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 system based service or component into the requirements of end users.

4.1 About C++

4.1.1 Classes and Objects

A record file is declared as a Voter ID Management System object with the V_ID, voter name, address as its data members. 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. The B Tree is declared as the class b tree, an object of which represents a node in the B Tree. Each node contains a count of the number of entries in the node and a reference to each descendant. The maximum number of descendants is 4 while the minimum is 2. Each node also has an array of objects, each an instance of the class type index. Each object of type index contains a VNO field and an address field, which stores the ASCII representation of the starting byte address at which the corresponding data record is stored in the data file. Class objects are created and allocated memory only at runtime.

4.1.2 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. Pointers are also data members of objects of type b tree node, and, an object's pointers are used to point to the descendants of the node represented by it. 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.

4.1.3 File Handling

Files form the core of this project and are used to provide for system 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.

4.1.4 Character Arrays and Character functions

Character arrays are used to store the PNO 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 and floating-point fields, that are written into 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).
- 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

The insertion operation is implemented by append() function which adds index objects to the B-Tree if the V_ID entered is not a duplicate. It makes calls to other recursive and non-recursive functions.

```
void append()
{
 student std;
 int flag=1, pos;
 fstream file("student.txt",ios::app);
 std.Input(0);
 file.seekp(0,ios::end);
```

```
  pos=file.tellp();
  flag=s.Insert(std.URN,pos);
  if(flag && std.Pack(file)) cout << "\n\t Done...\n";
  else cout << "\n\t Failure.";
  file.close();
}

int SSET :: Insert(char *val,int disp) // function to insert key in block
{
int i=0;
BK x=FindNode(val);
for(i=0;i<x->cnt;i++)
if(strcmpi(x->keys[i],val)==0)
            {
            cout<< " \nkey "<<val<<" is repeated \n"<<endl;
            return 0;
            }
if(x->cnt < 4)
{
for(i=0;i<x->cnt;i++)
if(strcmpi(x->keys[i],val)>0) break;
if(strcmpi(x->keys[i],val)==0){
cout<< "\n key "<<val<<" is repeated \n"<<endl;
return 0;
strcpy(x->keys[d],val);
x->disp[d]=disp;
x->cnt++;
if(x->cnt < 4)
{
for(i=0;i<x->cnt;i++)
if(strcmpi(x->keys[i],val)>=0) break;
if(strcmpi(x->keys[i],val)==0)
//cout<< " key "<<val<<" is repeated "<<endl;
cout<<"";
else   {
int d=i;
i=x->cnt-1;
while(i>=d)
{
strcpy(x->keys[i+1],x->keys[i]);
x->disp[i+1]=x->disp[i];
i--;
}
strcpy(x->keys[d],val);
x->disp[d]=disp;
x->cnt++;
}  }
bt.create();
}
return 1;
}
```

4.2.2 Display Module

The display() function displays the all the records in the order we have entered by 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;
  student std;
  if(p->isLeaf())
  {
            fstream file("student.txt",ios::in);
           if(file.fail())
{
           gotoxy(24,10);
           cout<<"!!! ...The File Is Empty... !!!";
```

```
            getch();
            return;
}

  cout<<"RECORDS : "<<p->cnt;
  cout<<"\n*******************************************************";
    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(std.Unpack(file))
             {
            std.Display();
            cout<< "\n\t\t\t\t Press any key ...\n";
  cout<<"\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 SSET :: display()            // frunction to display nodes
      {
                        int j=0;
      BK t;
      t=head;
      getch();
      cout<<"**********************************************";
      cout<<"\n Block Structure \n";
      cout<<"**********************************************";
          while(t != 0)
      {
            cout<<"\n Node :"<<j;
            for(int i=0;i<t->cnt;i++)
            {
             cout<<"\n keys["<<i<<"] : " <<t->keys[i];
             //   <<"\t disp["<<i<<"] : "<<t->disp[i]
//       cout<<"\n";
            }
            t=t->link;
            j++;
      }
      }
```

4.2.3 Deletion Module

The remove() function deletes values from the index file. It will ask for the voter name, if that product is present it will display details of that voter and asks to enter the Voter ID and delete that voter. If that voter is not present then record not found message will be displayed.

