VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JNANA SANGAMA”, Belagavi-590018

A Mini Project Report on

“Event Management System”

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR

THE AWARD OF DEGREE OF

BACHELOR OF ENGINEERING IN
INFORMATION SCIENCE AND ENGINEERING

SUBMITTED BY

PRATHEEK D (1JB18IS072)
PREETHI C S (1JB18IS075)

Under the Guidance of

Mr. Uday Shankar S V
Assistant Professor,
Dept. of ISE, SJBIT
Bengaluru-60

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

SJB INSTITUTE OF TECHNOLOGY

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

2020-2021

|| Jai Sri Gurudev ||
Sri Adichunchanagiri Shikshana Trust ®

SJB INSTITUTE OF TECHNOLOGY

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

Department of Information Science & Engineering

SJBIT

CERTIFICATE

Certified that the Mini-project work entitled **“Event Management System”,** is bonafide work carried out by **PRATHEEK D (1JB18IS072) and PREETHI C S (1JB18IS075),** students of **SJB Institute of Technology**, in partial fulfilment for 6th semester in **INFORMATION SCIENCE AND ENGINEERING** of the **VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI** during the academic year **2020-21.** 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.

Mr. Uday Shankar S V
Asst. Professor
Dept. of ISE, SJBIT

Dr. MOHAN H S
Professor & Head
Dept. of ISE, SJBIT

EXTERNAL VIVA

Name of the Examiners **Signature with Date**

1. ____________________ ____________________

2. ____________________ ____________________

ACKNOWLEDGEMENT

We would like to express our profound thanks to His Divine Soul **Padmabhushan Sri Sri Sri Dr. Balagangadharanatha Maha Swamiji** and His Holiness **Jagadguru Sri Sri Sri Dr. Nirmalanandanatha Maha Swamiji** for providing us an opportunity to pursue our academics in this esteemed institution.

We would also like to express our 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 mini project in this admired institution.

We express our gratitude to **Dr. Ajai Chandran C K**, Principal, SJB Institute of Technology, for providing excellent facilities and academic ambience, which have helped us in satisfactory completion of mini project work.

We extend our sincere thanks to **Dr. Mohan H.S**, Head of the Department, Information Science and Engineering, for providing an invaluable support throughout the period of mini project work.

We wish to express heartfelt gratitude to our guide**, Prof. Uday Shankar S V,** Assistant Professor, for her valuable guidance, suggestions and cheerful encouragement during the entire period of this work.

We express my truthful thanks to **Prof. Jeevraj R,** Mini Project Coordinator**,** who has been a source of inspiration to us and has extended his support throughout the mini project duration.

Finally, we take this opportunity to extend our earnest gratitude and respect to our parents, Teaching & Non-teaching staffs of the department, the library staff and all my friends, for their continuous support and encouragement.

PRATHEEK D (1JB18IS072)
PREETHI C S (1JB18IS075)

ABSTRACT

The event management system is a system where the user can perform actions on data of events. This system makes use of a technique called as the Hashing technique to store data entered by the user in file. The user can perform various actions such as Adding an event, deleting an event, searching for an event, etc. While adding the event, the user has to enter details such as event ID, Event name, Date of Conduction, Entry Fee and Prize amount. Once the user enters these, it is stored in the file. When the user wishes to display all the data of events stored in the file, the user selects the corresponding option to display the details. Along with this, the user can search for the details of a specific event by giving the event ID as the input. With the help of this ID, all details of the respective event is displayed. In addition, the user can delete a particular event if the user wishes to by passing the event ID as the input. If the event is present in the file, the record will be deleted. Otherwise, a message stating the record is not found is displayed. The other operation user can perform is to modify the event details. When the user selects the corresponding option to modify the details, the user can edit the details of the event by passing the event ID as the input, and then changing the details of the consequent event. These are the variety of operations that the user can perform in this system. The application prototype is implemented on Turbo C++.Turbo C++ is a C++ compiler and Integrated Development Environment and computer language originally from Borland. It includes a source code editor, a fast compiler, a linker and an offline help file for reference. Version 2 included a built-in debugger. C++ is a general-purpose object-oriented programming (OOP) language, developed by Bjarne Stroustrup, and is an extension of the C language.

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 might also 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 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. The problem was that, even with a balanced binary tree, dozens of accesses were 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 file block containing dozens, perhaps even hundreds, of records. Hashing is a good way to get what we 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 well with volatile, dynamic files. Extendible dynamic hashing can retrieve information with 1 or at most 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, bytes the 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 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, 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 the disk.

1.5Introduction to Project

This is the project about Event Management System. In this project we can easily maintain events and its details. The scope of project ‘Event management system’ is to Develop C++ based software using Hashing technique to store the details of the events and maintain all information of event related items. The primary aim of Event Management system is to improve accuracy and efficiency of tracking and keeping details of Events entered. This project has many facilities for users. We have a console window with 6 options that is Add an Event, search an Event, display all Events, delete an Event, Modify an Event and quit program.

