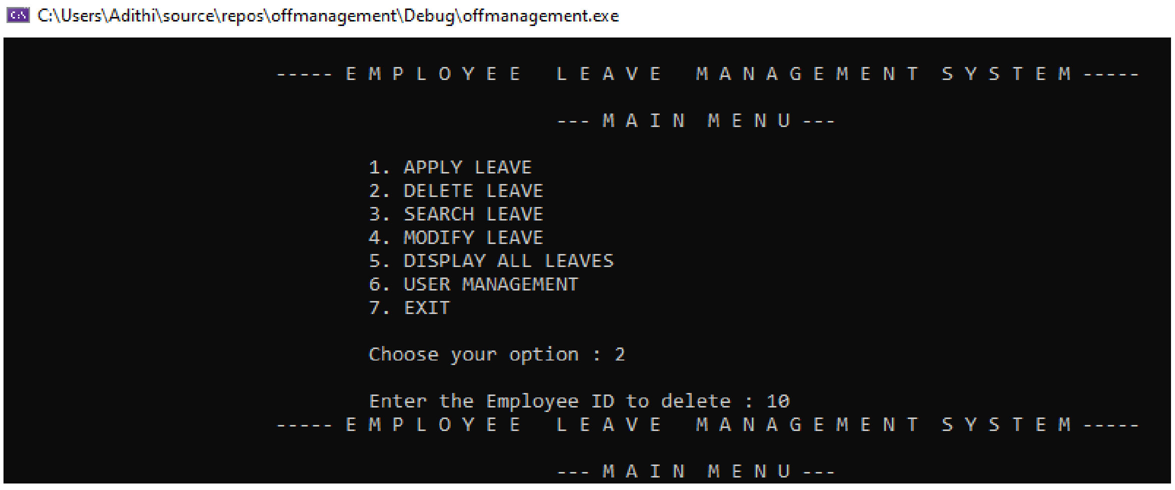
When the user wishes to modify the leave, option 4 is selected from the main menu and Figure 6.8 is seen where the user enters the ID which has to be modified.



```
C:\Users\Adithi\source\repos\offmanagement\Debug\offmanagement.exe
                        ----- E M P L O Y E E   L E A V E   M A N A G E M E N T  S Y S T E M -----

                                                --- M A I N  M E N U ---

                                1. APPLY LEAVE
                                2. DELETE LEAVE
                                3. SEARCH LEAVE
                                4. MODIFY LEAVE
                                5. DISPLAY ALL LEAVES
                                6. USER MANAGEMENT
                                7. EXIT

                                Choose your option : 5


Employee ID     Name            CL(20)          SL(15)          STL(10)         PL(40)          Salary          Start Date    End Date     NoofLeaves

10      adithi  20/20-0 15/15-0 0/10-5  40/40-0 10500   16/07/2022      21/07/2022      15
11      bhoom   20/20-0 15/15-0 10/10-0 40/40-0 16000   -       -       0
1101    satish  20/20-0 15/15-0 10/10-0 40/40-0 32000   -       -       0


Note : This table shows remaining leaves.
  CL : CASUAL LEAVE
  SL : SICK LEAVESEAVE
  STL : STUDY LEAVE
  PL : PARENTAL LEAVE
  Number after '-' sign indicates extra leaves.

                        ----- E M P L O Y E E   L E A V E   M A N A G E M E N T  S Y S T E M -----

                                                --- M A I N  M E N U ---
```

