ABSTRACT

The main goal of a hospital management system is to design a project that will improve patient care and reduce the expense of running a hospital. It aids in the registration of complete patient data, records and retains the patient’s medical history, treatment needs, past visit details, planned appointments, reports, insurance information, and more.

Hospital Management System is an organized computerized system designed and programmed to deal with day to day operations and management of the hospital activities. The program can look after inpatients, outpatients, records, database treatments, status illness, billings in the pharmacy and labs. It also maintains hospital information such as ward id, doctors in charge and department administering. The major problem for the patient nowadays to get report after consultation , many hospital managing reports in their system but it's not available to the patient when he / she is outside. In this project we are going to provide the extra facility to store the report in the database and make available from anywhere in the world .

Table of Contents

CHAPTER 1

INTRODUCTION

1.1 Overview

For more than a century now, hospital management system, as a discipline and practise in the management of people in an organisation, has evolved and developed into different areas The project Hospital Management system includes registration of patients, storing their details into the system, and also computerized billing in the pharmacy, and labs. The software has the facility to give a unique id for every patient and stores the details of every patient and the staff automatically. It includes a search facility to know the current status of each room. User can search availability of a doctor and the details of a patient using the id. The Hospital Management System can be entered using a username and password. It is accessible either by an administrator or receptionist. Only they can add data into the database. The data can be retrieved easily. The interface is very user-friendly. The data are well protected for personal use and makes the data processing very fast.

1.2 Problem Statement

It is very important to maintain efficient software to handle information of Hospital. This application provides a way to record this information and to access these in a simple way.

1.3 Existing System

The existing systems provide the basic functionalities needed to be handled in a hospital management environment. There is no intelligence of the software in such cases. In the existing system all the patient details, doctor availability details and regarding the tests done to the patients prescribed by the doctor is maintained manually by the receptionist. If a patient has to be admitted we need to check the availability of the bed which consumes lots of time if done manually. Also there is no proper search technique to check the patient information. It is a very difficult task to maintain all the finance management system, and the records which maintain doctor details, of the hospital by using this existing system. There are also many loopholes when we look at the security of the system. These are the main disadvantages of the existing systems that are overcome in the proposed model.

1.4 Proposed Systems

In our proposed system, we are going to provide solutions to all the above mentioned problems by automating the whole hospital management system by using an integrated software that handles the whole system. The proposed system provides One Integrated View to Patients for Billing, Collection, Discharge Detail, Patient Medical History. Easy Query Handling for instant decision of Bed Allocation for Patients, and request for the Bed

Transfers- Effective Search facility to search any type of information related to Patient history Certain actions are automated, for example, the date need not be entered explicitly by the user but instead the system adds it automatically while storing the data. One more advantage of this newly proposed

1.5 Advantages

- This software reduces paper work
- It is easy to handle record
- This software reduces time

1.6 Data flow diagram

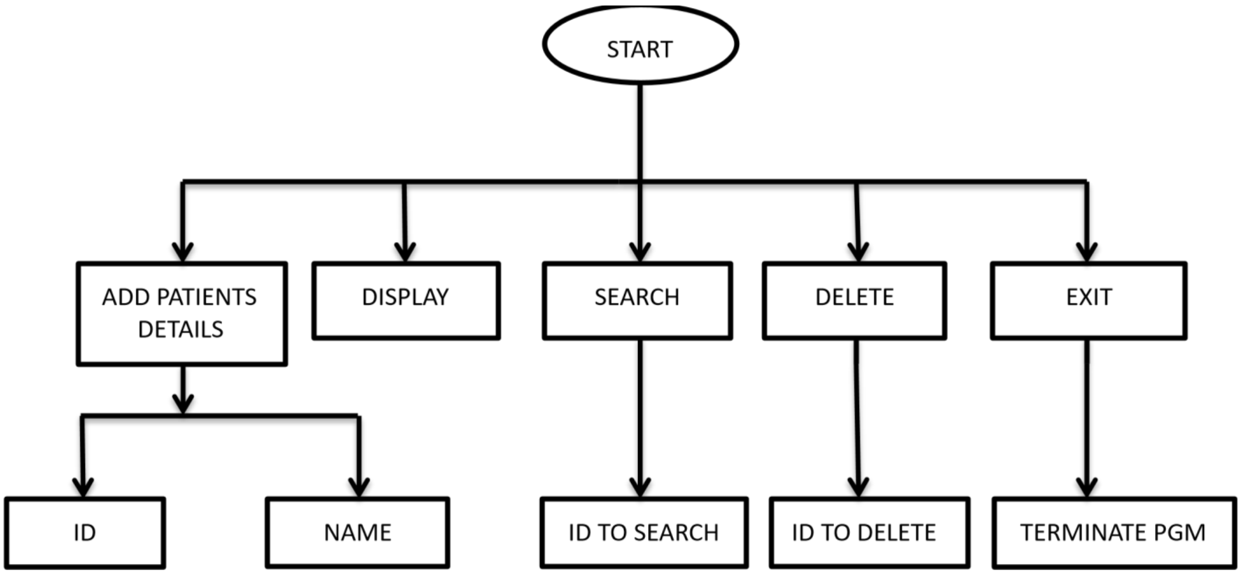


Fig 1.6.1 HOSPITAL MANAGEMENT SYSTEM

CHAPTER 2

SOFTWARE DESCRIPTIONS

2.1 SOFTWARE REQUIREMENTS

- Operating System: windows 11
- Coding language: C++
- IDE: Code ::Bocks(16.01) and Dev C++

2.2.1 Software Description

C++ (Language Used):

C++ was developed by Bjarne Stroustrup starting in 1979 at Bell Labs in Murray Hill, New Jersey, as an enhancement to the C language and originally named C with Classes but later it was renamed C++ in 1983. C++ is a statically typed, compiled, general-purpose, case sensitive, free-form programming language that supports procedural, object-oriented, and generic programming. C++ is regarded as a middle-level language, as it comprises a combination of both high-level and low-level language features.