```
void remove(char *key)
{
 int r=0,found=0,s;
 char del='N';
 student stds[100],std;
 fstream file("student.txt",ios::in);
 file.seekg(0,ios::beg);

 while(!file.fail())
  if(std.Unpack(file))
       if(strcmpi(std.URN,key)==0)
```

```
            {
           found=1;
           cout<<" \n Record :";
           std.Display();
           cout<<"\n\n Confirm permanent deletion:[Y/N]";
           cin>>del;
           if(!(del=='Y' || del=='y'))
           {
             stds[r].Clear();
             stds[r++].Assign(std);
           }
           else
           cout<<"\n\n\t Deleted : ";
            }
            else
            {
           stds[r].Clear();
           stds[r++].Assign(std);
            }
  file.clear();
  if(found==0) cout<<"\n\n\t Record not found.";
  else {
    file.close();
    file.open("student.txt",ios::out);
    file.seekp(0,ios::beg);
    for(s=0;s<r;s++)
           if(!(stds[s].Pack(file))) continue;
  }
  file.close();
}
```

4.2.4 Search Module Pseudocode

The search() function searches the secondary file, based on the secondary key which we have entered, if that voter is present it will display details of that voter and ask to enter the Voter ID and displays the details. If that voter is not present then record not found message will be displayed.

```
void search(char *key)
{
  student std;
  int found=0,i;
  block *dp;
  fstream file("student.txt",ios::in);
  file.seekg(ios::beg);
  dp=bt.search(key,found);
  if(found==0)
            cout<<"\n\n\t Record not found...!\n";
  else
  {
             found=0;
             for(i=0;i<dp->cnt;i++)
              if(strcmpi(key,dp->keys[i])==0)
              {
                       found = 1;
                       file.seekg(dp->disp[i],ios::beg);
             std.Unpack(file);
             cout<<"\n\n\t Record found : ";
             std.Display();
            }
    if(found==0) cout<<"\n\n\t Record not found ";
  }
  file.clear();
  file.close();
}
```

4.2.5 Update Record

A record can be updated using the primary key that is, the primary key. The primary key is entered and all the fields of the record are modified.

```
void update(char *key)
{
 student stds[100],std;
 int f=0,found=0,g;
 char upd='n';
 fstream file("student.txt",ios::in);
 file.seekg(0,ios::beg);
 while(!file.fail())
  if(std.Unpack(file))
          if(strcmpi(std.URN,key)==0) {
   found=1;
           cout<<"\n\tRecord:";
   std.Display();
   cout<<"\n\n Confirm permanent updation:[Y/N] ";
   cin>>upd;
   if(!(upd=='Y' || upd=='y')) {
          stds[f++].Assign(std);
   }
   else  {
          cout << "\n\t Enter the new record :\n";
          stds[f].Clear();
          stds[f++].Assign(std);
   }
          }
  else  {
           stds[f].Clear();
   stds[f++].Assign(std);
  }
  file.clear();
  if(found==0)
   cout<<"\n\n\t Record not found...!\n";
  else
  {
   file.close();
```

