
PROJECT REPORT

Course Code: CSEG1041

Course Title: Programming in C

Semester: [1]

Project Title Employee Management System.

Student Names :

- Nausheen Parveen
- Shourya Sharma

Roll Numbers:

[590026385 , 590027524]

Faculty: Vinod Kumar Sir.

1. ABSTRACT

This project implements a menu-driven Employee Management System using the C programming language. The system allows users to add, view, search, update, and delete employee records stored permanently in a file. It uses structures to represent employee data and demonstrates the application of file handling for data persistence. The project also applies modular programming concepts to build a clear, efficient, and maintainable management system.

2. OBJECTIVE

The objective of this project is to design and implement a simple, modular, and efficient Employee Management System using the C programming language. The system should automate key operations such as adding employee records, viewing details, searching for specific employees, updating information, and deleting entries, with all data stored persistently using file handling. The aim is to replace manual employee record-keeping with a faster, more reliable, and organized digital solution while reinforcing essential C programming concepts such as structures, functions, arrays, file operations, and modular program organization.

3. PROBLEM DEFINITION

In many small offices, shops, and institutions, employee information is still handled manually, which often leads to mistakes, difficulty in updating records, and loss of important data. There is a need for a simple, reliable, and efficient digital system to store and manage employee details.

The goal of this project is to build a console-based Employee Management System in C that automates basic record-handling tasks and improves accuracy and accessibility.

The system shall:

• Store employee details such as ID, Name, Age, Position, Salary, etc.

• Add new employee records

• Display all stored records

• Search for a specific employee by ID or name

• Update employee information when needed

• Delete employee records

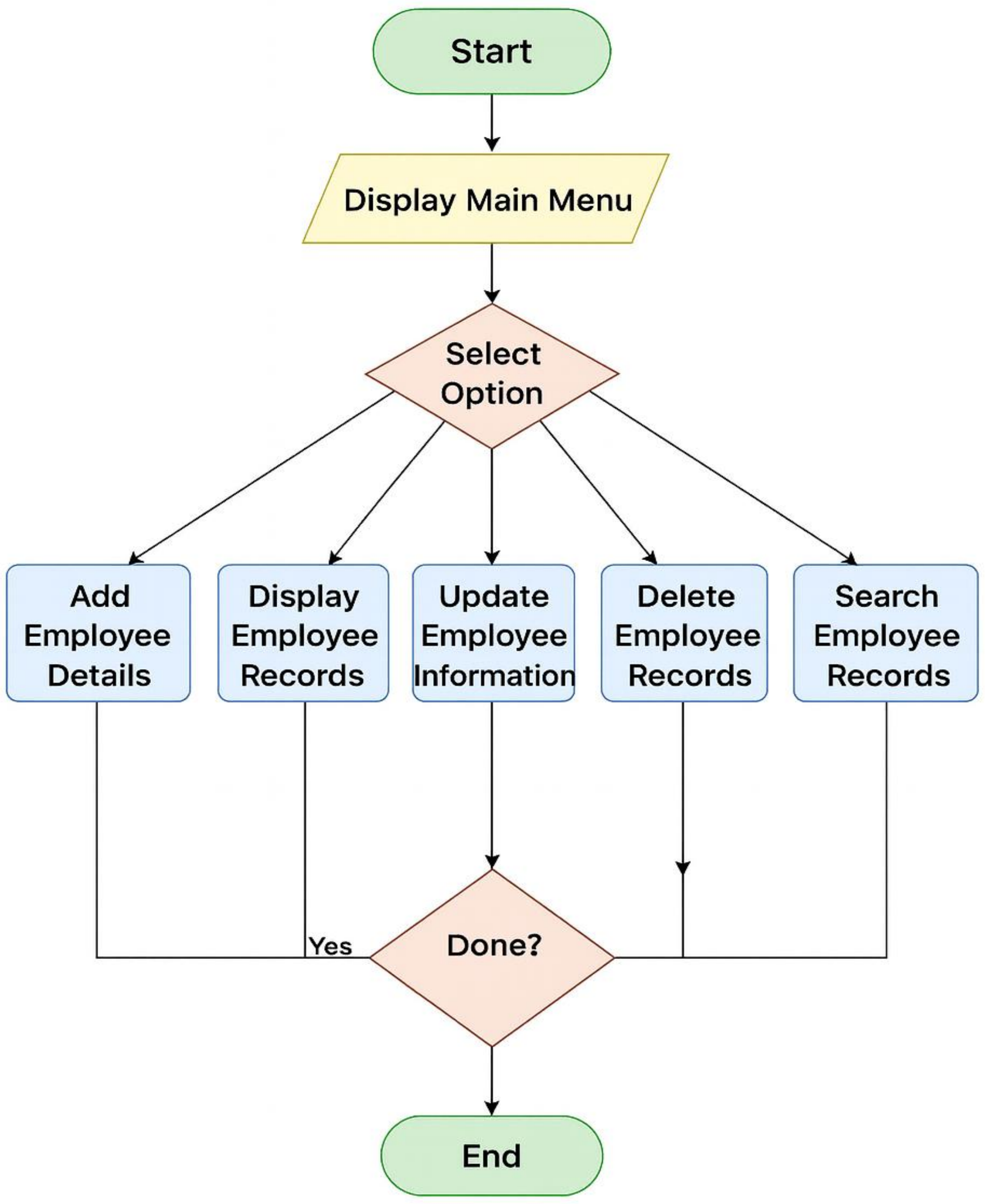
• Maintain all data permanently using file handling

