VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANASANGAMA , BELAGAVI -590018,KARNATAKA

FILE STRUCTURE MINI PROJECT REPORT

(18ISL67)

ON

B+ TREES

"HOSPITAL MANAGEMENT SYSTEM"

Submitted in partial fulfillment of the requirements for the 6th Semester

INFORMATION SCIENCE AND ENGINEERING

Submitted by

HITESH N

(1BI19IS021)

Under the guidance of

Dr. Roopa H

Associate Professor

Dept of ISE,BIT

2021-2022

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

BANGALORE INSTITUTE OF TECHNOLOGY

K. R. Road, V. V. Pura, Bengaluru-560004

BANGALORE INSTITUTE OF TECHNOLOGY

K. R. Road, V. V. Pura, Bengaluru-560004

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the implementation of **FILE STRUCTURE MINI PROJECT (18ISL67)** entitled **" HOSPITAL MANAGEMENT SYSTEM "** has been successfully completed by **HITESH N(1BI19IS021)** of VI semester B.E. for the partial fulfillment of the requirements for the Bachelor's degree in Information Science & Engineering of the **Visvesvaraya Technological University,Belagavi** during the academic year 2021-2022.

Lab In charge:

Mrs. Chethana M
Assistant professor
Dept of ISE,BIT

Head of Department:

Dr. Roopa H
Associate Professor
Dept of ISE,BIT

Name of Examiners:

Signature with date:

1.

2.

ABSTRACT

Hospital Management System is a system which maintains the information about the patient record present in the database, their details, referring to id, name, description and phone number. This is very difficult to organize manually. Maintenance of all this information manually is a very complex task. Owing to the advancement of technology, organization of a patient's records becomes much simpler. The Hospital management system has been designed to computerize and automate the operations performed over the information about the patients, their records and returns and all other operations. This computerization of the patient's records helps in many instances of its maintenance. It reduces the workload of management as most of the manual work done is reduced. Here we have used C++ programming language and B+ trees technique.

DECLARATION

I, **HITESH N**, bearing USN **1B119IS021** student of VI Semester **Department of Information Science and Engineering, Bangalore Institute of Technology, Bangalore** declare that the project work has been carried out by me and submitted partial fulfillment of the requirements for the **Bachelor's degree** in **Information Science & Engineering** of **Visvesvaraya Technological University, Belagavi** during the academic year 2021-2022.

Place: Bangalore

Name: HITESH N

USN: 1BI19IS021

ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped me in carrying out this project work. I would like to take this opportunity to thank them all.

I would like to thank **Dr. Roopa.H**, Professor and Head of the Department, Information Science and Engineering, BIT, Bangalore, for his valuable suggestions and expert advice.

I am also grateful to **Mrs.Chethana M,** Assistant Professor, Department of ISE, BIT, Bangalore, for providing advice and guidance regarding this project. I would like to thank all the teaching and non teaching staff of Department of Information Science & Engineering, BIT, and Bangalore for their constant support and encouragement.

HITESH N (1BI19IS021)

KUSHAL A C (1BI19IS028)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 GENERAL OVERVIEW

The objective of Hospital Management System is to allow the administrator of the hospital to edit and find out the details of a patient and allows the patient to keep up to date his profile. It'll also facilitate keeping all the records of patients, such as their Patients ID, Patients name, Description and Phone number. So, all the information about a Patient will be available in a few seconds.

1.2 PROBLEM DEFINITION

The Hospital Management System includes various information regarding the Patient's. The Hospital Management System needs details of the patient's such as ID, name, Description and phone number. So that they can keep track of their information. This project helps to handle this problem in an efficient manner. In these the patient's details are entered i.e., their Patients ID, Name, Description and Phone number are being stored as B + tree and in form of rft file. This project allows you to insert the patients details and Search Patients details and their details can be edited. This project also helps in insertion of data in such a way that when searched can be accessed faster.

1.3 OBJECTIVES

The main objective is to create a file so that the details of the patients can be recorded and maintained and used to view and search the details of the patients. This project is all about how to ease the problem of maintaining patients' data and storing it in a systematic way. The Hospital Management System facilitates the data storing, creating a file for that data, deleting and updating this data, adding other trees, searching the data etc... The main aim was to create a graphical user interface so that the user can effectively interact with the system and to implement the file handling using the concepts of B+ Tree so that the patient's details can be maintained and can be retrieved for the purpose of confirmation. The record contains the PID, Name, Description and phone_no for the storing, inserting, deleting, updating and searching.

CHAPTER 2

SOFTWARE AND HARDWARE REQUIREMENTS

The hideous method to search for the location and details of a suitable accommodation which is practically tedious in the current system is overcome by this project which will save the valuable time.

2.1 SOFTWARE REQUIREMENTS

1. OS: Windows XP,7,8,10
2. MinGW installer for C++ execution
3. Visual studio code

2.2 HARDWARE REQUIREMENTS

1. Pentium IV Processor, i3, i4, i5, i7
2. 4GB RAM
3. 40GB HDD
4. 1024 * 768 Resolution Color Monitor

- The hardware requirements specified are the hardware components/capacity of the system in which the application is developed and deployed.
- The above software requirements are the necessary software required to develop the application and the run the application.
- The project is developed in C++ language with concepts of File handling and B+ tree.

CHAPTER 3

DESIGN

3.1 CONTEXT-FLOW DIAGRAM

The context diagram is used to establish the context and boundaries of the system to be modelled which things are inside and outside of the system being modelled, and what is the relationship of the system with these external entities.

A context diagram, sometimes called a level 0 data-flow diagram, is drawn in order to define and clarify the boundaries of the software system. It identifies the flows of information between the system and external entities. The entire software system is shown as a single process.

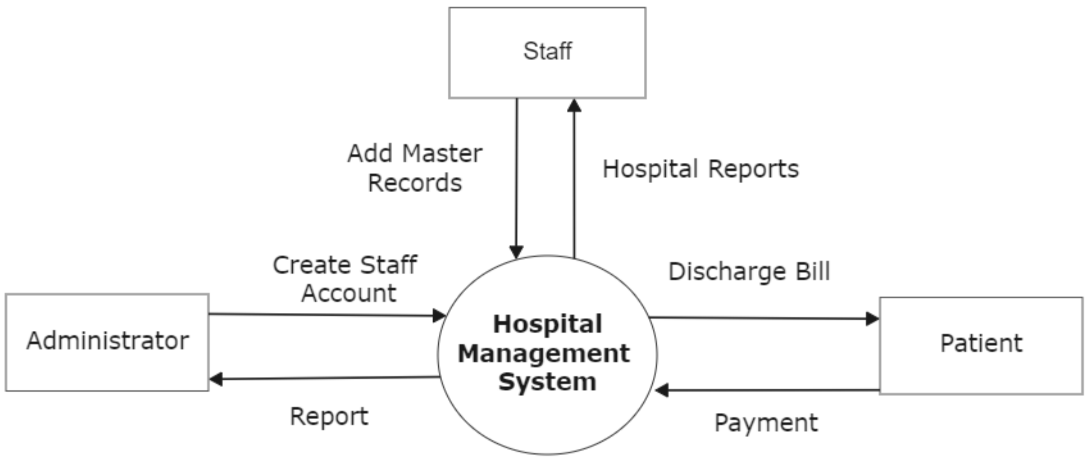


Figure 3.1 Context flow Diagram

3.2 SYSTEM ARCHITECTURE

System architecture is a solution, a "HOW TO" approach to the creation of a new system. It translates system requirements into ways by which they can be made operational. It is a translational from a user-oriented document to a document-oriented programmer. For that, it provides the understanding and procedural details necessary for the implementation. Systems design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements.

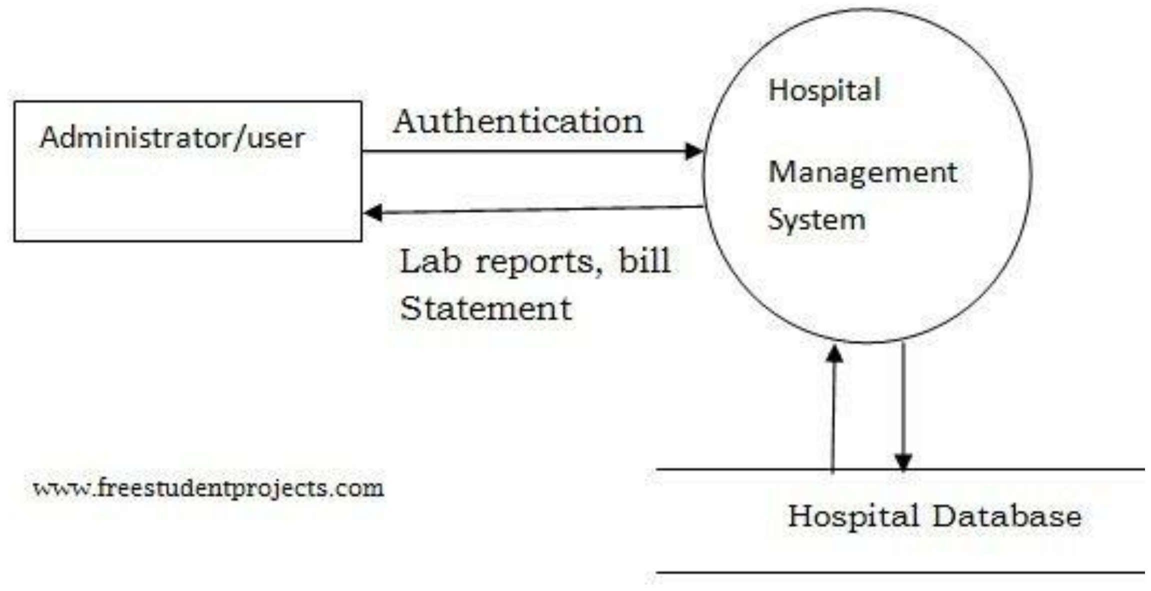

Administrator/user
Authentication
Lab reports, bill Statement
Hospital Management System
Hospital Database
www.freestudentprojects.com

Figure 3.2 Architecture of Hospital management system

3.3 DATA FLOW DIAGRAM

A data-flow diagram (DFD) is a way of representing a flow of a data of a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart. Data flow diagrams are used to graphically represent the flow of data in an information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation.

For each data flow, at least one of the endpoints (source and / or destination) must exist in a process. The refined representation of a process can be done in another data-flow diagram, which subdivides this process into sub-processes. The data-flow diagram is a tool that is part of structured analysis and data modeling. When using UML, the activity diagram typically takes over the role of the data-flow diagram. A special form of data-flow plan is a site-oriented data-flow plan. The DFD notation draws on graph theory, originally used in operational research to model workflow in organizations. DFD originated from the activity diagram used in the structured analysis and design technique methodology

3.3.1 DATA FLOW DIAGRAM FOR INSERTION

In the hospital management system, the admin can insert the details like name, pid, description, and phone number of the patient Then the data is validated, if the data is valid then the data is inserted into the file else insertion fails. The data that is saved in the file can be accessed by the admin later.