NOTE: A programming language is said to use static typing when type checking is performed during compile-time as opposed to run-time.

Object-Oriented Programming

C++ fully supports object-oriented programming, including the four pillars of object-oriented development.

- Encapsulation
- Data hiding
- Inheritance
- Polymorphism

2.2.2 Scope and importance of work

When a large number of files are maintainable, the necessity of maintaining the index is increased. Indexing increases the utility of filling by providing an easy reference to the files. The very purpose of maintaining the index is that it is easy and quicker to find location of the files. The advantage of index lies in the fact is that index makes search operation perform very fast. So adding an index to a column will allow you to query based on a column faster.

2.2.3 System analysis

When the program is executed, two index files (primary and secondary index files) are created. Then the user is presented with a menu choice comprising of three operations namely: **INSERT, SEARCH,** and **DELETE** which can perform on both primary and secondary indexes respectively.

When **DELETE** operation is performed, it will insert an asterisk (*) symbol to replace the first character in the record and this operation will remove that particular record from the resprctive index list.

CHAPTER 3
FILE OPERATIONS

3.1 FILE STRUCTURE

Operation	Input	Output
1.Insertion	We will insert all the details of the patient which will be a sorted in a respective file Record.	Patient Registered Successfully…..
2.To Display	We will select an option display for obtaining the details. We search by an index.	All the patients records will be displayed.
3.Search	When we need a particular patient's details to be obtained, we search by the ID of that patient.	The corresponding patient's details are obtained.
4.Delete	When some patient's details needed to be removed, we provide the ID to delete that particular patient.	The particular patient will be removed from the record.

3.2 IMPLEMENTATION

The implementation of the entire process has been briefly described in the following stages below. Simple indexing on Hospital management record creates two index files namely, primary index file and secondary index file.

BUILDING INDEXING FILES

3.2.1 Primary Index File

Index on Sequential File, also called Primary Index, when the Index is associated to a Data File which is in turn sorted with respect to the search key.

- A Primary Index forces a sequential file organization on the Data File.
- Since a Data File can have just one order there can be just one Primary Index for Data File. Usually used when the search key is also the primary key of the relation. Usually, these indexes fit in main memory.

Index on sequential files can be:

1. Dense: One entry in the index file for every record in the data file.
2. Sparse: One entry in the index file for each block of the data file.

3.3 TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that a software system meets its requirements and user expectation does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement

3.3.1 Types of testing

- Unit Testing.
- Integrated Testing.
- System Testing.

1.Unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage producers, and operating procedures, are tested to determine whether they are fit for use. The primary goal of unit testing is to take the smallest piece of testable software in the application, isolate it from the reminder of the code, and determine whether it behaves exactly as you expect. Each unit is tested separately before integrating them into modules to test the interfaces between modules.

Unit testing has proven its value in that a large percentage of defects are identified during its use.

2.Integration testing is the logical extension of unit testing. In its simplest form, two units that have already been tested are combined into a component and the interface between them is tested. A component, in this sense, refers to an integrated aggregate of more than one unit. In a realistic scenario, many units are combined into components, which are in turn aggregated into even larger parts of the program. The idea is to test combinations of pieces and eventually expand the process to test your modules with those of other groups. Eventually all the modules making up the process are tested together. Beyond that, if the program is composed of more than one process, they should be tested in pairs rather than all at once.

3.The process of performing a variety of tests on a system to explore functionality or to identify problems is called System Testing. It is usually required before and after a system is put in place. A series of systematic procedures are referred to while testing is being performed. These procedures tell the tester how the system should perform and where common mistakes may be formed. Testers usually try to “Break the system” by entering data that may cause the system to malfunction or return incorrect information. For example, A tester may put in a city in a search engine design to accept only states, to see how the system responds to the incorrect inputs.

3.4 TEST CASES

Test case is a set of test inputs, execution and expected results developed for a particular objective.

An excellent test case satisfies the following criteria:

- Reasonable probability of catching errors.
- Does interesting things.
- Doesn’t do unnecessary things.
- Neither too simple nor too complex.
- Allows isolation and identification of errors.