This project also reinforces key C programming concepts such as structures for organizing data, functions for modular design, arrays for storing multiple records, and text-file operations for persistent data management.

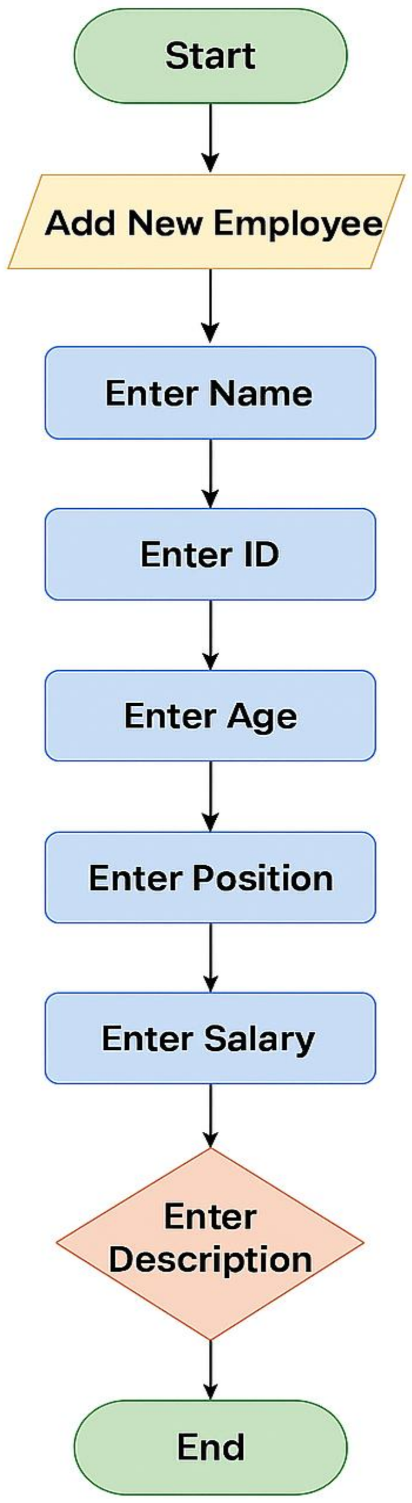
4. SYSTEM DESIGN AND ALGORITHM(FLOWCHART, ALGORITHM) :

