Chapter 4

IMPLEMENTATION

Implementation is the process of defining how the system should be built, ensuring that it is operational and meets quality standards. It is a systematic and structured approach for effectively integrating a software-based service or component into the requirements of end users.

4.1 About C++

Classes and Objects

A file is declared as a student object with the USN, and the name of the student and the company along with the package which is offered. An object of this class type is used to store the values entered by the user, for the fields represented by the above data members, to be written to the data file.

Dynamic Memory Allocation and Pointers

Memory is allocated for nodes of the B-Tree dynamically, using the method malloc(), which returns a pointer (or reference), to the allocated block. File handles used to store and retrieve data from files, act as pointers. The free() function is used to release memory, that had once been allocated dynamically. malloc() and free() are defined in the header file alloc.h.

File Handling

Files form the core of this project and are used to provide persistent storage for user entered information on disk. The open() and close() methods, as the names suggest, are defined in the C++ Stream header file fstream.h, to provide mechanisms to open and close files. The physical file handles used to refer to the logical filenames, are also used to ensure files exist before use and that existing files aren't overwritten unintentionally.
The 2 types of files used are data files and index files open() and close() are invoked on the file handle of the file to be opened/closed open() takes 2 parameters- the filename and the mode of access close() takes no parameters.

Character Arrays and Character functions

Character arrays are used to store the USN fields to be written to data files and stored in a B- Tree index object. They are also used to store the ASCII representations of the integer address, that are written to the data file, and, the starting byte offsets of data records, to be written to the index file. Character functions are defined in the header file ctype.h. Some of the functions used include:

- toupper() – used to convert lowercase characters to uppercase characters.
- itoa() – to store ASCII representations of integers in character arrays
- isdigit() – to check if a character is a decimal digit (returns non-zero value) or not (returns zero value)
- isalpha() - to check if a character is an alphabet (returns non-zero value) or not (returns zero value)
- atoi() – to convert an ASCII value into an integer.
- atof() – converts an ASCII value into a floating-point number.

4.2 Pseudocode

4.2.1 Insertion Module Pseudocode

The insertion operation is implemented by append() function which adds index objects to the B^+ TREE if the section_no entered is not a duplicate. Values are inserted into a node until it is full, after which the node is split and a new root node is created for the 2 child nodes formed. It makes calls to other recursive and non-recursive functions.

void append() {

```
    pms aids;  int flag=1, pos,k;
   fstream file.open("pms.txt",ios::app);              //open a file in append mode
   aids.Input(0);
   for(i=indsize;i>0;i--){
          if(strcmp(aids.aid,id[i-
   1].aid)<0)
          id[i]=id[i-1];
          else
                  break;
            }
   file.seekp(0,ios::end);                              //seeking file pointer
pos=file.tellp();                                       //putting file pointer position to pos
k=file.tellg();
```

```
 flag=s.Insert(aids.aid,pos);
strcpy(id[i].aid,aids.section_no);              //section no. in index file
itoa(k,id[i].addr,10);
indsize++;
if(flag && aids.Pack(file))                     //condition to check for flag
        cout << "\n\t Done...\n";
else
        cout << "\n\t Failure.";   file.clear();
file.close();
}
```

4.2.2 Display Module Pseudocode

The traversal() function traverses the B^+ TREE in ascending order accessing each index object stored at each node. The byte offsets in the objects are used to retrieve and display the corresponding data record fields.

```
void display(node * p){
int i,j;
 pms aids;
 if(p->isLeaf())  {                             //check for the pointer
            fstream file("pms.txt",ios::in);    //open file in input mode
           if(file.fail()) {
                      gotoxy(24,10); cout<<"!!! ...The File Is Empty... !!!";
                    getch();
                     return;
}
cout<<"COUNT : "<<p->cnt;                       //displaying the count
for(i=0;i<p->cnt;i++) {
                      block * t=p->ptr[i];
for(j=0;j<t->cnt;j++){
           file.seekg(t->disp[j],ios::beg);
```

```
            if(aids.Unpack(file)){
            aids.Display();
            cout<< "\n\t\t\t\t Press any key ...\n"; getch();
                        }
                else break;
                        }
            }
  file.clear();
  file.close();
   }
   for(i=0;i<p->cnt;i++)
                    if(p->dptrs[i]!=0)
                                    display(p->dptrs[i]);
}
```

Void

person::Display()

```
{
        cout << "\n\n\t Section number        : " << section_no  << "\n\n\t Chapter number
        : " << chapter_no << "\n\n\t Offence   : " << offence
        << "\n\n\t Punishment : " << punishment << "\n\n\t Congizance    : " <<
        congizance << "\n\n\t Trial by : "<<trial_by;
}
```