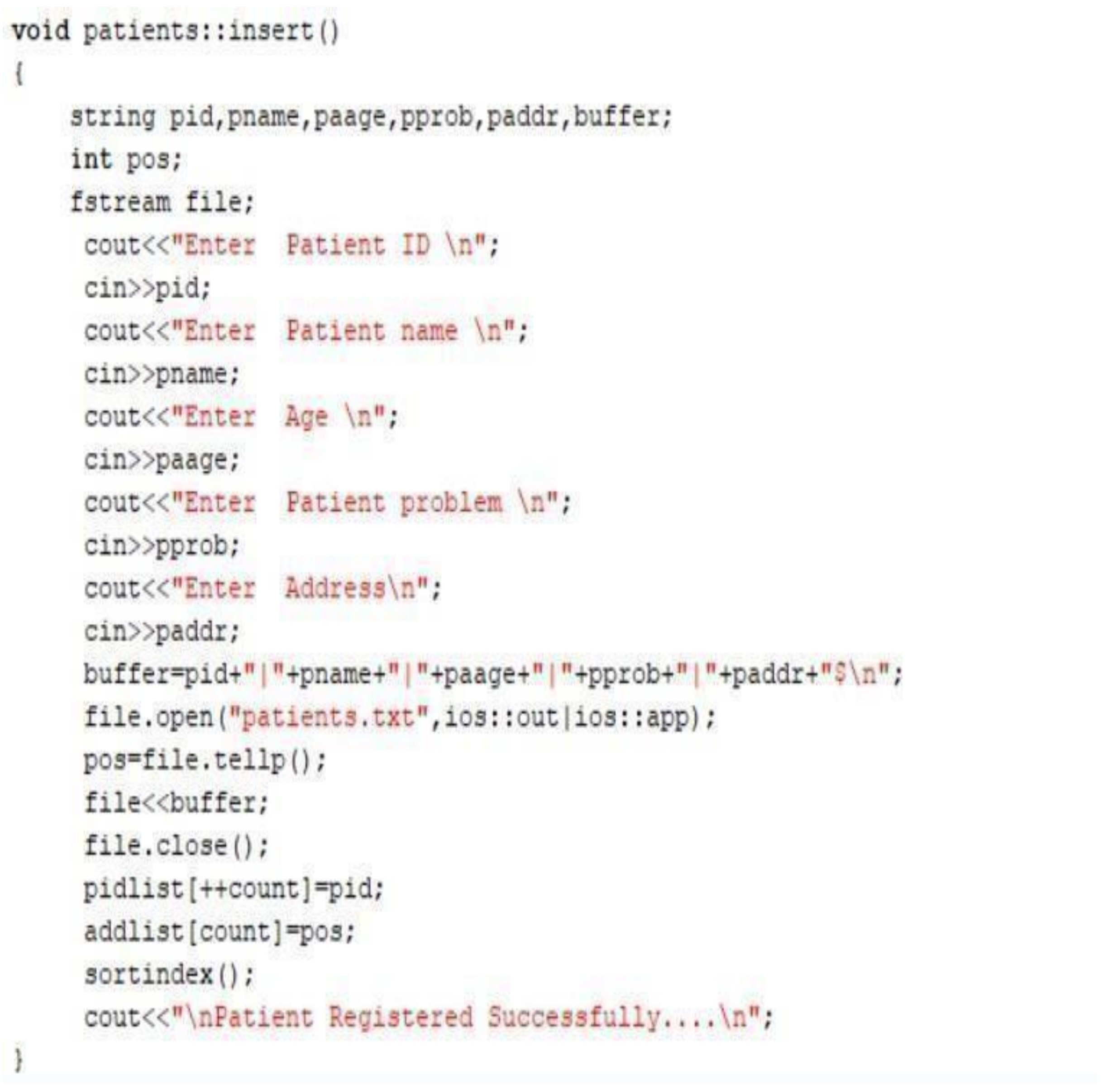
3.4.1 Unit Testing

TEST CASE ID	CASE NAME	TEST CASE CONDITION	INPUT	EXPECTED RESULT	ACTUAL RESULT	RESULT
TC 01	Insertion	Entering patients ID	When choice is entered	patients id and name should be display	All patients id is entered	Pass
TC 02	Display	Validating patients	When choice is entered	All patients details should be displayed	Patients details should display	Pass
TC 03	Search	Validating patients details	When choice is entered	Displays the Searched patients id	Displays the Searched id	Pass
TC 04	Delete	Validating patients details	When choice is entered	Delete the entered id	Delete the entered id	Pass
TC 05	Search	Valid patient ID	When wrong ID is entered	Display the searched patient details	Record not found	Fail

3.5 MODULES

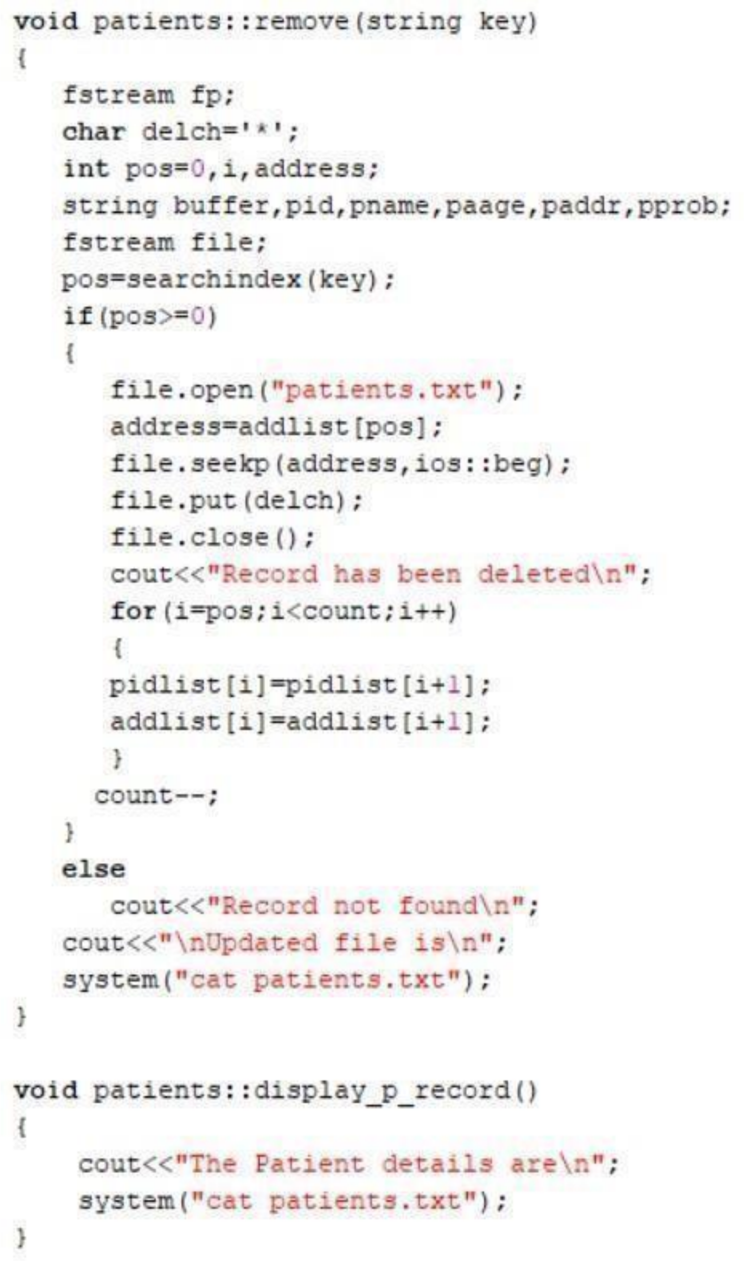
The user is given with an option to insert the records into the record file. Upon execution of the insertion operation, the new record will entered into the record file, primary-index file. This record will be inserted at the end of the file, that is, it will appended to these files. After insertion index files are rebuilt again.