Firstly Add an Event, this option helps in storing the details of the event like Event id, Event name, Date of conduction, Entry Fee and Prize Amount. In Search an event we can search the details of the event using Event id. Display all Events, in this option we will get to display of all the events. Delete an Event, in this option we are allowed to delete any event using its Event id. In Modify an Event, we can modify the details of the event by giving the respective Event Id. Finally the quit program this option terminates the program.

i

Chapter 2

SYSTEM REQUIREMENTS 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 Event Managment System are stated as follows:

2.1 Software Requirements

- **Software's used:**
  - o Operating System – Windows OS
  - o Back End – Turbo C++ compiler
- **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 by Bjarne Stroustrup 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 previously to have been compiled into a machine language program. A compiler is a special program that processes statements written in a particular programming language and turns them into machine language or "code" that a computer's processor uses. Typically, a programmer writes language 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 any language, 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 very strictly 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, the test 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 wants the 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 as well as the implicit requirements desired by the customer.

3.1 Operations Performed on a File

- **Insertion:**

The system is initially used to add products containing product id, name and price into the file. Records with duplicate product id fields are not allowed to be inserted. The length of the product name is checked to see whether it contains less than 50 characters.

- **Display:**

The system can then be used to display existing records. The records are displayed based on the way we inserted. It contains product id, name and price of the product**.**

- **Search:**

The system can then be used to search for existing records. The user is prompted for a product ID. If product ID is matched then entire details of the existing record will be displayed. If product 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 item or all items in his cart. The user is prompted for a product 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 product 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 Diagrams:

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-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyse an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually "say" things that would be hard to explain in words, and they work for both technical and 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 took off 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 analyse and design an application or system.

- Structured Systems Analysis and Design Method (SSADM), a waterfall method to analyse and design information systems. This rigorous documentation approach contrasts with modern agile 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.

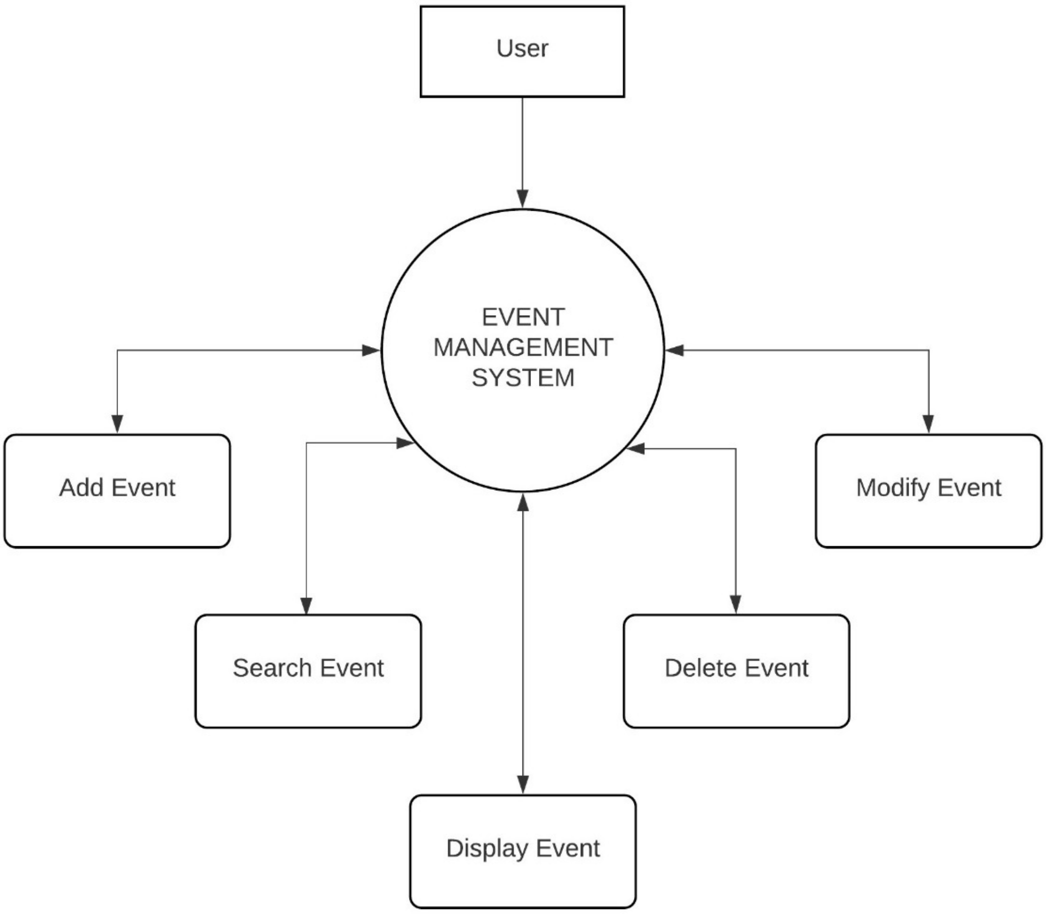


Fig.3.1: Zero level DFD

The above figure shows zero level data flow diagram of sales management system. Zero level data flow diagrams concentrate mainly on overview of the whole system or process being analysed 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 and its constraints on implementation, design of methods to achieve the changeover, an evaluation of 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 possible hazards.