Figure 6.9 Display all Leaves

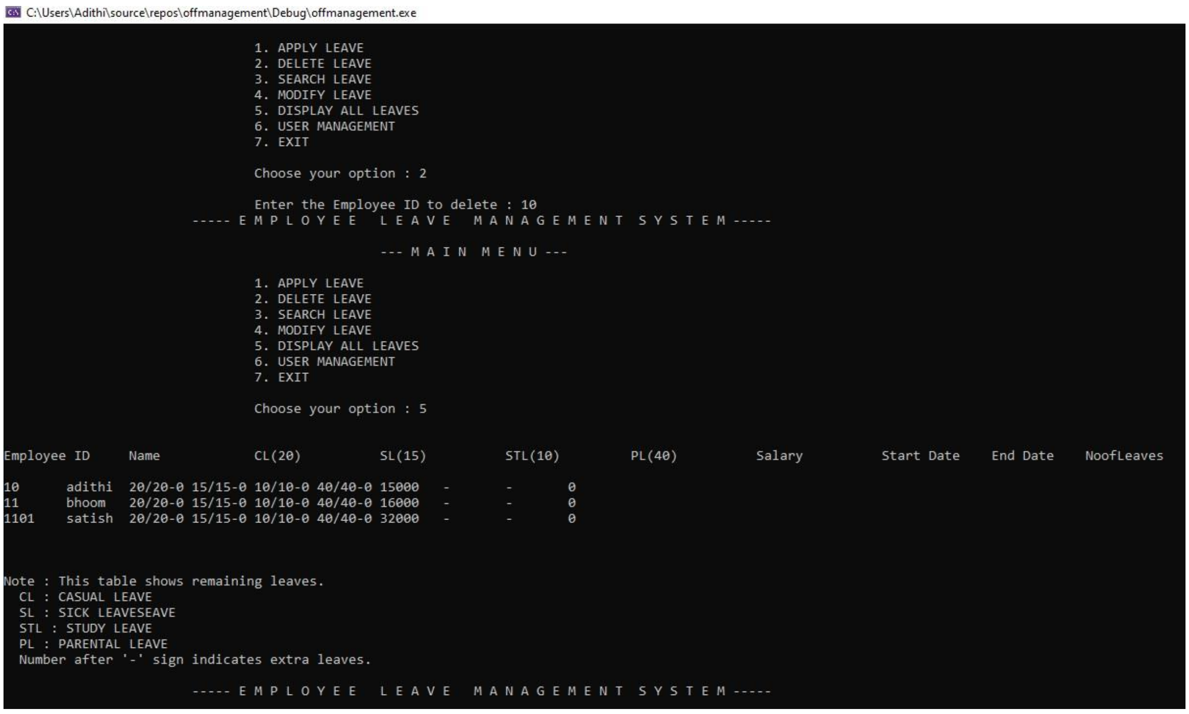
When the user wishes to display all the leaves, option 5 is selected from the main menu .Figure 6.9 is seen before a particular leave is deleted.



```
C:\Users\Adithi\source\repos\offmanagement\Debug\offmanagement.exe
                        ----- E M P L O Y E E   L E A V E   M A N A G E M E N T  S Y S T E M -----

                                                --- M A I N  M E N U ---

                                1. APPLY LEAVE
                                2. DELETE LEAVE
                                3. SEARCH LEAVE
                                4. MODIFY LEAVE
                                5. DISPLAY ALL LEAVES
                                6. USER MANAGEMENT
                                7. EXIT

                                Choose your option : 2

                                Enter the Employee ID to delete : 10
                        ----- E M P L O Y E E   L E A V E   M A N A G E M E N T  S Y S T E M -----

                                                --- M A I N  M E N U ---
```

