Project Title: Employee Record Management System

Table of Content

1. Introduction

In today's digital era, efficient management of employee data is critical for every organization. The "Employee Record Management System" project offers a computerized solution to store, process, manipulate, and access employee details dynamically. This system supports sorting, searching, updating, and displaying employee records based on various criteria such as ID, Name, and Salary, streamlining HR operations and minimizing manual errors.

2. Team Members

SNO	NAME	ENROLL NO	RESPONSIBILITY
1	Yeshwanth Kumar	2403031460982	Searching algorithm by Name
2	Deekshith Reddy	2403031460279	Display and update functions
3	Pattem Jaiyant	2403031461525	Main function and declarations

4	Narra Kartik	2403031461154	Testing and validation
5	Nallani Manikanta	2403031461152	Sorting by Salary and criteria
6	Chitti Babu	2403031461151	Sorting by Name and criteria

3. Objective

''To develop a C program that:
- Stores a list of employee records.
- Sorts employees based on Salary, Name, or ID using sorting algorithms.
- Searches for a specific employee by Name or ID.
- Updates, adds, or deletes employee records efficiently.

4. Problem Statement

Developing an Employee Record Management System that can process employee information (such as Name, Employee ID, Department, Salary, Join Date). The system must enable sorting and searching functionalities, updating existing records, adding/removing records, and displaying all employee information efficiently.

5. System Requirements

Software: Turbo C++ / Dev C++
Text Editor (e.g., VS Code)
Hardware: Minimum 2GB RAM, 1.0GHz Processor, Laptop with keyboard & monitor

6. Data Structures Used

- Arrays: Used array of structures to store multiple employee information records.
- Sorting (Bubble Sort): Mechanism used to sort employee lists by ID, Name, or Salary.
- Searching (Linear Search): Mechanism used to search a specific employee by Name or ID.

7. Modules and Description

MODULE	DESCRIPTION
Input	Accepts and validates employee records (ID, Name, Salary, etc.)

Display	Displays all stored employee records
Sorting	Sorts employees by Salary, Name, or ID using sorting algorithms
Searching	Searches specific employee record by Name/ID (case-insensitive)
Update/Delete	Updates or deletes existing employee data
Add	Adds new employee records

8. Algorithms Used

A. Sorting: Bubble Sort by Salary

```
// Sorts employees by descending Salary, then ascending Name
void sortBySalary(Employee e[], int n) {
  Employee temp;
  for (int i = 0; i < n-1; i++) {
    for (int j = 0; j < n-i-1; j++) {
      if ((e[j].salary < e[j+1].salary) ||
        (e[j].salary == e[j+1].salary && strcmp(e[j].name, e[j+1].name) > 0)) {
        temp = e[j];
        e[j] = e[j+1];
        e[j+1] = temp;
      }
    }
  }
}
```

B. Sorting: Bubble Sort by Name

```
// Sorts employees by Name (ascending), then Salary (descending)
void sortByName(Employee e[], int n) {
  Employee temp;
  for (int i = 0; i < n-1; i++) {
    for (int j = 0; j < n-i-1; j++) {
      if ((strcasecmp(e[j].name, e[j+1].name) > 0) ||
        (strcasecmp(e[j].name, e[j+1].name) == 0 && e[j].salary < e[j+1].salary)) {
        temp = e[j];
        e[j] = e[j+1];
        e[j+1] = temp;
      }
    }
  }
}
```

C. Searching: Linear Search by Name

```
// Case-insensitive search for employee by name
int searchByName(Employee e[], int n, char target[]) {
   for (int i = 0; i < n; i++) {
      if (strcasecmp(e[i].name, target) == 0) {
         return i;
      }
   }
   return -1;
}
```

9. Sample Input/Output

Input:
Enter number of employees: 4
Enter employee 1 ID: 1001
Enter employee 1 name: Yeshwanth
Enter employee 1 salary: 55000
Enter employee 2 ID: 1002
Enter employee 2 name: Deekshith

Enter employee 2 salary: 65000
Enter employee 3 ID: 1003
Enter employee 3 name: Jaiyant
Enter employee 3 salary: 65000
Enter employee 4 ID: 1004
Enter employee 4 name: Kartik
Enter employee 4 salary: 48000
Select sorting criteria ("salary", "name", or "id"): salary
Enter employee name to search: Vikas

Stored Output:
Employee Records:
ID: 1001, Name: Yeshwanth, Salary: 55000
ID: 1002, Name: Deekshith, Salary:65000
ID: 1003, Name: Jaiyant Salary: 65000
ID: 1004, Name: Kartik Salary: 48000
Sorting by Salary:
Rank 1: Yeshwanth- 65000
Rank 2: Deekshith- 65000
Rank 3: Jaiyant - 55000
Rank 4: Kartik - 48000
Searching for Name "Yeshwanth":

Employee Yeshwanth found at rank 3.
ID: 1001, Salary: 55000

10. Test Cases

Test Case	Input	Operation	Expected Output
1	1001,Yeshwanth,55000; 1002, Deekshith,65000; 1003,Jaiyant,65000; 1004,Kartik,48000	Sorting by Salary	Yeshwanth,Deekshith,Jaiyant,Kartik (sorted by salary descending)
2	Above employees	Search Name "Jaiyant"	Found at rank 2, ID: 1003, Salary:65000
3	Above employees	Search Name "Manoj"	Not found
4	Add New Employee 1005,Sai,62000	Add	Record added successfully; included in future sorting/search
5	Edit Salary of Yeshwanth To 60000	Update	Salary updated; reflected in display, sorting

11. Future Enhancements

- Store and load employee data from files
- Support floating-point salaries
- Advanced search (partial name, department, join date)
- Enhanced sorting options (multi-field, date order)
- Graphical representation of employee stats
- Proper rank assignment for salary ties
- Multiple designations per employee
- User interface: menu systems (add, sort, search, display all, delete, update, exit)
- Export to CSV/Excel for reporting

12. Conclusion

The Employee Record Management System is a basic but robust application for recording, updating, and managing employee information in a digital format. It assists HR departments by providing fast, reliable access to structured data, efficient searching & sorting, and ensures streamlined organizational workflow.

13. References

1. Data Structures classes by Jashwanth Sir
2. Chat-GPT
3. ByteXL-App