# Contents

[Contents 1](#_Toc48666798)

[1. INTRODUCTION 3](#_Toc48666799)

[1.1 General Overview 3](#_Toc48666800)

[1.2 Problem definition 3](#_Toc48666801)

[1.3 Objectives 4](#_Toc48666802)

[2. HARDWARE AND SOFTWARE REQUIREMENTS 5](#_Toc48666803)

[2.1 Hardware requirements 5](#_Toc48666804)

[2.2 Software requirements 5](#_Toc48666805)

[3. SYSTEM DESIGN 6](#_Toc48666806)

[3.1 Context-flow Diagram 6](#_Toc48666807)

[3.2 SYSTEM DESIGN 6](#_Toc48666808)

[3.3 Data-flow Diagram 7](#_Toc48666809)

[3.3.1 Data-flow diagram for insertion 8](#_Toc48666810)

[3.3.2 Data-flow diagram for search 9](#_Toc48666811)

[3.3.3 Data-flow diagram for display 10](#_Toc48666812)

[3.3.4 Data-flow diagram for delete 11](#_Toc48666813)

[3.3.5 Data-flow diagram for Update 12](#_Toc48666814)

[4. IMPLEMENTATION 13](#_Toc48666815)

[4.1 B+TREE 13](#_Toc48666816)

[4.2 Rules for B+ Tree 13](#_Toc48666817)

[4.3 Why use B+ Tree 14](#_Toc48666818)

[4.4 B+ Tree vs. B Tree 14](#_Toc48666819)

[4.5 Searching a record in B+ Tree 15](#_Toc48666820)

[4.5.1 Search Operation Algorithm 15](#_Toc48666821)

[Output: 15](#_Toc48666822)

[4.6 Insertion in B+ tree 15](#_Toc48666823)

[4.6.1 Insert Operation Algorithm 16](#_Toc48666824)

[5. SOFTWARE TESTING 18](#_Toc48666825)

[5.1 Unit Testing 18](#_Toc48666826)

[5.2 Unit Testing Table 20](#_Toc48666827)

[6. RESULTS 21](#_Toc48666828)

[7. Source code 28](#_Toc48666829)

[7.1 B+Tree Class 28](#_Toc48666830)

[7.2 Window/Interface Code 38](#_Toc48666831)

# INTRODUCTION

## 1.1 General Overview

The objective of Student information Management System is to allow the administrator of any organization to edit and find out the personal details of a student and allows the student to keep up to date his profile .It’ll also facilitate keeping all the records of students, such as their USN, name, Marks, Attendance etc. So all the information about an student will be available in a few seconds.

Overall, it’ll make Student Information Management an easier job for the administrator and the student of any organization. The main purpose of this project is intended to help any organization to maintain and manage its student’s personal data

## 1.2 Problem definition

The Student Information Management System includes various information regarding the Student’s. The Student Information Management System needs details of the Student’s such as the USN, Student’s name, Marks and Attendance. So that they can be kept track of their information.

This project helps to handle this problem in a efficient manner. In this the Student’s details are entered i.e. their USN, name, Marks and attendance are being stored us b+tree and in form of text file. This project allow you to insert the student details and searching student details and their details can be edited.

This project also helps to in insertion of data in such 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 students can be recorded and maintained and used to view and search the detail of the students. This project is all about how to ease the problem of maintaining student data and store in it in systemic way.

The Student Information Management System facilitates the data storing, creating a file for that data , deleting and updating this data , adding other tree ,searching the data , etc . The main aim was to create a graphical user interface so that the user can effectively interact with system and to implement the file handling using the concepts of B+Tree so that the Student’s details can be maintained and can be retrieved for the purpose of confirmation.

The record contains the USN, Student’s name, Marks and Attendance for the store ,inserting ,deleting ,updating and searching.

# HARDWARE AND SOFTWARE REQUIREMENTS

## 2.1 Hardware requirements

1. Processor: Pentium IV or Core i3, i5, i7, i9.

2. RAM: 2GB or higher.

3. HDD: 40GB or higher.

4. Display: 1024 \* 768 Resolution Color Monitor.

5. Keyboard: 104 keys.

## 2.2 Software requirements

1. Operating System: Windows XP, 7, 8, 10.

2. IDE: Microsoft Visual Studio 2010 or higher version.

* 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’s required to develop the application and the run the application.
* Microsoft Visual Studio is used to develop the application on windows platform.
* The project is developed in C++ language with concepts of File handling and B-Trees.

# SYSTEM DESIGN

## 3.1 Context-flow Diagram

The context diagram is used to establish the context and boundaries of the system to be modeled: which things are inside and outside of the system being modeled, 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.

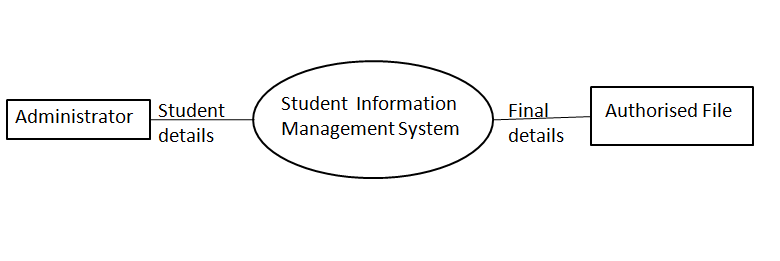


Fig 3.1: Context flow Diagram

## 3.2 SYSTEM DESIGN

System design 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.

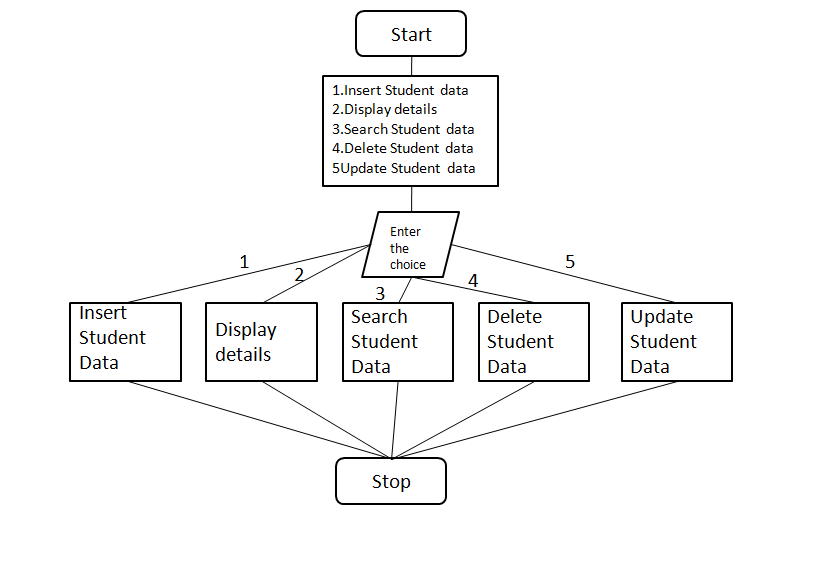


Fig 3.2:System Design for Student Information Management System

## 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.

### Data-flow diagram for insertion

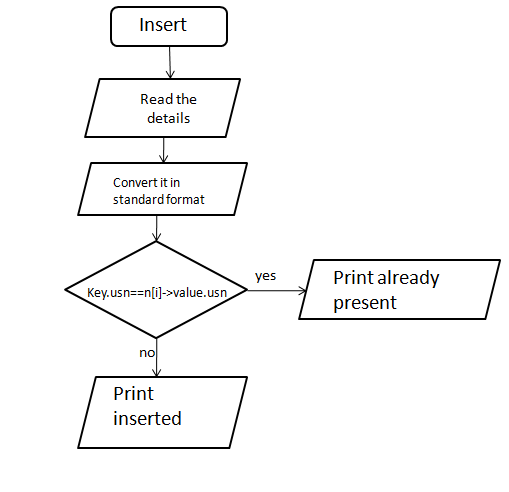


Fig. 3.3: Data-flow diagram for insertion

In the Student Information Management System, the Administrator can insert the details of student’s like USN, Student name, Marks and Attendance. Then the data is validated and checked for errors and converted in standard form, if there are no errors or same usn is not there in list then the data is saved to the tree. The data that is saved in the tree can be accessed later for other use.