FLOWCHARTS

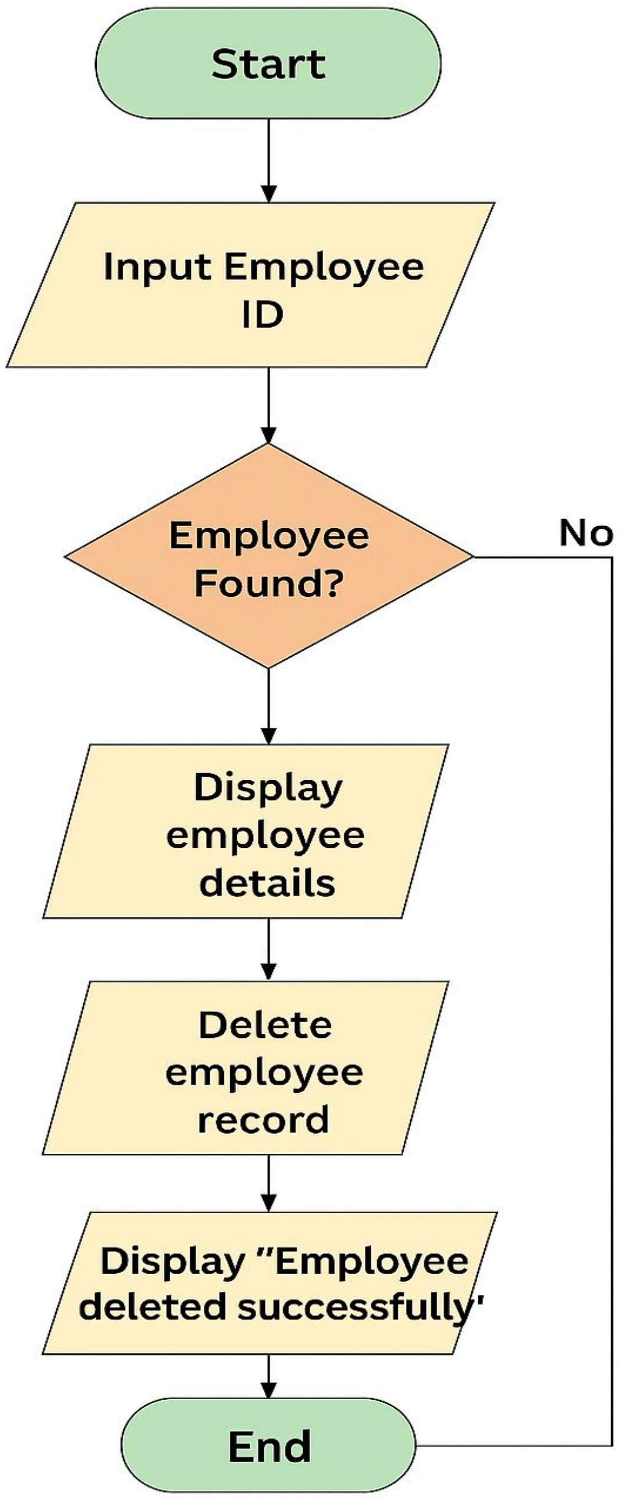
FLOWCHART 1 :- DISPLAY OF MAIN MENU



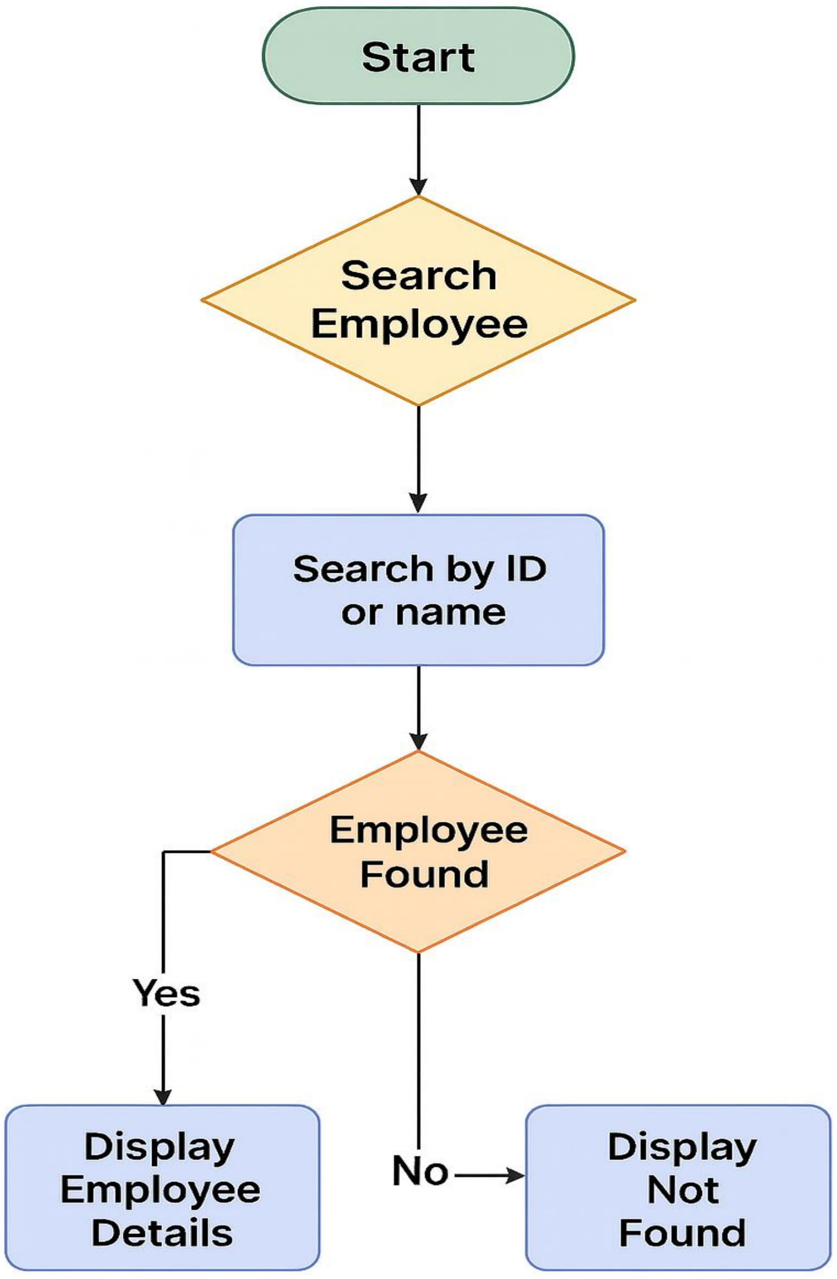
FLOWCHART 2 :- ADDING NEW EMPLOYEE



FLOWCHART 3 :- DELETE EMPLOYEE RECORD



FLOWCHART 4 :- SEARCH EMPLOYEE RECORD



ALGORITHMS

Algorithm 1: Add Employee

Step 1: Read the next available Employee ID from the file.
Step 2: Display the generated Employee ID to the user.
Step 3: Take input for employee name, age, position, salary, and description.
Step 4: Set default values for any additional fields if needed (e.g., status = active).
Step 5: Save the employee details to the file using save_employee().
Step 6: Display “Employee added successfully”.

Algorithm 2: List Employees

Step 1: Open the employee data file using fopen("r").
Step 2: Read each employee record using fread().
Step 3: Store each record into an array for processing.
Step 4: Display all employee records one by one.

Algorithm 3: Search Employee (By ID or Name)

Step 1: Display search menu with options:

- Search by ID
- Search by Name

Step 2: Read user choice.
Step 3: If choice = ID → ask for Employee ID.