Inserting the records:



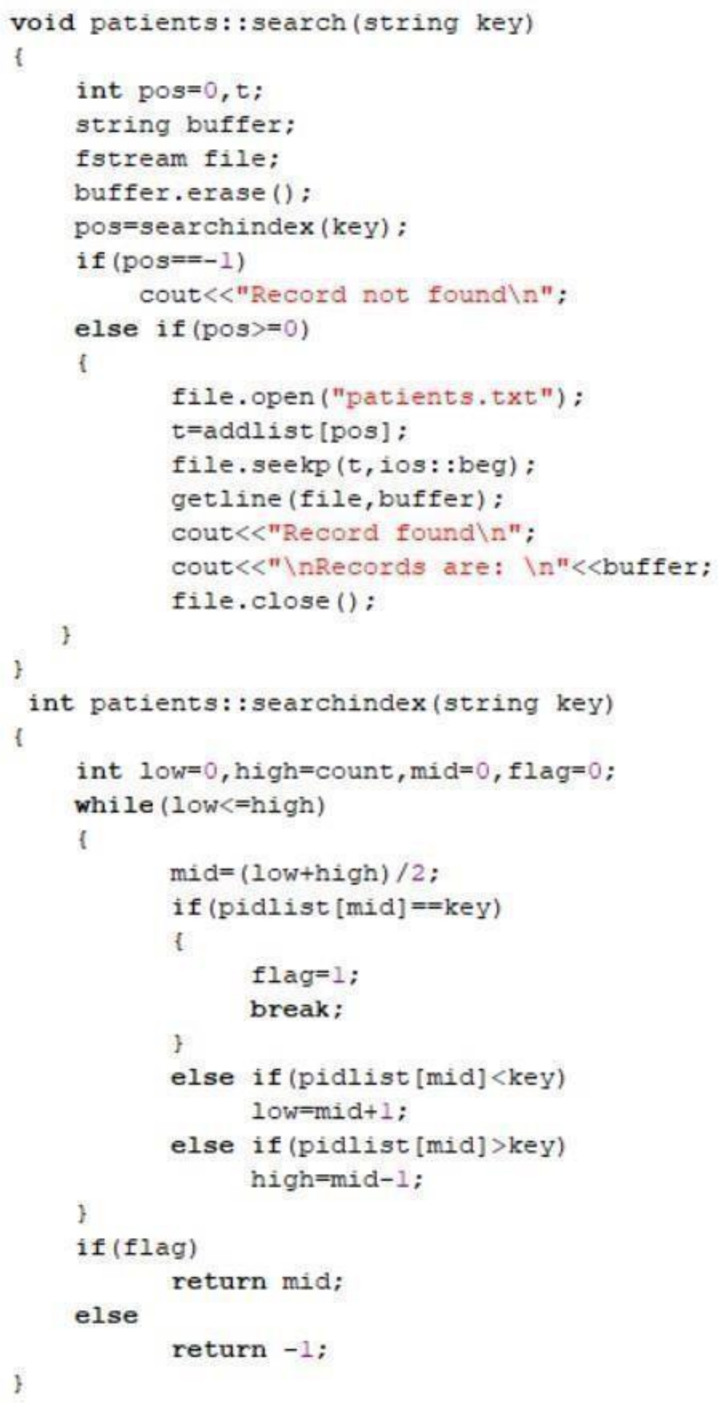
```
void patients::insert()
{
    string pid,pname,paage,pprob,paddr,buffer;
    int pos;
    fstream file;
     cout<<"Enter  Patient ID \n";
     cin>>pid;
     cout<<"Enter  Patient name \n";
     cin>>pname;
     cout<<"Enter  Age \n";
     cin>>paage;
     cout<<"Enter  Patient problem \n";
     cin>>pprob;
     cout<<"Enter  Address\n";
     cin>>paddr;
     buffer=pid+"|"+pname+"|"+paage+"|"+pprob+"|"+paddr+"$\n";
     file.open("patients.txt",ios::out|ios::app);
     pos=file.tellp();
     file<<buffer;
     file.close();
     pidlist[++count]=pid;
     addlist[count]=pos;
     sortindex();
     cout<<"\nPatient Registered Successfully....\n";
}
```

Deleting the records:



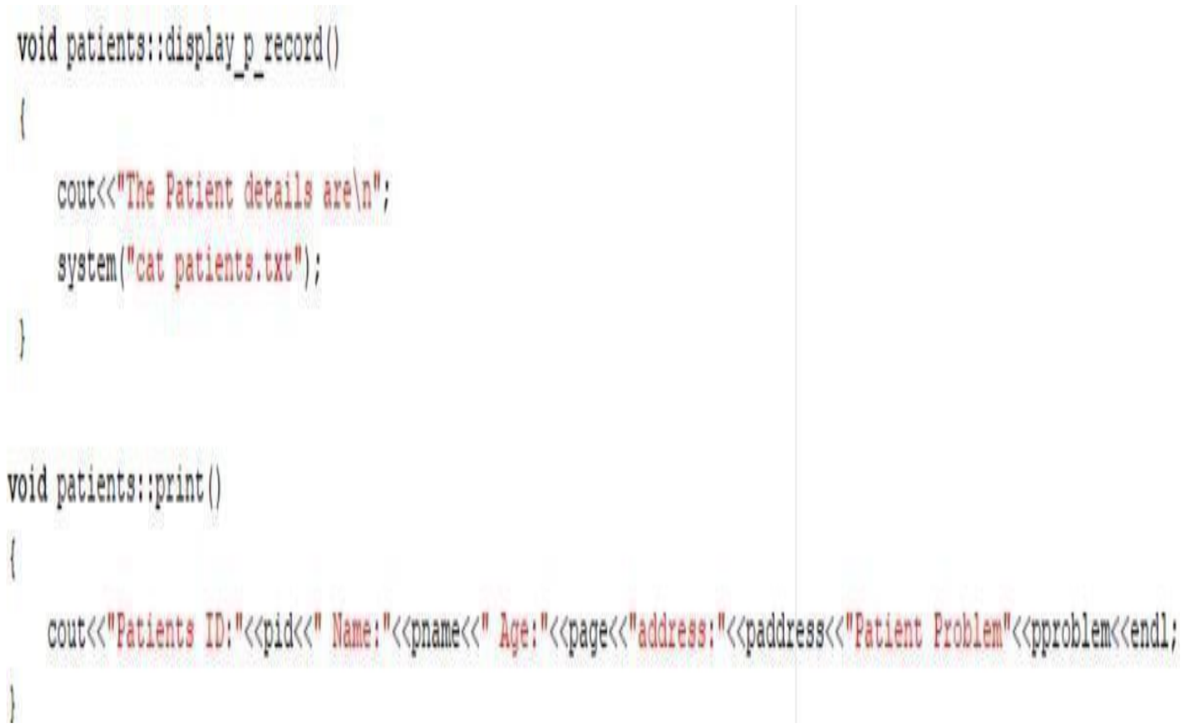
```
void patients::remove(string key)
{
   fstream fp;
   char delch='*';
   int pos=0,i,address;
   string buffer,pid,pname,paage,paddr,pprob;
   fstream file;
   pos=searchindex(key);
   if(pos>=0)
   {
      file.open("patients.txt");
      address=addlist[pos];
      file.seekp(address,ios::beg);
      file.put(delch);
      file.close();
      cout<<"Record has been deleted\n";
      for(i=pos;i<count;i++)
      {
      pidlist[i]=pidlist[i+1];
      addlist[i]=addlist[i+1];
      }
     count--;
   }
   else
      cout<<"Record not found\n";
   cout<<"\nUpdated file is\n";
   system("cat patients.txt");
}

void patients::display_p_record()
{
    cout<<"The Patient details are\n";
    system("cat patients.txt");
}
```

Searching the records:



```
void patients::search(string key)
{
    int pos=0,t;
    string buffer;
    fstream file;
    buffer.erase();
    pos=searchindex(key);
    if(pos==-1)
        cout<<"Record not found\n";
    else if(pos>=0)
    {
            file.open("patients.txt");
            t=addlist[pos];
            file.seekp(t,ios::beg);
            getline(file,buffer);
            cout<<"Record found\n";
            cout<<"\nRecords are: \n"<<buffer;
            file.close();
   }
}
 int patients::searchindex(string key)
{
    int low=0,high=count,mid=0,flag=0;
    while(low<=high)
    {
            mid=(low+high)/2;
            if(pidlist[mid]==key)
            {
                 flag=1;
                 break;
            }
            else if(pidlist[mid]<key)
                 low=mid+1;
            else if(pidlist[mid]>key)
                 high=mid-1;
    }
    if(flag)
            return mid;
    else
            return -1;
}
```

Displaying the records:

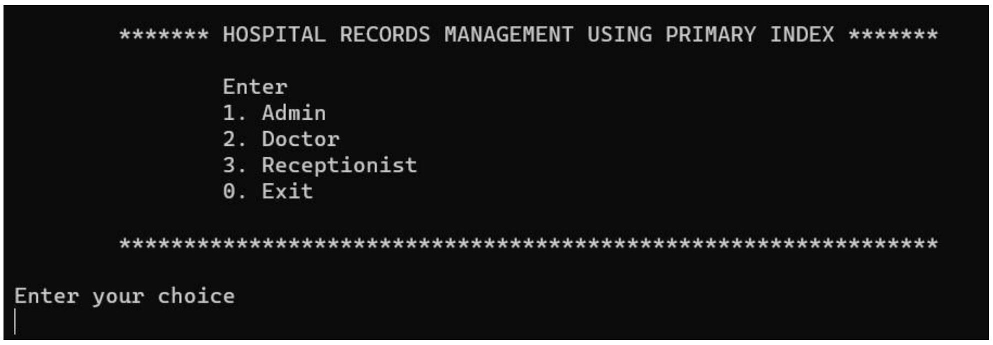


```
void patients::display_p_record()
{
    cout<<"The Patient details are\n";
    system("cat patients.txt");
}

void patients::print()
{
    cout<<"Patients ID:"<<pid<<" Name:"<<pname<<" Age:"<<page<<"address:"<<paddress<<"Patient Problem"<<pproblem<<endl;
}
```

