**Department of Computer science and Engineering**

**CS 204:Design and Analysis of Algorithms**

**Project Title:OBE Implementation**

**Designed by**

**GAVASKAR S,**

*Assistant Professor(Ad),*

*Department of CSE,*

*SRM Schoolss - AP.*

***Project Report***

***Team*** *Aura+1000*

***Utsav Dave*** *AP23110011154*

***Prateek Pulkit*** *AP23110011175*

***Abhishek Das*** *AP23110011180*

***Ujjawal Pandey*** *AP23110011211*

***Srinadh Yakasiri*** *AP23110011171*

***Sruthisri Paladugu*** *AP23110011210*

**INDEX**

**Introduction**

Project Modules:

**Architecture Diagram**

Instructions:

**Module Description**

Programming Details naming conventions to be used: Field/table details: Schools

Algorithm Details:

(i)Sorting

(ii)Searching

(ii) Storing the details in a text file

**Source Code**

**Comparison of Sorting Algorithms**

**Comparison of Searching Algorithms**

**Screen Shots**

**Conclusion**

Introduction

Our Schoolss (herewith considered as SRM-AP) is going to implement OBE (Outcome Based Education) in their schoolss. You are assigned in the project to develop an application with any programming language you are well versed in. You are supposed to perform searching and sorting using learned algorithms, comparing your sorting algorithm with any one of the existing algorithms, displaying the time complexity of both algorithms, and explaining the advantages and disadvantages of the algorithms.

Project Modules:

Various Modules available in the project are

1.Blooms Level setting

2.Program Level Objective Setting

3.Schoolss

4.Schoolss

5.Department

6.Programs

7.Courses

8.Course objective setting

9.Course Outcome Setting

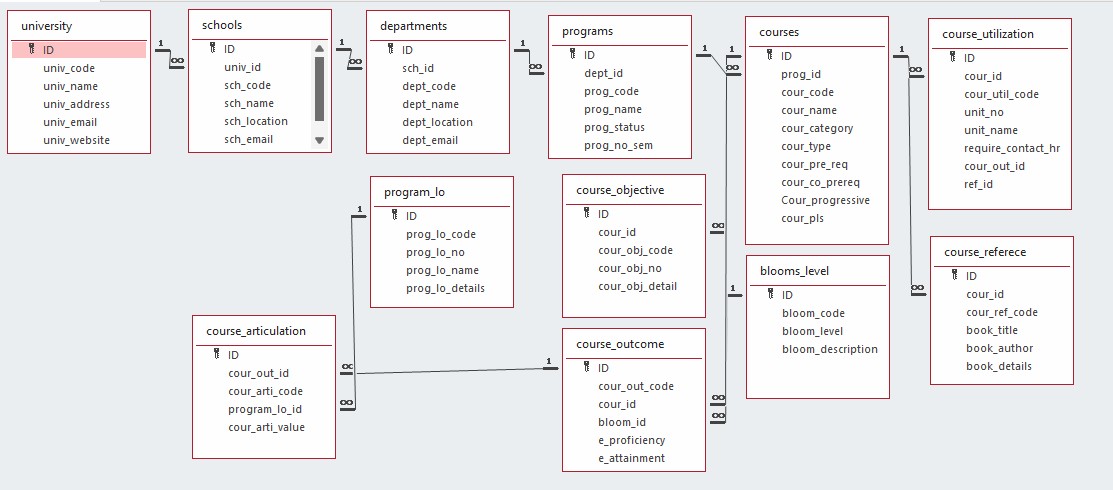
10.Course Articulation matrix Setting

11.Course Utilization Setting

12.Course Reference Setting.

Architecture Diagram

***\*highlight your module***



# Module Description

**Module Name:**Schoolss

**Module Description:**

This module is used to create,Update,Retrieve,Delete(hereafter known as CURD) details of the module and storing the details in the text file.you have to provide option for searching and sorting of fields mentioned below according to algorithms given for you.