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 finds the 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

- Add an event
- Search event details
- Display all events
- Delete an event
- Modify an event
- Quit program

4.2 Code Snippets

4.2.1 Main Screen

```
cout<<"MAIN MENU"

cout<<"*";

for(i=0;i<50;i++) cout<<"*";

cout<<"* "<<"1. Add an event"<<endl;

cout<<"* "<<"2. Search event details"<<endl;

cout<<"* "<<"3. Display all event"<<endl;

cout<<"* "<<"4. Delete an event "<<endl;

cout<<"* "<<"5. Modify an event"<<endl;

cout<<"* "<<"6. Quit Program\n";

for(i=0;i<50;i++) cout<<"*";
```

4.2.2 Add an Event

```
cout<<"ADD AN EVENT";

for(i=0;i<50;i++) cout<<"*";

cout<<"ENTER EVENT DETAILS:"<<endl;

cout<<"\tEnter the event id:(ev__)"; gets(eid);

cout<<"\tEnter the event name:"; gets(ename);

cout<<"\tEnter the time of conduct:"; gets(toc);

cout<<"\tEnter the registeration fee:"; gets(rfee);

cout<<"\tEnter the prize money:"; gets(pamt);

strcpy(buffer,eid);        strcat(buffer,"|");

strcat(buffer,ename);      strcat(buffer,"|");

strcat(buffer,toc);        strcat(buffer,"|");
strcat(buffer,rfee);           strcat(buffer,"|");

strcat(buffer,pamt);          strcat(buffer,"|");
```

4.2.3 Search event details

```
void student::retrieve(int addr,char k[])
{
 int found=0,i;
 char dummy[10];
 i=addr;
 file.open(studentfile,ios::in|ios::out);
 do
 {
           file.seekg(i*recsize,ios::beg);
           file.getline(dummy,5,'\n');
           if(strcmp(dummy,"####")==0)
           break;
           file.seekg(i*recsize,ios::beg);
           file.getline(eid,15,'|');
           if(strcmp(eid,k)==0)
           {
                    found=1;
                    textcolor(RED);
                    cout<<"\n";
```

```
          gotoxy(20,12);
          cout<<"RECORD FOUND!!\n";
          cout<<endl;
          file.getline(ename,20,'|');
           file.getline(toc,10,'|');
          file.getline(rfee,8,'|');
          file.getline(pamt,8,'|');
          gotoxy(20,14);
          cout<<"ID:"<<eid<<endl;
          gotoxy(20,15);
          cout<<"NAME:"<<ename;
          gotoxy(20,16);
          cout<<"Time Of Conduct:"<<toc;
          gotoxy(20,17);
          cout<<"Registeration Fee:"<<rfee;
          gotoxy(20,18);
          cout<<"Prize Amount:"<<pamt;
          break;
}
else
{
          i++;
          if(i%max==0)
```

```
                    i=0;
            }
 } while(i!=addr);
 if(found==0)
 cout<<"\n\t\t\trecord does not exist in hash file\n";
```

4.2.4 Display all events

```
void student::display()
{
   clrscr();
 char dummy[80];
 ifile.open(datafile,ios::in);
 cout<<setiosflags(ios::left);
 cout<<"\t\t\t\tEVENT DETAILS"<<endl;
cout<<"++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++"<<endl;
 cout<<"\t"<<setw(10)<<"ID"<<setw(20)<<"NAME"<<setw(15)<<"CONDUCT
TIME"<<setw(10)<<"FEE"<<setw(10)<<"PRIZE"<<endl;
cout<<"++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++"<<endl;
while(1)
 {
            ifile.getline(eid,15,'|');
            ifile.getline(ename,20,'|');
            ifile.getline(toc,10,'|');

            ifile.getline(rfee,8,'|');

            ifile.getline(pamt,8,'|');

            ifile.getline(dummy,80,'\n');

            if(ifile.eof())

                    break;

            if(eid[0]!='$')

            {

           cout<<"\t"<<setw(10)<<eid<<setw(20)<<ename<<setw(15)<<toc<<setw(10)<<rfee<
<setw(10)<<pamt<<"\n";

             }

}
```