```
    file.open("student.txt",ios::out);
    file.seekp(0,ios::beg);
    for(g=0;g<f;g++)
        if(!(stds[g].Pack(file))) continue;
    file.clear();
  }
  file.close();
  }
```

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

It is a level of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc.

In object-oriented programming, the smallest unit is the method, which may belong to base or super class, abstract class or derived or child class. Unit testing frameworks, drivers, stubs, and mock/fake objects are used to assist in unit testing. It is performed by using white box testing method.

Unit testing is the first level of software testing and is performed prior to integration testing. It is normally performed by software developers themselves or their peers. In rare cases, it may be performed by independent software testers.

Table 4.1 Unit test cases for input

Case_id	Description	Input data	Expected o/p	Actual o/p	Status
1	Opening a file to insert the data	Insert option is	File should be opened in append mode	File opened in append	Pass

		selected	without any error messages	mode	
2	Input V_ID	ABC1324345	Accept the V_ID and prompt for Name will be displayed	“Enter the Name:”	Pass
3	Input V_ID	ABC132435	Invalid V_ID and displays the message “V_ID already exist”	“V_ID already exist”	Pass
4	Enter the V_ID: Enter the name: Enterthe address: Enter the dob: Enter the sex:	ABC12367 RAJ Banglore 06/04/1999 Male	It should accept all the fields and it will return to the main menu	“Press any key to return to main menu”	Pass
5	File Updation	-	It should pack all the fields and will be appended to the data file	Data will be updated both in index and data file	Pass
6	Closing a file	-	File should be closed without any error	File is closed	Pass

4.3.2 Integration Testing

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

Table 4.2 Integration testing for all modules

Case_Id	Description	Input	Expected o/p	Actual o/p	Status
1	To display the entered records of the data file	Enter the option 1 in the menu	Should display the record one after the other	Display the record one after the other	Pass
2	To add the new records into the data file	Enter the option 2 in menu	Should display the record entry form	Display the record entry form	Pass
3	To search for particular record in the file	Enter the option 3 in menu and should enter the V_ID:ABC1432437	Should display record not found	Record not found	Pass
4	To search for a particular record in the file	Enter the option 3 in the menu and should enter the V_ID:ABC1234567	Should display the record is found and contents of searched record	The record is found. Contents will be displayed	Pass
5	To delete a particular record in the file	Enter the option 4 in menu and should enter V_ID:2	Should display record not found	Record not found	Pass
6	To delete a particular record in the file	Enter the option 4 in menu and should enter V_ID:ABC1234567	Deletes the record	The record is deleted	Pass

7	To update a particular record in file	Enter the option 5 in menu and should enter V_ID:ABCXYZ1234	Should display record not found	Record not found	Pass
8	To update a particular record in the file	Enter the option 5 in menu and should enter V_ID:ABC1234567	Record is found and allows the user to re-enter the details in the record for updation	Record is found and allows the user to re-enter the details in the record for updation	Pass
9	To display all the entered records in the data file	Enter the option 3 in menu	Displays all the saved records	Display all the saved records	Pass