CHAPTER 4

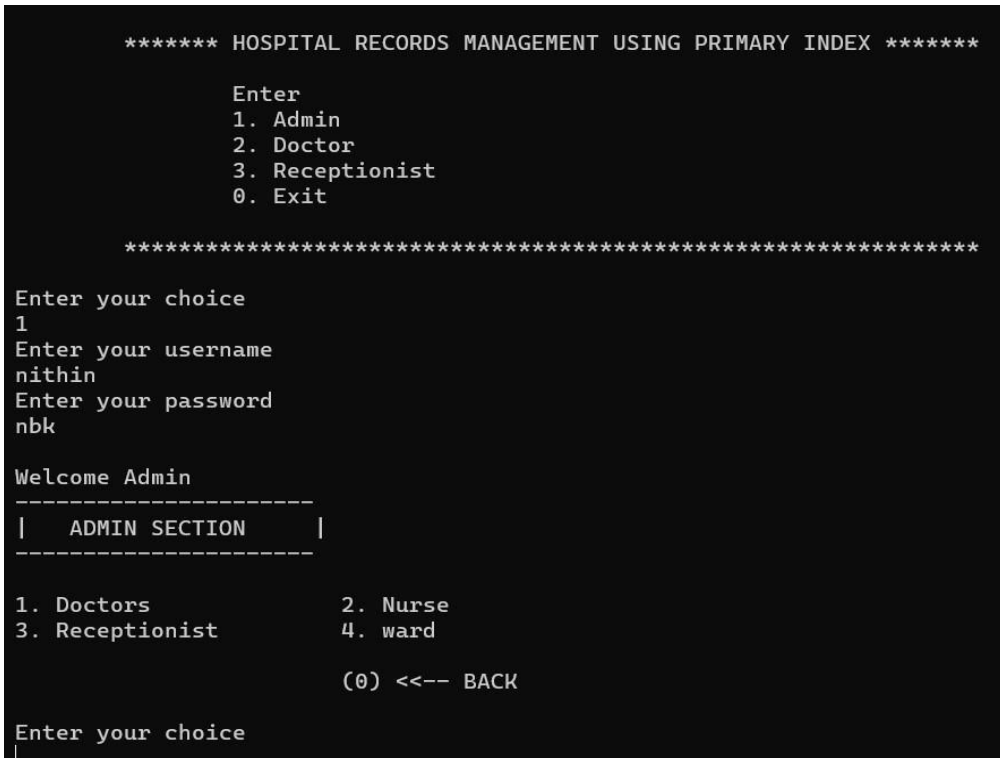
SNAPSHOTS :

Snap 1: User's Role.



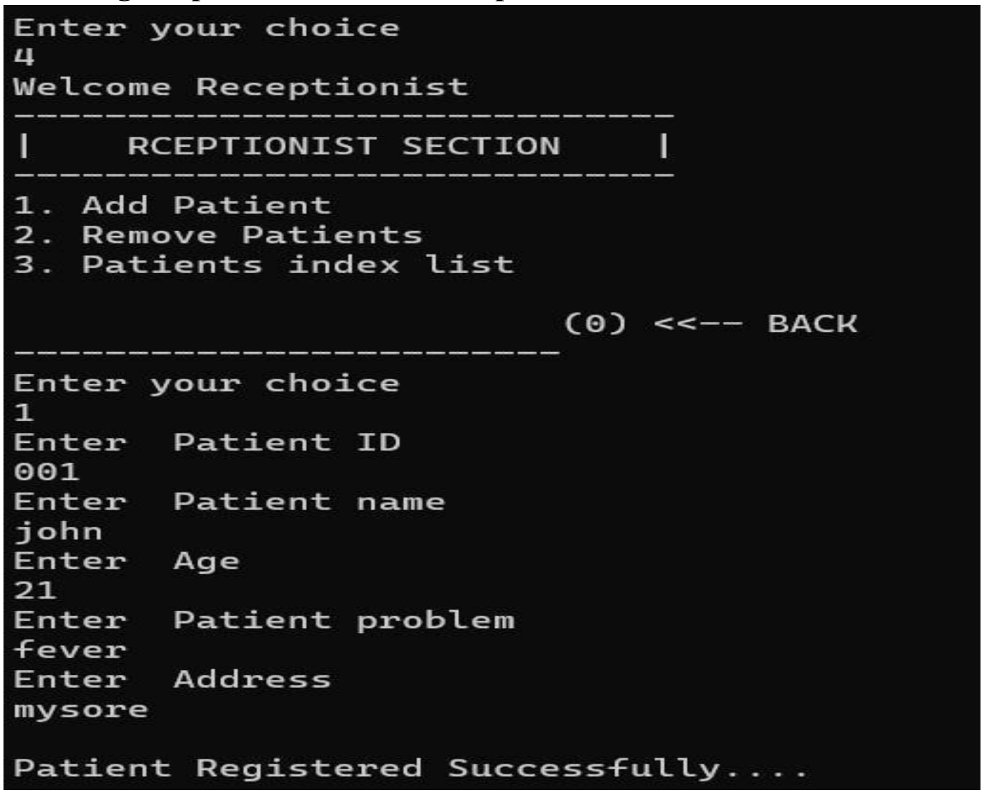
```
       ******* HOSPITAL RECORDS MANAGEMENT USING PRIMARY INDEX *******

               Enter
               1. Admin
               2. Doctor
               3. Receptionist
               0. Exit

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

Enter your choice
|
```

Snap 2: Operation performed by admin.