### Data-flow diagram for search

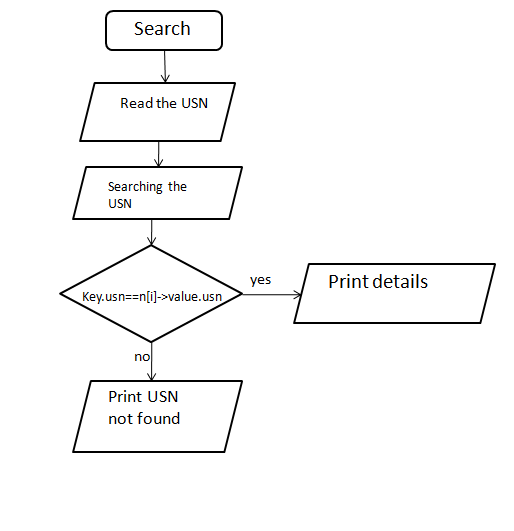


Fig. 3.4: Data-flow diagram for search

In the Student Information Management System, the administration can search the details of student by using UNS, if the USN is present then the system displays that the Student details and displays a message suitably and if the USN is not present then the system will display suitable error message that is ‘USN not Found‘ has to be displayed.

### Data-flow diagram for display

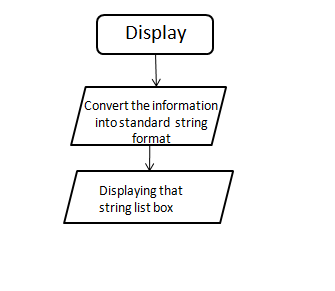


Fig. 3.5: Data-flow diagram for display

In the Student Information management System, the Administer can view all the details of the Student. First data is converted into a string and that string is than add to the list box for the displaying the Contain in GUI mode.

### Data-flow diagram for delete

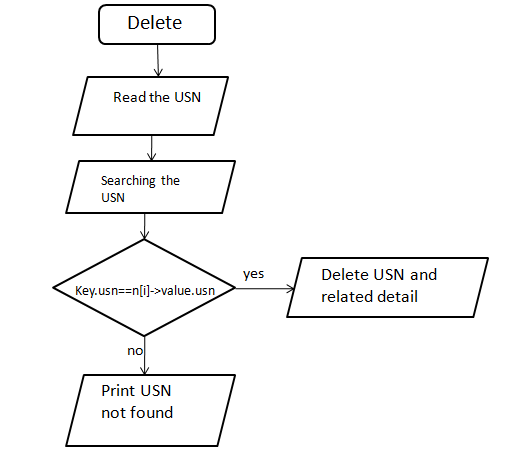


Fig. 3.6: Data-flow diagram for delete

In the Student Information Management System, the administration can delete the details of student by using UNS, if the USN is present then the system deletes that the Student details and displays a message suitably and if the USN is not present then the system will display suitable error message that is ‘USN not Found‘ has to be displayed.

### Data-flow diagram for Update

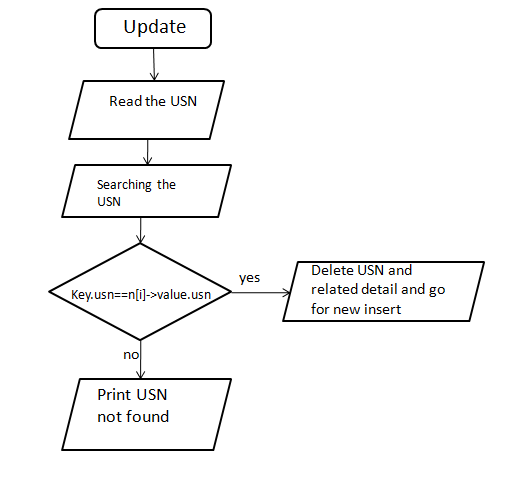


Fig. 3.7 Data-flow diagram for Update

In the Student Information Management System, the administration can Update the details of student by using UNS, if the USN is present then the system replace that the Student details and displays a message suitably and if the USN is not present then the system will display suitable error message that is ‘USN not Found‘ has to be displayed.

# IMPLEMENTATION

## 4.1 B+TREE

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 are linked together in the form of singly linked lists to make the search queries more efficient.

B+ Tree are used to store the 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 are often called index nodes. A B+ tree of order 3 is shown in the following figure.

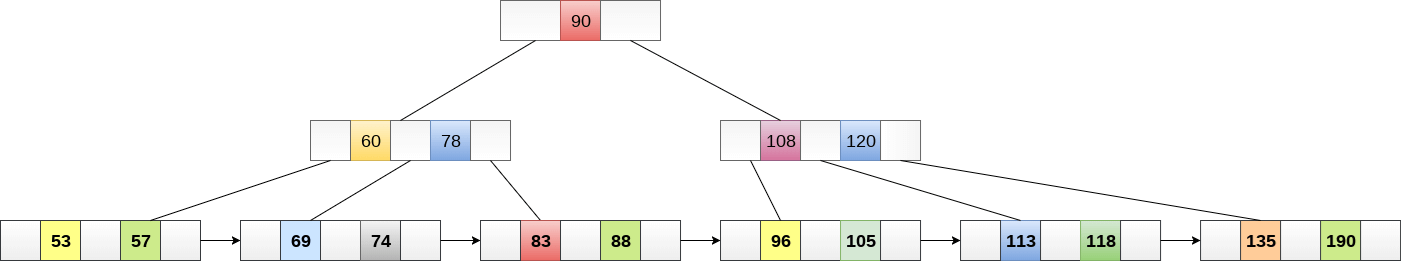


Fig. 4.1:B+ tree

## 4.2 Rules for B+ Tree

Here are essential rules for B+ Tree.

* Leaves are used to store data records.
* It 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.
* The root has a minimum of two children.

## 4.3 Why use B+ Tree

Here, are reasons for using B+ Tree:

* Key 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 B+ Tree vs. B Tree

Here, are the main differences between B+ Tree vs. B Tree

|  |  |
| --- | --- |
| **B + Tree** | **B Tree** |
| Search keys can be repeated. | Search keys cannot be redundant. |
| Data is only saved on the leaf nodes. | Both leaf nodes and internal nodes can store data |
| Data stored on the leaf node makes the search more accurate and faster. | Searching is slow due to data stored on Leaf and internal nodes. |
| Deletion is not difficult as an element is only removed from a leaf node. | Deletion of elements is a complicated and time-consuming process. |
| Linked leaf nodes make the search efficient and quick. | You cannot link leaf nodes. |

## ****4.5 Searching a record in B+ Tree****

Suppose we want to search 65 in the below B+ tree structure. First we will fetch for the intermediary node which will direct to the leaf node that can contain record for 65. So we find branch between 50 and 75 nodes in the intermediary node. Then we will be redirected to the third leaf node at the end. Here DBMS will perform sequential search to find 65. Suppose, instead of 65, we have to search for 60. What will happen in this case? We will not be able to find in the leaf node. No insertions/update/delete is allowed during the search in B+ tree.

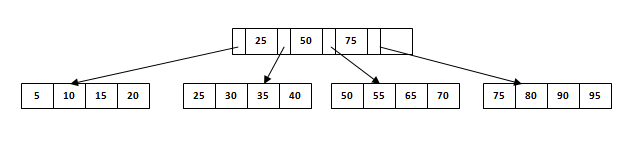


Fig. 4.2: Searching a record in B+ Tree

### 4.5.1 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."

### Output:

The matched record set against the exact key is displayed to the user; otherwise, a failed attempt is shown to the user.