4.2.3 Search Module Pseudocode

The search() function traverses the B^+ TREE, based on the values of objects in the root node.

void search(char *key) {

```
  student aids;
```

```
 int found=0,i;
block *dp;
 fstream file("student.txt",ios::in);
 file.seekg(ios::beg);
 dp=bt.search(key,found);                    //search function to find the key value
 if(found==0)
             cout<<"\n\n\t Record not found...!\n";
            else{
                        found=0;
                         for(i=0;i<dp->cnt;i++)
           if(strcmp(key,dp->keys[i])==0){
            found = 1;
             file.seekg(dp->disp[i],ios::beg);
             aids.Unpack(file);                          //file unpack
            cout<<"\n\n\t Record found : ";
                                aids.Display();                //display the contents
                        }
   if(found==0)
            cout<<"\n\n\t Record not found ";
  }
  file.clear();
file.close();
}
```

4.2.4 Deletion Module Pseudocode

The delrec() function deletes values from nodes and balances the B^+ TREE after, by merging or redistributing objects in the nodes. It makes calls to other recursive and non recursive functions.

```
void delrec(char *key){

 int r=0,found=0,s;

char del='N';
pms aids[100],aid;
fstream file("pms.txt",ios::in);                    //open file in input mode
 file.seekg(0,ios::beg);
while(!file.fail())
if(aids.Unpack(file))
if(strcmpi(aids.aid,key)==0) {                      //if the key is found
found=1;
cout<<" \n Record :";
aids.Display();
 cout<<"\n\n Confirm permanent deletion:[Y/N]";
cin>>del;                                           //take value to be deleted
if(!(del=='Y' || del=='y')) {
        aids[r].Clear();                            //content deleted
        aids[r++].Assign(aids);                     //rearranging
                        }

else

cout<<"\n\n\t Deleted : \n\n";

 }

else {

                    aids[r].Clear();

                aidss[r++].Assign(aids);

        }

file.clear();

if(found==0)
cout<<"\n\n\t Record not found.";                          //if record not found
```

```
 else  {
        file.close();
 file.open("pms.txt",ios::out);
 file.seekp(0,ios::beg);
for(s=0;s<r;s++)
if(!(aidss[s].Pack(file))) continue;                    //pack contents of the file
 file.clear();
}
 file.close();

}
```

4.2.5 Indexing Pseudocode

The B^+ TREE of indexes is written to the index file after every insertion, deletion, and modification operation, using the write()and initial()functions to keep index file up-todate and consistent.

void index::initial() {

```
 indfile.open(indexfile,ios::in)                          //open index file in input mode
if(!indfile){
indsize=0;                                                //file not found
return;
 }

 for(indsize=0;;indsize++){
                    indfile.getline(id[indsize].aid,15,'|');
                    indfile.getline(id[indsize].addr,5,'\n');
                              if(indfile.eof())
                    break;
            }

            indfile.close();

}
```

```
void index::write(){
        opener(indfile,indexfile,ios::out);              //open file in output mode
        for(i=0;i<indsize;i++)
         indfile<<id[i].aid<<"|"<<id[i].addr<<"\n";  //writing the content in output file
        indfile.close();
}
```

Software Testing is the process used to help identify the correctness, completeness, security and quality of the developed computer software. Testing is the process of technical investigation and includes the process of executing a program or application with the intent of finding errors.

4.3 Testing

Software Testing is the process used to help identify the correctness, completeness, security and quality of the developed computer software. Testing is the process of technical investigation and includes the process of executing a program or application with the intent of finding errors.

4.3.1 Unit Testing

1. USN input it checked to see if it is in alphanumeric form: USN should accept integers and alphabets.

Table 4.1 Unit test case for USN Input Check

Sl No. of test case:	1
Name of test:	Check test
Item / Feature being tested:	Input for USN field
Sample Input:	USN='1RN16I@050'. Upon press of ENTER key.
Expected output:	Prompt for USN with 'Enter USN:' with "USN accepts only alphanumeric" message.
Actual output:	'Enter USN:' with "USN accepts only alphanumeric" message displayed.
Remarks:	Test succeeded

2. Name input it checked to see if it contains onlycharacters:

Table 4.2 Unit test case for name Input Check

Sl No. of test case:	2
Name of test:	Check test
Item / Feature being tested:	Input for name field
Sample Input:	Name='ban34'. Upon press of ENTER key.
Expected output:	Prompt for name with 'Enter name:' with "name accepts only characters" message.
Actual output:	'Enter name:' with "name accepts only characters" message displayed.
Remarks:	Test succeeded