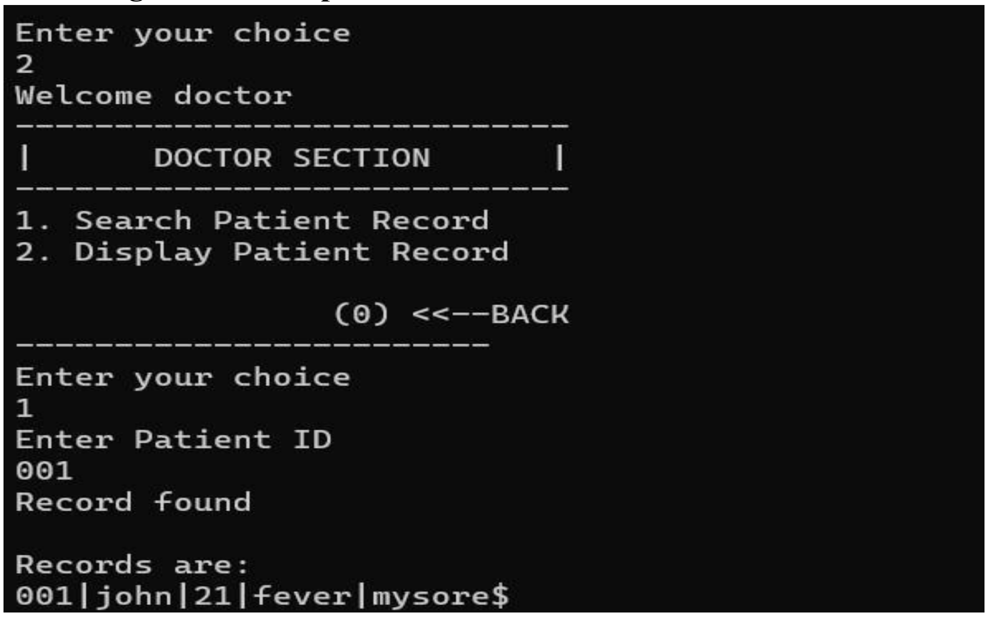
```
        ******* HOSPITAL RECORDS MANAGEMENT USING PRIMARY INDEX *******

                Enter
                1. Admin
                2. Doctor
                3. Receptionist
                0. Exit

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

Enter your choice
1
Enter your username
nithin
Enter your password
nbk

Welcome Admin
----------------------
|   ADMIN SECTION    |
----------------------

1. Doctors                2. Nurse
3. Receptionist           4. ward

                          (0) <<-- BACK

Enter your choice
```

Snap 3: Inserting the patients details in receptionist section



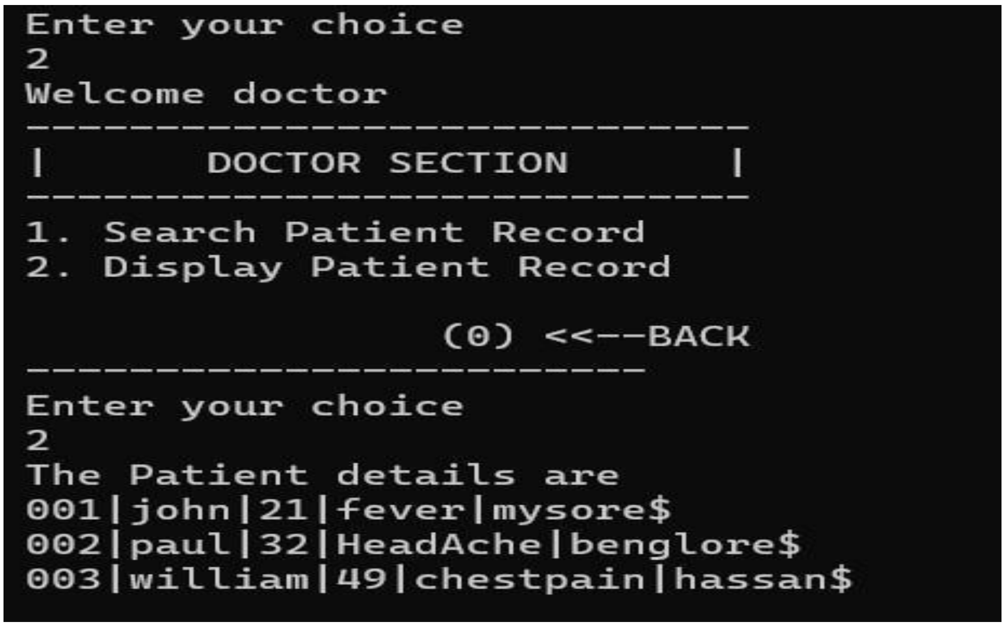
```
Enter your choice
4
Welcome Receptionist
-----------------------------
|     RCEPTIONIST SECTION     |
-----------------------------
1. Add Patient
2. Remove Patients
3. Patients index list

                        (0) <<-- BACK
------------------------
Enter your choice
1
Enter  Patient ID
001
Enter  Patient name
john
Enter  Age
21
Enter  Patient problem
fever
Enter  Address
mysore

Patient Registered Successfully....
```