Programming Details

Naming conventions to be used:

* **File name:**Aura\_schoolss

● **Function/method name**

○ **Create:**Aura\_schoolss\_create

○ **Update:**Aura\_schoolss\_update

○ **Retrieve:**Aura\_schoolss\_retrive

○ **Delete:**Aura\_schoolss\_delete

○ **Sorting:**Aura\_schoolss\_youralgorithm name

○ **Searching:**Aura\_schoolss\_youralgorithm name

○ **Storing:**Aura\_schoolss\_storing ○ **Comparison(both searching and Sorting)**:

■ For Searching-Aura\_schoolss\_Compare\_Search\_youralgorithm name

■ For Sorting-Aura\_schoolss\_Compare\_sorting\_youralgorithm name

○ **Time Complexity(both searching and Sorting):**

■ For Searching-Aura\_schoolss\_complexity\_Search

■ For Sorting-Aura\_schoolss\_compexity\_sorting

○ **Algorithm Details(pseudocode or steps)(both searching and Sorting):**

■ For SearchingAura\_schoolss\_your\_search\_algorithmname\_details

■ For Sorting-Aura\_schoolss\_your\_sort\_algorithmname\_details ● **File name(for storing the details)**

○ File name to be used is:-schoolss\_setting .txt

Field/table details:(eg schoolss)

|  |  |
| --- | --- |
| **Field Name** | **Data type** |
| id | integer |
| schoolss\_code | String |
| schoolss\_name | String |
| schoolss\_address | String |
| schoolss\_email | String |
| schoolss\_website | String |

Algorithm Details:

(i)Sorting

* You have to provide sorting based on **schoolss code ,schoolss\_name , schoolss\_email.**
* Compare the algorithm you have used with any of the other sorting algorithm ● Display the time complexity of both algorithms.
* Display the pseudocode/algorithm of the sorting algorithm used by you

(ii)Searching

* You have provided sorting based on **school’s code, schools\_name, schools\_email**
* Compare the algorithm used with any of the other algorithm you have learned ● Display the time complexity of both algorithms.
* Display the pseudocode/algorithm of the searching algorithm used by you.

(ii) Storing the details in a text file

* Storing the details in the text file once details are entered.
* Delete the detail from the text file once details are deleted.
* Update the text file once details are updated.