3. Company name input it checked to see if it contains onlycharacters:

Table 4.3 Unit test case for company name Input Check

Sl No. of test case:	3
Name of test:	Check test
Item / Feature being tested:	Input for cname field
Sample Input:	cname='ban34'. Upon press of ENTER key.
Expected output:	Prompt for cname with 'Enter depot name:' with "cname accepts only characters" message.
Actual output:	'Enter cname:' with "cname accepts only characters" message displayed.
Remarks:	Test succeeded

4. Package input it checked to see if it contains onlynumeric:

Table 4.4 Unit test case for package Input Check

Sl No. of test case:	4
Name of test:	Check test
Item / Feature being tested:	Input for package field
Sample Input:	package='ban34'. Upon press of ENTER key.
Expected output:	Prompt for package with 'Enter package:' with "package accepts only numeric" message.
Actual output:	'Enter package:' with "package accepts only numeric" message displayed.
Remarks:	Test succeeded

4.3.2 Integration Testing

1. The insertion function is checked to see if a duplicate record is attempted to be inserted and that the data files and secondary index files are correctly updated with the appropriate records. It is also checked to see if validation of input values is performed correctly.

Table 4.5 Integration test case for Insertion module

Sl No. of test case:	1
Name of test:	Check test
Feature being tested:	Insertion module
Sample Input:	USN='1RN16IS001'. Upon press of ENTER key.
Expected output:	USN and all other inputs accepted, followed by 'Record inserted' message and a 'Press any key to go back to Menu' message displayed.
Actual output:	USN and all other fields accepted, followed by 'Record inserted' message and a 'Press any key to go back to Menu' message is displayed.
Remarks:	Test succeeded

Table 4.6 Integration test case for Insertion module

Sl No. of test case:	2
Name of test:	Check test
Feature being tested:	Insertion module
Sample Input:	USN=’1RN16IS001’. Upon press of ENTER key.
Expected output:	“Duplicate Entry Re-Enter the USN Value” followed by a prompt to enter another USN.
Actual output:	“Duplicate Entry Re-Enter the USN Value” followed by a prompt to enter another USN.
Remarks:	Test succeeded

2. The deletion function is checked to see if validation of input values is performed correctly, the correct results are returned based on whether a file contains records or not and whether desired record is present in a file or not, only existing matching records are deleted and that the result of the deletion is reflected in the data and index files.

Table 4.7 Integration test case for Deletion module

Sl No. of test case:	3
Name of test:	Check test
Item / Feature being tested:	Deletion module
Sample Input:	USN=’1RN15IS050’. Upon press of ENTER key.
Expected output:	“Record deleted” message followed by “Press any key to go back to Menu” message displayed.
Actual output:	“Record deleted” message followed by “Press any key to go back to Menu” message is displayed.
Remarks:	Test succeeded

Table 4.8 Integration test case for Deletion module

Sl No. of test case:	4
Name of test:	Check test
Item / Feature being tested:	Deletion module
Sample Input:	USN=’1RN15IS050’. Upon press of ENTER key.
Expected output:	“Record not found” message followed by “Press any key to go back to Menu” message displayed.
Actual output:	“Record not found” message followed by “Press any key to go back to Menu” message is displayed.
Remarks:	Test succeeded

3. The search function is checked to see if validation of input values is performed correctly, correct results are returned based on whether a file contains records or not and whether desired record is present in a file or not.

Table 4.9 Integration test case for Search module

Sl No. of test case:	5
Name of test:	Check test
Item / Feature being tested:	Search module
Sample Input:	cname=’infosys’. Upon press of ENTER key.
Expected output:	“Record found” message followed by list of USN comes under that particular company and prompt to enter USN.
Actual output:	“Record found” message followed by list of USN comes under that particular company and prompt to enter USN.
Remarks:	Test succeeded

Table 4.10 Integration test case for Search module

Sl No. of test case:	6
Name of test:	Check test
Item / Feature being tested:	Search module
Sample Input:	USN=’11’. Upon press of ENTER key.
Expected output:	“Record found” message followed by a message “Successful search”.
Actual output:	“Record found” message followed by a message “Successful search”.
Remarks:	Test succeeded

Table 4.11 Integration test case for Search module

Sl No. of test case:	7
Name of test:	Check test
Item / Feature being tested:	Search module
Sample Input:	USN=’1rn15is050’. Upon press of ENTER key.
Expected output:	“Record not found” message followed by a message “press any key to main menu”.
Actual output:	“Record not found” message followed by a message “press any key to main menu”.
Remarks:	Test succeeded

4.3.1 System Testing

System Testing is a level of the software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system’s compliance with the specified requirements. The application is run to check if all the modules (functions) can be executed concurrently, if each return correct results of the operations performed by them, and if the data and index files are left in consistent states by each module.