Figure 6.10 Delete Leave

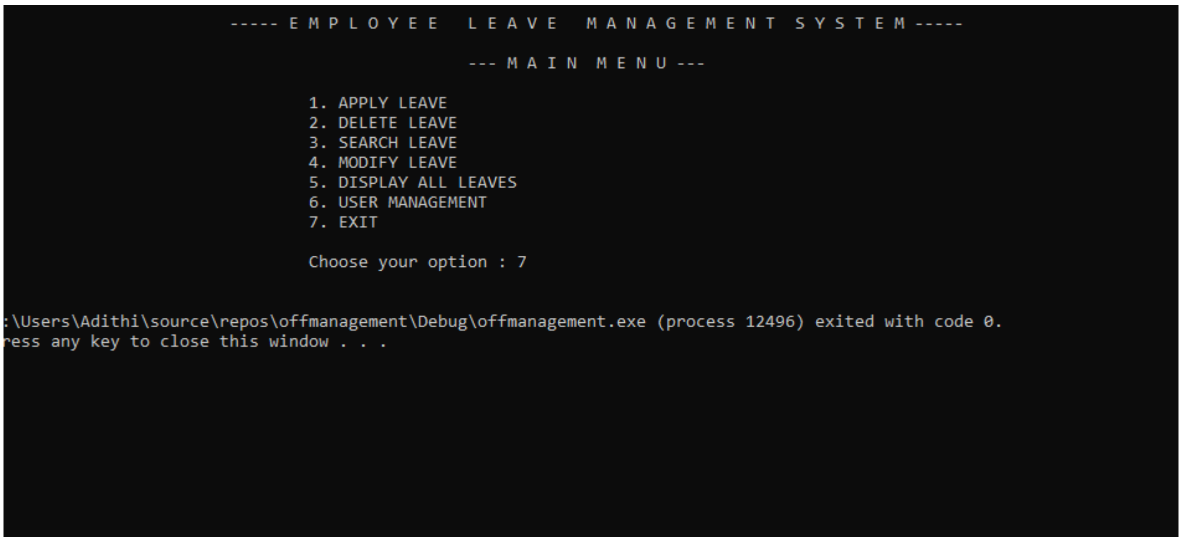
When the user wishes to delete the leave, option 2 is selected from the main menu and Figure 6.10 is seen where the user enters the ID which has to be deleted.



```
C:\Users\Adithi\source\repos\offmanagement\Debug\offmanagement.exe
                                1. APPLY LEAVE
                                2. DELETE LEAVE
                                3. SEARCH LEAVE
                                4. MODIFY LEAVE
                                5. DISPLAY ALL LEAVES
                                6. USER MANAGEMENT
                                7. EXIT

                                Choose your option : 2

                                Enter the Employee ID to delete : 10
                        ----- E M P L O Y E E   L E A V E   M A N A G E M E N T  S Y S T E M -----

                                                --- M A I N  M E N U ---

                                1. APPLY LEAVE
                                2. DELETE LEAVE
                                3. SEARCH LEAVE
                                4. MODIFY LEAVE
                                5. DISPLAY ALL LEAVES
                                6. USER MANAGEMENT
                                7. EXIT

                                Choose your option : 5


Employee ID     Name            CL(20)          SL(15)          STL(10)         PL(40)          Salary          Start Date     End Date     NoofLeaves

10      adithi  20/20-0 15/15-0 10/10-0 40/40-0 15000   -       -       0
11      bhoom   20/20-0 15/15-0 10/10-0 40/40-0 16000   -       -       0
1101    satish  20/20-0 15/15-0 10/10-0 40/40-0 32000   -       -       0


Note : This table shows remaining leaves.
  CL : CASUAL LEAVE
  SL : SICK LEAVESEAVE
  STL : STUDY LEAVE
  PL : PARENTAL LEAVE
  Number after '-' sign indicates extra leaves.

                        ----- E M P L O Y E E   L E A V E   M A N A G E M E N T  S Y S T E M -----
```

Figure 6.11 Display all Leaves

When the user wishes to display all the leaves, option 5 is selected from the main menu .Figure 6.11 is seen after a particular leave is deleted.



```
                        ----- E M P L O Y E E   L E A V E   M A N A G E M E N T  S Y S T E M -----

                                                --- M A I N  M E N U ---

                                1. APPLY LEAVE
                                2. DELETE LEAVE
                                3. SEARCH LEAVE
                                4. MODIFY LEAVE
                                5. DISPLAY ALL LEAVES
                                6. USER MANAGEMENT
                                7. EXIT

                                Choose your option : 7


:\Users\Adithi\source\repos\offmanagement\Debug\offmanagement.exe (process 12496) exited with code 0.
ress any key to close this window . . .
```

Figure 6.12 Quit Program

When the user enters option 7, the program will be terminated.

CONCLUSION AND FUTURE ENHANCEMENT

Conclusion

Off management System project has been successfully developed using Turbo C++ under Windows platform. The User can apply leave, delete leave from the list based on user ID , user can display, search leaves based on the same user ID. User can also modify the entered leave.

Advantages

• Fast retrieval of information.

• Easy access.

• The user can keep track of the events easily.

All these goals are achieved and now here we are with a Event Management System.

Future Enhancement

For any system, present satisfaction is important, but it is also necessary to see and visualize the future scope. Future enhancement is necessary for any system as the limitations that cannot be denied by anybody. These limitations can be overcome by

• Better technologies.

• Adding Feedback and suggestion option.

The main goals of this mini-project are:

• To learn accessing data from file system and displaying, fetching, retrieving, inserting, deleting to it using different techniques available.

• To write Dataflow Diagram and Flow-Chart for the same file system.

REFERENCES

Books

• File Structures: An Object-Oriented Approach with C++ 3rd Edition by Michael J. Folk (Author), Bill Zoellick (Author), Greg Riccardi (Author).

• The C++ Programming Language, 4th Edition by Bjarne Stroustrup (Author).

• The Waite Group's C Programming Using Turbo C+/Book and Disk Subsequent Edition by Robert Lafore (Author**).**

Online Websites

• For Data Flow Diagram and Flowchart, Lucid chart is a web-based commercial service to create flowcharts, organizational charts, website wireframes, and other things.

• https://www.lucidchart.com/documents/edit/6830d573-57de-4424- 83b2- 660f3108bd9b- for Data Flow Diagram • www.stackoverflow.com/files/

• Turbo C++ tutorial in YouTube: https://www.youtube.com/watch?vR15KpkibkR0 www.codebook.com/files/