4.2.5 Delete an event

```
void student::edelete(int addr,char k[])

{

 int found=0,i,j,fn;

 char dummy[10],temp[80];

 i=addr;

 ifile.open(datafile,ios::in|ios::out);

 while(!(ifile.eof()))

 {      fn = ifile.tellg();

           ifile.getline(eid,15,'|');
```

```
          ifile.getline(temp,80,'\n');
          if(strcmp(eid,k)==0)
          {
                    ifile.seekg(fn,ios::beg);
                    ifile.put('$');
          }
}
ifile.close();
file.open(studentfile,ios::in|ios::out);
do
{
          file.seekg(i*recsize,ios::beg);
          file.getline(dummy,5,'\n');
          if(strcmp(dummy,"####")==0)
          break;
          file.seekg(i*recsize,ios::beg);
          file.getline(eid,15,'|');
          if(strcmp(eid,k)==0)
          {
                    found=1;
                    cout<<"\n\t\t\t\tRECORD FOUND!!\n";
                    cout<<endl;
                    file.getline(ename,20,'|');
```

```
                cout<<"\t\t\t\tkey="<<eid<<endl<<"\n\t\t\t\tname="<<ename<<"\n";
                file.seekg(i*recsize,ios::beg);
                for(j=0;j<recsize-2;j++)
                        file<<"#";
                file<<endl;
                break;
        }
        else
        {
                i++;
                if(i%max==0)
                i=0;
        }
} while(i!=addr);
if(found==0)
cout<<"\n\t\t\trecord does not exist in hash file\n";
getch();
return;
}
```

4.2.6 Quit program

o Prompts the user to enter option 6 when he/she is in the console window.

o When user enter 6 then the program will be terminated.

Chapter 5

Testing and Debugging

5.1Testing

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 complexity are 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

o 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.

o 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.

○ 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 behaviour of the software is the different from the specified behaviour. 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

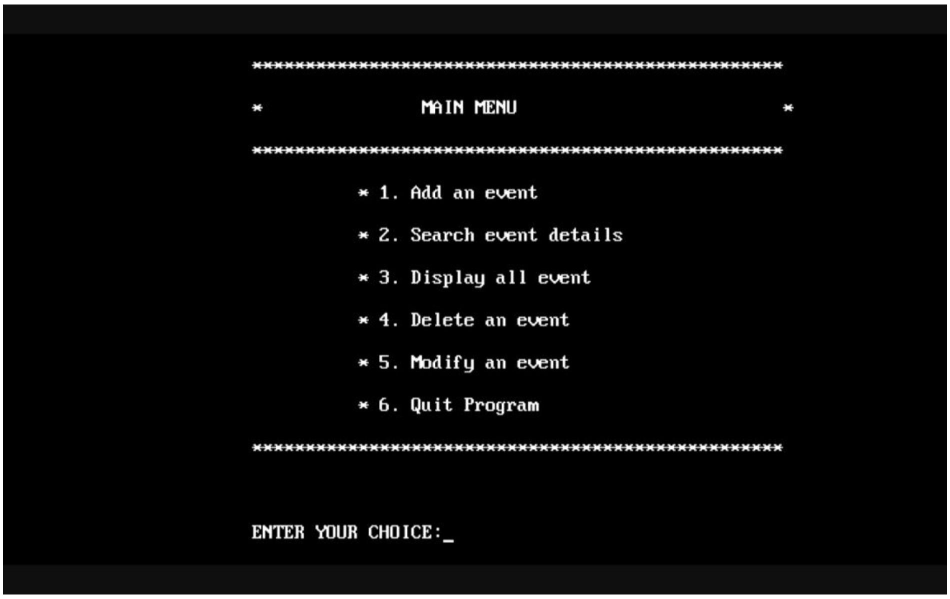
Structure tests are concerned with exercising the internal logic of a program and traversing particular execution paths. Some authors refer collectively to functional performance and stress testing as “black box” testing. While structure testing is referred to as “white box” or “glass box” testing. The 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

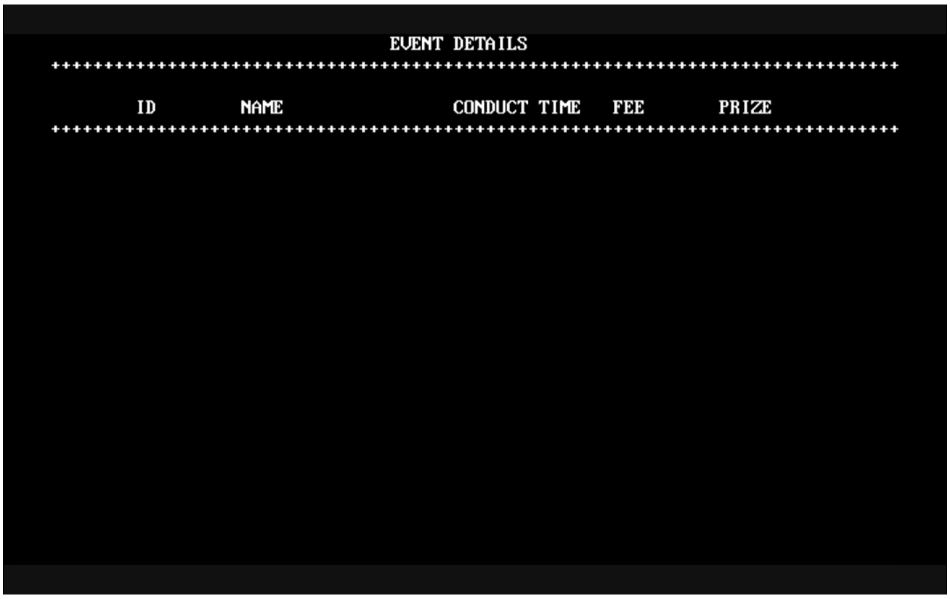
RESULTS



```
**************************************************
*               MAIN MENU                         *
**************************************************
          * 1. Add an event
          * 2. Search event details
          * 3. Display all event
          * 4. Delete an event
          * 5. Modify an event
          * 6. Quit Program
**************************************************


ENTER YOUR CHOICE:_
```