Table 4.15 System test case for Placement Statistics System

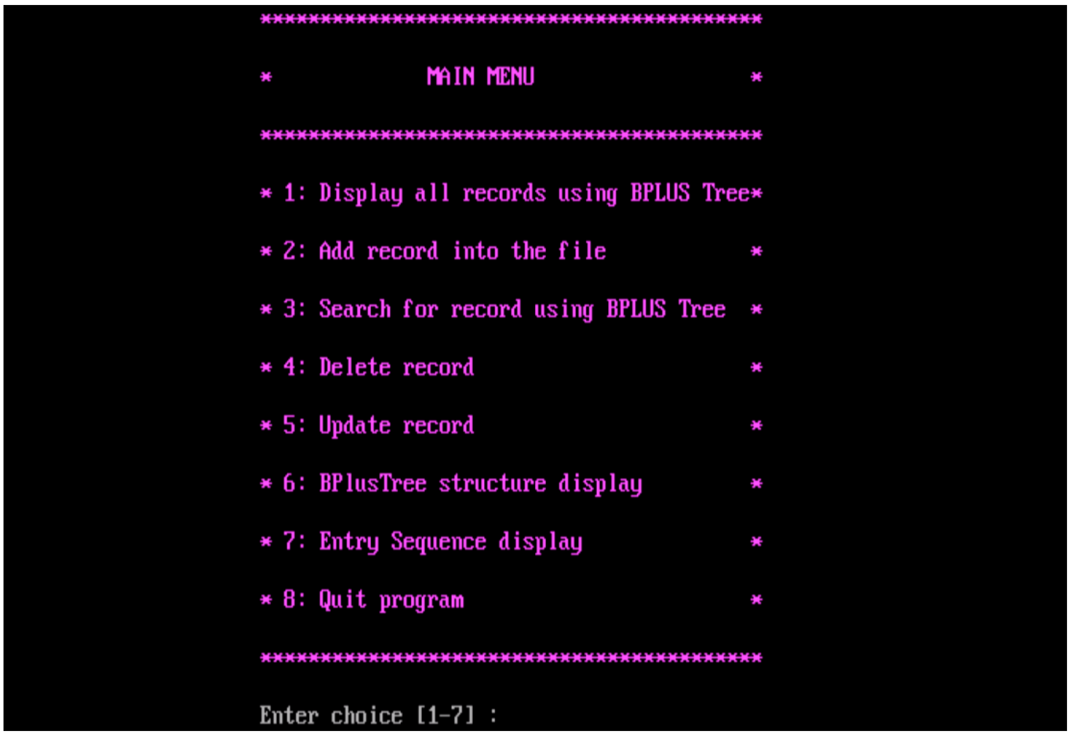
Sl No. of test case:	1
Name of test:	Check test
Item / Feature being tested:	Placement Statistics System
Sample Input:	Choices entered in the order: 1 1 2 3 4 5 2 1
Expected output:	Screens displayed (except menu screen) in order for: Main menu
Actual output:	Screens are displayed in order
Remarks:	Test succeeded

4.4 Discussion of Results

All the menu options provided in the application and its operations have been presented in as screenshots from Figure 4.1 to 4.8

4.4.1 Menu Options

The snapshot shows the menu which has number of options to choose from and gives an idea to the user of the system as shown in Figure 4.1.

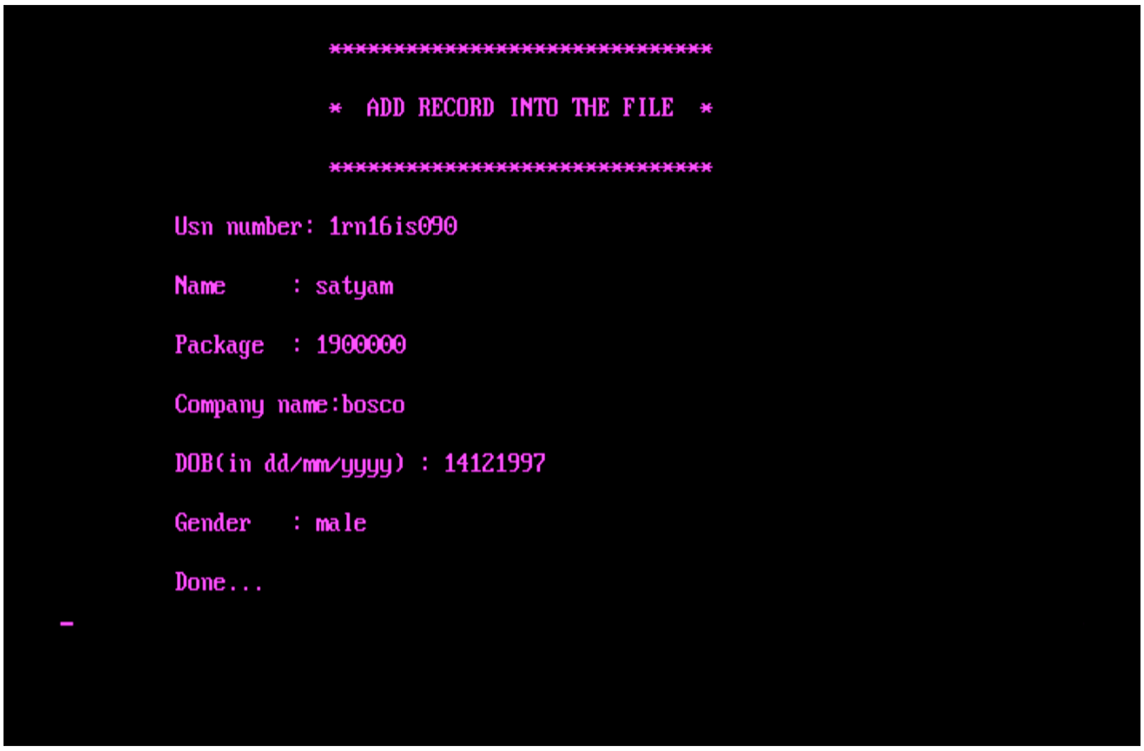


```
*******************************************
*             MAIN MENU                   *
*******************************************
* 1: Display all records using BPLUS Tree*
* 2: Add record into the file             *
* 3: Search for record using BPLUS Tree   *
* 4: Delete record                        *
* 5: Update record                        *
* 6: BPlusTree structure display          *
* 7: Entry Sequence display               *
* 8: Quit program                         *
*******************************************
Enter choice [1-7] :
```

