Chapter 3

SYSTEM DESIGN

3.1 Design of fields and records

- usn[15]: the usn field of size 15 holds the usn number of the student which is a unique value that identifies each student.
- name[20]: the name field of size 20, a variable length type char that is used to store the name of the student.
- dept[5]: the dept field of size 5, a variable length type char that is used to store the department to which the student belongs to.
- cname[20]: the cname field of size 20, a variable length type char that is used to store the company name in which the student is placed in.
- package[15]: the package field of size 15, a variable length type char that is used to store the annual package the company is offering.

```
class Student
{
public:
char usn[30],name[30],dept[30],cname[30],package[30];
        Student()
         {
           usn[0]=0;
           name[0]=0;
           dept[0]=0;
           cname[0]=0;
           package[0]=0;

         }
```

3.2 User Interface

The User Interface or UI refers to the interface between the application and the user. Here, the UI is menu-driven, that is, a list of options (menu) is displayed to the user and the user is prompted to enter an integer corresponding to the choice, that represents the desired operation to be performed.

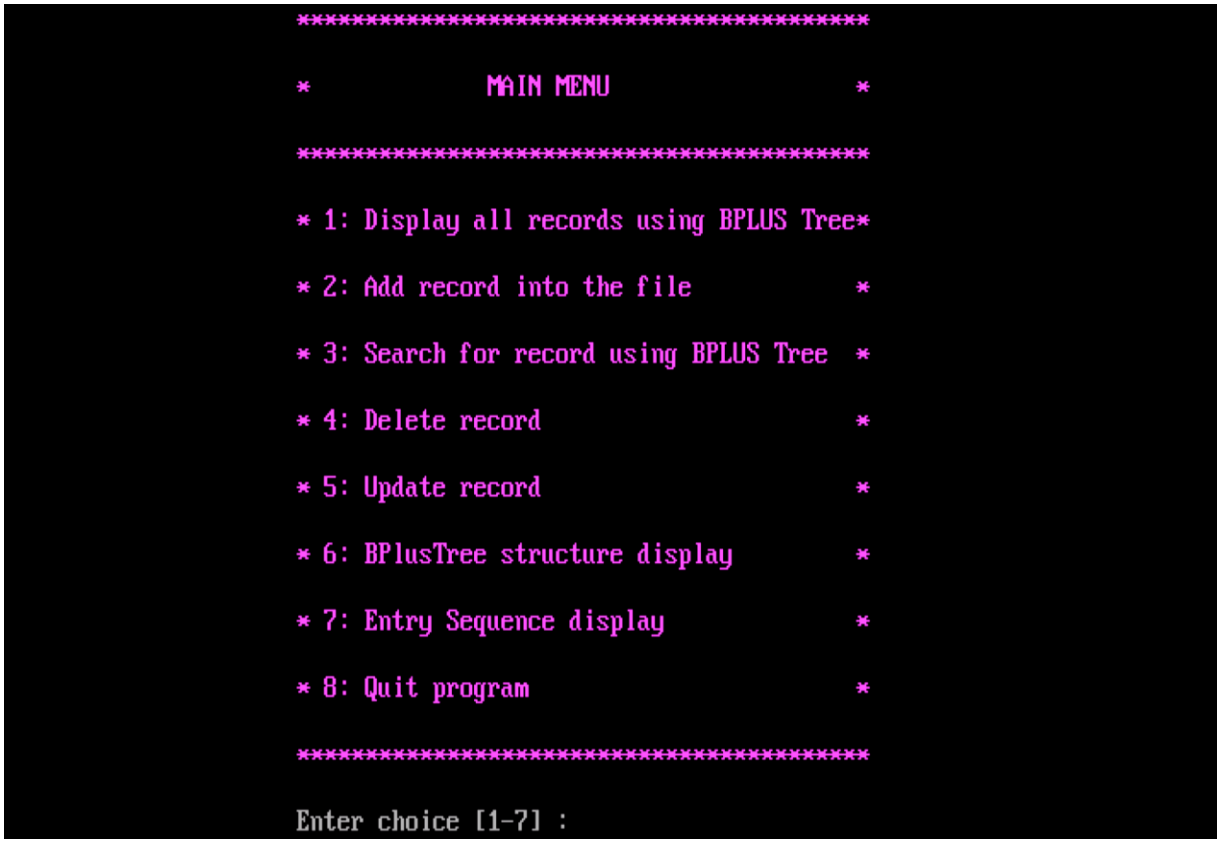

Figure 3.1 User Interface main screen

3.2.1 Insertion of a Record

If the operation desired is Insertion, the user is required to enter 1 as his/her choice, from the menu displayed, after which a new screen is displayed. Next, the user is prompted to enter the USN for which the record is to be inserted. Finally, the user must enter the name, followed by the department, company name in which he/she is placed and the package offered. After all the values are accepted, a "record inserted" message is displayed and the user is prompted to press any key to return back to the menu screen.

3.2.2 Display of a Record

If the operation desired is Display of Records, the user is required to enter 2 as his/her choice, from the menu displayed, after which a new screen is displayed. If there are no records in any file "no records found" message is displayed. For the placement files with at least 1 record, details of each record within the file, with suitable headings, is displayed. In each case, the user is then prompted to press any key to return back to the menu screen.

3.2.3 Update a Record

If the operation desired is Modify, the user is required to enter 3 as his/her choice, from

the menu displayed, after which a new screen is displayed. Next, the user is prompted to enter the usn, whose file from which a record is to be modified. If there are no records in the file, a "no records found" message is displayed, and the user is prompted to press any key to return back to the menu screen. Then prompts would be displayed for updating the value of various fields. After all the values are accepted, a "record inserted" message is displayed and the user is prompted to press any key to return back to the menu screen.

3.2.4 Deletion of a Record