Fig 6.1 Main Menu

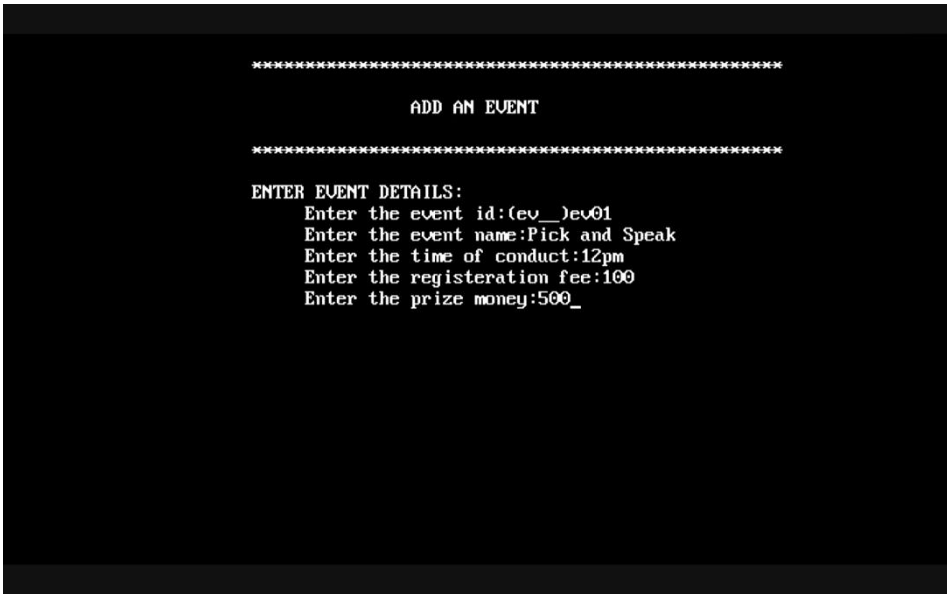
When the user runs the program, Fig 6.1 appears initially giving the user a range of options.



```
                                EVENT DETAILS
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        ID        NAME                CONDUCT TIME   FEE       PRIZE
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Fig 6.2 Event details list initially

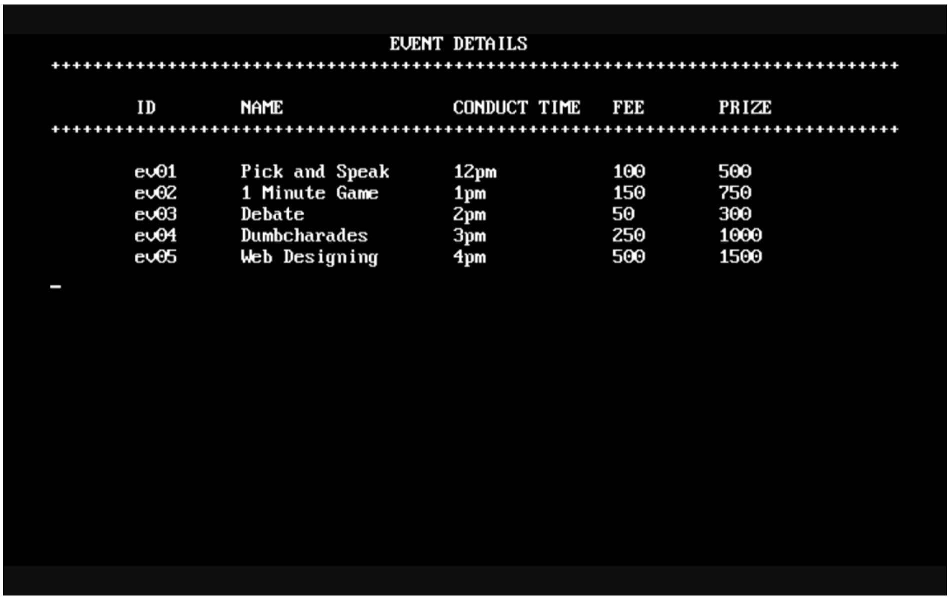
Fig 6.2 shows list of events initially when the user runs the program and hasn’t added an event.