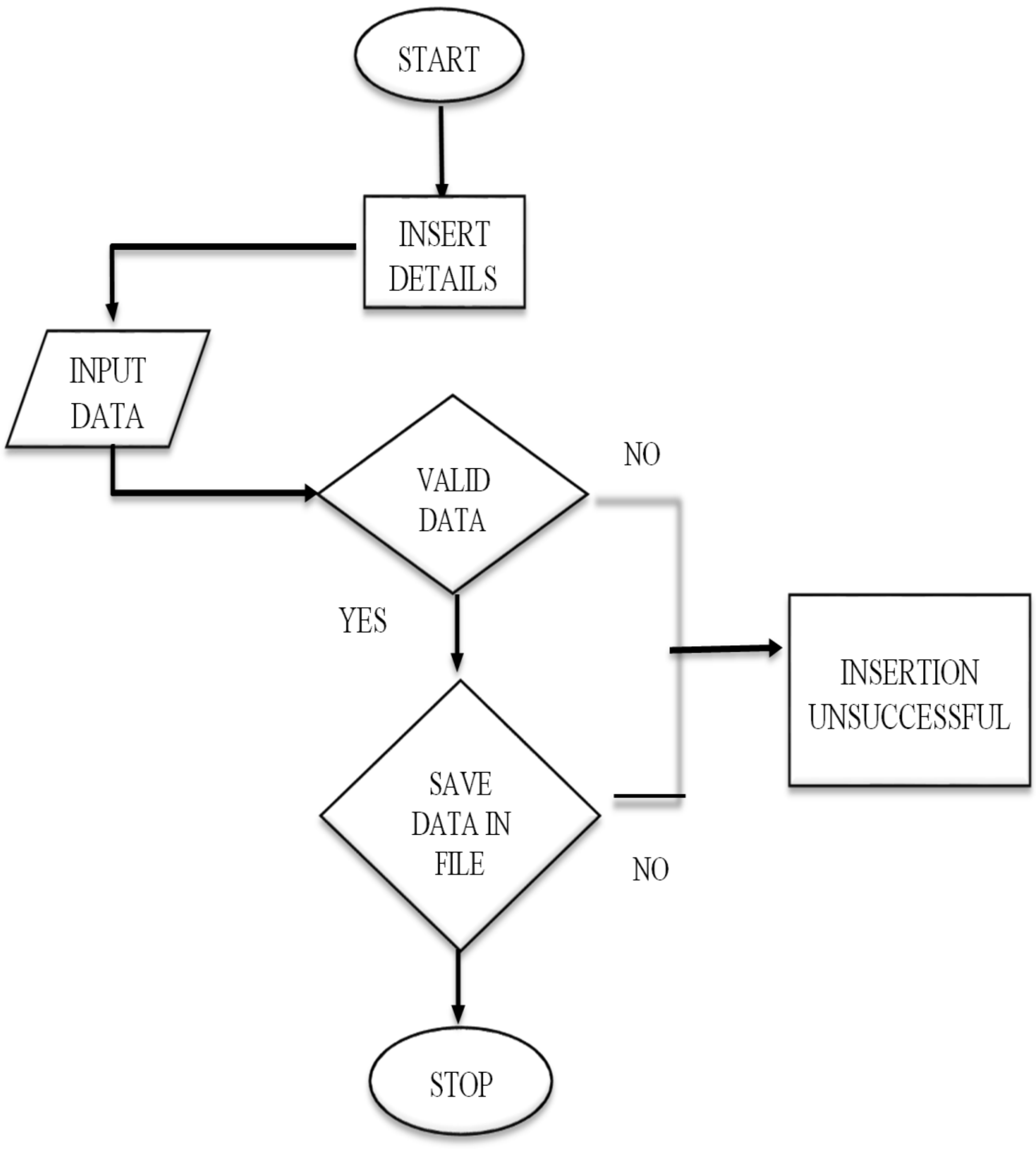


Figure 3.3 Dataflow diagram for insertion

3.3.2 DATA FLOW DIAGRAM FOR MODIFY DATA

In the Hospital management system, the admin can modify any patient's record by changing any of the patient details. The given patient ID is searched, if the patient is found then the data is to be modified by entering the new data for that patient else patient is not found and modify fails. The new data is saved to the file and can be accessed by the user later.

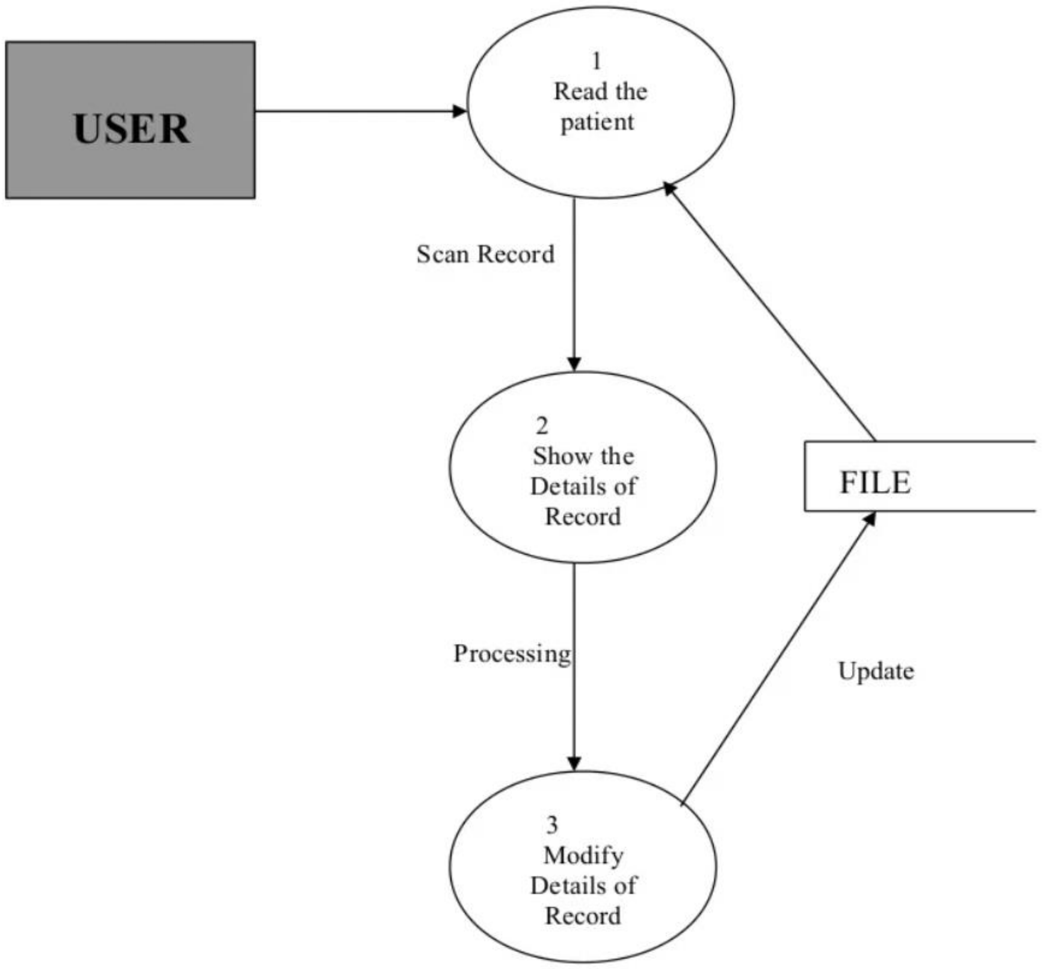


Figure 3.4 Dataflow diagram for modify

3.3.3 DATA FLOW DIAGRAM FOR SEARCH

In the Hospital management system, the admin can search for any patient's data by giving the pid. Then the respective record is searched, if the record is found then the data is displayed else search fails.

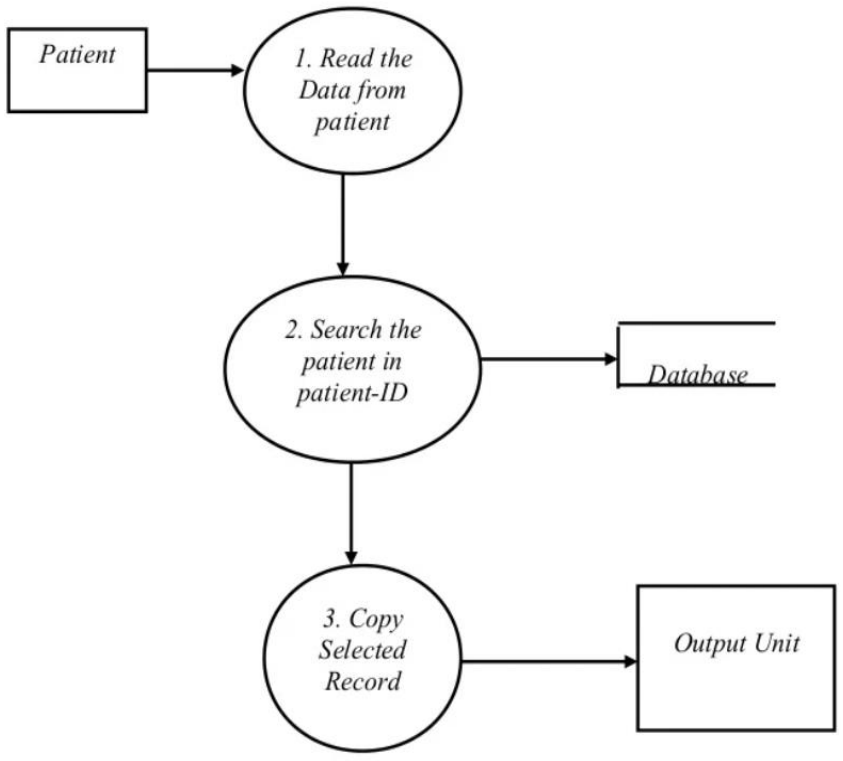

Patient
1. Read the Data from patient
2. Search the patient in patient-ID
Database
3. Copy Selected Record
Output Unit

Figure 3.5 Search for the patient's record

3.3.4 DATA FLOW DIAGRAM FOR DISPLAY

In the Hospital management system, the admin can view all the patients records i.e., the records the admin has inserted, modified, deleted and can view the record by using display function.

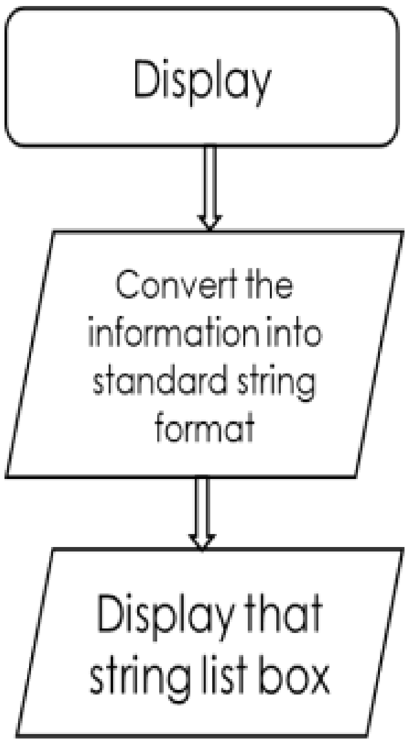

Display
Convert the information into standard string format
Display that string list box

Figure 3.6 Dataflow diagram for display

CHAPTER 4
IMPLEMENTATION

A B+ tree is an m-ary tree with a variable but often large number of children per node. A B+ tree consists of a root, internal nodes and leaves. The root may be either a leaf or a node with two or more children.

4.1 B+ Tree Introduction

B+ Tree is an extension of B Tree which allows efficient insertion, deletion and search operations. In B Tree, Keys and records both can be stored in the internal as well as leaf nodes. Whereas, in B+ tree, records (data) can only be stored on the leaf nodes while internal nodes can only store the key values. The leaf nodes of a B+ tree is linked together in the form of singly linked lists to make the search queries more efficient. B+ Tree is used to store a large amount of data which cannot be stored in the main memory. Due to the fact that, size of main memory is always limited, the internal nodes (keys to access records) of the B+ tree are stored in the main memory whereas leaf nodes are stored in the secondary memory.
The internal nodes of B+ tree is often called index nodes. A B+ tree of order 3 is shown in the following figure.

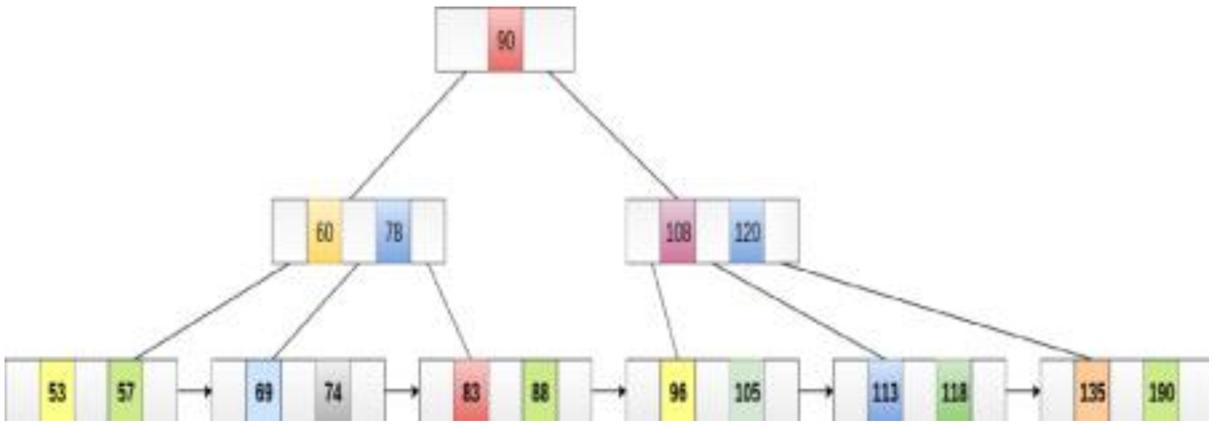


Figure 4.1 B+ tree

4.2 Rules for B+ Tree

Here are essential rules for B+ Tree.

- Leaves are used to store data records.
- It is stored in the internal nodes of the Tree.
- If a target key value is less than the internal node, then the point just to its left side is followed.

- If a target key value is greater than or equal to the internal node, then the point just to its right side is followed.