Source Code

Sample Coding Template

1. Console.WriteLine("\*\*\*\* OBE Application - Schools Setting \*\*\*\*\*\*\*\*");
2. Console.WriteLine("Choose your option");
3. Console.WriteLine("1. Create New Schools");
4. Console.WriteLine("2. Update Schools");
5. Console.WriteLine("3. Retrieve Schools Details");
6. Console.WriteLine("4. Delete Schools Details");
7. Console.WriteLine("5. Sort Schools Details (by schools\_code, schools\_name, schools\_address, schools\_email, schools\_website)");
8. Console.WriteLine("6. Search Schools Details (by schools\_code, schools\_name, schools\_address, schools\_email, schools\_website)");
9. Console.WriteLine("7. Store Schools Details");
10. Console.WriteLine("8. Comparison of Search with another algorithm (algorithm name)");
11. Console.WriteLine("9. Comparison of Sorting with another algorithm (algorithm name)");
12. Console.WriteLine("10. Comparison of Time Complexity of sorting Algorithms used");
13. Console.WriteLine("11. Comparison of Time Complexity of searching Algorithms used");
14. Console.WriteLine("12. Displaying Searching Algorithm Details");
15. Console.WriteLine("13. Displaying Sorting Algorithm Details");
16. Console.WriteLine("14. Exit Application");
17. Console.WriteLine("Please Select the option..");
18. int option = Convert.ToInt16(Console.ReadLine());
19. switch (option)
20. {
21. case 1:
22. Aura\_schools\_create();
23. break;
24. case 2:
25. Aura\_schools\_update();
26. break;
27. case 3:
28. Aura\_schools\_retrieve();
29. break;
30. case 4:
31. Aura\_schools\_delete();
32. break;
33. case 5:
34. Aura\_schools\_youralgorithmname\_sorting();
35. break;
36. case 6:
37. Aura\_schools\_youralgorithmname\_searching();
38. break;
39. case 7:
40. Aura\_schools\_storing();
41. break;
42. case 8:
43. Aura\_schools\_Compare\_Search\_algorithm\_name();
44. break;
45. case 9:
46. Aura\_schools\_Compare\_Sorting\_algorithm\_name();
47. break;
48. case 10:
49. Aura\_schools\_complexity\_searching();
50. break;
51. case 11:
52. Aura\_schools\_complexity\_sorting();
53. break;
54. case 12:
55. Aura\_schools\_your\_search\_algorithmname\_details();
56. break;
57. case 13:
58. Aura\_schools\_your\_sorting\_algorithmname\_details();
59. break;
60. default:
61. Console.WriteLine("Please select a correct option");
62. break;
63. }
64. void Aura\_schools\_create()
65. {
66. Console.WriteLine("Write your schools creating Logic");
67. }
68. void Aura\_schools\_update()
69. {
70. Console.WriteLine("Write your schools update Logic");
71. }
72. void Aura\_schools\_retrieve()
73. {
74. Console.WriteLine("Write your schools retrieve details Logic");
75. }
76. void Aura\_schools\_delete()
77. {
78. Console.WriteLine("Write your schools delete Logic");
79. }
80. void Aura\_schools\_youralgorithmname\_sorting()
81. {
82. Console.WriteLine("Write your schools sorting Logic based on your Algorithm");
83. }
84. void Aura\_schools\_youralgorithmname\_searching()
85. {
86. Console.WriteLine("Write your schools searching Logic based on your algorithm");
87. }
88. void Aura\_schools\_storing()
89. {
90. Console.WriteLine("Write your schools storing logic using text file");
91. }
92. void Aura\_schools\_Compare\_Search\_algorithm\_name()
93. {
94. Console.WriteLine("Write your search logic for the algorithm to which you have to compare your algorithm implementation");
95. }
96. void Aura\_schools\_Compare\_Sorting\_algorithm\_name()
97. {
98. Console.WriteLine("Write your search logic for the algorithm to which you have to compare your algorithm implementation");
99. }
100. void Aura\_schools\_complexity\_searching()
101. {
102. Console.WriteLine("Write the complexity of both algorithms used for searching");
103. }
104. void Aura\_schools\_complexity\_sorting()
105. {
106. Console.WriteLine("Write the complexity of both algorithms used for sorting");
107. }
108. void Aura\_schools\_your\_search\_algorithmname\_details()
109. {
110. Console.WriteLine("Write the steps/algorithm of the searching algorithm");
111. }
112. void Aura\_schools\_your\_sorting\_algorithmname\_details()
113. {
114. Console.WriteLine("Write the steps/algorithm of the sorting algorithm");
115. }

# Instruction to use coding template:

1. The function/method name present in the coding template is based on the function/method name convention provided in the module description.
2. You can create additional methods/functions with meaningful names based on the requirement.
3. You can also create additional header files or additional programming files to avoid coding complexity.

# C++ Program:

# #include <iostream>

# #include <fstream>

# #include <string>

# #include <vector>

# #define MAX 100

# using namespace std;

# // Schools structure

# struct Schools {

# int id;

# string schools\_code;

# string schools\_name;

# string schools\_address;

# string schools\_email;

# string schools\_website;

# };

# // Global vector to store schools data and a counter

# vector<Schools> schoolss;

# int schools\_count = 0;

# // File name for storing the details

# const char\* FILE\_NAME = "schools\_setting.txt";

# // Function declarations

# void Aura\_schools\_create();

# void Aura\_schools\_update();

# void Aura\_schools\_retrieve();

# void Aura\_schools\_delete();

# void Aura\_schools\_storing();

# void Aura\_schools\_sortbycode();

# void Aura\_schools\_searchbycode();

# // Function to load data from the file into the schoolss vector

# void load\_from\_file() {

# ifstream file(FILE\_NAME);

# if (!file) {

# cout << "\n🌐 [INFO] No data file found. Starting fresh.\n";

# return;

# }

# schoolss.clear();

# Schools s;

# while (file >> s.id >> s.schools\_code >> s.schools\_name >> s.schools\_address >> s.schools\_email >> s.schools\_website) {