```
*************************************************

                ADD AN EVENT

*************************************************

ENTER EVENT DETAILS:
     Enter the event id:(ev__)ev01
     Enter the event name:Pick and Speak
     Enter the time of conduct:12pm
     Enter the registeration fee:100
     Enter the prize money:500_
```

Fig 6.3 Option 1 in main menu

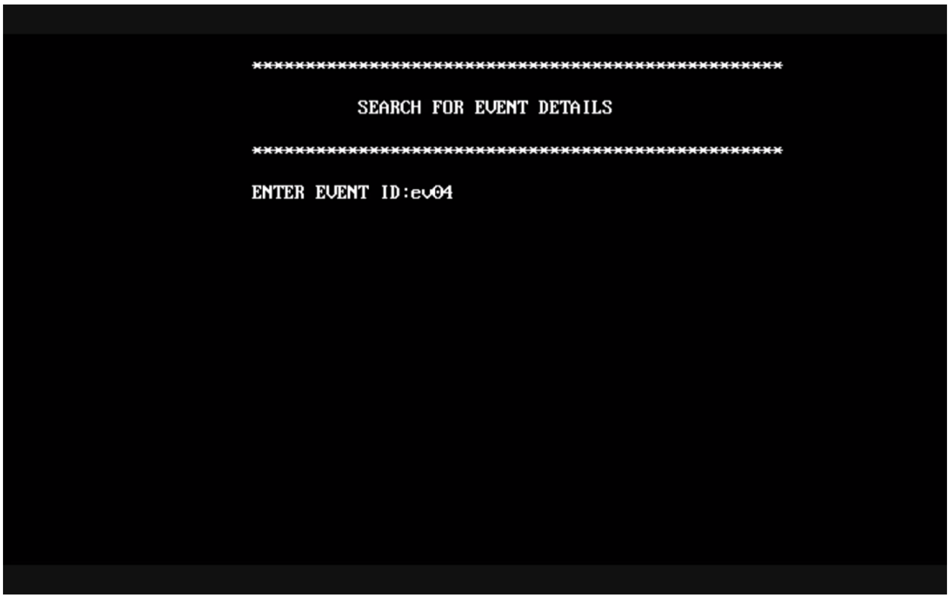
Fig 6.3 appears when the user wants to add an event, by clicking 1 in Main Menu. In addition, user has to give details of event ID, name, time of conduct, registration fee and prize money. Adding an event would store the event details in a file which is used for other functions.



```
                                EVENT DETAILS
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

        ID        NAME                CONDUCT TIME   FEE       PRIZE
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

        ev01      Pick and Speak      12pm           100       500
        ev02      1 Minute Game       1pm            150       750
        ev03      Debate              2pm            50        300
        ev04      Dumbcharades        3pm            250       1000
        ev05      Web Designing       4pm            500       1500
_
```

Fig 6.4 Option 3 in main menu

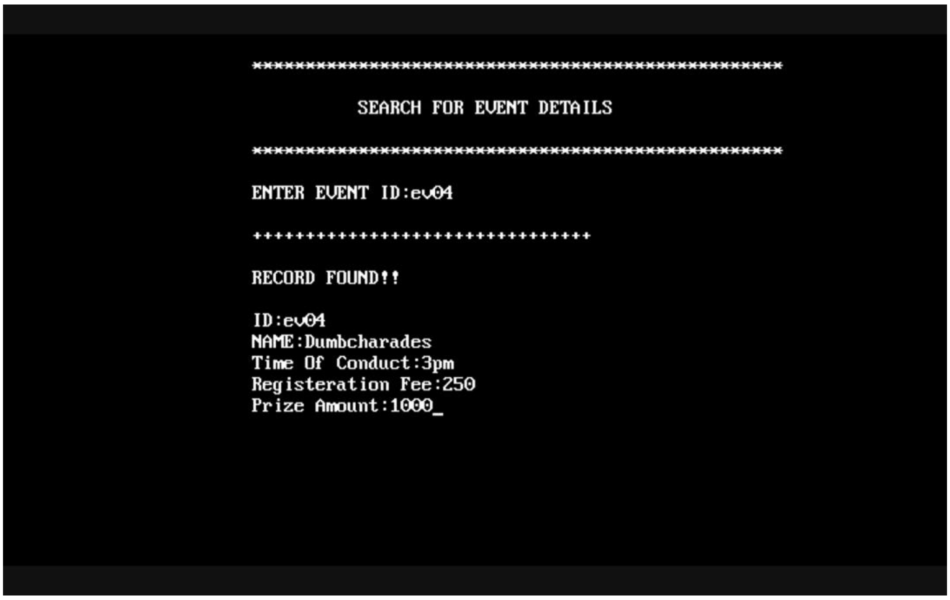
After adding five events, Fig 6.4 is seen when the user selects option 3 from the main menu to display all the events that is currently stored in file.