Step 4: If choice = Name → ask for Employee Name.
Step 5: Scan each record in the file and compare with search input.

Step 6: If a matching employee is found → display details.
Step 7: If no match found → display "Employee not found".

Algorithm 4: Update Employee

Step 1: Ask the user to enter Employee ID to update.
Step 2: Search employee using the ID.
Step 3: If employee not found → display "Record not found" and stop.
Step 4: If found → display current details.
Step 5: Ask user which fields to update (name, age, position, salary, description).
Step 6: Modify the selected fields in the employee record.
Step 7: Save updated record back to the file.
Step 8: Display "Employee updated successfully".

Algorithm 5: Delete Employee

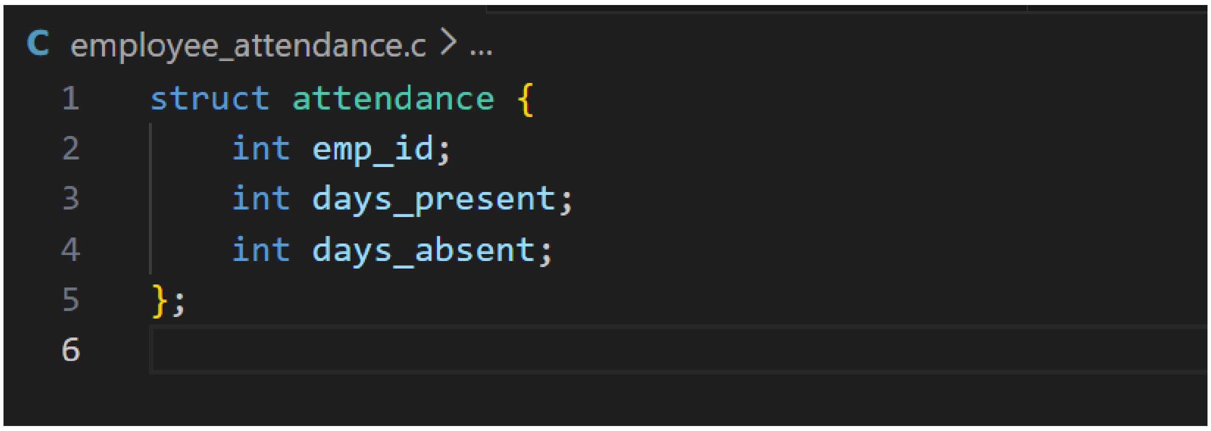
Step 1: Ask the user to enter Employee ID to delete.
Step 2: Search employee using the ID.
Step 3: If employee not found → display "Record not found".
Step 4: If found → ask for confirmation (Yes/No).
Step 5: If Yes → remove employee record from file (copy all others to a new file).
Step 6: Replace old file with updated file.
Step 7: Display "Employee deleted successfully".