## 4.6 ****Insertion in B+ tree****

Suppose we have to insert a record 60 in below structure. It will go to 3rd leaf node after 55. Since it is a balanced tree and that leaf node is already full, we cannot insert the record there. But it should be inserted there without affecting the fill factor, balance and order. So the only option here is to split the leaf node. But how do we split the nodes?

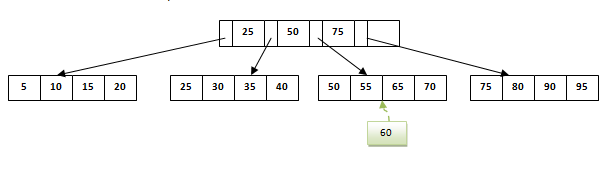


Fig. 4.3: Inserting in B+ tree

The 3rd leaf node should have values (50, 55, 60, 65, 70) and its current root node is 50. We will split the leaf node in the middle so that its balance is not altered. So we can group (50, 55) and (60, 65, 70) into 2 leaf nodes. If these two has to be leaf nodes, the intermediary node cannot branch from 50. It should have 60 added to it and then we can have pointers to new leaf node.

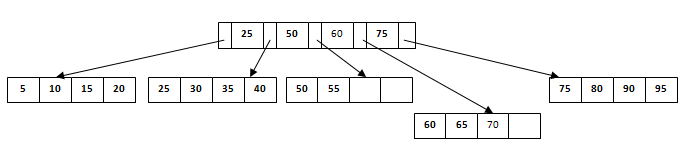


Fig. 4.4: Inserting in B+ tree

This is how we insert a new entry when there is overflow. In normal scenario, it is simple to find the node where it fits and place it in that leaf node.

### ****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.

i. Similarly, insert a key in the center of the top-level node in the hierarchy of the Tree.

d. 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.

**Output**:

The algorithm will determine the element and successfully insert it in the required leaf node.

# SOFTWARE TESTING

Software Testing is an empirical investigation conducted to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which respect to the context in which it is intended to operate. Software Testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks at implementation of the software. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs. It can also be stated as the process of validating and verifying that a software program/application/product meets the business and technical requirements that guide its design and development, so that it works as expected and can be implemented with the same characteristics.

## 5.1 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. The various controls are tested to ensure that each performs its action as required.

Sometimes software developers attempt to save time by doing minimal unit testing. This is a myth because skipping on unit testing leads to higher Defect fixing costs during System Testing, Integration Testing and even Beta Testing after the application is completed. Proper unit testing done during the development stage saves both time and money in the end.

## 5.2 Unit Testing Table

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO | INPUT | OUTPUT | Remake |
| 1 | As the insert operation is used with null and not null characters. | Display appropriate message for null and not null characters. | PASS |
| 2 | As the display operation is used with insert and without insert. | Display appropriate message for insert and without insert display. | PASS |
| 3 | As the delete operation is used with null and not null characters. | Display appropriate message for null and not null characters. | PASS |
| 4 | As the update operation is used with null and not null characters. | Display appropriate message for null and not null characters. | PASS |
| 5 | As the search operation is used with null and not null characters. | Display appropriate message for null and not null characters. | PASS |

Table 5‑1 Unit Testing Table

# RESULTS

The project is compiled and executed on Microsoft Visual C++. We have put in few screen shots in here to show the working of our Application.

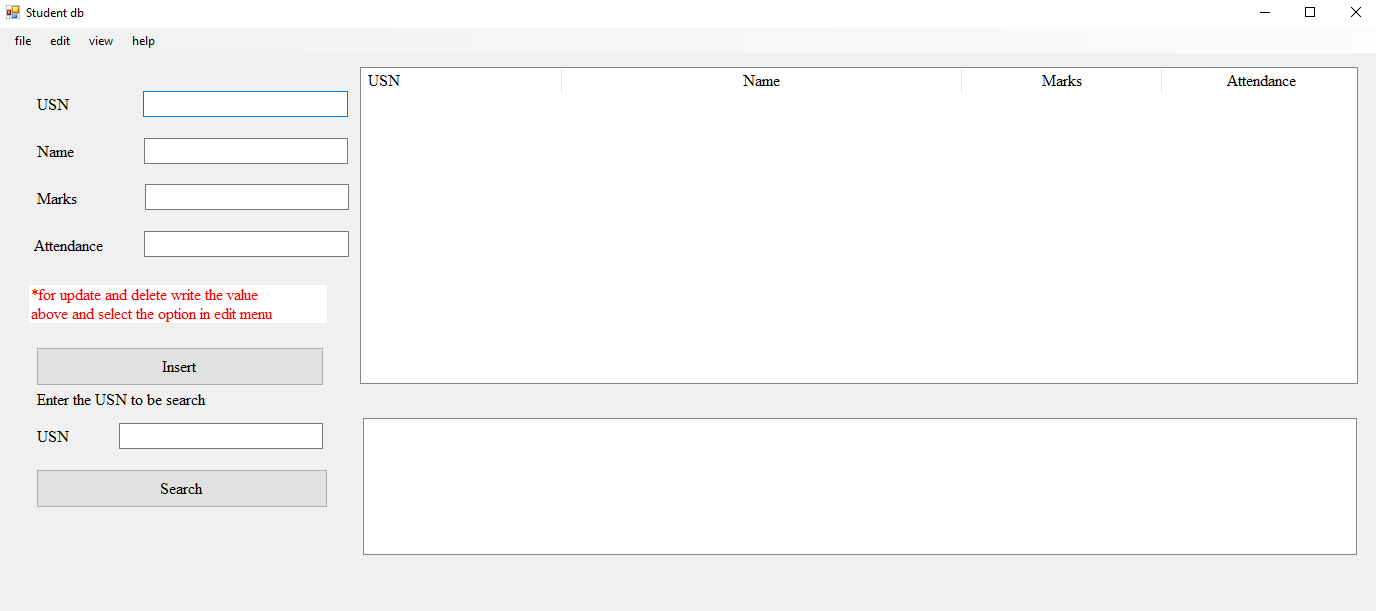


Fig. 6.1: Student Information Management System Main display

In the Student Information Management System main display page, the administrator can perform various options such as insert a new Student record , searching a student record, display all the record , deleting a student record, update a student record and add a tree to a b +tree record. The administrator can save, open and save as record in .rft format in text file.

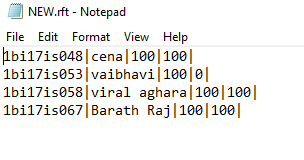


Fig. 6.2: File in which the record is stored

In this way the record gets stored in file.