4.3 Why use B+ Tree

Here, are reasons for using B+ Tree:

- Keys are primarily utilized to aid the search by directing to the proper Leaf.
- B+ Tree uses a "fill factor" to manage the increase and decrease in a tree.
- In B+ trees, numerous keys can easily be placed on the page of memory because they do not have the data associated with the interior nodes. Therefore, it will quickly access tree data that is on the leaf node.
- A comprehensive full scan of all the elements is a tree that needs just one linear pass because all the leaf nodes of a B+ tree are linked with each other.

4.4 Searching a record in B+ Tree

Searching in a B+ tree is similar to searching in a BST. If the current value is less than the searching key, then traverse the left subtree, and if greater than first traverse this current bucket(node) and then check where the ideal location is.

Consider the below representation of a B+ tree to understand the searching procedure. Suppose we want to search for a key with a value equal to 59 in the below given B+ tree.

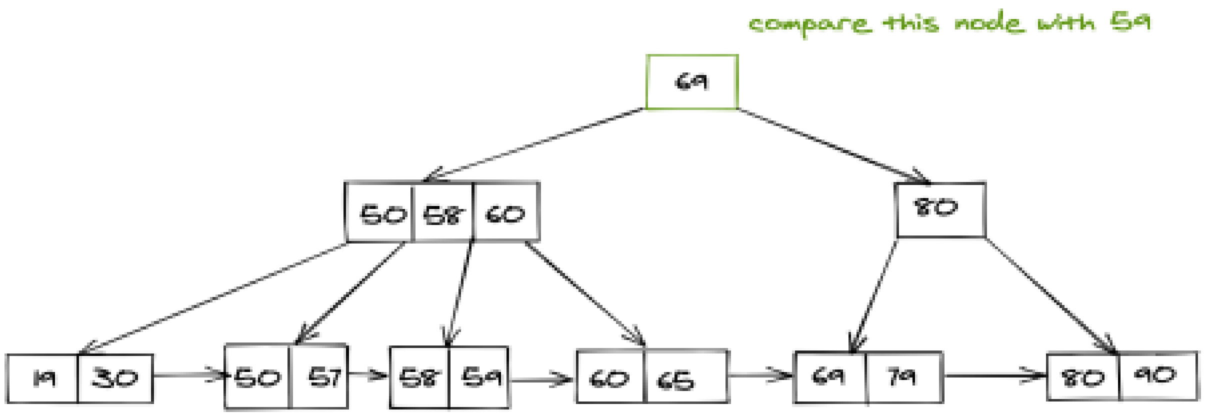


Figure 4.2 Key 59 lies between 58 and 60 in B+ Tree

Now we have found the internal pointer that will point us to our required search value.

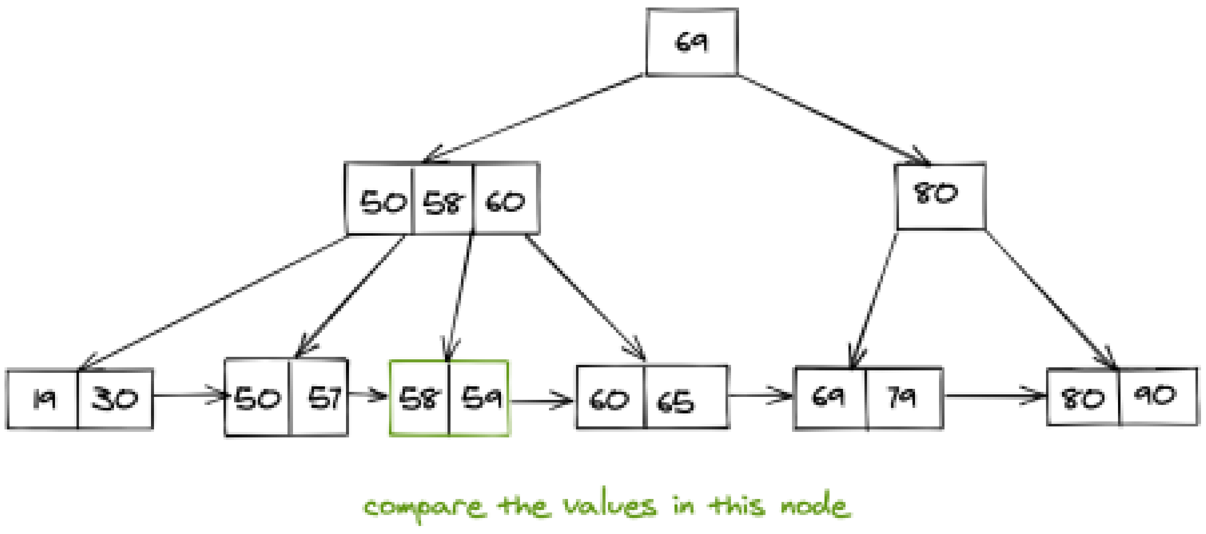


Figure 4.3 Internal pointer pointing to key 59 in B+ Tree

Finally, we traverse this bucket in a linear fashion to get to our required search value.

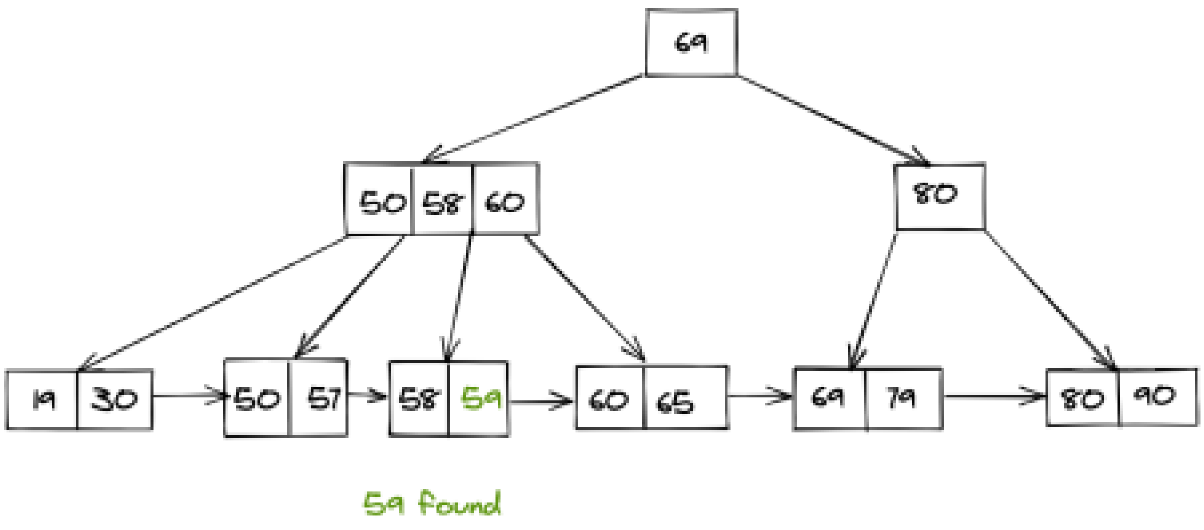


Figure 4.4 Key 59 is found in B+ Tree

4.5 Search Operation Algorithm

1. Call the binary search method on the records in the B+ Tree.
2. If the search parameters match the exact key

   The accurate result is returned and displayed to the user

   Else, if the node being searched is the current and the exact key is not found by the algorithm

   Display the statement "Record cannot be found."

4.6 Insertion in B+ tree

1. Perform a search operation in the B+ tree to check the ideal bucket location where this new node should go to.
2. If the bucket is not full (does not violate the B+ tree property), then add that node into this bucket.
3. Otherwise split the nodes into two nodes and push the middle node (median node to be precise) to the parent node and then insert the new node. Repeat the above steps.

4.6.1 Insert Operation Algorithm

1.Even inserting at-least 1 entry into the leaf container does not make it full then add the record

2. Else, divide the node into more locations to fit more records.

   a. Assign a new leaf and transfer 50 percent of the node elements to a new placement in the tree
   b. The minimum key of the binary tree leaf and its new key address are associated with the top-level node.
   c. Divide the top-level node if it gets full of keys and addresses.
   d. Similarly, insert a key in the center of the top-level node in the hierarchy of the Tree.
   e. Continue to execute the above steps until a top-level node is found that does not need to be divided anymore.

3. Build a new top-level root node of 1 Key and 2 indicators.

4.7 Deletion in B+ Tree

Deletion is a bit complicated process, two case arises:

1. It is only present at the leaf level
2. or, it contains a pointer from an internal node also.

If it is present only as a leaf node position, then we can simply delete it, for which we first do the search operation and then delete it.

4.7.1 Deletion Operation Algorithm

1) Start at the root and go up to leaf node containing the key K

2) Find the node n on path from the root to the leaf node containing A. If n is root, remove K

a. if root has more than one key, done

b. if root has only K

i) if any of its child nodes can lend a node

Borrow key from the child and adjust child links

ii) Otherwise merge the children nodes. It will be a new root

c. If n is an internal node, remove K

i) If n has at least ceil(m/2) keys, done!

ii) If n has less than ceil(m/2) keys,

If a sibling can lend a key,

Borrow key from the sibling and adjust keys in n and the parent node Adjust child links else

Merge n with its sibling

Adjust child links

d. If n is a leaf node, remove K

i) If n has at least ceiled(M/2) elements, done!

In case the smallest key is deleted, push up the next key

ii) If n has less than ceil(m/2) elements

If the sibling can lend a key

Borrow key from a sibling and adjust keys in n and its parent node Else

Merge n and its sibling

Adjust keys in the parent node

CHAPTER 5

SOFTWARE TESTING

Software testing is the stage of implementation, which is aimed at ensuring that the software works accurately and efficiently before live operation commences. Testing is the process of executing the program with the intent of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied. The ultimate aim is quality assurance. Tests are carried out and the results are compared with the expected document. In the case of erroneous results, debugging is done. Using detailed testing strategies, a test plan is carried out on each module. The various tests performed are unit testing, integration testing and user acceptance testing.

5.1 UNIT TESTING

Unit Testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the 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 a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.) Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing. The software units in a system are modules and routines that are assembled and integrated to perform a specific function. Unit testing focuses first on modules, independently of one another, to locate errors. This enables, to detect errors in coding and logic that are contained within each module. This testing includes entering data and ascertaining if the value matches to the type and size supported by java. The various controls are tested to ensure that each performs its action as required.

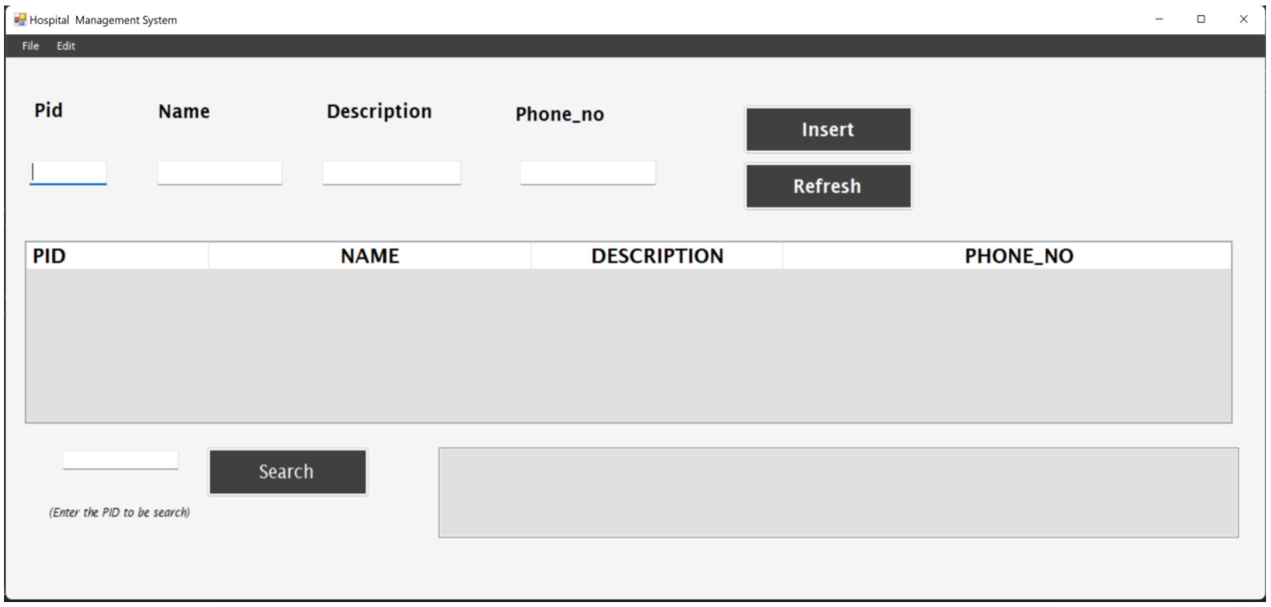
Table 5.1 Unit testing

S.NO	INPUT	OUTPUT	REMARKS
1	As the insert operation is selected, the valid data is entered and inserted.	Insertion successful	Pass
2	As the insert operation is selected, sufficient data is not entered.	Insertion failed	Pass
3	As the insert operation is selected, the valid data is entered when the file is full.	Storage full	Pass
4	As the insert operation is selected, data is entered for the same patient.	Already exist	Pass
5	As the search operation is selected, the valid patient ID is entered.	Record found	Pass
6	As the search operation is selected, the patient ID which is not present in the file is entered.	Record not found	Pass
7	As the delete operation is selected, the valid ID is entered.	Deletion successful	Pass
8	As the search operation is selected, the patient ID which is not present in the file is entered.	Record not found	Pass