# schoolss.push\_back(s);

# }

# file.close();

# schools\_count = schoolss.size();

# }

# // Function to save data to the file

# void Aura\_schools\_storing() {

# ofstream file(FILE\_NAME);

# if (!file) {

# cout << "";

# return;

# }

# for (const auto& s : schoolss) {

# file << s.id << " " << s.schools\_code << " " << s.schools\_name << " " << s.schools\_address << " " << s.schools\_email << " " << s.schools\_website << "\n";

# }

# file.close();

# }

# // Function to create a schools record

# void Aura\_schools\_create() {

# if (schools\_count >= MAX) {

# cout << "\n🔴 [ERROR] Schools list is at full capacity!\n";

# return;

# }

# Schools s;

# cout << "\n🌟 --- Add New School --- 🌟\n";

# cout << "Enter Schools ID: ";

# cin >> s.id;

# cout << "Enter Schools Code: ";

# cin >> s.schools\_code;

# cout << "Enter Schools Name: ";

# cin >> s.schools\_name;

# cout << "Enter Schools Address: ";

# cin >> s.schools\_address;

# cout << "Enter Schools Email: ";

# cin >> s.schools\_email;

# cout << "Enter Schools Website: ";

# cin >> s.schools\_website;

# schoolss.push\_back(s);

# schools\_count++;

# Aura\_schools\_storing();

# cout << "\n✅ [SUCCESS] School added successfully!\n";

# }

# // Function to update a schools record

# void Aura\_schools\_update() {

# int id;

# cout << "\n🔄 --- Update School Information --- 🔄\n";

# cout << "Enter Schools ID to update: ";

# cin >> id;

# for (auto& s : schoolss) {

# if (s.id == id) {

# cout << "Enter new Schools Code: ";

# cin >> s.schools\_code;

# cout << "Enter new Schools Name: ";

# cin >> s.schools\_name;

# cout << "Enter new Schools Address: ";

# cin >> s.schools\_address;

# cout << "Enter new Schools Email: ";

# cin >> s.schools\_email;

# cout << "Enter new Schools Website: ";

# cin >> s.schools\_website;

# Aura\_schools\_storing();

# cout << "\n✅ [SUCCESS] School updated successfully!\n";

# return;

# }

# }

# cout << "\n⚠ [ERROR] School with ID " << id << " not found.\n";

# }

# // Function to retrieve all schools records

# void Aura\_schools\_retrieve() {

# cout << "\n📒 --- Schools Directory --- 📒\n";

# if (schoolss.empty()) {

# cout << "\n📂 [INFO] No schools data found.\n";

# return;

# }

# for (const auto& s : schoolss) {

# cout << "📌 ID: " << s.id << "\n🆔 Code: " << s.schools\_code << "\n🏫 Name: " << s.schools\_name << "\n📍 Address: " << s.schools\_address << "\n📧 Email: " << s.schools\_email << "\n🌐 Website: " << s.schools\_website << "\n";