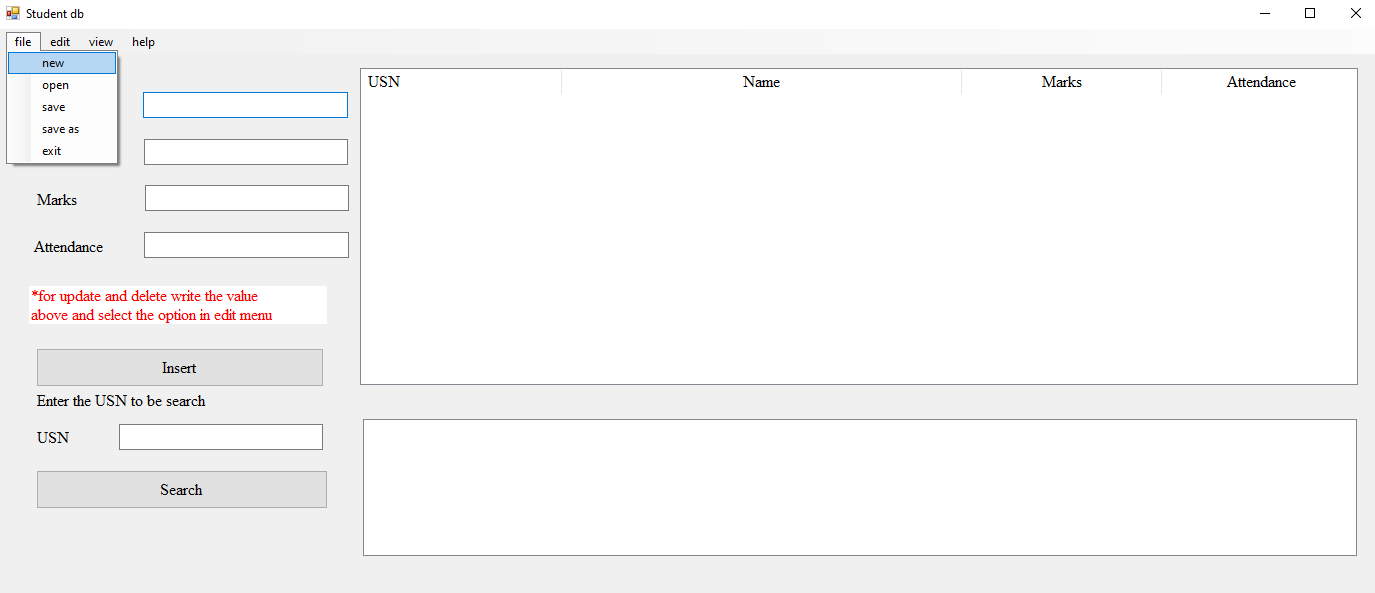


Fig. 6.3: File tab

This is file tab in menu. It contains the options new ,open ,save ,save as and exit option are present .

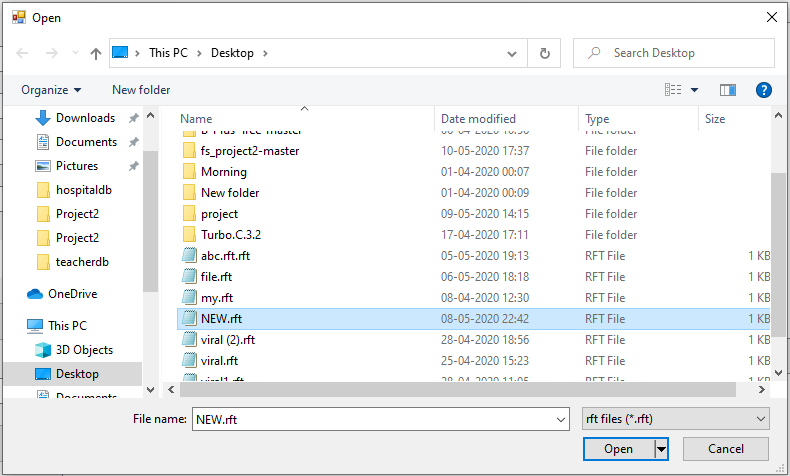


Fig. 6.4 : Open Dialog

This is a open dialog, it only allows the user to select only .rft this open that file in system and insert all the data in b + tree and can be viewed by content display in view tab

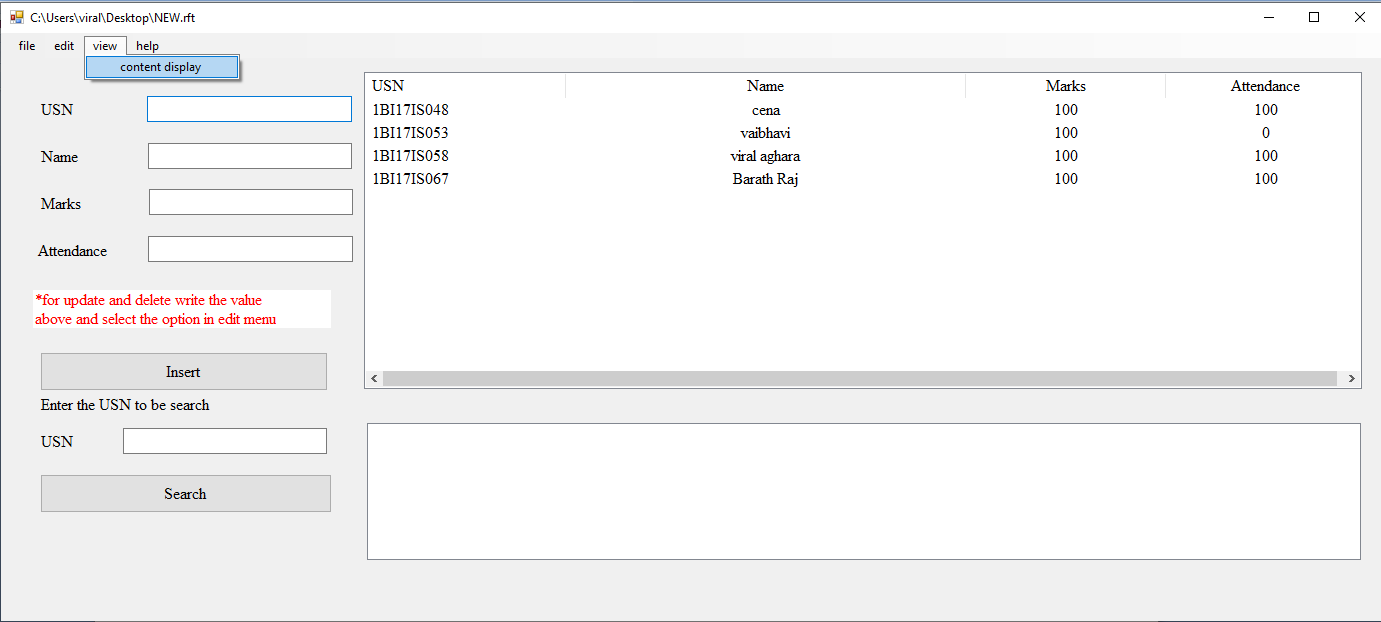


Fig. 6.5: content display

In this way the content can be seen by view->content display on clicking the record in file can be seen.

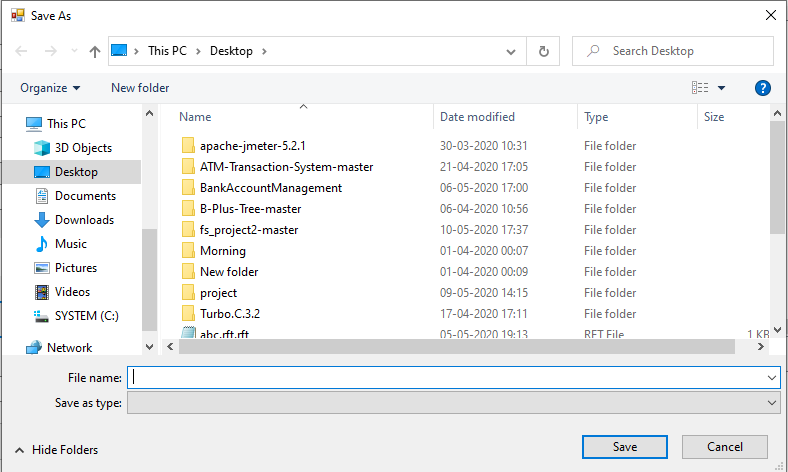


Fig. 6.6: Save as dialog

This can be seen when file is not opened you go for save or when you go for save as directly. This will save your file in .rft format. The Exit in file tab exit button exits the software.