9	As the modify operation is selected, the patient ID is entered.	Modified & inserted	Pass
10	As the modify operation is selected, the patient ID which is not present in the file is entered.	Record not found	Pass
11	As the display operation is selected.	Displays all records	Pass
12	As the display operation is selected, when no record has been inserted.	No records found to display	Pass

CHAPTER 6

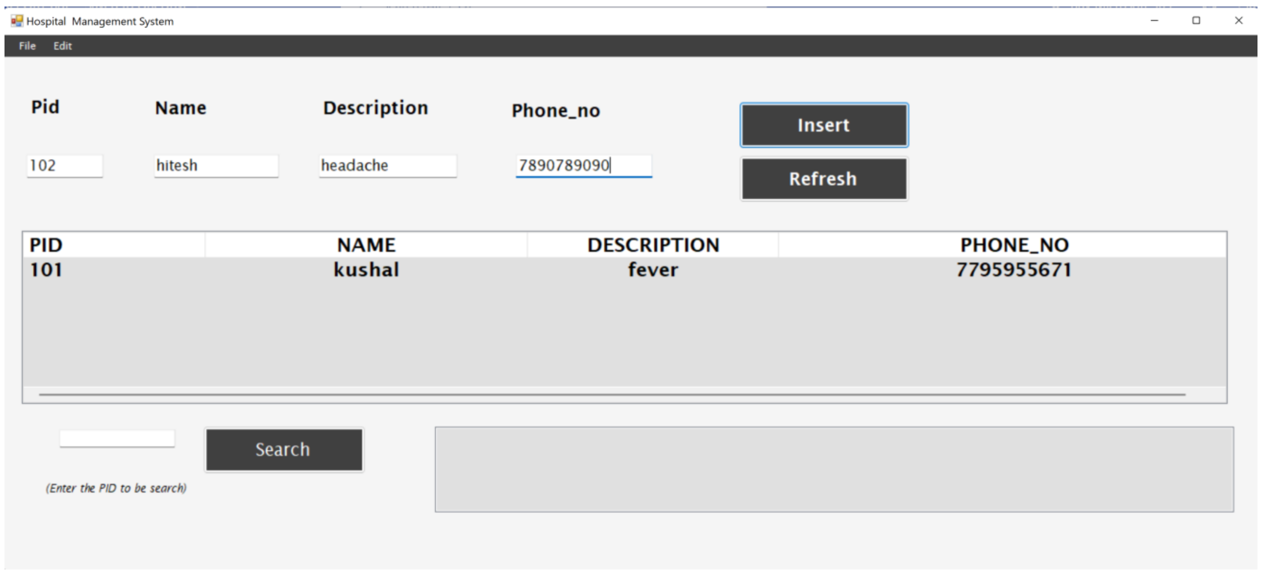
SNAPSHOTS



PID	NAME	DESCRIPTION	PHONE_NO

Figure 6.1 Default output page

When the project is executed, the above shown figure 6.1 is the first page that appears where the User can enter the details of patients in their respective field provided in the above diagram

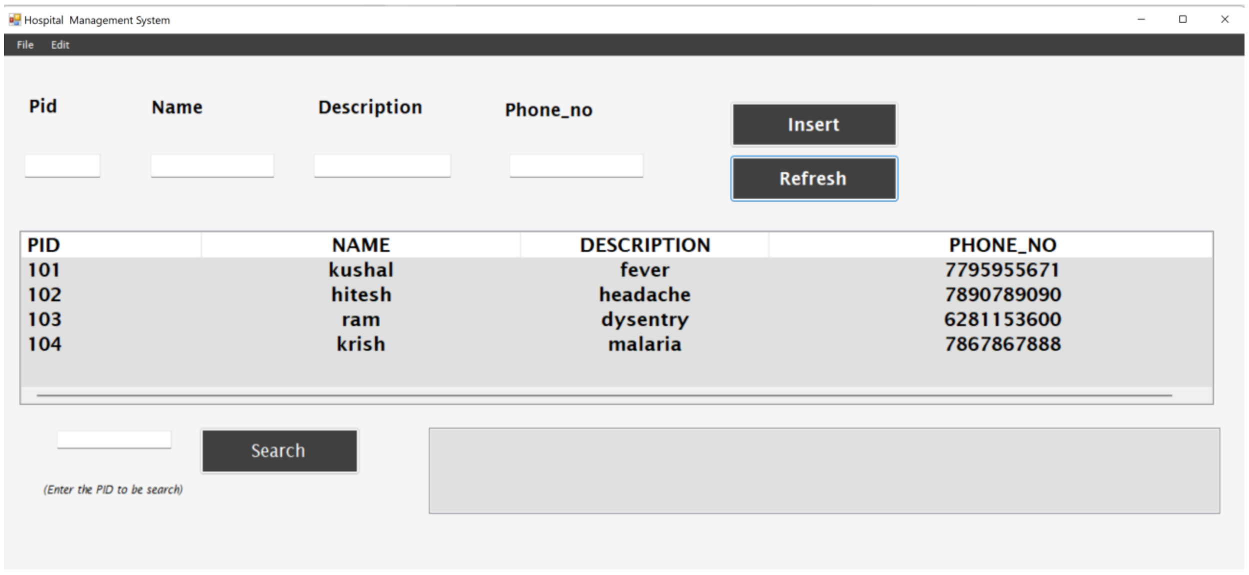


Pid: 102
Name: hitesh
Description: headache
Phone_no: 7890789090

PID	NAME	DESCRIPTION	PHONE_NO
101	kushal	fever	7795955671

Figure 6.2 Insertion

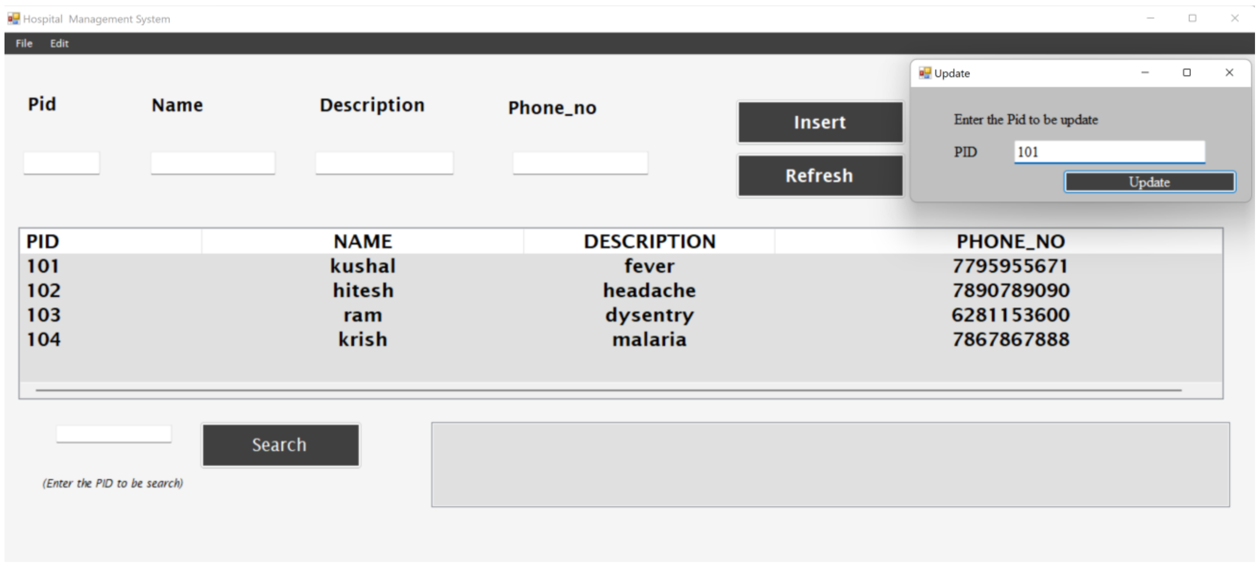
The above shown figure 6.2 is the snapshot of the insertion which is opened when the user wants enters the details of patients such as patients pid, name, description of his ailment or any other medical history ...and also his valid phone number.



PID	NAME	DESCRIPTION	PHONE_NO
101	kushal	fever	7795955671
102	hitesh	headache	7890789090
103	ram	dysentry	6281153600
104	krish	malaria	7867867888

Figure 6.3 Display

The above figure 6.3 shows the display page where all the inserted records present in the file are displayed. List of all the patient's records that are available are displayed.

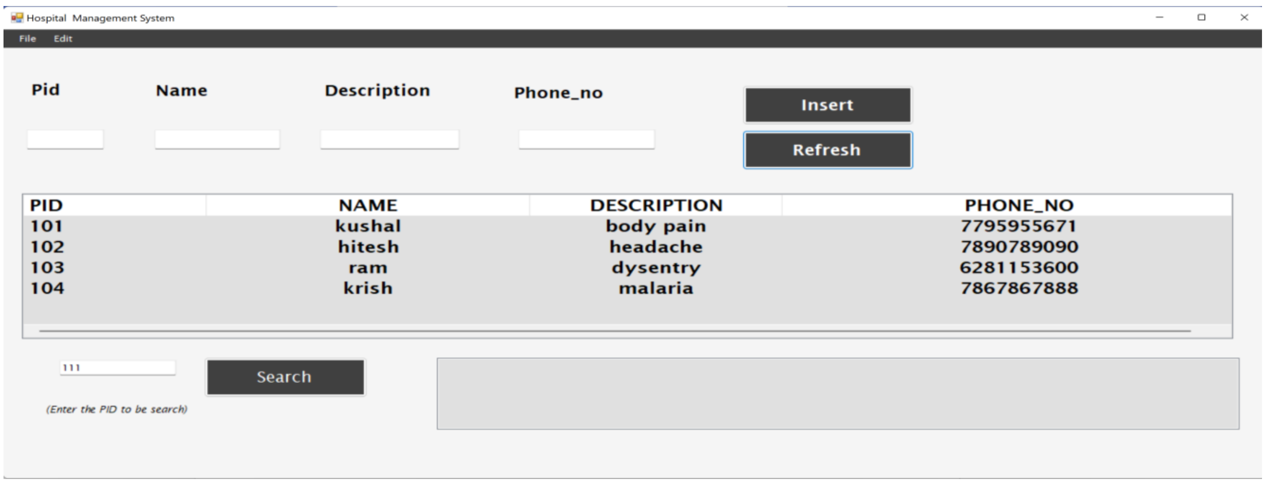

Hospital Management System
File
Edit
Pid
Name
Description
Phone_no
Insert
Refresh
Update
Enter the Pid to be update
Update
Search
(Enter the PID to be search)

PID 101

PID	NAME	DESCRIPTION	PHONE_NO
101	kushal	fever	7795955671
102	hitesh	headache	7890789090
103	ram	dysentry	6281153600
104	krish	malaria	7867867888

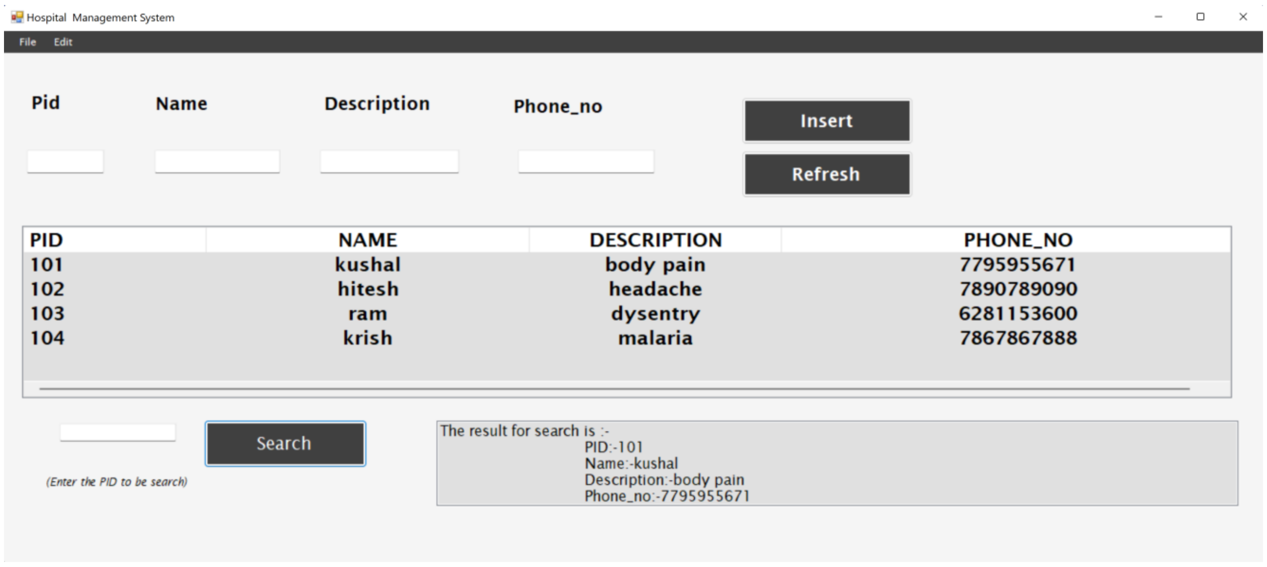
Figure 6.4 Before Update

The above figure 6.4 depicts the updating of record by providing all the values again in the respective fields or text boxes and again entering the valid pid for updating which will be displayed immediately after clicking refresh button as shown below in the figure 6.5.