Figure 4.1 User Main menu

4.4.2 Insertion

The snapshot insertion page which gives the data insertion format to the user of the system to insert the new record as shown in Figure 4.2.

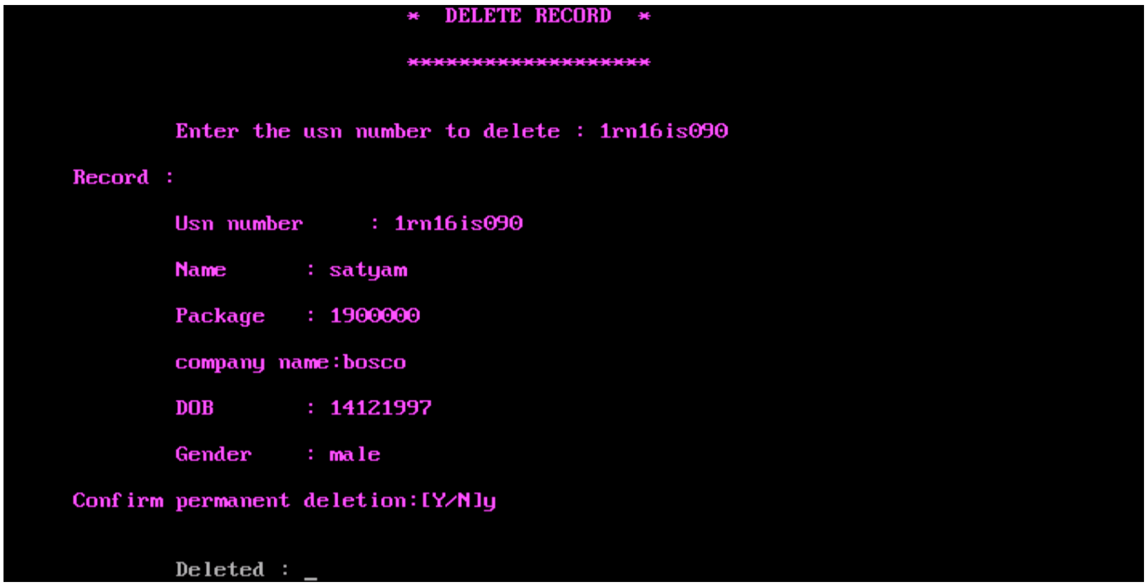


```
                    ******************************

                    *  ADD RECORD INTO THE FILE  *

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

          Usn number: 1rn16is090

          Name     : satyam

          Package  : 1900000

          Company name:bosco

          DOB(in dd/mm/yyyy) : 14121997

          Gender   : male

          Done...
   _
```

Figure 4.2 Insertion of a student record

4.4.3 Deletion

The snapshot is the deletion page to delete the unwanted records. To delete the record we should specify the correct player_id as shown in Figure 4.3.