10	To quit the program	Enter the option 5 in the menu	Exit the program	Exit the program	Pass

4.3.3 System Testing

System testing is defined as testing of a complete and fully integrated software product. This testing falls in black –box testing where in knowledge of the inner design of the code is not a pre-requisite and is done by the testing team. System testing is done after integration testing is complete. System testing should test functional and non-functional requirements of the software. It is the third level of software testing performed after integration tesing and before acceptance testing. It is the process of testing an integrated systems to verify that it meets specified requirements. Usually, black box testing method is used.

Table 4.3 System Testing for voter record details

Case id	Description	Input data	Expected o/p	Actual o/p	Status
1	To display	Display records for	Record is	Record is	Pass

Case id	Description	Input data	Expected o/p	Actual o/p	Status
	the records	valid V_ID=‘ABC1234567’(present)and invalid V_ID =’ABC1223345’(not present)	displayed for valid and record not found for invalid V_ID	displayed for valid V_ID and record not found for invalid V_ID	
2	To search the records	Display records for valid V_ID=‘ABC1234567’(present)and invalid V_ID =’XYZ2225557’(not present)	Record is displayed for valid and record not found for invalid	Record is displayed for valid and record not found for invalid	Pass
3	To delete the records	Delete records for valid V_ID = ‘ABC1234567’(present) and invalid V_ID=’CSK1122334’ (not present)	Record is deleted for valid and record not found for invalid	Record is deleted for valid and record not found for invalid	Pass
4	To update the records	Update records for valid V_ID = ‘ABC1234567’(present) and invalid V_ID=’PSK4455667’ (not present)	Record is Updated for valid and record not found for invalid	Record is Updated for valid and record not found for invalid	Pass

4.3.4 Acceptance Testing

Acceptance Testing is a level of software testing where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery. It is performed by:

- Internal Acceptance Testing (Also known as Alpha Testing) is performed by members of the organization that developed the software but who are not directly involved in the project (Development or Testing). Usually, it is the members of Product Management, Sales and/or Customer Support.

- External Acceptance Testing is performed by people who are not employees of the organization that developed the software.

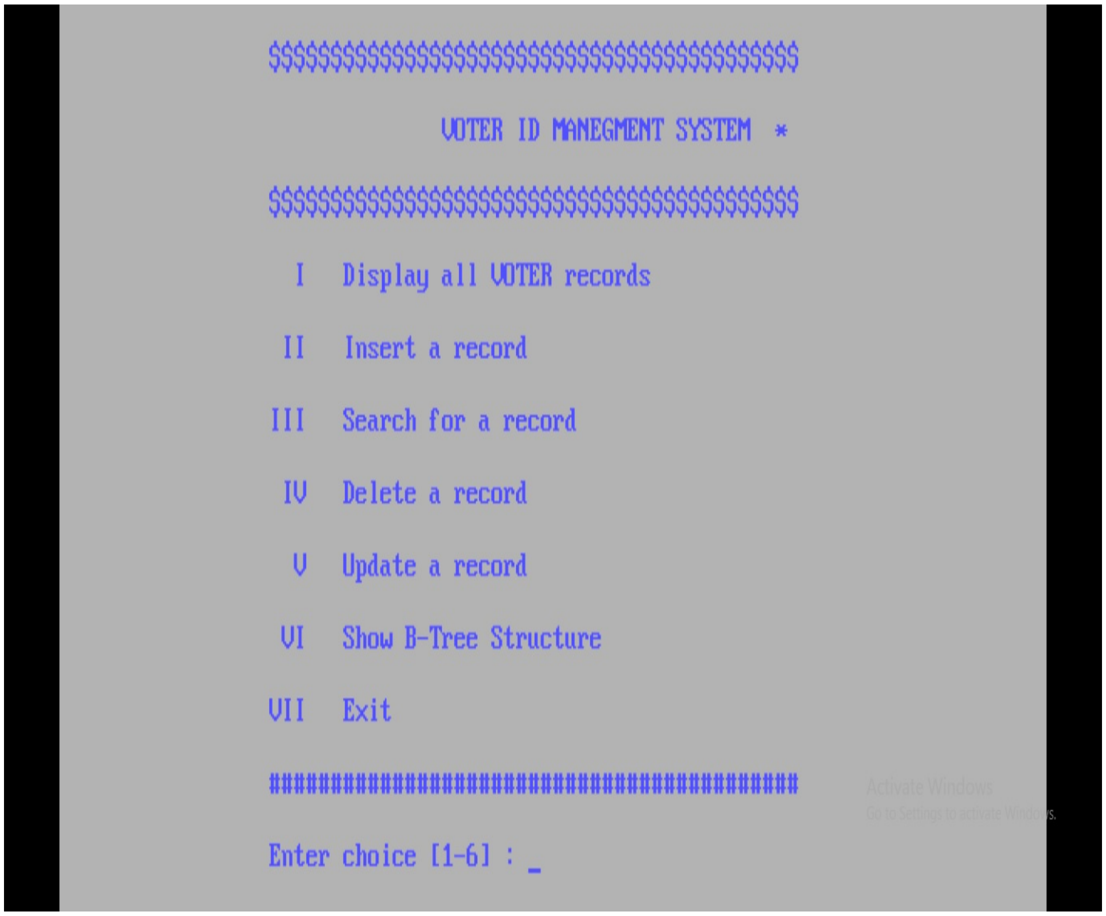
Table 4.4 Acceptance testing for voter record details

Case ID	Description	Input Data	Expected o/p	Actual o/p	Status
1.	Insertion Module	Inserting valid data	Record added successfully	Record added successfully	Pass
2.	Deletion Module	Enter valid VOTER id.	Record deleted successfully	Record deleted successfully	Pass
3.	Modification Module	Enter valid VOTER id.	Record updated successfully	Record updated successfully	Pass

4.4 Discussion of Results

All the menu options provided in the application and its operations have been presented in as screen shots from Fig 4.1 to 4.5

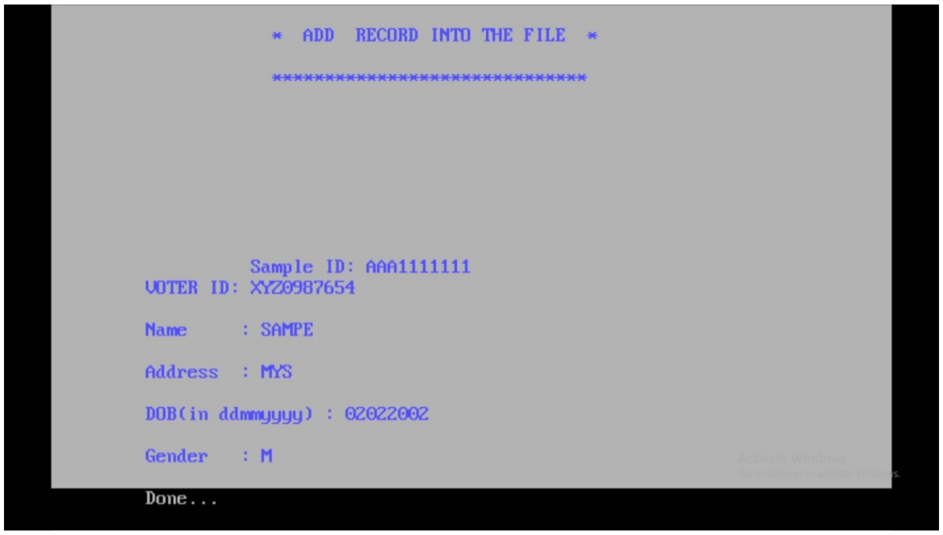
4.4.1 Menu Options: This is the home screen through which user interacts with the system. It gives the user various options to access the records using operations such as insertion, deletion, modification, searching, and display of voter's details. The user can implement insertion, deletion and traversal in the file using b-tree to index the user records. To exit the system at any time, user can press option 7 to return back.