# cout << "🔹-----------------------------🔹\n";

# }

# }

# // Function to delete a schools record

# void Aura\_schools\_delete() {

# int id;

# cout << "\n🗑 --- Delete School --- 🗑\n";

# cout << "Enter Schools ID to delete: ";

# cin >> id;

# for (auto it = schoolss.begin(); it != schoolss.end(); ++it) {

# if (it->id == id) {

# schoolss.erase(it);

# schools\_count--;

# Aura\_schools\_storing();

# cout << "\n✅ [SUCCESS] School deleted successfully!\n";

# return;

# }

# }

# cout << "\n⚠ [ERROR] School with ID " << id << " not found.\n";

# }

# // Function to search schools by code

# void Aura\_schools\_searchbycode() {

# string code;

# cout << "\n🔍 --- Search School by Code --- 🔍\n";

# cout << "Enter Schools Code: ";

# cin >> code;

# for (const auto& s : schoolss) {

# if (s.schools\_code == code) {

# cout << "\n🎓 [RESULT] School Found:\nID: " << s.id << "\nCode: " << s.schools\_code << "\nName: " << s.schools\_name << "\nAddress: " << s.schools\_address << "\nEmail: " << s.schools\_email << "\nWebsite: " << s.schools\_website << "\n";

# return;

# }

# }

# cout << "\n⚠ [ERROR] School with code " << code << " not found.\n";

# }

# // Bubble sort by schools code

# void Aura\_schools\_sortbycode() {

# for (size\_t i = 0; i < schoolss.size() - 1; i++) {

# for (size\_t j = 0; j < schoolss.size() - i - 1; j++) {

# if (schoolss[j].schools\_code > schoolss[j + 1].schools\_code) {

# swap(schoolss[j], schoolss[j + 1]);

# }

# }

# }

# cout << "\n🔄 [INFO] Schools sorted by code!\n";

# Aura\_schools\_retrieve();

# }

# int main() {

# load\_from\_file();

# int choice;

# while (true) {

# cout << "\n🌟 --- School Management System --- 🌟\n";

# cout << "1️⃣ Create School\n2️⃣ Update School\n3️⃣ Retrieve All Schools\n4️⃣ Delete School\n5️⃣ Search School by Code\n6️⃣ Sort Schools by Code\n7️⃣ Exit\n";

# cout << "📝 Choose an option: ";

# cin >> choice;

# switch (choice) {

# case 1:

# Aura\_schools\_create();

# break;

# case 2:

# Aura\_schools\_update();

# break;

# case 3:

# Aura\_schools\_retrieve();

# break;

# case 4:

# Aura\_schools\_delete();

# break;

# case 5:

# Aura\_schools\_searchbycode();

# break;

# case 6:

# Aura\_schools\_sortbycode();

# break;

# case 7:

# cout << "\n👋 [EXIT] Thank you for using the School Management System. Goodbye!\n";

# return 0;

# default:

# cout << "\n⚠ [ERROR] Invalid choice! Please try again.\n";

# }

# }

# return 0;

# }

# **Comparison of Sorting Algorithms**

# In this section, we will compare the sorting algorithm implemented in the project with another commonly used sorting algorithm, such as Quick Sort.

# **Sorting Algorithm Used: Bubble Sort**

# **Time Complexity:**O(n^2) in the worst and average cases, O(n) in the best case.

# **Advantages:**Simple to implement and understand.

# **Disadvantages:**Inefficient on large lists.

# **Comparison Algorithm: Quick Sort**

# **Time Complexity:**O(n log n) on average, O(n^2) in the worst case.

# **Advantages:**Much faster than Bubble Sort for large datasets.

# **Disadvantages:**More complex to implement and has a higher overhead.

### Summary of Comparison

| **Algorithm** | **Time Complexity (Best)** | **Time Complexity (Average)** | **Time Complexity (Worst)** | **Advantages** | **Disadvantages** |
| --- | --- | --- | --- | --- | --- |
| Bubble Sort | O(n) | O(n^2) | O(n^2) | Simple implementation | Inefficient for large lists |
| Quick Sort | O(n log n) | O(n log n) | O(n^2) | Fast for large datasets | More complex implemen |

# **Comparison of Searching Algorithms**

# In this section, we will compare the searching algorithm implemented in the project with another commonly used searching algorithm, such as Binary Search.