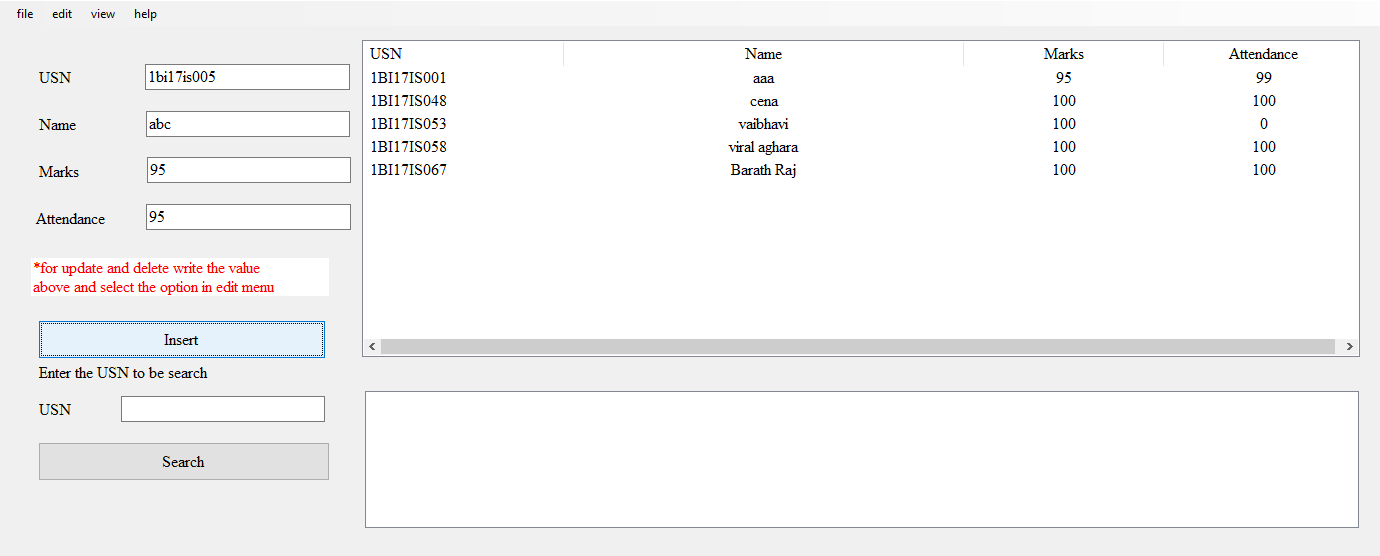


Fig. 6.7: Insert Function

When we press the insert it read the value from textbox are converted in standard form and insert in the B + tree . The key used for splitting and sorting is USN. Once inserted can be seen using contain display button.

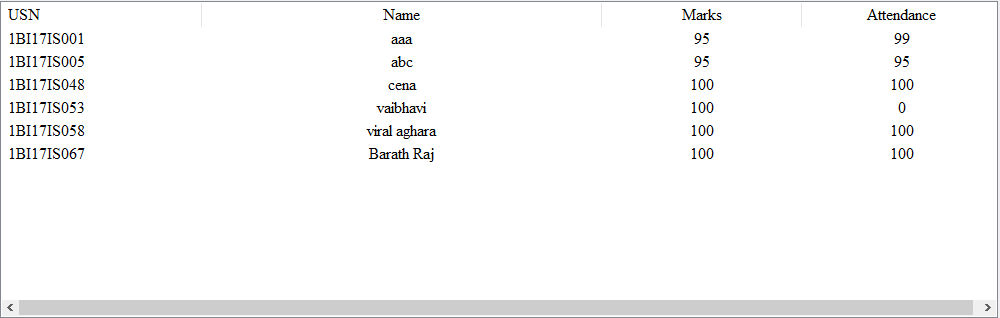


Fig. 6.8: Display for insert

You can see when value is inserted it gets sorted and the tree gets balanced when a value is inserted.

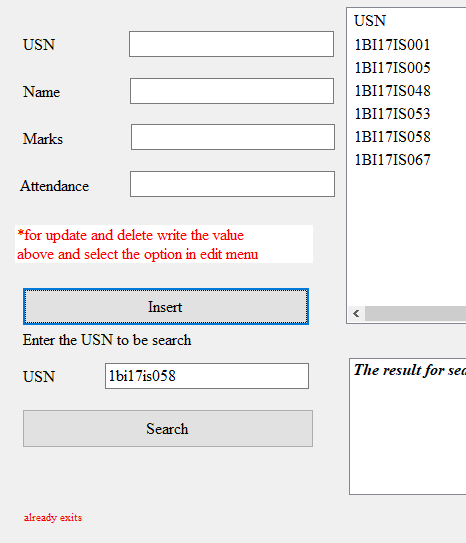


Fig. 6.9:Error of insert

When an exiting usn is inserted it display error already exiting.



Fig. 6.10: Search Function

When a usn is searched it will display student detail in second display box.

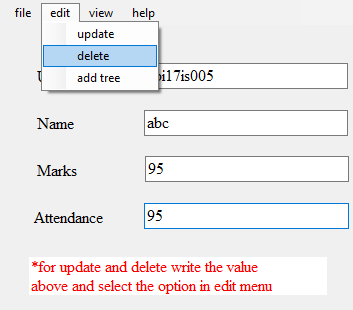


Fig. 6.11:Delete Function

The delete function will read all the value from textboxs and finds the usn and deletes that record.

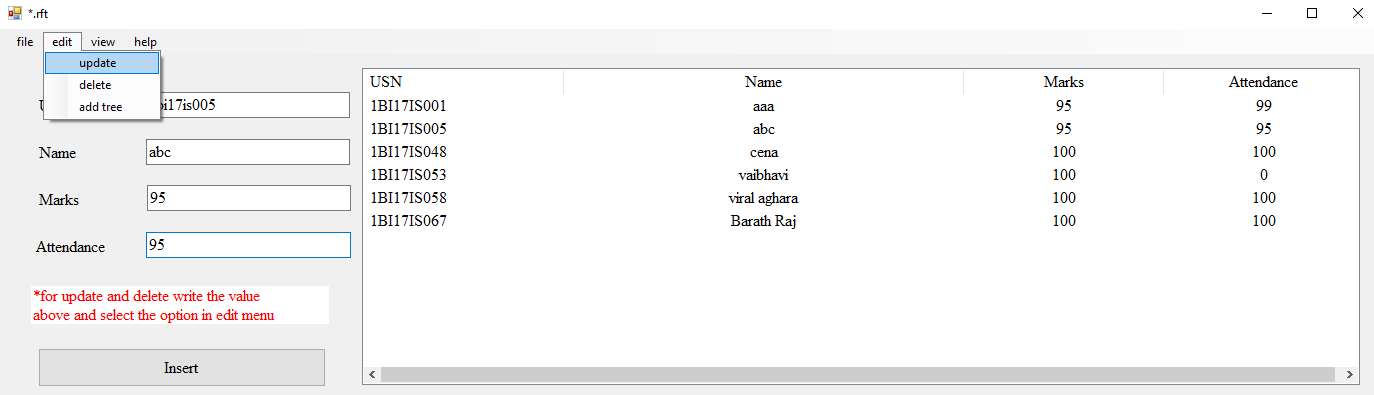


Fig. 6.12: Update Function

It works same as the delete fuction.

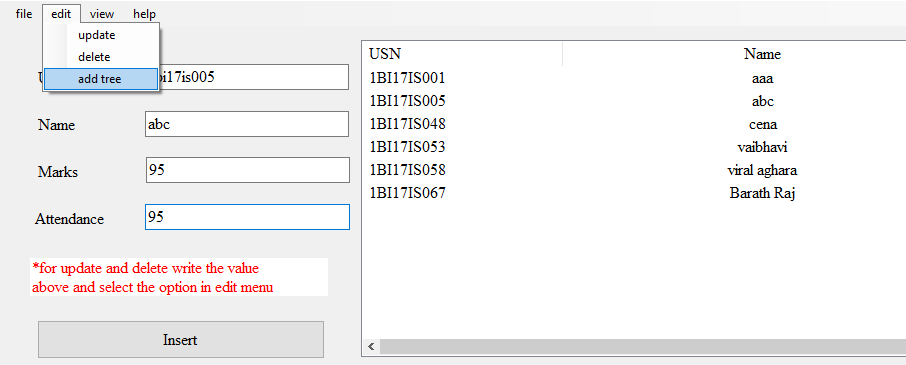


Fig. 6.13: Add tree function

The add tree function works same as the open but it file and add value form file in exits tree.