```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

             VOTER ID MANEGMENT SYSTEM  *

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  I   Display all VOTER records

 II   Insert a record

III   Search for a record

 IV   Delete a record

  V   Update a record

 VI   Show B-Tree Structure

VII   Exit

############################################

Enter choice [1-6] : _
```

Figure 4.1 Main menu

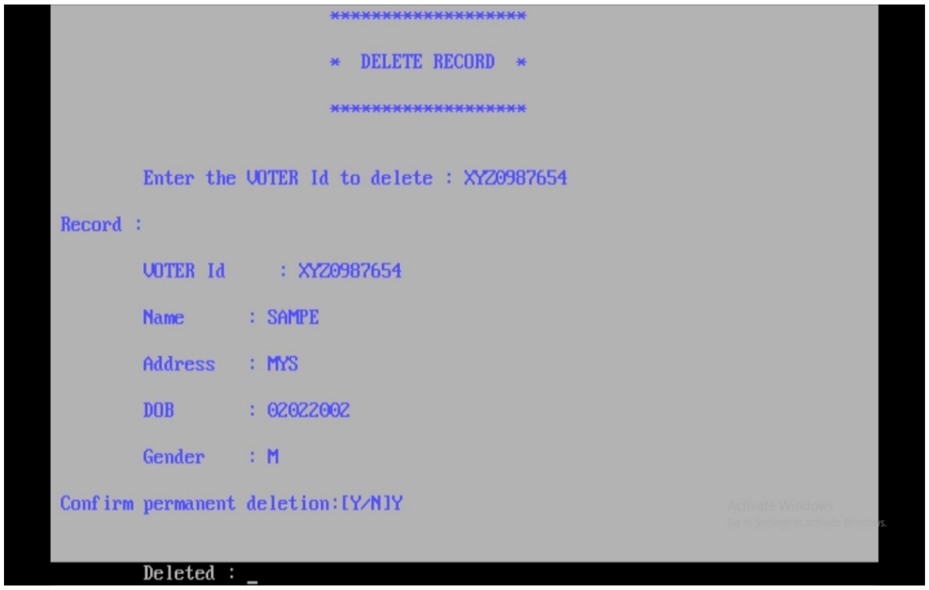
4.4.2 Insertion: Inserting a new voter record. Each voter record should have unique Voter ID as shown in figure 4.2.