```
**************************************************

          SEARCH FOR EVENT DETAILS

**************************************************

ENTER EVENT ID:ev04
```

Fig 6.5 Option 2 in main menu

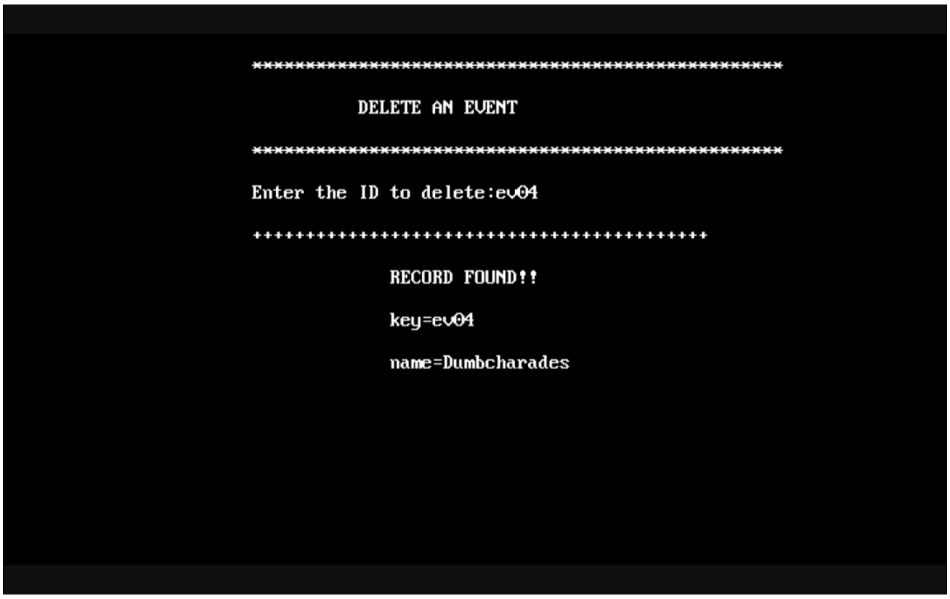
Fig 6.5 is seen on the screen when the user wishes to search for the event details.



```
**************************************************

          SEARCH FOR EVENT DETAILS

**************************************************

ENTER EVENT ID:ev04

++++++++++++++++++++++++++++++++

RECORD FOUND!!

ID:ev04
NAME:Dumbcharades
Time Of Conduct:3pm
Registeration Fee:250
Prize Amount:1000_
```

Fig 6.6 Details of the searched event

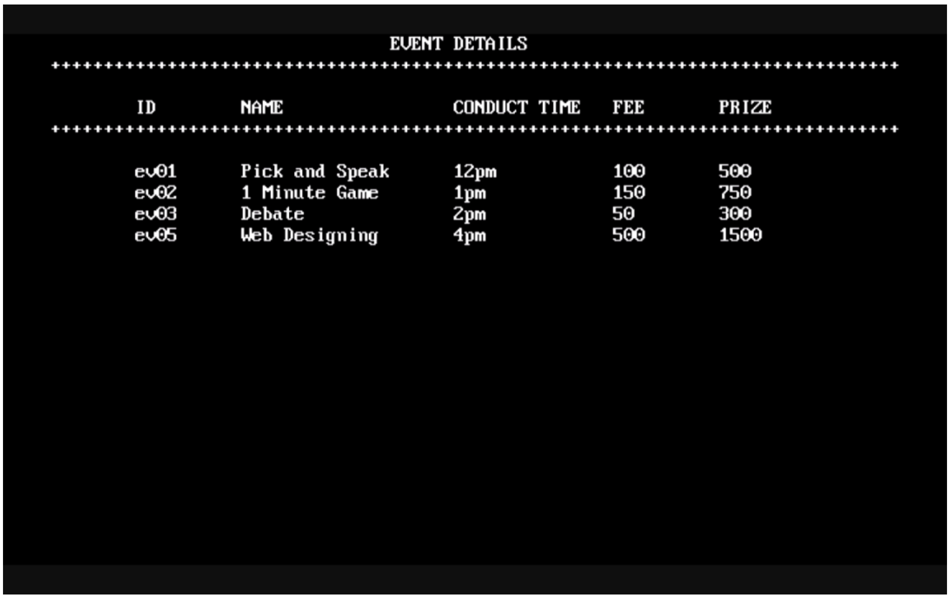
When the user enters the event ID of which the details are too be searched for, Fig 6.6 is seen and all the corresponding details of the event is shown.



```
***************************************************

          DELETE AN EVENT

***************************************************

Enter the ID to delete:ev04

++++++++++++++++++++++++++++++++++++++++++++

             RECORD FOUND!!