# Source code

## 7.1 B+Tree Class

|  |
| --- |
| #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 student  {  public:  string usn;  string name;  string marks;  string attendance;  string Buf;  char buf[100];  };  istream& operator >> (istream& in, student& s)  {  cout << "Enter usn:-";  in >> s.usn;  cout << "Enter name:-";  in >> s.name;  cout << "Enter marks.:-";  in >> s.marks;  cout << "Enter attendance.:-";  in >> s.attendance;  return in;  }  ostream& operator << (ostream& out, student& s)  {  out << s.usn << "\t" << s.name << "\t" << s.marks << "\t" << s.attendance << "\t";  return out;  }  class node  {  public:  student value[4];  node\* child[4];  node\* next, \* per;  node\* parent;  int size;  node();  };  class btree  {  public:  node\* head;  int flag, flage,ds;  string st[1000];  student sd[1000];  void insert(student key);  void search(student key);  btree update(student key, student key2);  void tree\_view(node\* n);  btree del(student key);  void write();  void read();  void display();  btree();  private:  node\* split(node\* n);  node\* tchilds(node\* n);  node\* findlevel(student key, node\* n);  };  node::node() {  size = 4;  for (int i = 0; i < size; i++) {  value[i].usn = "";  value[i].name = "";  value[i].marks = "";  value[i].attendance = "";  child[i] = NULL;  }  size = 0;  next = NULL;  per = NULL;  parent = NULL;  }  btree::btree() {  head = NULL;  flag = 0;  flage = 0;  }  void btree::insert(student key) {  ds = 0;  transform(key.usn.begin(),key.usn.end(), key.usn.begin(), ::toupper);  cout << key.usn << 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.usn, n->value[i].usn)) {    st[ds++]="already exits";  return;  }  if (key.usn < n->value[i].usn) {  break;  }  }  if (i == n->size) {  n->value[i] = key;  n->size++;  }  else {  student 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\* btree::findlevel(student key, node\* n) {  node\* ptr = n;  int i;  for (i = 0; i < ptr->size; i++) {  if (key.usn < ptr->value[i].usn) {  if (ptr->child[i] != NULL) {  return(findlevel(key, ptr->child[i]));  }  else {  return ptr;  }  }  }  if (i == ptr->size && key.usn > ptr->value[i].usn) {  if (ptr->child[i] != NULL) {  return(findlevel(key, ptr->child[i]));  }  else  {  return ptr;  }  }  return ptr;  }  btree btree::del(student key) {  transform(key.usn.begin(), key.usn.end(), key.usn.begin(), ::toupper);  node\* n = head;  int x = 0;  btree b;  b.ds = 0;  flage = 0;  student 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.usn, n->value[i].usn)) == 0) {  a[x++] = n->value[i];  }  else {  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;  }  return b;  }  btree btree::update(student key, student key2) {  transform(key.usn.begin(), key.usn.end(), key.usn.begin(), ::toupper);  transform(key2.usn.begin(), key2.usn.end(), key2.usn.begin(), ::toupper);  node\* n = head;  int x = 0;  btree b;  b.ds = 0;  flage = 0;  student 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.usn, n->value[i].usn))==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\* btree::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].usn < ptr->value[x].usn) {  break;  }  }  student temp = n->value[m];  student 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 btree::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].usn == "")  continue;  sd[ds++]=n->value[i];  }  n = n->next;  }  return;  }  void btree::search(student key) {  transform(key.usn.begin(), key.usn.end(), key.usn.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.usn, ptr->value[i].usn))==1) {  transform(key.usn.begin(), key.usn.end(), key.usn.begin(), ::toupper);  cout << key.usn<<endl;  flag = 1;  sd[ds++] = ptr->value[i];  return;  }  }  break;  }  else  {  int i;  for (i = 0; i < ptr->size; i++) {  if (key.usn < ptr->value[i].usn) {  break;  }  }  ptr = ptr->child[i];  cout << x++ << "\t";  cout << "->";  }  }  st[0] = "usn not found";  return;  }  void btree::tree\_view(node\* n) {  cout << "|";  for (int i = 0; i < n->size; i++) {  cout << n->value[i];  }  cout << "|" << endl;  while (n != NULL) {}  }  void btree::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].usn == "")  continue;  st[ds++] = n->value[i].usn + "|" + n->value[i].name + "|" + n->value[i].marks + "|" + n->value[i].attendance + "|";  }  n = n->next;  }  return;  }  void btree::read() {  char dl = '|';  student s;  cout << ds<<endl;  for (int j = 0; j < ds; j++) {  cout << st[j]<<j<<endl;  vector<string> res = splitStrings(st[j], dl);  s.usn = res[0];  s.name = res[1];  s.marks = res[2];  s.attendance = res[3];  int temp = ds;  insert(s);  ds = temp;  }  } |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## 7.2 Window/Interface Code

#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>

btree b;

student 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^ displayusn;

private: System::Windows::Forms::ColumnHeader^ displayname;

private: System::Windows::Forms::ColumnHeader^ displaymarks;

private: System::Windows::Forms::ColumnHeader^ dispalyattendance;

private: System::Windows::Forms::Label^ label7;

private: System::Windows::Forms::ListBox^ sdisplay;

private: System::Windows::Forms::OpenFileDialog^ openFileDialog1;

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^ usn;

private: System::Windows::Forms::Button^ insert;

protected:

private: System::Windows::Forms::MenuStrip^ menuStrip1;

private: System::Windows::Forms::ToolStripMenuItem^ fileToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ newToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ openToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ saveToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ saveAsToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ exitToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ editToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ updateToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ deleteToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ addTreeToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ viewToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ contentDisplayToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ heToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ aboutToolStripMenuItem;

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::TextBox^ name;

private: System::Windows::Forms::TextBox^ marks;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::TextBox^ attendance;

private: System::Windows::Forms::TextBox^ susn;

private: System::Windows::Forms::Label^ label5;

private: System::Windows::Forms::Label^ label6;

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->usn = (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->newToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->openToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->saveToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->saveAsToolStripMenuItem = (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->deleteToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->addTreeToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->viewToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->contentDisplayToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->heToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->aboutToolStripMenuItem = (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->marks = (gcnew System::Windows::Forms::TextBox());

this->label3 = (gcnew System::Windows::Forms::Label());

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

this->attendance = (gcnew System::Windows::Forms::TextBox());

this->susn = (gcnew System::Windows::Forms::TextBox());

this->label5 = (gcnew System::Windows::Forms::Label());

this->label6 = (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->listBox1 = (gcnew System::Windows::Forms::ListBox());

this->display = (gcnew System::Windows::Forms::ListView());

this->displayusn = (gcnew System::Windows::Forms::ColumnHeader());

this->displayname = (gcnew System::Windows::Forms::ColumnHeader());

this->displaymarks = (gcnew System::Windows::Forms::ColumnHeader());

this->dispalyattendance = (gcnew System::Windows::Forms::ColumnHeader());

this->label7 = (gcnew System::Windows::Forms::Label());

this->sdisplay = (gcnew System::Windows::Forms::ListBox());

this->openFileDialog1 = (gcnew System::Windows::Forms::OpenFileDialog());

this->menuStrip1->SuspendLayout();

this->SuspendLayout();

//

// usn

//

this->usn->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->usn->Location = System::Drawing::Point(146, 63);

this->usn->Margin = System::Windows::Forms::Padding(4, 4, 4, 4);

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

this->usn->Size = System::Drawing::Size(205, 26);

this->usn->TabIndex = 0;

//

// insert

//

this->insert->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->insert->Location = System::Drawing::Point(39, 319);

this->insert->Margin = System::Windows::Forms::Padding(4, 4, 4, 4);

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

this->insert->Size = System::Drawing::Size(288, 39);

this->insert->TabIndex = 1;

this->insert->Text = L"Insert";

this->insert->UseVisualStyleBackColor = true;

this->insert->Click += gcnew System::EventHandler(this, &MyForm::button1\_Click);