```
                 * ADD  RECORD INTO THE FILE  *
                 ******************************




             Sample ID: AAA1111111
    VOTER ID: XYZ0987654

    Name     : SAMPE

    Address  : MYS

    DOB(in ddmmyyyy) : 02022002

    Gender   : M

    Done...
```

Figure 4.2 Insertion of record

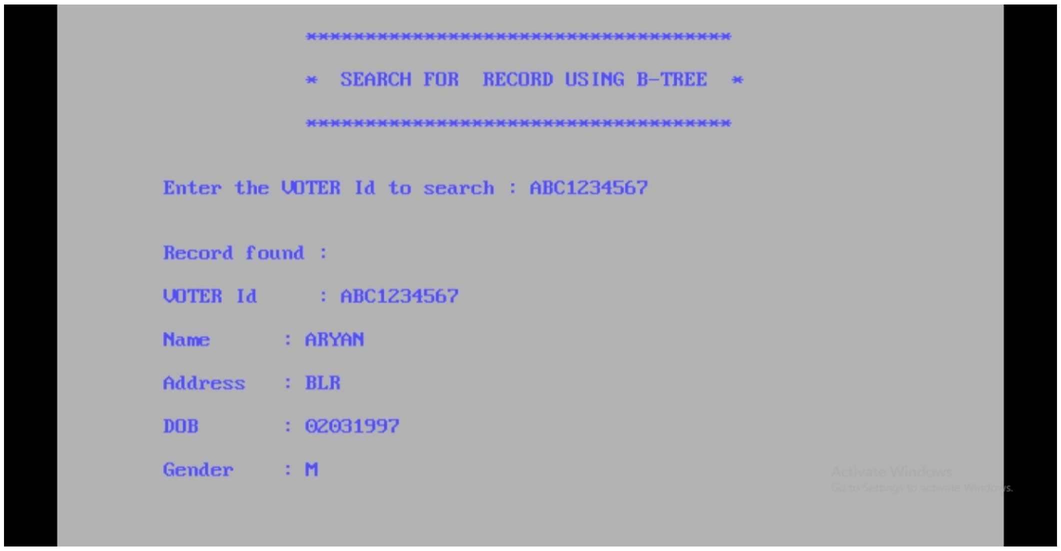
4.4.3 Deletion of a product: Deleting the voter record by giving voter name as input as shown in figure 4.3.



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

                           *  DELETE RECORD  *

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

        Enter the VOTER Id to delete : XYZ0987654

Record :

        VOTER Id     : XYZ0987654

        Name      : SAMPE

        Address   : MYS

        DOB       : 02022002

        Gender    : M

Confirm permanent deletion:[Y/N]Y

        Deleted : _
```

Figure 4.3 Deletion of Record

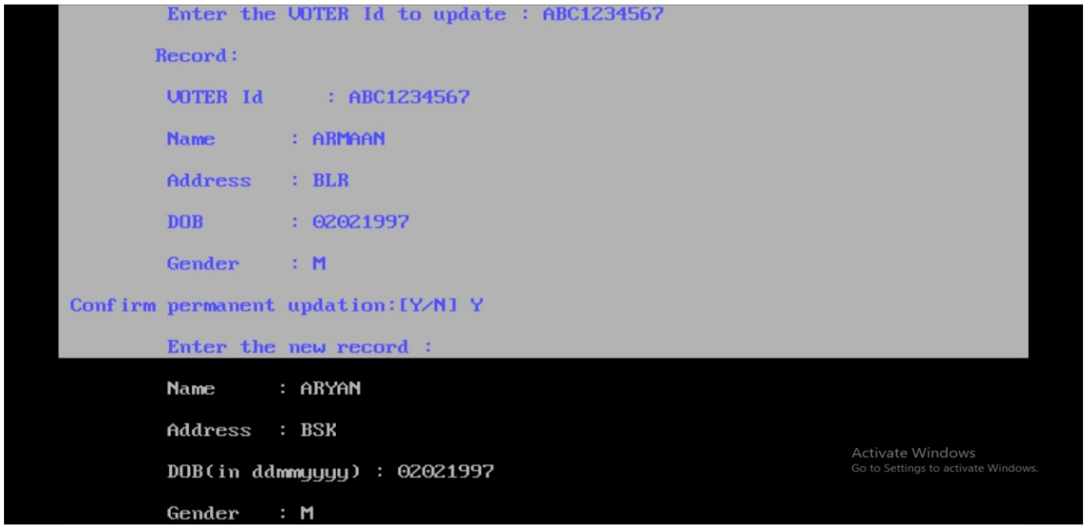
4.4.4 Searching of a Product: Searching the voter record by giving voter's name as searching key as shown in figure 4.4.



```
        ************************************
        *  SEARCH FOR  RECORD USING B-TREE  *
        ************************************

Enter the VOTER Id to search : ABC1234567

Record found :
VOTER Id      : ABC1234567
Name       : ARYAN
Address    : BLR
DOB        : 02031997
Gender     : M
```