PID	NAME	DESCRIPTION	PHONE_NO
101	kushal	body pain	7795955671
102	hitesh	headache	7890789090
103	ram	dysentry	6281153600
104	krish	malaria	7867867888

Figure 6.5 After Update

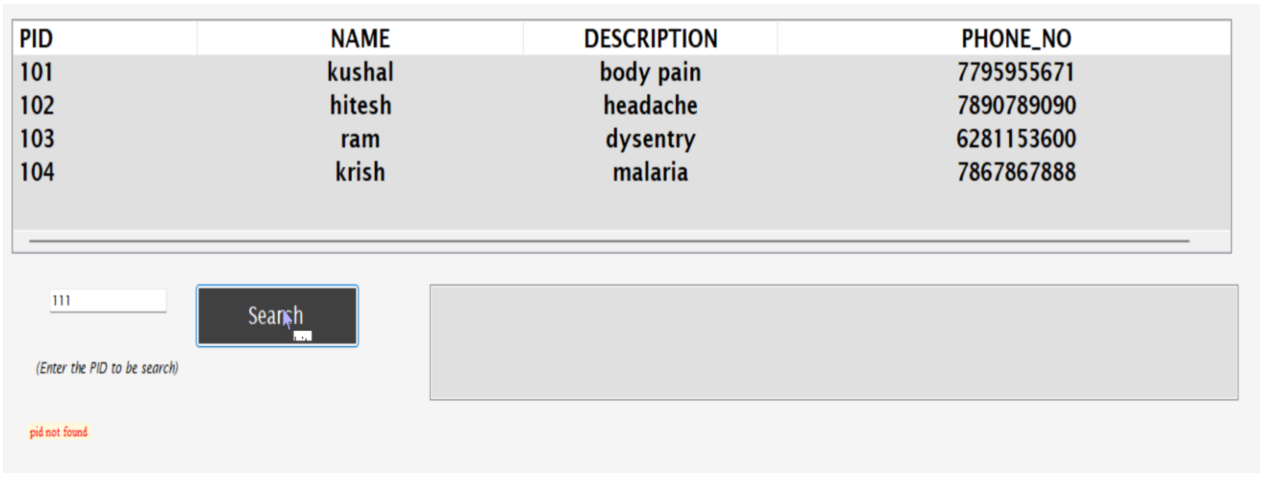


PID	NAME	DESCRIPTION	PHONE_NO
101	kushal	body pain	7795955671
102	hitesh	headache	7890789090
103	ram	dysentry	6281153600
104	krish	malaria	7867867888

```
The result for search is :-
                    PID:-101
                    Name:-kushal
                    Description:-body pain
                    Phone_no:-7795955671
```

Figure 6.6 Search

The above figure 6.6 shows the search where the user is able to search the desired record by providing appropriate pid in the textbox shown in above, the required details of patient like pid, name, description and phone number is displayed in the text area beside as shown above. If the given pid doesn’t exist it shows as pid not found as shown in the figure 6.7.



PID	NAME	DESCRIPTION	PHONE_NO
101	kushal	body pain	7795955671
102	hitesh	headache	7890789090
103	ram	dysentry	6281153600
104	krish	malaria	7867867888

111

(Enter the PID to be search)

pid not found

Figure 6.7 Search Failed

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

This project is developed successfully and the performance is found to be satisfactory. This Project is designed to meet the requirements of the user. It has been developed in C + + and the Database has been built using file structure hashing technique. The user will be able to store and manage the patient's data in a file. We have designed the project to provide the user with easy retrieval of data, details of patients are necessary operations can be performed on the data as well. In this project, the user is patient he can go to the patient's corner to view their information or if the user is admin, he can go to the admin portal by entering the respective details where various actions has been performed get access to the data. B+ tree technique was used to patients' data records in a file since it is one of the most useful and efficient file structure.

CHAPTER 8

REFERENCES

1. Michael J. Folk, Bill Zoellick, Greg Riccardi: File Structures-An Object-Oriented Approach with C++,3rd Edition, Pearson Education, 1998.
2. Scot Robert Ladd: C++ Components and Algorithms, BPB Publications, 1993
3. Geeks for Geeks www.geeksforgeeks.com
4. Stack overflow www.stackoverflow.com
5. www.cppbuzz.com

APPENDIX A

BACKEND CODE

Header.h

```
#pragma once
#include<iostream>
#include<string>
#include<fstream>
#include <vector>
#include <algorithm>
using namespace std;
int case_insensitive_match(string s1, string s2)
{
//convert s1 and s2 into lower case strings
transform(s1.begin(), s1.end(), s1.begin(), ::toupper);
transform(s2.begin(), s2.end(), s2.begin(), ::toupper);
cout << s1 << "\t" << s2 << endl;
if (s1.compare(s2) == 0)
return 1; //The strings are same
return 0; //not matched
}
vector<string> splitStrings(string str, char dl)
{
string word = "";
int num = 0;
str = str + dl;
int l = str.size();
vector<string> substr_list;
for (int i = 0; i < l; i++) {
if (str[i] != dl)
word = word + str[i];
else {
if ((int)word.size() != 0)
substr_list.push_back(word);
word = "";
}
}
return substr_list;
}
class patient
{
public:
string pid;
string name;
string description;
string phone_no;
string Buf;
char buf[100];
};
istream& operator >> (istream& in, patient& s)
{
cout << "Enter emid:-";
in >> s.pid;
cout << "Enter name:-";
in >> s.name;
cout << "Enter designation:-";
in >> s.description;
cout << "Enter address:-";
in >> s.phone_no;
return in;
}
ostream& operator << (ostream& out, patient& s)
{
    out << s.pid << "\t" << s.name << "\t" << s.description << "\t" << s.phone_no << "\t";
return out;
}
class node
```

```
{
public:
patient value[4];
node* child[4];
node* next, * per;
node* parent;
int size;
node();
};
class bplustree
{
public:
node* head;
int flag, flage, ds;
string st[1000];
patient sd[1000];
void insert(patient key);
void search(patient key);
bplustree update(patient key, patient key2);
void tree_view(node* n);
void write();
void read();
void display();
bplustree();
private:
node* split(node* n);
node* tchilds(node* n);
node* findlevel(patient key, node* n);
};
node::node() {
size = 4;
for (int i = 0; i < size; i++) {
value[i].pid = "";
```

```
value[i].name = "";
value[i].description = "";
value[i].phone_no = "";
child[i] = NULL;
}
size = 0;
next = NULL;
per = NULL;
parent = NULL;
}
bplustree::bplustree() {
head = NULL;
flag = 0;
flage = 0;
}
void bplustree::insert(patient key) {
ds = 0;
transform(key.pid.begin(), key.pid.end(), key.pid.begin(), ::toupper);
cout << key.pid << endl;
if (head == NULL) {
head = new node;
head->value[head->size] = key;
head->size++;
return;
}
node* n = findlevel(key, head);
int i;
for (i = 0; i < n->size; i++) {
if (case_insensitive_match(key.pid, n->value[i].pid)) {
st[ds++] = "already exits";
return;

}
```

```
if (key.pid < n->value[i].pid) {
break;
}
}
if (i == n->size) {
n->value[i] = key;
n->size++;
}
else {
patient temp;
for (; i < n->size; i++) {
temp = n->value[i];
n->value[i] = key;
key = temp;
}
n->value[i] = key;
n->size++;
}
if (n->size > 3) {
node* x = split(n);
}
return;
}
node* bplustree::findlevel(patient key, node* n) {
node* ptr = n;
int i;
for (i = 0; i < ptr->size; i++) {
if (key.pid < ptr->value[i].pid) {
if (ptr->child[i] != NULL) {
return(findlevel(key, ptr->child[i]));
}
else {
return ptr;
```

```
}
}
}
if (i == ptr->size && key.pid > ptr->value[i].pid) {
if (ptr->child[i] != NULL) {
return(findlevel(key, ptr->child[i]));
}
else
{
return ptr;
}
}
return ptr;
}

bplustree bplustree::update(patient key, patient key2) {
transform(key.pid.begin(), key.pid.end(), key.pid.begin(), ::toupper);
transform(key2.pid.begin(), key2.pid.end(), key2.pid.begin(), ::toupper);
node* n = head;
int x = 0;
bplustree b;
b.ds = 0;
flage = 0;
patient a[1000];
while (n->child[0] != NULL) {
n = n->child[0];
}
while (n != NULL) {
for (int i = 0; i < n->size; i++) {
if ((case_insensitive_match(key.pid, n->value[i].pid)) == 0)
a[x++] = n->value[i];
else
{
```

```
a[x++] = key2;
flage = 1;
}
}
n = n->next;
}
if (flage == 0) {
b.st[0] = "element not found\n";
b.ds++;
b.head = head;
return b;
}
for (int i = 0; i <= x; i++) {
b.insert(a[i]);
b.flage = 1;

}

n = b.head;
while (n->child[0] != NULL) {
n = n->child[0];
}
for (int i = 0; i < n->size; i++)
cout << n->value[i];
return b;
}
node* bplustree::split(node* n) {
int m = (n->size / 2);
int x = 0;
node* ptr;
if (n->parent == NULL) {
head = new node;
ptr = head;
ptr->value[x] = n->value[m];
ptr->size++;
```

```
}
else {
ptr = n->parent;
for (x = 0; x < ptr->size; x++) {
if (n->value[m].pid < ptr->value[x].pid) {
break;
}
}
patient temp = n->value[m];
patient temp2;
node* ad1, * ad2;
ad1 = ptr->child[x];
int y = x;
for (; x < ptr->size; x++) {
temp2 = ptr->value[x];
ptr->value[x] = temp;
temp = temp2;
ad2 = ptr->child[x + 1];
ptr->child[x + 1] = ad1;
ad1 = ad2;
}
ptr->value[x] = temp;
ptr->child[x + 1] = ad1;
x = y;
ptr->size++;
}
node* c1 = new node;
node* c2 = new node;
ptr->child[x] = c1;
ptr->child[x + 1] = c2;
c1->next = c2;
c2->per = c1;
if (x != 0) {
```

```
if (ptr->child[x - 1] != NULL) {

ptr->child[x - 1]->next = c1;
c1->per = ptr->child[x - 1];
}
}
if (ptr->child[x + 2] != NULL) {
c2->next = ptr->child[x + 2];
ptr->child[x + 2]->per = c2;
}
if (n->child[0] == NULL) {
for (int i = 0; i < m; i++) {
c1->value[i] = n->value[i];
c1->size++;
}
for (int i = m, x = 0; i < n->size; i++, x++) {
c2->value[x] = n->value[i];
c2->size++;
}
if (n->next != NULL) {
c2->next = n->next;
}
if (n->per != NULL)
n->per->next = c1;
}
else {
c1->child[0] = n->child[0];
c1->child[0]->parent = c1;
for (int i = 0; i < m; i++) {
c1->value[i] = n->value[i];
c1->child[i + 1] = n->child[i + 1];
c1->child[i + 1]->parent = c1;
c1->size++;
}
c2->child[0] = n->child[m + 1];

c2->child[0]->parent = c2;
for (int i = m + 1, x = 0; i < n->size; i++, x++) {
c2->value[x] = n->value[i];
c2->child[x + 1] = n->child[i + 1];
c2->child[x + 1]->parent = c2;
c2->size++;
}
}
c1->parent = ptr;
c2->parent = ptr;
if (ptr->size > 3) {
ptr = split(ptr);
}
return c2;
}
void bplustree::display() {
node* n = head;
ds = 0;
while (n->child[0] != NULL) {
n = n->child[0];
}
while (n != NULL) {
for (int i = 0; i < n->size; i++) {
if (n->value[i].pid == "")
continue;
sd[ds++] = n->value[i];
}
n = n->next;
}
return;
```

```
}
void bplustree::search(patient key) {
transform(key.pid.begin(), key.pid.end(), key.pid.begin(), ::toupper);
node* ptr = head;
ds = 0;
int x = 0;
flag = 0;
while (ptr != NULL) {
if (ptr->child[0] == NULL) {
for (int i = 0; i < ptr->size; i++) {
if ((case_insensitive_match(key.pid, ptr->value[i].pid)) == 1)
{
transform(key.pid.begin(), key.pid.end(),
key.pid.begin(), ::toupper);
cout << key.pid << endl;
flag = 1;
sd[ds++] = ptr->value[i];
return;
}
}
break;
}
else
{
int i;
for (i = 0; i < ptr->size; i++) {
if (key.pid < ptr->value[i].pid) {
break;
}
}
ptr = ptr->child[i];
cout << x++ << "\t";
cout << "->";
}
}
st[0] = "pid not found";
return;
}
void bplustree::tree_view(node* n) {
cout << "|";
for (int i = 0; i < n->size; i++) {
cout << n->value[i];
}
cout << "|" << endl;
while (n != NULL) {}
}
void bplustree::write() {
node* n = head;
ds = 0;
for (int i = 0; i < 1000; i++)
{
st[i] = "";
}
while (n->child[0] != NULL) {
n = n->child[0];
}
while (n != NULL) {
for (int i = 0; i < n->size; i++) {
if (n->value[i].pid == "")
continue;
st[ds++] = n->value[i].pid + "|" + n->value[i].name + "|" + n->value[i].description + "|" + n-
>value[i].phone_no + "|";
}
n = n->next;
}
return;
```

```
}
void bplustree::read() {
char dl = '|';
patient s;
cout << ds << endl;
for (int j = 0; j < ds; j++) {
cout << st[j] << j << endl;
vector<string> res = splitStrings(st[j], dl);
s.pid = res[0];
s.name = res[1];
s.description = res[2];
s.phone_no = res[3];
int temp = ds;
insert(s);
ds = temp;
}
}
```