//

// menuStrip1

//

this->menuStrip1->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(4) {

this->fileToolStripMenuItem,

this->editToolStripMenuItem, this->viewToolStripMenuItem, this->heToolStripMenuItem

});

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(1382, 25);

this->menuStrip1->TabIndex = 2;

this->menuStrip1->Text = L"menuStrip1";

//

// fileToolStripMenuItem

//

this->fileToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(5) {

this->newToolStripMenuItem,

this->openToolStripMenuItem, this->saveToolStripMenuItem, this->saveAsToolStripMenuItem, this->exitToolStripMenuItem

});

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

this->fileToolStripMenuItem->Size = System::Drawing::Size(35, 19);

this->fileToolStripMenuItem->Text = L"file";

this->fileToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::fileToolStripMenuItem\_Click);

//

// newToolStripMenuItem

//

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

this->newToolStripMenuItem->Size = System::Drawing::Size(111, 22);

this->newToolStripMenuItem->Text = L"new";

this->newToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::newToolStripMenuItem\_Click);

//

// openToolStripMenuItem

//

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

this->openToolStripMenuItem->Size = System::Drawing::Size(111, 22);

this->openToolStripMenuItem->Text = L"open";

this->openToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::openToolStripMenuItem\_Click);

//

// saveToolStripMenuItem

//

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

this->saveToolStripMenuItem->Size = System::Drawing::Size(111, 22);

this->saveToolStripMenuItem->Text = L"save";

this->saveToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::saveToolStripMenuItem\_Click);

//

// saveAsToolStripMenuItem

//

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

this->saveAsToolStripMenuItem->Size = System::Drawing::Size(111, 22);

this->saveAsToolStripMenuItem->Text = L"save as";

this->saveAsToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::saveAsToolStripMenuItem\_Click);

//

// exitToolStripMenuItem

//

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

this->exitToolStripMenuItem->Size = System::Drawing::Size(111, 22);

this->exitToolStripMenuItem->Text = L"exit";

this->exitToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::exitToolStripMenuItem\_Click);

//

// editToolStripMenuItem

//

this->editToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(3) {

this->updateToolStripMenuItem,

this->deleteToolStripMenuItem, this->addTreeToolStripMenuItem

});

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

this->editToolStripMenuItem->Size = System::Drawing::Size(39, 19);

this->editToolStripMenuItem->Text = L"edit";

//

// updateToolStripMenuItem

//

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

this->updateToolStripMenuItem->Size = System::Drawing::Size(117, 22);

this->updateToolStripMenuItem->Text = L"update ";

this->updateToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::updateToolStripMenuItem\_Click);

//

// deleteToolStripMenuItem

//

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

this->deleteToolStripMenuItem->Size = System::Drawing::Size(117, 22);

this->deleteToolStripMenuItem->Text = L"delete";

this->deleteToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::deleteToolStripMenuItem\_Click);

//

// addTreeToolStripMenuItem

//

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

this->addTreeToolStripMenuItem->Size = System::Drawing::Size(117, 22);

this->addTreeToolStripMenuItem->Text = L"add tree";

this->addTreeToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::addTreeToolStripMenuItem\_Click);

//

// viewToolStripMenuItem

//

this->viewToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(1) { this->contentDisplayToolStripMenuItem });

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

this->viewToolStripMenuItem->Size = System::Drawing::Size(43, 19);

this->viewToolStripMenuItem->Text = L"view";

//

// contentDisplayToolStripMenuItem

//

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

this->contentDisplayToolStripMenuItem->Size = System::Drawing::Size(155, 22);

this->contentDisplayToolStripMenuItem->Text = L"content display";

this->contentDisplayToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::contentDisplayToolStripMenuItem\_Click);

//

// heToolStripMenuItem

//

this->heToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(1) { this->aboutToolStripMenuItem });

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

this->heToolStripMenuItem->Size = System::Drawing::Size(42, 19);

this->heToolStripMenuItem->Text = L"help";

//

// aboutToolStripMenuItem

//

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

this->aboutToolStripMenuItem->Size = System::Drawing::Size(105, 22);

this->aboutToolStripMenuItem->Text = L"about";

this->aboutToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::aboutToolStripMenuItem\_Click);

//

// 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(36, 67);

this->label1->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

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

this->label1->Size = System::Drawing::Size(41, 19);

this->label1->TabIndex = 3;

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

//

// label2

//

this->label2->AutoSize = true;

this->label2->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label2->Location = System::Drawing::Point(36, 114);

this->label2->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(46, 19);

this->label2->TabIndex = 5;

this->label2->Text = L"Name";

//

// name

//

this->name->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->name->Location = System::Drawing::Point(147, 110);

this->name->Margin = System::Windows::Forms::Padding(4, 4, 4, 4);

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

this->name->Size = System::Drawing::Size(204, 26);

this->name->TabIndex = 6;

//

// marks

//

this->marks->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->marks->Location = System::Drawing::Point(148, 156);

this->marks->Margin = System::Windows::Forms::Padding(4, 4, 4, 4);

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

this->marks->Size = System::Drawing::Size(204, 26);

this->marks->TabIndex = 8;

//

// label3

//

this->label3->AutoSize = true;

this->label3->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label3->Location = System::Drawing::Point(36, 161);

this->label3->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(49, 19);

this->label3->TabIndex = 9;

this->label3->Text = L"Marks";

//

// 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(33, 208);

this->label4->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(78, 19);

this->label4->TabIndex = 10;

this->label4->Text = L"Attendance";

//

// attendance

//

this->attendance->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->attendance->Location = System::Drawing::Point(147, 203);

this->attendance->Margin = System::Windows::Forms::Padding(4, 4, 4, 4);

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

this->attendance->Size = System::Drawing::Size(205, 26);

this->attendance->TabIndex = 11;

//

// susn

//

this->susn->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->susn->Location = System::Drawing::Point(122, 395);

this->susn->Margin = System::Windows::Forms::Padding(4, 4, 4, 4);

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

this->susn->Size = System::Drawing::Size(204, 26);

this->susn->TabIndex = 12;

//

// label5

//

this->label5->AutoSize = true;

this->label5->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label5->Location = System::Drawing::Point(36, 362);

this->label5->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label5->Name = L"label5";

this->label5->Size = System::Drawing::Size(177, 19);

this->label5->TabIndex = 13;

this->label5->Text = L"Enter the USN to be search";

//

// label6

//

this->label6->AutoSize = true;

this->label6->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label6->Location = System::Drawing::Point(36, 399);

this->label6->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label6->Name = L"label6";

this->label6->Size = System::Drawing::Size(41, 19);

this->label6->TabIndex = 14;

this->label6->Text = L"USN";

//

// openFD

//

this->openFD->FileName = L"openFileDialog";

//

// button1

//

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->Location = System::Drawing::Point(39, 441);

this->button1->Margin = System::Windows::Forms::Padding(4, 4, 4, 4);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(292, 39);

this->button1->TabIndex = 17;

this->button1->Text = L"Search";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &MyForm::search);

//

// label8

//

this->label8->AutoSize = true;

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, 19);

this->label8->TabIndex = 18;

//

// listBox1

//

this->listBox1->BackColor = System::Drawing::Color::White;

this->listBox1->BorderStyle = System::Windows::Forms::BorderStyle::None;

this->listBox1->ForeColor = System::Drawing::Color::Red;

this->listBox1->FormattingEnabled = true;

this->listBox1->ItemHeight = 19;

this->listBox1->Items->AddRange(gcnew cli::array< System::Object^ >(2) { L"\*for update and delete write the value ", L"above and select the option in edit menu" });

this->listBox1->Location = System::Drawing::Point(32, 257);

this->listBox1->Margin = System::Windows::Forms::Padding(4, 4, 4, 4);

this->listBox1->Name = L"listBox1";

this->listBox1->Size = System::Drawing::Size(298, 38);

this->listBox1->TabIndex = 19;