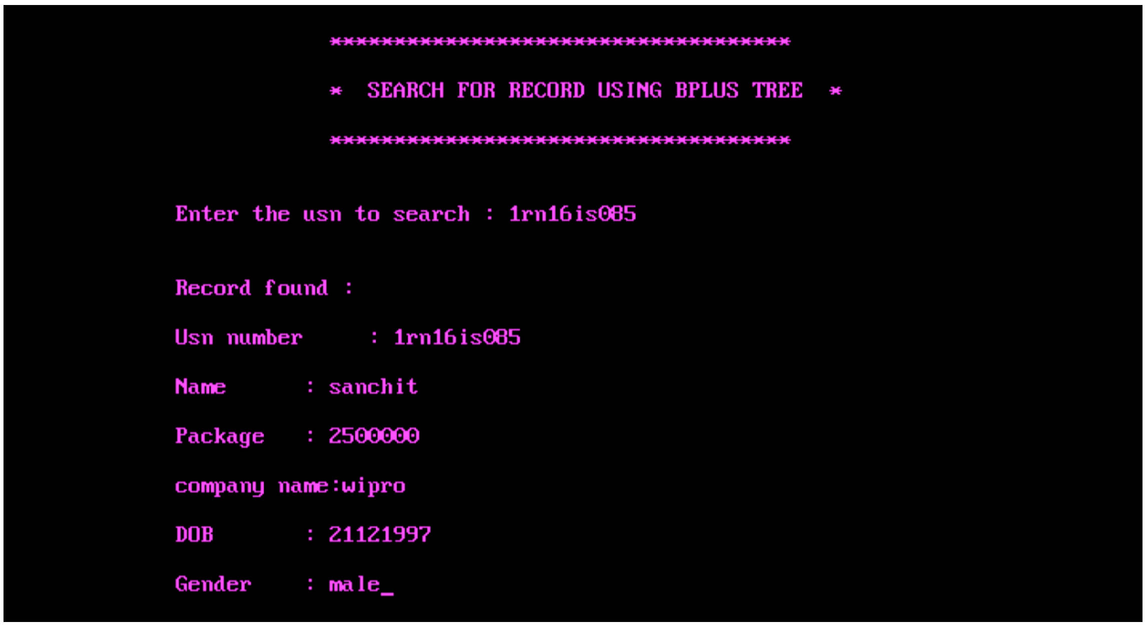


```
                              *  DELETE RECORD  *

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


        Enter the usn number to delete : 1rn16is090

   Record :

        Usn number      : 1rn16is090

        Name       : satyam

        Package    : 1900000

        company name:bosco

        DOB        : 14121997

        Gender     : male

   Confirm permanent deletion:[Y/N]y

        Deleted : _
```

Figure 4.3 Deletion of a record

4.4.4 Searching a Record

The snapshot is the searching page to search the particular record in the data file. To search the particular record we have to provide the correct player_id as shown in Figure 4.4.

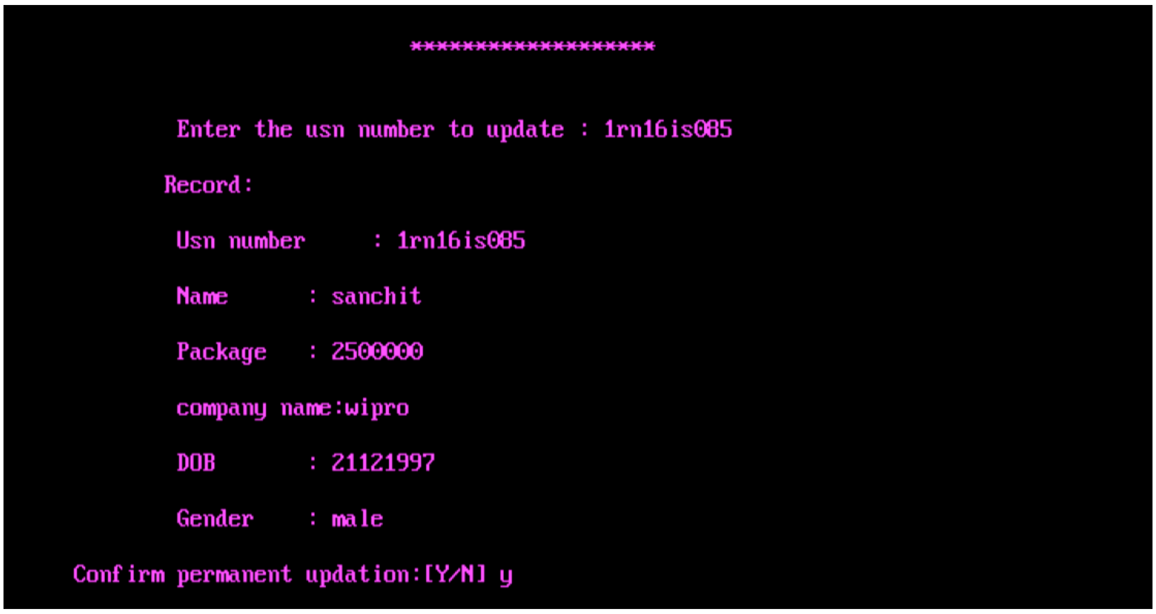


```
                    ************************************
                    *  SEARCH FOR RECORD USING BPLUS TREE  *
                    ************************************

        Enter the usn to search : 1rn16is085

        Record found :
        Usn number      : 1rn16is085
        Name       : sanchit
        Package    : 2500000
        company name:wipro
        DOB        : 21121997
        Gender     : male_
```