Figure 4.4 Searching Record

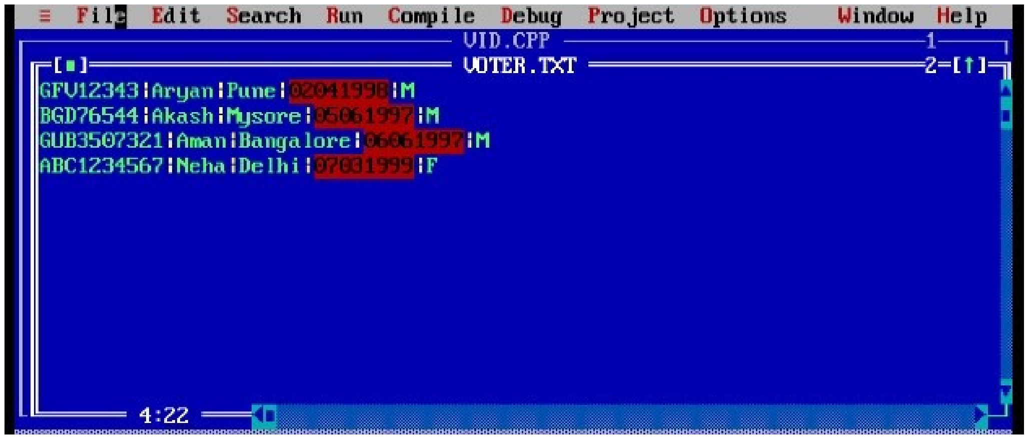
4.4.5 Updating of a Details: Updating the voters details using V_ID as a primary key as shown in figure 4.5.



```
        Enter the VOTER Id to update : ABC1234567
       Record:
        VOTER Id      : ABC1234567
        Name       : ARMAAN
        Address    : BLR
        DOB        : 02021997
        Gender     : M
Confirm permanent updation:[Y/N] Y
        Enter the new record :
        Name      : ARYAN
        Address   : BSK
        DOB(in ddmmyyyy) : 02021997
        Gender    : M
```

Figure 4.5 Updating of a Record

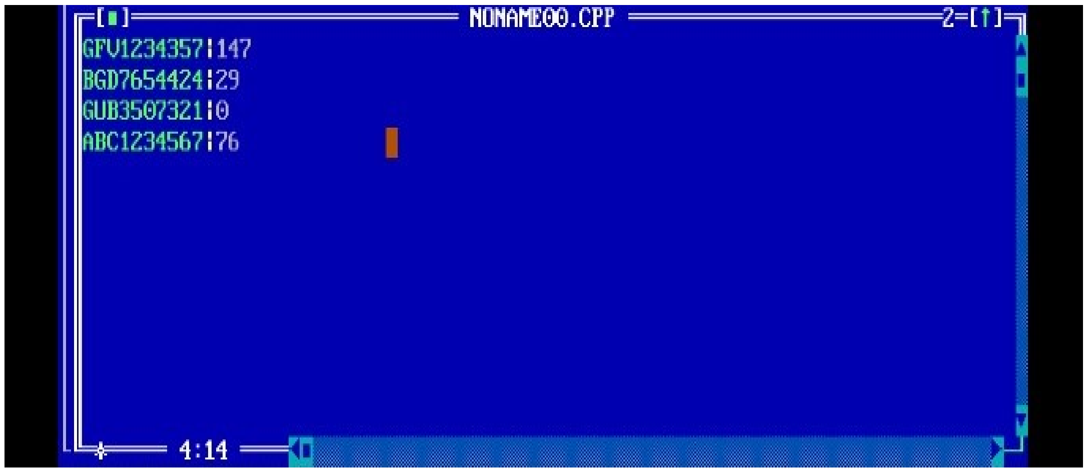
4.4.6 Data file contents: Stored details of voter record in datafile. `$` indicates deleted and modified voter record as shown in figure 4.6.


≡ File Edit Search Run Compile Debug Project Options Window Help
VID.CPP
VOTER.TXT
4:22

```
GFV12343|Aryan|Pune|02041998|M
BGD76544|Akash|Mysore|05061997|M
GUB3507321|Aman|Bangalore|06061997|M
ABC1234567|Neha|Delhi|07031999|F
```

Figure 4.6 Data file details

4.4.7 Index File Contents: Index file contains secondary key and primary reference in it as shown in figure 4.7.



```
GFV1234357|147
BGD7654424|29
GUB3507321|0
ABC1234567|76
```

Figure 4.7 Index file details