//

// display

//

this->display->Columns->AddRange(gcnew cli::array< System::Windows::Forms::ColumnHeader^ >(4) {

this->displayusn, this->displayname,

this->displaymarks, this->dispalyattendance

});

this->display->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->display->HideSelection = false;

this->display->Location = System::Drawing::Point(363, 39);

this->display->Margin = System::Windows::Forms::Padding(4, 4, 4, 4);

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

this->display->Size = System::Drawing::Size(998, 317);

this->display->TabIndex = 20;

this->display->UseCompatibleStateImageBehavior = false;

this->display->View = System::Windows::Forms::View::Details;

//

// displayusn

//

this->displayusn->Text = L"USN";

this->displayusn->Width = 200;

//

// displayname

//

this->displayname->Text = L"Name";

this->displayname->TextAlign = System::Windows::Forms::HorizontalAlignment::Center;

this->displayname->Width = 400;

//

// displaymarks

//

this->displaymarks->Text = L"Marks";

this->displaymarks->TextAlign = System::Windows::Forms::HorizontalAlignment::Center;

this->displaymarks->Width = 200;

//

// dispalyattendance

//

this->dispalyattendance->Text = L"Attendance";

this->dispalyattendance->TextAlign = System::Windows::Forms::HorizontalAlignment::Center;

this->dispalyattendance->Width = 200;

//

// label7

//

this->label7->AutoSize = true;

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, 14);

this->label7->TabIndex = 23;

//

// sdisplay

//

this->sdisplay->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, static\_cast<System::Drawing::FontStyle>((System::Drawing::FontStyle::Bold | System::Drawing::FontStyle::Italic)),

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(0)));

this->sdisplay->FormattingEnabled = true;

this->sdisplay->ItemHeight = 19;

this->sdisplay->Location = System::Drawing::Point(366, 390);

this->sdisplay->Margin = System::Windows::Forms::Padding(4, 4, 4, 4);

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

this->sdisplay->Size = System::Drawing::Size(994, 137);

this->sdisplay->TabIndex = 24;

this->sdisplay->SelectedIndexChanged += gcnew System::EventHandler(this, &MyForm::sdisplay\_SelectedIndexChanged);

//

// openFileDialog1

//

this->openFileDialog1->FileName = L"openFileDialog1";

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(9, 19);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(1382, 588);

this->Controls->Add(this->sdisplay);

this->Controls->Add(this->label7);

this->Controls->Add(this->display);

this->Controls->Add(this->listBox1);

this->Controls->Add(this->label8);

this->Controls->Add(this->button1);

this->Controls->Add(this->label6);

this->Controls->Add(this->label5);

this->Controls->Add(this->susn);

this->Controls->Add(this->attendance);

this->Controls->Add(this->label4);

this->Controls->Add(this->label3);

this->Controls->Add(this->marks);

this->Controls->Add(this->name);

this->Controls->Add(this->label2);

this->Controls->Add(this->label1);

this->Controls->Add(this->insert);

this->Controls->Add(this->usn);

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, 4, 4, 4);

this->MaximumSize = System::Drawing::Size(1398, 627);

this->MinimumSize = System::Drawing::Size(1398, 627);

this->Name = L"MyForm";

this->Text = L"Student db";

this->FormClosing += gcnew System::Windows::Forms::FormClosingEventHandler(this, &MyForm::MyForm\_FormClosing);

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 (usn->Text==""|| name->Text == "" || marks->Text == "" || attendance->Text == "")

{

MessageBox::Show("Enter Value Please.", "Error");

}

else

{

msclr::interop::marshal\_context con;

s.usn= con.marshal\_as< std::string >(usn->Text);

s.name= con.marshal\_as<std::string>(name->Text);

s.marks= con.marshal\_as<std::string>(marks->Text);

s.attendance= con.marshal\_as<std::string>(attendance->Text);

b.insert(s);

if (b.ds != 0) {

label7->Text = gcnew String(b.st[0].c\_str());

usn->Text = ""; name->Text = ""; marks->Text = ""; attendance->Text = "";

return;

}

flags = 0;

usn->Text = ""; name->Text = ""; marks->Text =""; attendance->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 = btree();

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].usn).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].marks).c\_str()));

dis->SubItems->Add(gcnew String((b.sd[i].attendance).c\_str()));

display->Items->Add(dis);

}

label7->Text = "";

}

private: System::Void search(System::Object^ sender, System::EventArgs^ e) {

sdisplay->Items->Clear();

label7->Text = "";

for (int i = 0; i < 1000; i++) {

b.st[i] = "";

}

b.ds = 0;

if (susn->Text == "")

{

MessageBox::Show("Enter Value Please.", "Error");

}

else

{

msclr::interop::marshal\_context con;

s.usn = con.marshal\_as<std::string>(susn->Text);

cout << s.usn << endl;

s.name = "";

s.marks = "";

s.attendance = "";

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 :-";

sdisplay->Items->Add(s);

s="\t\tUSN:-"+gcnew String((b.sd[i].usn).c\_str());

sdisplay->Items->Add(s);

s = "\t\tName:-" + gcnew String((b.sd[i].name).c\_str());

sdisplay->Items->Add(s);

s = "\t\tMarks:-" + gcnew String((b.sd[i].marks).c\_str());

sdisplay->Items->Add(s);

s = "\t\tAttendance:-" + gcnew String((b.sd[i].attendance).c\_str());

sdisplay->Items->Add(s);

}

b.ds = 0;

susn->Text = "";

}

}

private: System::Void newToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

b = btree();

s = student();

filename = "";

flags = 0;

}

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 saveAsToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

savefd->FileName = "";

savefd->ShowDialog();

if (savefd->FileName == "")

return;

filename = savefd->FileName;

this->Text = filename;

StreamWriter^ out = File::CreateText(filename);

b.write();

for (int i = 0; i < b.ds; i++) {

String^ stud = gcnew String(b.st[i].c\_str());

out->WriteLine(stud);

}

out->Close();

}

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?", "project2", MessageBoxButtons::YesNoCancel);

if (r == System::Windows::Forms::DialogResult::Yes) {

}

else if (r == System::Windows::Forms::DialogResult::No) {

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);

}

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 aboutToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

System::Windows::Forms::DialogResult r;

r = MessageBox::Show("Made by viral @copyrigth", "project2", MessageBoxButtons::OK);

}

private: System::Void updateToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

if (usn->Text == "" || name->Text == "" || marks->Text == "" || attendance->Text == "")

{

MessageBox::Show("Enter Value Please.", "Error");

}

else

{

msclr::interop::marshal\_context con;

s.usn = con.marshal\_as< std::string >(usn->Text);

s.name = con.marshal\_as<std::string>(name->Text);

s.marks = con.marshal\_as<std::string>(marks->Text);

s.attendance = con.marshal\_as<std::string>(attendance->Text);

usn->Text = ""; name->Text = ""; marks->Text = ""; attendance->Text = "";

Project2::MyForm1 f;

f.ShowDialog();

StreamReader^ r = File::OpenText("project.txt");

s2.usn = 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 deleteToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

if (usn->Text == "" || name->Text == "" || marks->Text == "" || attendance->Text == "")

{

MessageBox::Show("Enter Value Please.", "Error");

}

else

{

msclr::interop::marshal\_context con;

s.usn = con.marshal\_as< std::string >(usn->Text);

s.name = con.marshal\_as<std::string>(name->Text);

s.marks = con.marshal\_as<std::string>(marks->Text);

s.attendance = con.marshal\_as<std::string>(attendance->Text);

b=(b.del(s));

cout << b.ds << endl << b.st[0] << endl;

if (b.ds != 0) {

label7->Text = gcnew String(b.st[0].c\_str());

return;

}

flags = 0;

usn->Text = ""; name->Text = ""; marks->Text = ""; attendance->Text = "";

}

}

private: System::Void sdisplay\_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e) {

}

};

}