Figure 4.4 Searching of a record

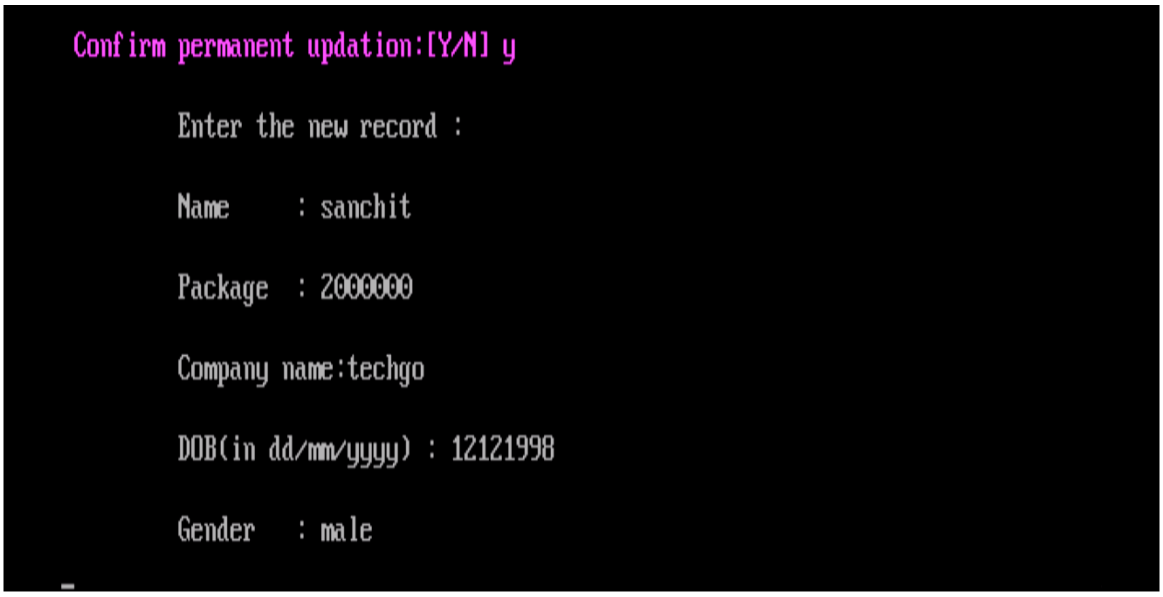
4.4.5 Before and after Modifying a Record

The snapshot is updation page to update the existing record. To update the record we have to provide exact player_id of the record as shown in Figure 4.5 and Figure 4.6.



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

          Enter the usn number to update : 1rn16is085
         Record:
          Usn number      : 1rn16is085
          Name       : sanchit
          Package    : 2500000
          company name:wipro
          DOB        : 21121997
          Gender     : male
     Confirm permanent updation:[Y/N] y
```

Figure 4.5 Before modifying a record

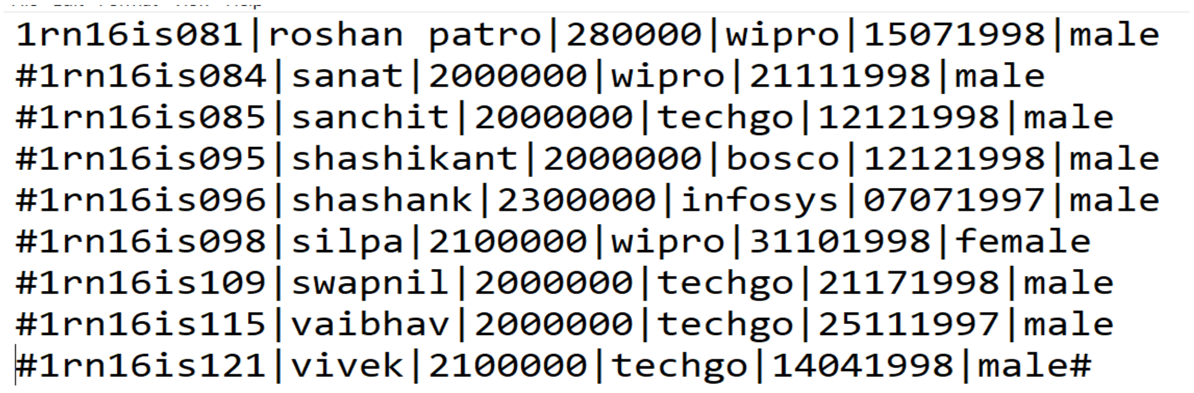