MyForm.cpp

```
#include "MyForm.h"

using namespace System;
using namespace System::Windows::Forms;
[STAThreadAttribute]
void main(cli::array<String^>^ args) {
Application::EnableVisualStyles();
Application::SetCompatibleTextRenderingDefault(false);
Project2::MyForm form;
Application::Run(% form);
}
```

APPENDIX B

Frontend Code

MyForm.h

```
#pragma once
#include"Header.h"
#include"update.h"
#include<string.h>
#include <msclr/marshal.h>
#include <msclr/marshal_cppstd.h>
namespace Project2 {
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
using namespace msclr::interop;
using namespace std;
using namespace System::IO;
/// <summary>
/// Summary for MyForm
/// </summary>
bplustree b;
patient s, s2;
public ref class MyForm : public System::Windows::Forms::Form
{
public:
int flags = 1;
private: System::Windows::Forms::SaveFileDialog^ savefd;
private: System::Windows::Forms::Label^ label8;
private: System::Windows::Forms::ListBox^ listBox1;
private: System::Windows::Forms::ListView^ display;
private: System::Windows::Forms::ColumnHeader^ displaypid;
```

```
private: System::Windows::Forms::ColumnHeader^ displayname;
private: System::Windows::Forms::ColumnHeader^ displaydescription;
private: System::Windows::Forms::ColumnHeader^ displayphone_no;
private: System::Windows::Forms::Label^ label7;
private: System::Windows::Forms::ListBox^ Search_display;
private: System::Windows::Forms::OpenFileDialog^ openFileDialog1;
private: System::Windows::Forms::Button^ button2;
public:
public:
String^ filename = "";
MyForm(void)
{
InitializeComponent();
//
//TODO: Add the constructor code here
//
}
protected:
/// <summary>
/// Clean up any resources being used.
/// </summary>
~MyForm()
{
if (components)
{
delete components;
}
}
private: System::Windows::Forms::TextBox^ pid;
private: System::Windows::Forms::Button^ insert;
protected:
private: System::Windows::Forms::MenuStrip^ menuStrip1;
private: System::Windows::Forms::ToolStripMenuItem^ fileToolStripMenuItem;
```

```
private: System::Windows::Forms::ToolStripMenuItem^ openToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^ saveToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^ exitToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^ editToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^ updateToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^ addTreeToolStripMenuItem;
private: System::Windows::Forms::Label^ label1;
private: System::Windows::Forms::Label^ label2;
private: System::Windows::Forms::TextBox^ name;
private: System::Windows::Forms::TextBox^ description;
private: System::Windows::Forms::Label^ label3;
private: System::Windows::Forms::Label^ label4;
private: System::Windows::Forms::TextBox^ phone_no;
private: System::Windows::Forms::TextBox^ spid;
private: System::Windows::Forms::Label^ label5;
private: System::Windows::Forms::OpenFileDialog^ openFD;
private: System::Windows::Forms::Button^ button1;
private:
protected:
private:
/// <summary>
/// Required designer variable.
/// </summary>
System::ComponentModel::Container^ components;
#pragma region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
void InitializeComponent(void)
{
this->pid = (gcnew System::Windows::Forms::TextBox());
this->insert = (gcnew System::Windows::Forms::Button());
```

```
                    this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());
this->fileToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->openToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->saveToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->exitToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->editToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->updateToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->addTreeToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->label1 = (gcnew System::Windows::Forms::Label());
this->label2 = (gcnew System::Windows::Forms::Label());
this->name = (gcnew System::Windows::Forms::TextBox());
this->description = (gcnew System::Windows::Forms::TextBox());
this->label3 = (gcnew System::Windows::Forms::Label());
this->label4 = (gcnew System::Windows::Forms::Label());
this->phone_no = (gcnew System::Windows::Forms::TextBox());
this->spid = (gcnew System::Windows::Forms::TextBox());
this->label5 = (gcnew System::Windows::Forms::Label());
this->openFD = (gcnew System::Windows::Forms::OpenFileDialog());
this->button1 = (gcnew System::Windows::Forms::Button());
this->savefd = (gcnew System::Windows::Forms::SaveFileDialog());
this->label8 = (gcnew System::Windows::Forms::Label());
this->display = (gcnew System::Windows::Forms::ListView());
this->displaypid = (gcnew System::Windows::Forms::ColumnHeader());
this->displayname = (gcnew System::Windows::Forms::ColumnHeader());
this->displaydescription = (gcnew System::Windows::Forms::ColumnHeader());
this->displayphone_no = (gcnew System::Windows::Forms::ColumnHeader());
this->label7 = (gcnew System::Windows::Forms::Label());
this->Search_display = (gcnew System::Windows::Forms::ListBox());
this->openFileDialog1 = (gcnew System::Windows::Forms::OpenFileDialog());
this->button2 = (gcnew System::Windows::Forms::Button());
this->menuStrip1->SuspendLayout();
this->SuspendLayout();
//
```

```
// pid
//
this->pid->Font = (gcnew System::Drawing::Font(L"Lucida Sans", 12,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(0)));
this->pid->Location = System::Drawing::Point(26, 131);
this->pid->Margin = System::Windows::Forms::Padding(4);
this->pid->Name = L"pid";
this->pid->Size = System::Drawing::Size(85, 55);
this->pid->TabIndex = 0;
//
// insert
//
this->insert->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(64)),
static_cast<System::Int32>(static_cast<System::Byte>(64)),
static_cast<System::Int32>(static_cast<System::Byte>(64)));
this->insert->Font = (gcnew System::Drawing::Font(L"Lucida Sans", 14.1F, System::Drawing::
FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(0)));
this->insert->ForeColor = System::Drawing::Color::White;
this->insert->Location = System::Drawing::Point(809, 73);
this->insert->Margin = System::Windows::Forms::Padding(4);
this->insert->Name = L"insert";
this->insert->Size = System::Drawing::Size(188, 52);
this->insert->TabIndex = 1;
this->insert->Text = L"Insert";
this->insert->UseVisualStyleBackColor = false;
this->insert->Click += gcnew System::EventHandler(this, &MyForm::button1_Click);
//
// menuStrip1
//
this->menuStrip1->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(64)),
static_cast<System::Int32>(static_cast<System::Byte>(64)),
static_cast<System::Int32>(static_cast<System::Byte>(64)));
this->menuStrip1->Font = (gcnew System::Drawing::Font(L"Lucida Sans", 8.1F, System::Drawing::
FontStyle::Regular, System::Drawing::GraphicsUnit::Point,static_cast<System::Byte>(0)));
this->menuStrip1->GripMargin = System::Windows::Forms::Padding(2, 2, 0, 2);
this->menuStrip1->ImageScalingSize = System::Drawing::Size(40, 40);
this->menuStrip1->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^
>(2) {
this->fileToolStripMenuItem,
        this->editToolStripMenuItem
});
this->menuStrip1->Location = System::Drawing::Point(0, 0);
this->menuStrip1->Name = L"menuStrip1";
this->menuStrip1->Padding = System::Windows::Forms::Padding(9, 3, 0, 3);
this->menuStrip1->Size = System::Drawing::Size(1366, 60);
this->menuStrip1->TabIndex = 2;
this->menuStrip1->Text = L"menuStrip1";
//
// fileToolStripMenuItem
//
this->fileToolStripMenuItem->BackColor = System::Drawing::Color::FromArgb (static_cast <System
::Int32> (static_cast<System::Byte>(64)),static_cast<System::Int32>(static_cast<System:: Byte>(64)),
static_cast<System::Int32>(static_cast<System::Byte>(64)));
this->fileToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::
Forms::ToolStripItem^ >(3) {

this->openToolStripMenuItem, this->saveToolStripMenuItem, this->exitToolStripMenuItem
});
this->fileToolStripMenuItem->ForeColor = System::Drawing::Color::White;
this->fileToolStripMenuItem->Name = L"fileToolStripMenuItem";
this->fileToolStripMenuItem->Size = System::Drawing::Size(86, 54);
this->fileToolStripMenuItem->Text = L"File";
this->fileToolStripMenuItem->Click += gcnew System::EventHandler(this,
&MyForm::fileToolStripMenuItem_Click);
//
```

```
// openToolStripMenuItem
//
this->openToolStripMenuItem->BackColor = System::Drawing::Color::Silver;
this->openToolStripMenuItem->Name = L"openToolStripMenuItem";
this->openToolStripMenuItem->Size = System::Drawing::Size(248, 54);
this->openToolStripMenuItem->Text = L"Open";
this->openToolStripMenuItem->Click += gcnew System::EventHandler(this,
&MyForm::openToolStripMenuItem_Click);
//
// saveToolStripMenuItem
//
this->saveToolStripMenuItem->BackColor = System::Drawing::Color::Silver;
this->saveToolStripMenuItem->Name = L"saveToolStripMenuItem";
this->saveToolStripMenuItem->Size = System::Drawing::Size(248, 54);
this->saveToolStripMenuItem->Text = L"Save";
this->saveToolStripMenuItem->Click += gcnew System::EventHandler(this,
&MyForm::saveToolStripMenuItem_Click);
//
// exitToolStripMenuItem
//
this->exitToolStripMenuItem->BackColor = System::Drawing::Color::Silver;
this->exitToolStripMenuItem->Name = L"exitToolStripMenuItem";
this->exitToolStripMenuItem->Size = System::Drawing::Size(248, 54);
this->exitToolStripMenuItem->Text = L"Exit";
this->exitToolStripMenuItem->Click += gcnew System::EventHandler(this,
&MyForm::exitToolStripMenuItem_Click);
//
// editToolStripMenuItem
//
this->editToolStripMenuItem->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(64)),
static_cast<System::Int32>(static_cast<System::Byte>(64)),static_cast<System::Int32>(static_cast<Sy
stem::Byte>(64)));
this->editToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::
Forms::ToolStripItem^  >(2) {
```

```
this->updateToolStripMenuItem,
 this->addTreeToolStripMenuItem
});
this->editToolStripMenuItem->ForeColor = System::Drawing::Color::White;
this->editToolStripMenuItem->Name = L"editToolStripMenuItem";
this->editToolStripMenuItem->Size = System::Drawing::Size(91, 54);
this->editToolStripMenuItem->Text = L"Edit";
//
// updateToolStripMenuItem
//
this->updateToolStripMenuItem->BackColor = System::Drawing::Color::Silver;
this->updateToolStripMenuItem->Name = L"updateToolStripMenuItem";
this->updateToolStripMenuItem->Size = System::Drawing::Size(300, 54);
this->updateToolStripMenuItem->Text = L"Update ";
this->updateToolStripMenuItem->Click += gcnew System::EventHandler(this,
&MyForm::updateToolStripMenuItem_Click);
//
// addTreeToolStripMenuItem
//
this->addTreeToolStripMenuItem->BackColor = System::Drawing::Color::Silver;
this->addTreeToolStripMenuItem->Name = L"addTreeToolStripMenuItem";
this->addTreeToolStripMenuItem->Size = System::Drawing::Size(300, 54);
this->addTreeToolStripMenuItem->Text = L"Add Tree";
this->addTreeToolStripMenuItem->Click += gcnew System::EventHandler(this,
&MyForm::addTreeToolStripMenuItem_Click);
//
// label1
//
this->label1->AutoSize = true;
this->label1->Font = (gcnew System::Drawing::Font(L"Lucida Sans", 14.1F, System::Drawing::
FontStyle::Bold, System::Drawing::GraphicsUnit::Point,static_cast<System::Byte>(0)));
this->label1->Location = System::Drawing::Point(27, 68);
this->label1->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);
this->label1->Name = L"label1";
```

```
this->label1->Size = System::Drawing::Size(98, 54);
this->label1->TabIndex = 3;
this->label1->Text = L"Pid";
//
// label2
//
this->label2->AutoSize = true;
this->label2->Font = (gcnew System::Drawing::Font(L"Lucida Sans", 14.1F,System::Drawing::
FontStyle::Bold, System::Drawing::GraphicsUnit::Point,static_cast<System::Byte>(0)));
this->label2->Location = System::Drawing::Point(163, 69);
this->label2->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);
this->label2->Name = L"label2";
this->label2->Size = System::Drawing::Size(162, 54);
this->label2->TabIndex = 5;
this->label2->Text = L"Name";
//
// name
//
this->name->Font = (gcnew System::Drawing::Font(L"Lucida Sans", 12, System::Drawing::FontStyle
::Regular, System::Drawing::GraphicsUnit::Point,static_cast<System::Byte>(0)));
this->name->Location = System::Drawing::Point(166, 131);
this->name->Margin = System::Windows::Forms::Padding(4);
this->name->Name = L"name";
this->name->Size = System::Drawing::Size(138, 55);
this->name->TabIndex = 6;
//
// description
//
this->description->Font = (gcnew System::Drawing::Font(L"Lucida Sans", 12, System::Drawing::
FontStyle ::Regular, System::Drawing::GraphicsUnit::Point,static_cast<System::Byte>(0)));
this->description->Location = System::Drawing::Point(347, 131);
this->description->Margin = System::Windows::Forms::Padding(4);
this->description->Name = L"description";
this->description->Size = System::Drawing::Size(153, 55);
```

```
this->description->TabIndex = 8;
//
// label3
//
this->label3->AutoSize = true;
this->label3->Font = (gcnew System::Drawing::Font(L"Lucida Sans", 14.1F, System::Drawing::
FontStyle::Bold, System::Drawing::GraphicsUnit::Point,static_cast<System::Byte>(0)));
this->label3->Location = System::Drawing::Point(348, 69);
this->label3->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);
this->label3->Name = L"label3";
this->label3->Size = System::Drawing::Size(302, 54);
this->label3->TabIndex = 9;
this->label3->Text = L"Description";
//
// label4
//
this->label4->AutoSize = true;
this->label4->Font = (gcnew System::Drawing::Font(L"Lucida Sans", 14.1F, System::Drawing::
FontStyle::Bold, System::Drawing::GraphicsUnit::Point,static_cast<System::Byte>(0)));
this->label4->Location = System::Drawing::Point(555, 72);
this->label4->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);
this->label4->Name = L"label4";
this->label4->Size = System::Drawing::Size(256, 54);
this->label4->TabIndex = 10;
this->label4->Text = L"Phone_no";
//
// phone_no
//
this->phone_no->Font = (gcnew System::Drawing::Font(L"Lucida Sans", 12, System::Drawing::
FontStyle::Regular, System::Drawing::GraphicsUnit::Point,static_cast<System::Byte>(0)));
this->phone_no->Location = System::Drawing::Point(564, 131);
this->phone_no->Margin = System::Windows::Forms::Padding(4);
this->phone_no->Name = L"phone_no";
this->phone_no->Size = System::Drawing::Size(150, 55);
```

```
this->phone_no->TabIndex = 11;
//
// spid
//
this->spid->Font = (gcnew System::Drawing::Font(L"Lucida Sans", 8.1F, System::Drawing::
FontStyle::Regular, System::Drawing::GraphicsUnit::Point,static_cast<System::Byte>(0)));
this->spid->Location = System::Drawing::Point(62, 433);
this->spid->Margin = System::Windows::Forms::Padding(4);
this->spid->Name = L"spid";
this->spid->Size = System::Drawing::Size(128, 39);
this->spid->TabIndex = 12;
//
// label5
//
this->label5->AutoSize = true;
this->label5->Font = (gcnew System::Drawing::Font(L"Lucida Sans", 9, System::Drawing::