             key=ev04

             name=Dumbcharades
```

Fig 6.7 Option 4 in main menu

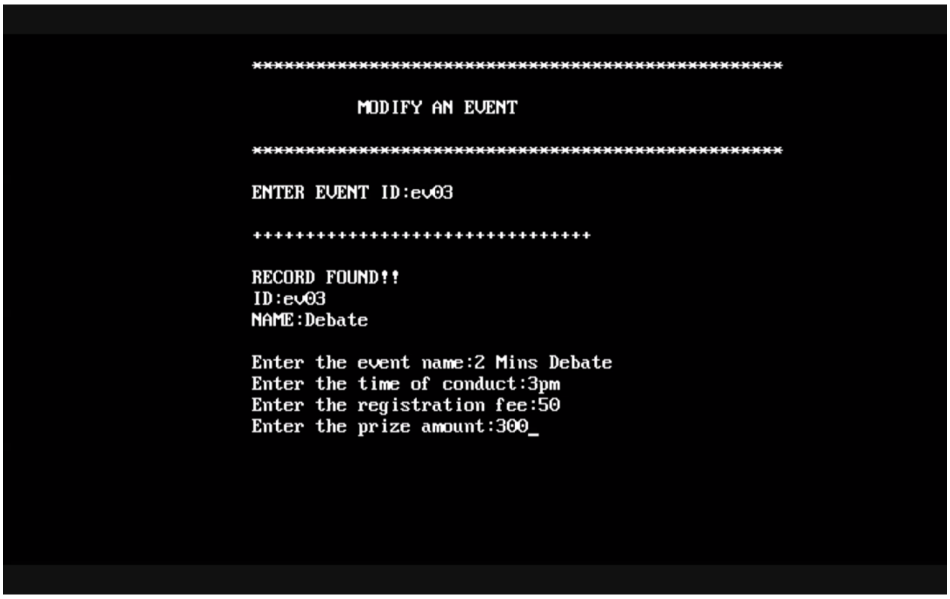
When the user wishes to delete an event, option 4 is selected from the main menu and Fig 6.7 is seen where the user enters the ID of the event which has to be deleted.



```
                                EVENT DETAILS
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

        ID        NAME                CONDUCT TIME   FEE       PRIZE
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

        ev01      Pick and Speak      12pm           100       500
        ev02      1 Minute Game       1pm            150       750
        ev03      Debate              2pm            50        300
        ev05      Web Designing       4pm            500       1500
```

Fig 6.8 List of events after deletion

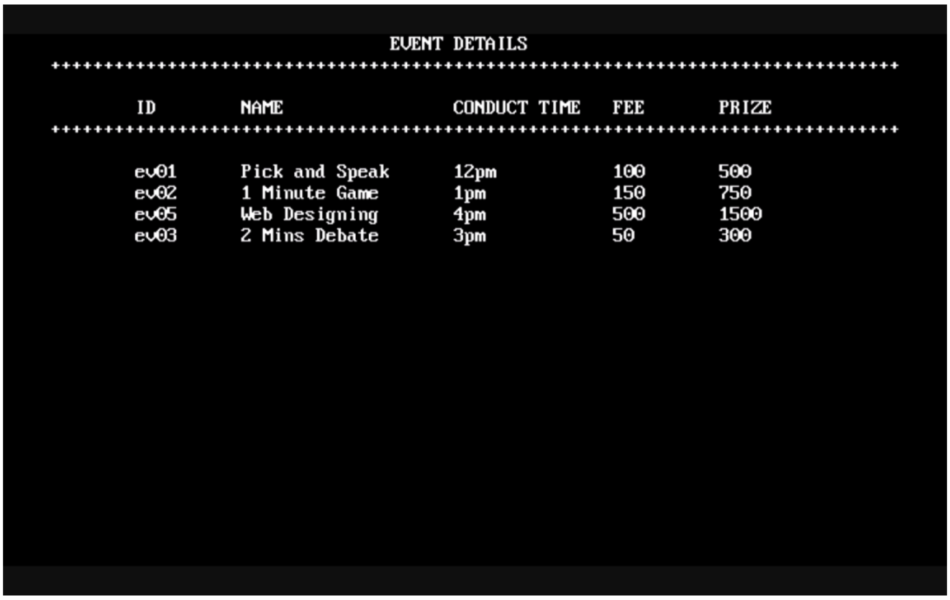
Once the user deletes an event, the event is deleted from the file where data is stored and when the option to display data in the file in file is selected, Fig 6.8 is seen and can be observed that the corresponding event is deleted from the list.



```
**************************************************

          MODIFY AN EVENT

**************************************************

ENTER EVENT ID:ev03

++++++++++++++++++++++++++++++++

RECORD FOUND!!
ID:ev03
NAME:Debate

Enter the event name:2 Mins Debate
Enter the time of conduct:3pm
Enter the registration fee:50
Enter the prize amount:300_
```

Fig 6.9 Option 5 in main menu

When the user wishes to edit the information of any event in the list, option 5 is selected in the main menu and the user is asked to enter the event ID of which the details has to be edited. Fig 6.9 is seen once the ID is entered and the modified details are given.



```
                              EVENT DETAILS
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

        ID        NAME                CONDUCT TIME   FEE       PRIZE
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

        ev01      Pick and Speak      12pm           100       500
        ev02      1 Minute Game       1pm            150       750
        ev05      Web Designing       4pm            500       1500
        ev03      2 Mins Debate       3pm            50        300
```

Fig 6.10 List of events after modification

Once the user finishes modifying the details of an event and selects the option to display the list of events in the file, Fig 6.10 is seen where the details of the respective event is modified and updated in the list.

CONCLUSION AND FUTURE ENHANCEMENT

Conclusion

Event management System project has been successfully developed using Turbo C++ under Windows platform. The User can add events in the list. And can delete event from the list based on event id , user can display, search the events based on event id only. User can also update or modify the entered event.

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/