If the operation desired is Deletion, the user is required to enter 4 as his/her choice, from the menu displayed, after which a new screen is displayed. Next, the user is prompted to enter the company name from which the record has to be deleted. If there are no records in the file, a "no records to delete" message is displayed, and the user is prompted to press any key to return back to the menu screen. If there is at least 1 record in the file, the user is prompted for the USN, whose matching record is to be deleted. The USN entered is used as a key to search for a matching record. If none is found, a "record not found" message is displayed. If one is found, a "record deleted" message is displayed. In each case, the user is then prompted to press any key to return back to the menu screen.

3.2.5 Search for a Record

If the operation desired is Search, the user is required to enter 2 as his/her choice, from the indexing menu displayed, after which a new screen is displayed. Next, the user is prompted to enter the secondary key that is company name, whose file the record is to be searched for. If there are no records in the file, a "no records to search" message is displayed, and the user is prompted to press any key to return back to the menu screen. If there is at least 1 record in the file, the user is prompted for the company name, whose matching record is to be searched for. The company name entered is used as a key to search for a matching record. If none is found, a "record not found" message is displayed. If one is found, the details of the record, with suitable headings, are displayed. In each case, the user is then prompted to press any key to return back to the menu screen.

3.2.6 Design of index

The index table is stored in a separate file called the index file.The index file is separate from the encoded file data for the same reasons and to allow us to manage each part separately and simply. The index file starts with a word that encodes flags and the number of pages in the corresponding original data file. We reserve the lower 12 bits for special flags such as whether the index file encodes a file in a 32-bit or a 64-bit file system,

whether fast tails were encoded in this file etc. The very first bit of these flags, and therefore the first bit in the index file, determines if the file encoded is part of a 32-bit or a 64-bit file system. This way, just by reading the first bit we can determine how to interpret the rest of the index file: 4 bytes to encode page offsets on 32-bit file systems or 8 bytes to encode page offsets on 64-bit file systems.

```
void  Student::Index()
{
        int size=0,length=75;
        char uns[15]="0",IBuff[75]="";
        count=0;
        ifstream ofile("list.txt",ios::in);
        ofile.seekg(0,ios::beg);
        fstream nfile("index.txt",ios::out);
        if(ofile.fail())
                cout<<"file not exist";
        else
                while(1)
                 {
                        ofile.read(IBuff,length);
                        if(ofile.fail())
                        break;
                        for(int i=0;IBuff[i]!='|';i++)
                        uns[i]=IBuff[i];
                        count++;
                        nfile.seekp(size);
                        nfile<<uns<<"|"<<count<<"|";
                        size=size+15;
                 }
         nfile.close();
         ofile.close();
}
```