FontStyle::Italic, System::Drawing::GraphicsUnit::Point,static_cast<System::Byte>(0)));
this->label5->ForeColor = System::Drawing::Color::Black;
this->label5->Location = System::Drawing::Point(44, 490);
this->label5->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);
this->label5->Name = L"label5";
this->label5->Size = System::Drawing::Size(417, 34);
this->label5->TabIndex = 13;
this->label5->Text = L"(Enter the PID to be search)";
//
// openFD
//
this->openFD->FileName = L"openFileDialog";
//
// button1
//
this->button1->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(64)),
static_cast<System::Int32>(static_cast<System::Byte>(64)),static_cast<System::Int32>(static_cast<Sy
stem::Byte>(64)));
```

```
this->button1->Font = (gcnew System::Drawing::Font(L"Lucida Sans", 14.1F,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(0)));
this->button1->ForeColor = System::Drawing::Color::White;
this->button1->Location = System::Drawing::Point(220, 429);
this->button1->Margin = System::Windows::Forms::Padding(4);
this->button1->Name = L"button1";
this->button1->Size = System::Drawing::Size(179, 53);
this->button1->TabIndex = 17;
this->button1->Text = L"Search";
this->button1->UseVisualStyleBackColor = false;
this->button1->Click += gcnew System::EventHandler(this, &MyForm::search);
//
// label8
//
this->label8->AutoSize = true;
this->label8->Font = (gcnew System::Drawing::Font(L"Lucida Sans", 15.9F, System::Drawing::
FontStyle::Bold, System::Drawing::GraphicsUnit::Point,static_cast<System::Byte>(0)));
this->label8->Location = System::Drawing::Point(34, 270);
this->label8->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);
this->label8->Name = L"label8";
this->label8->Size = System::Drawing::Size(0, 61);
this->label8->TabIndex = 18;
//
// display
//
this->display->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(224)),
static_cast<System::Int32>(static_cast<System::Byte>(224)),static_cast<System::Int32>(static_cast<S
ystem::Byte>(224)));
this->display->Columns->AddRange(gcnew cli::array< System::Windows::Forms::ColumnHeader^
>(4) {
this->displaypid, this->displayname,
this->displaydescription, this->displayphone_no
});
this->display->Font = (gcnew System::Drawing::Font(L"Lucida Sans", 15.9F, System::Drawing::
FontStyle::Bold, System::Drawing::GraphicsUnit::Point,static_cast<System::Byte>(0)));
this->display->HideSelection = false;
this->display->Location = System::Drawing::Point(21, 215);
this->display->Margin = System::Windows::Forms::Padding(4);
this->display->Name = L"display";
this->display->Size = System::Drawing::Size(1325, 190);
this->display->TabIndex = 20;
this->display->UseCompatibleStateImageBehavior = false;
this->display->View = System::Windows::Forms::View::Details;
//
// displaypid
//
this->displaypid->Text = L"PID";
this->displaypid->Width = 200;
//
// displayname
//
this->displayname->Text = L"NAME";
this->displayname->TextAlign = System::Windows::Forms::HorizontalAlignment::Center;
this->displayname->Width = 354;
//
// displaydescription
//
this->displaydescription->Text = L"DESCRIPTION";
this->displaydescription->TextAlign = System::Windows::Forms::HorizontalAlignment::Center;
this->displaydescription->Width = 276;
//
// displayphone_no
//
this->displayphone_no->Text = L"PHONE_NO";
this->displayphone_no->TextAlign = System::Windows::Forms::HorizontalAlignment::Center;
this->displayphone_no->Width = 519;
```

```
//
// label7
//
this->label7->AutoSize = true;
this->label7->BackColor = System::Drawing::Color::Cornsilk;
this->label7->Font = (gcnew System::Drawing::Font(L"Times New Roman", 8.25F, System::Drawing
::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,static_cast<System::Byte>(0)));
this->label7->ForeColor = System::Drawing::Color::Red;
this->label7->Location = System::Drawing::Point(38, 542);
this->label7->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);
this->label7->Name = L"label7";
this->label7->Size = System::Drawing::Size(0, 33);
this->label7->TabIndex = 23;
//
// Search_display
//
 this->Search_display->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(224)),
static_cast<System::Int32>(static_cast<System::Byte>(224)),
static_cast<System::Int32>(static_cast<System::Byte>(224)));
this->Search_display->Font = (gcnew System::Drawing::Font(L"Lucida Sans", 12, System::Drawing
::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,static_cast<System::Byte>(0)));
this->Search_display->FormattingEnabled = true;
this->Search_display->ItemHeight = 45;
this->Search_display->Location = System::Drawing::Point(475, 430);
this->Search_display->Margin = System::Windows::Forms::Padding(4);
this->Search_display->Name = L"Search_display";
this->Search_display->Size = System::Drawing::Size(878, 94);
this->Search_display->TabIndex = 24;
this->Search_display->SelectedIndexChanged += gcnew System::EventHandler(this,
&MyForm::sdisplay_SelectedIndexChanged);
//
// openFileDialog1
//
this->openFileDialog1->FileName = L"openFileDialog1";
```

```
//
// button2
//
this->button2->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(64)),
static_cast<System::Int32>(static_cast<System::Byte>(64)),
static_cast<System::Int32>(static_cast<System::Byte>(64)));
this->button2->Font = (gcnew System::Drawing::Font(L"Lucida Sans", 14.1F, System::Drawing
::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,static_cast<System::Byte>(0)));
this->button2->ForeColor = System::Drawing::Color::White;
this->button2->Location = System::Drawing::Point(809, 132);
this->button2->Name = L"button2";
this->button2->Size = System::Drawing::Size(188, 52);
this->button2->TabIndex = 25;
this->button2->Text = L"Refresh";
this->button2->UseVisualStyleBackColor = false;
  this->button2->Click += gcnew System::EventHandler(this, &MyForm::button2_Click);
//
// MyForm
//
this->AutoScaleDimensions = System::Drawing::SizeF(22, 46);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->BackColor = System::Drawing::Color::WhiteSmoke;
this->ClientSize = System::Drawing::Size(1366, 539);
this->Controls->Add(this->button2);
this->Controls->Add(this->Search_display);
this->Controls->Add(this->label7);
this->Controls->Add(this->display);
this->Controls->Add(this->label8);
this->Controls->Add(this->button1);
this->Controls->Add(this->label5);
this->Controls->Add(this->spid);
this->Controls->Add(this->phone_no);
this->Controls->Add(this->label4);
```

```
this->Controls->Add(this->label3);
this->Controls->Add(this->description);
this->Controls->Add(this->name);
this->Controls->Add(this->label2);
this->Controls->Add(this->label1);
this->Controls->Add(this->insert);
this->Controls->Add(this->pid);
this->Controls->Add(this->menuStrip1);
this->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing
::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,static_cast<System::Byte>(0)));
this->MainMenuStrip = this->menuStrip1;
this->Margin = System::Windows::Forms::Padding(4);
this->MaximumSize = System::Drawing::Size(1398, 627);
this->MinimumSize = System::Drawing::Size(1398, 627);
this->Name = L"MyForm";
this->Text = L"Hospital  Management System";
this->FormClosing += gcnew System::Windows::Forms::FormClosingEventHandler(this, &MyForm
::MyForm_FormClosing);
this->Load += gcnew System::EventHandler(this, &MyForm::MyForm_Load);
this->menuStrip1->ResumeLayout(false);
this->menuStrip1->PerformLayout();
this->ResumeLayout(false);
this->PerformLayout();
}
#pragma endregion
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{
if (pid->Text == "" || name->Text == "" || description->Text == "" || phone_no->Text == "")
{
MessageBox::Show("Enter Value Please.", "Error");
}
else
{
msclr::interop::marshal_context con;
s.pid = con.marshal_as< std::string >(pid->Text);
s.name = con.marshal_as<std::string>(name->Text);
s.description = con.marshal_as<std::string>(description->Text);
s.phone_no = con.marshal_as<std::string>(phone_no->Text);
b.insert(s);
if (b.ds != 0) {
label7->Text = gcnew String(b.st[0].c_str());
pid->Text = ""; name->Text = ""; description->Text = "";
phone_no->Text = "";
return;
}
flags = 0;
pid->Text = ""; name->Text = ""; description->Text = ""; phone_no->Text = "";
}
}
private: System::Void fileToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e)
{
}
private: System::Void exitToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e)
{


this->Close();
}
private: System::Void openToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e)
{
openFD->Filter = "rft files (*.rft)|*.rft";
openFD->FileName = "";
openFD->ShowDialog();
if (openFD->FileName == "")
return;
filename = openFD->FileName;
this->Text = filename;
msclr::interop::marshal_context con;
string s = con.marshal_as<std::string>(filename);
```

```
cout << s << endl;
b = bplustree();
if (b.head == NULL) {
cout << "new";
}
for (int i = 0; i < 1000; i++) {
b.st[i] = "";
}
b.ds = 0;
StreamReader^ r = File::OpenText(filename);
int x = 0;
String^ line;
while ((line = r->ReadLine()) != nullptr) {
b.st[x++] = con.marshal_as<std::string>(line);
}
cout << x << endl;
b.ds = x;
b.read();
r->Close();
}
private: System::Void contentDisplayToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e) {
if (b.head == NULL)
{
MessageBox::Show("NO Value Inserted", "Error");
return;
}
display->Items->Clear();
for (int i = 0; i < 1000; i++) {
b.st[i] = "";
}
b.ds = 0;
b.display();
for (int i = 0; i < b.ds; i++) {
```

```
String^ s = gcnew String((b.sd[i].pid).c_str());
ListViewItem^ dis = gcnew ListViewItem(s);
dis->SubItems->Add(gcnew String((b.sd[i].name).c_str()));
dis->SubItems->Add(gcnew String((b.sd[i].description).c_str()));
dis->SubItems->Add(gcnew String((b.sd[i].phone_no).c_str()));
display->Items->Add(dis);
}
label7->Text = "";
}
private: System::Void search(System::Object^ sender, System::EventArgs^ e) {
Search_display->Items->Clear();
label7->Text = "";
for (int i = 0; i < 1000; i++) {
b.st[i] = "";
}
b.ds = 0;
if (spid->Text == "")

{
MessageBox::Show("Enter Value Please.", "Error");
}
else
{
msclr::interop::marshal_context con;
s.pid = con.marshal_as<std::string>(spid->Text);
cout << s.pid << endl;
s.name = "";
s.description = "";
s.phone_no = "";
b.search(s);
if (b.flag == 0) {
label7->Text = gcnew String(b.st[0].c_str());
return;
```

```
}
for (int i = 0; i < b.ds; i++) {
String^ s = "The result for search is :-";
Search_display->Items->Add(s);
s = "\t\tPID:-" + gcnew String((b.sd[i].pid).c_str());
Search_display->Items->Add(s);
s = "\t\tName:-" + gcnew String((b.sd[i].name).c_str());
Search_display->Items->Add(s);
s = "\t\tDescription:-" + gcnew String((b.sd[i].description).c_str());
Search_display->Items->Add(s);
s = "\t\tPhone_no:-" + gcnew String((b.sd[i].phone_no).c_str());
Search_display->Items->Add(s);
}
b.ds = 0;
spid->Text = "";
}
}


private: System::Void saveToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e)
{
if (flags != 1) {
if (filename == "") {
savefd->FileName = "";
savefd->ShowDialog();
if (savefd->FileName == "")
return;
filename = savefd->FileName;
this->Text = filename;
StreamWriter^ outp = File::CreateText(filename);
b.write();
for (int i = 0; i < b.ds; i++) {
String^ stud = gcnew String(b.st[i].c_str());
cout << b.st[i];
```

```
outp->WriteLine(stud);
}
flags = 1;
outp->Close();
}
else {
StreamWriter^ out = File::CreateText(filename);
b.write();
for (int i = 0; i < b.ds; i++) {
String^ stud = gcnew String(b.st[i].c_str());
cout << b.st[i];
out->WriteLine(stud);

}
out->Close();
flags = 1;
}
}
}
 private: System::Void MyForm_FormClosing(System::Object^ sender,System:: Windows::Forms::
FormClosingEventArgs^ e) {
if (flags != 1) {
System::Windows::Forms::DialogResult r;
r = MessageBox::Show("Do you really want to close the program without saving ? ", "project",
MessageBoxButtons::YesNoCancel);
if (r == System::Windows::Forms::DialogResult::Yes) {
}
else if (r == System::Windows::Forms::DialogResult::No) {
if (filename == "") {
savefd->FileName = "*.rft";
savefd->ShowDialog();
if (savefd->FileName == "")
return;
filename = savefd->FileName;
```

```
this->Text = filename;
StreamWriter^ outp = File::CreateText(filename);
b.write();
for (int i = 0; i < b.ds; i++) {
String^ stud = gcnew String(b.st[i].c_str());
cout << b.st[i];
outp->WriteLine(stud);
}
outp->Close();
flags = 1;
}
else {
StreamWriter^ out = File::CreateText(filename);
b.write();
for (int i = 0; i < b.ds; i++) {
String^ stud = gcnew String(b.st[i].c_str());
cout << b.st[i];
out->WriteLine(stud);
}
out->Close();
flags = 1;
}
}
else {
e->Cancel = true;
}
}
}
private: System::Void addTreeToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e) {
openFD->Filter = "rft files (*.rft)|*.rft";
openFD->FileName = "";
openFD->ShowDialog();
```

```
if (openFD->FileName == "")
return;
String^ filename1 = openFD->FileName;
msclr::interop::marshal_context con;
string s = con.marshal_as<std::string>(filename1);
cout << s << endl;
for (int i = 0; i < 1000; i++) {
b.st[i] = "";
}
StreamReader^ r = File::OpenText(filename1);
int x = 0;
String^ line;
while ((line = r->ReadLine()) != nullptr) {
b.st[x++] = con.marshal_as<std::string>(line);
}
b.ds = x;
b.read();
r->Close();
}
private: System::Void updateToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e) {

if (pid->Text == "" || name->Text == "" || description->Text == "" || phone_no->Text == "")
{
MessageBox::Show("Enter Value Please.", "Error");
}
else
{
msclr::interop::marshal_context con;
s.pid = con.marshal_as< std::string >(pid->Text);
s.name = con.marshal_as<std::string>(name->Text);
s.description = con.marshal_as<std::string>(description->Text);
s.phone_no = con.marshal_as<std::string>(phone_no->Text);
```

```
                    pid->Text = ""; name->Text = ""; description->Text = ""; phone_no->Text ="";
Project2::MyForm1 f;
f.ShowDialog();
StreamReader^ r = File::OpenText("project.txt");
s2.pid = con.marshal_as< std::string >(r->ReadLine());
b = b.update(s2, s);
if (b.ds != 0) {
label7->Text = gcnew String(b.st[0].c_str());
return;
}
r->Close();
flags = 0;
}
}
private: System::Void sdisplay_SelectedIndexChanged(System::Object^ sender,
System::EventArgs^ e) {
}
private: System::Void MyForm_Load(System::Object^ sender, System::EventArgs^ e) {
}
private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e) {
if (b.head == NULL)
{
MessageBox::Show("NO Value Inserted", "Error");
return;
}
display->Items->Clear();
for (int i = 0; i < 1000; i++) {
b.st[i] = "";
}
b.ds = 0;
b.display();
for (int i = 0; i < b.ds; i++) {
String^ s = gcnew String((b.sd[i].pid).c_str());
```

```
ListViewItem^ dis = gcnew ListViewItem(s);
dis->SubItems->Add(gcnew String((b.sd[i].name).c_str()));
dis->SubItems->Add(gcnew String((b.sd[i].description).c_str()));
dis->SubItems->Add(gcnew String((b.sd[i].phone_no).c_str()));
display->Items->Add(dis);
}
label7->Text = "";
}
};
}
```