```
Confirm permanent updation:[Y/N] y

        Enter the new record :

        Name     : sanchit

        Package  : 2000000

        Company name:techgo

        DOB(in dd/mm/yyyy) : 12121998

        Gender   : male
```

Figure 4.6 After modifying the record

4.4.6 File Contents

The below snapshot is the data file where it shows the inserted records of the user as shown in Figure 4.7.

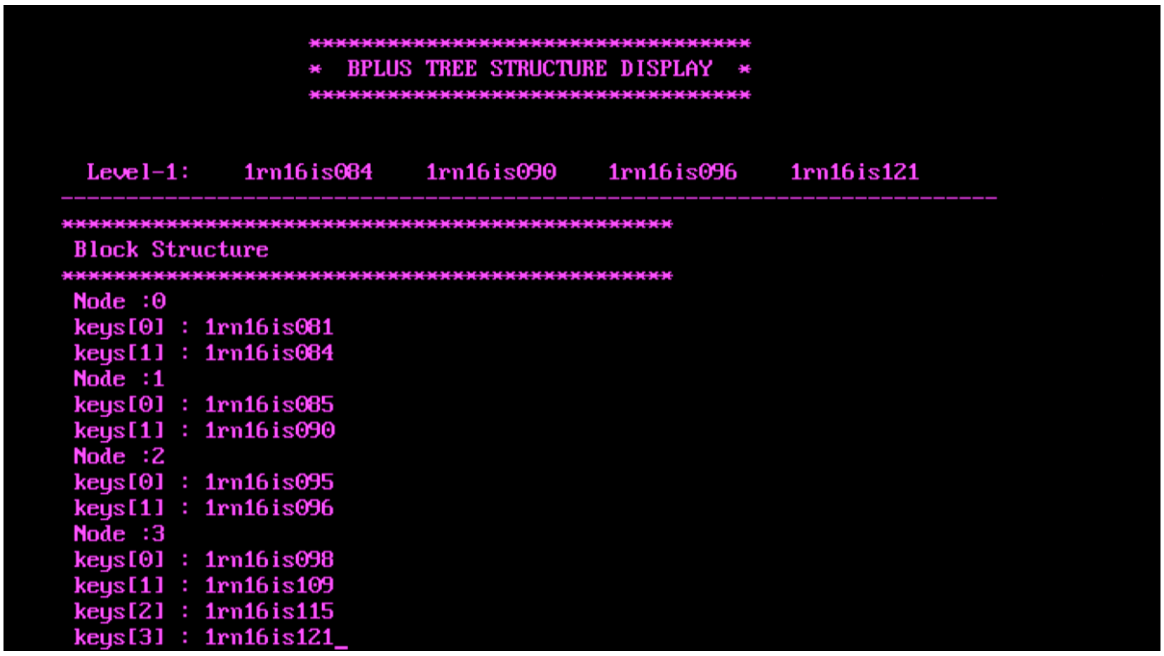


```
1rn16is081|roshan patro|280000|wipro|15071998|male
#1rn16is084|sanat|2000000|wipro|21111998|male
#1rn16is085|sanchit|2000000|techgo|12121998|male
#1rn16is095|shashikant|2000000|bosco|12121998|male
#1rn16is096|shashank|2300000|infosys|07071997|male
#1rn16is098|silpa|2100000|wipro|31101998|female
#1rn16is109|swapnil|2000000|techgo|21171998|male
#1rn16is115|vaibhav|2000000|techgo|25111997|male
#1rn16is121|vivek|2100000|techgo|14041998|male#
```

Figure 4.7 Datafile contents

4.4.7 B$^+$ Tree Contents

The snapshot is the Index file contents it gives the address of the record where it is stored in the memory as shown in Figure 4.8.



```
                    ***********************************
                    *  BPLUS TREE STRUCTURE DISPLAY  *
                    ***********************************


  Level-1:    1rn16is084    1rn16is090    1rn16is096    1rn16is121
--------------------------------------------------------------------------
************************************************
 Block Structure
************************************************
 Node :0
 keys[0] : 1rn16is081
 keys[1] : 1rn16is084
 Node :1
 keys[0] : 1rn16is085
 keys[1] : 1rn16is090
 Node :2
 keys[0] : 1rn16is095
 keys[1] : 1rn16is096
 Node :3
 keys[0] : 1rn16is098
 keys[1] : 1rn16is109
 keys[2] : 1rn16is115
 keys[3] : 1rn16is121_
```

Figure 4.8 B$^+$ Tree Contents