Snap 4: Searching the inserted patient record



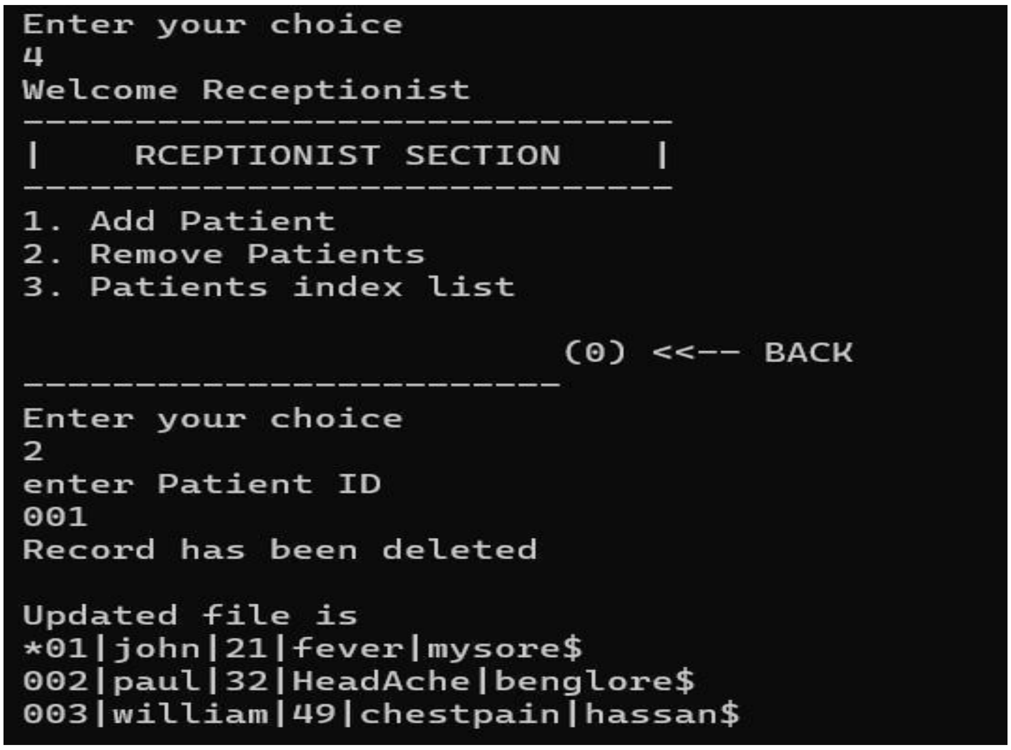
```
Enter your choice
2
Welcome doctor
----------------------------
|      DOCTOR SECTION      |
----------------------------
1. Search Patient Record
2. Display Patient Record

                (0) <<--BACK
------------------------
Enter your choice
1
Enter Patient ID
001
Record found

Records are:
001|john|21|fever|mysore$
```

Snap 5: Displaying the inserted patient record



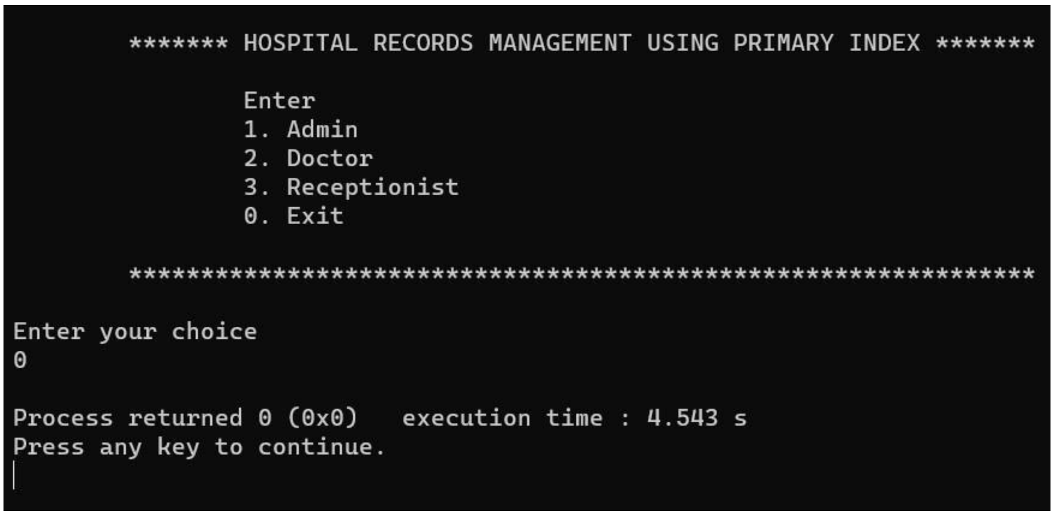
```
Enter your choice
2
Welcome doctor
------------------------------
|       DOCTOR SECTION       |
------------------------------
1. Search Patient Record
2. Display Patient Record

                (0) <<--BACK
------------------------
Enter your choice
2
The Patient details are
001|john|21|fever|mysore$
002|paul|32|HeadAche|benglore$
003|william|49|chestpain|hassan$
```

Snap 6: Deleting the record



```
Enter your choice
4
Welcome Receptionist
-----------------------------
|    RCEPTIONIST SECTION    |
-----------------------------
1. Add Patient
2. Remove Patients
3. Patients index list

                        (0) <<-- BACK
------------------------
Enter your choice
2
enter Patient ID
001
Record has been deleted

Updated file is
*01|john|21|fever|mysore$
002|paul|32|HeadAche|benglore$
003|william|49|chestpain|hassan$
```

Snap 7: Exit



```
        ******* HOSPITAL RECORDS MANAGEMENT USING PRIMARY INDEX *******

                Enter
                1. Admin
                2. Doctor
                3. Receptionist
                0. Exit

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

Enter your choice
0

Process returned 0 (0x0)   execution time : 4.543 s
Press any key to continue.
```

CHAPTER 5

○ FUTURE ENHANCEMENTS:

In this project, building the index files takes up a lot of time depending on the size of the record file. The larger the size of the record file, the more time it takes to build the index files. In case when the size of the index file is too large and exceeds the size of the main memory, then handling the index files itself will not be possible. Therefore, we can rely on other data structures such as B-trees, Hashing, Extensible hashing etc to build index files for large record files efficiently and within short period of time.

○ CONCLUSION:

The project is to digitalize the database of the staff in an organisation and enabling administration to have benefit from the computers. Software acts as Information System between Employees and administrations. Here the user can keep his\her database secure and safe for unlimited period of time. Software provides Employee management System for inserting, updating, Searching and deleting records with ease and simplicity. We will provide a fresh new approach to our esteemed users to search for records and make database in digital way.

REFERENCES

1. https://www.geeksforgeeks.org/

2. https://subramanyaks.github.io/

3. https://youtu.be/SAVKLLzXy8M

4. Michael J. Flok, Bill Zoellick, Greg Riccardi: File Structure-An Object.

5. Oriented Approach with C++, 3rd Edition, Pearson Education, 1998.