Update.h

```
#pragma once
namespace Project2 {
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
using namespace System::IO;
/// <summary>
/// Summary for MyForm1
/// </summary>
public ref class MyForm1 : public System::Windows::Forms::Form
{
public:
MyForm1(void)
{
InitializeComponent();
//
//TODO: Add the constructor code here
//
```

```
}
protected:
/// <summary>
/// Clean up any resources being used.
/// </summary>
~MyForm1()
{

if (components)
{
delete components;
}
}
private: System::Windows::Forms::TextBox^ pid;
private: System::Windows::Forms::Label^ label1;
private: System::Windows::Forms::Button^ button1;
private: System::Windows::Forms::Label^ label4;
protected:
protected:
private:
/// <summary>
/// Required designer variable.
/// </summary>
        System::ComponentModel::Container^ components;
#pragma region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
void InitializeComponent(void)
{
this->pid = (gcnew System::Windows::Forms::TextBox());
```

```
this->label1 = (gcnew System::Windows::Forms::Label());

this->button1 = (gcnew System::Windows::Forms::Button());

this->label4 = (gcnew System::Windows::Forms::Label());

this->SuspendLayout();

//

// pid

//

this->pid->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::
FontStyle ::Regular, System::Drawing::GraphicsUnit::Point,static_cast<System::Byte>(0)));

this->pid->Location = System::Drawing::Point(304, 141);

this->pid->Margin = System::Windows::Forms::Padding(8, 7, 8, 7);

this->pid->Name = L"pid";

this->pid->Size = System::Drawing::Size(553, 53);

this->pid->TabIndex = 0;

//

// label1

//

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::
FontStyle ::Regular, System::Drawing::GraphicsUnit::Point,static_cast<System::Byte>(0)));

this->label1->Location = System::Drawing::Point(117, 148);

this->label1->Margin = System::Windows::Forms::Padding(8, 0, 8, 0);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(84, 46);

this->label1->TabIndex = 2;

this->label1->Text = L"PID";

this->label1->TextAlign = System::Drawing::ContentAlignment::MiddleCenter;

//

// button1

//

this->button1->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(64)),
static_cast<System::Int32>(static_cast<System::Byte>(64)),static_cast<System::Int32>(static_cast<System::Byte>(64)));
this->button1->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,static_cast<System::Byte>(0)));
this->button1->ForeColor = System::Drawing::Color::White;
this->button1->Location = System::Drawing::Point(445, 217);
this->button1->Margin = System::Windows::Forms::Padding(8, 7, 8, 7);
this->button1->Name = L"button1";
this->button1->Size = System::Drawing::Size(512, 64);
this->button1->TabIndex = 4;
this->button1->Text = L"Update";
this->button1->UseVisualStyleBackColor = false;
this->button1->Click += gcnew System::EventHandler(this, &MyForm1::button1_Click);
//
// label4
//
this->label4->AutoSize = true;
this->label4->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,static_cast<System::Byte>(0)));
this->label4->Location = System::Drawing::Point(117, 62);
this->label4->Margin = System::Windows::Forms::Padding(8, 0, 8, 0);
this->label4->Name = L"label4";
this->label4->Size = System::Drawing::Size(466, 46);
this->label4->TabIndex = 6;
this->label4->Text = L"Enter the Pid to be update ";
//
// MyForm1
//
this->AutoScaleDimensions = System::Drawing::SizeF(16, 31);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->BackColor = System::Drawing::Color::Silver;
this->ClientSize = System::Drawing::Size(997, 303);
this->Controls->Add(this->label4);
```

```
this->Controls->Add(this->button1);
this->Controls->Add(this->label1);
this->Controls->Add(this->pid);
this->Margin = System::Windows::Forms::Padding(8, 7, 8, 7);
this->Name = L"MyForm1";
this->Text = L"Update";
this->ResumeLayout(false);
this->PerformLayout();

}
#pragma endregion
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{
StreamWriter^ outp = File::CreateText("project.txt");
outp->WriteLine(pid->Text);
outp->Close();
this->Close();
}
};
}
```