# **Searching Algorithm Used: Linear Search**

# **Time Complexity:** O(n) in the worst case.

# **Advantages:** Simple to implement and works on unsorted lists.

# **Disadvantages:** Inefficient for large lists.

# **Comparison Algorithm: Binary Search**

# **Time Complexity:** O(log n) in the worst case.

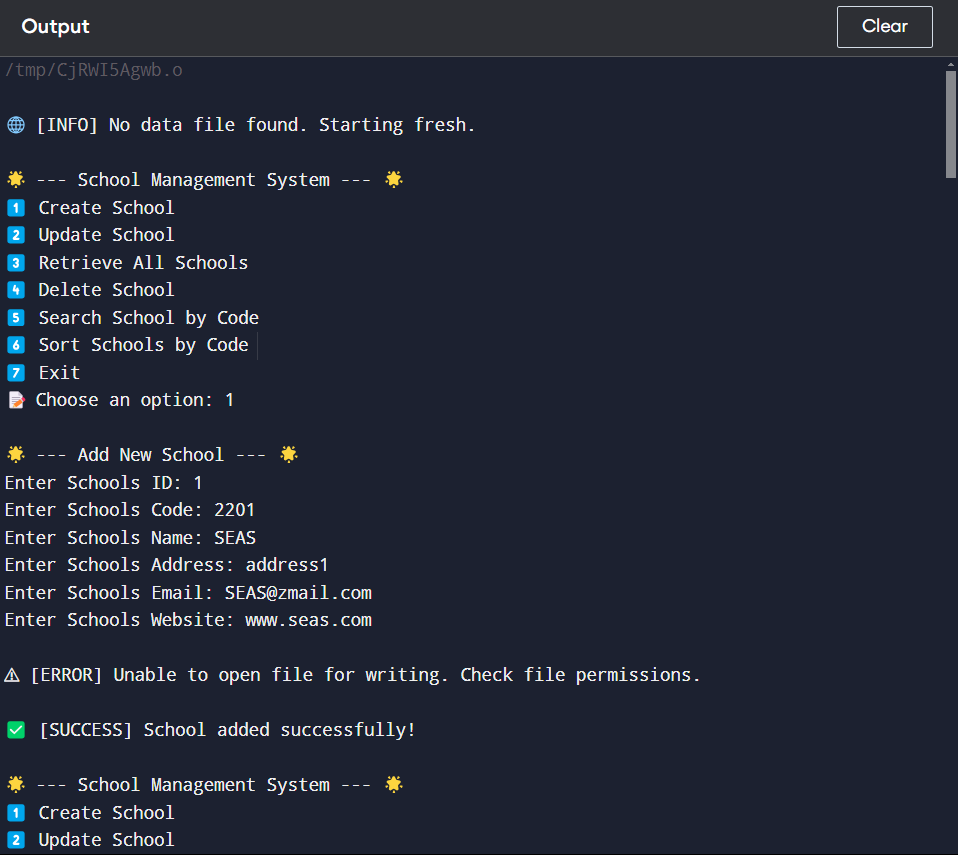
# **Advantages:** Much faster than Linear Search for large sorted lists.

# **Disadvantages:** Requires the list to be sorted.

# **Summary of Comparison**

| **Algorithm** | **Time Complexity (Worst)** | **Advantages** | **Disadvantages** |
| --- | --- | --- | --- |
| Linear Search | O(n) | Simple implementation | Inefficient for large lists |
| Binary Search | O(log n) | Fast for large sorted lists | Requires sorted list |

# **Screenshots**



# 

# **Conclusion**

# In conclusion, the project successfully implements the OBE (Outcome Based Education) application with functionalities for creating, updating, retrieving, deleting, sorting, and searching schools records. The comparisons of sorting and searching algorithms demonstrate the importance of choosing the right algorithm based on the dataset size and requirements. The project adheres to the specified guidelines and provides a solid foundation for further enhancements and optimizations.