---

5. IMPLEMENTATION DETAILS (WITH SNIPPETS):

Structure for Attendance Record :

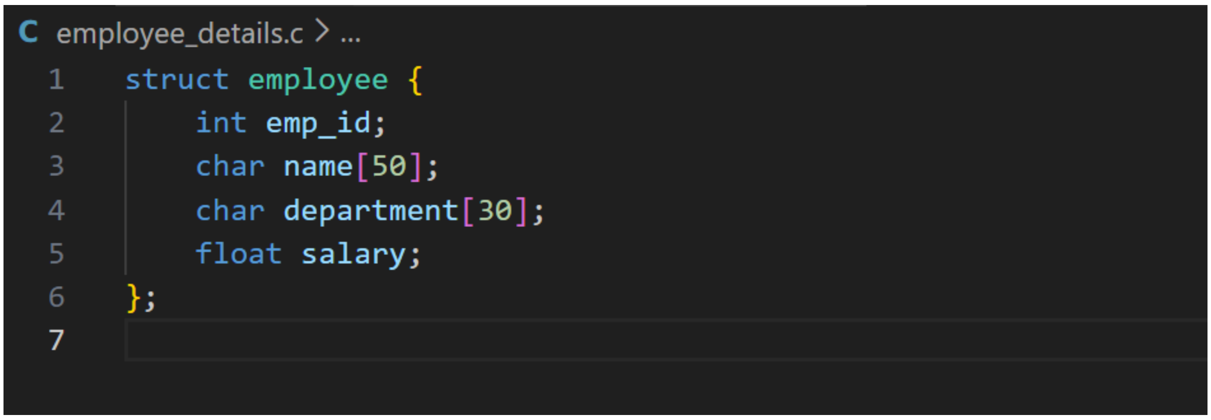
This structure tracks the attendance status of employees based on their unique IDs.



```c
struct attendance {
    int emp_id;
    int days_present;
    int days_absent;
};
```

Structure for Employee Details :

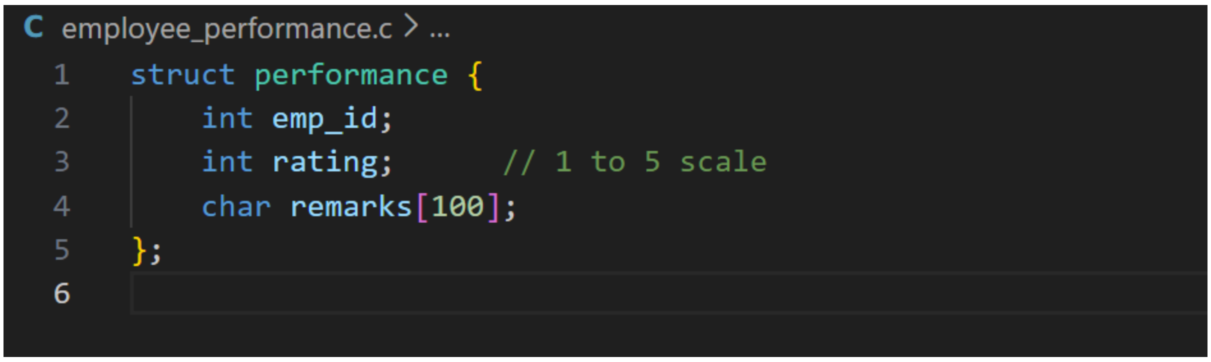
This structure holds the essential details of each employee, including identification, personal information, and salary.



```c
struct employee {
    int emp_id;
    char name[50];
    char department[30];
    float salary;
};
```

Structure for Employee Performance Evaluation :

This structure is used to store performance ratings and feedback for each employee.


employee_performance.c > ...

```c
struct performance {
    int emp_id;
    int rating;      // 1 to 5 scale
    char remarks[100];
};
```

6. PROBLEM FACED BY OUR GROUP:

While working on our Employee Management System project as a team of two, we came across several challenges that tested our coordination and technical understanding. Since both of us were handling multiple roles, dividing the tasks in a balanced way took some trial and error. We also struggled at times with aligning our coding styles and integrating our individual modules without conflicts. Debugging became especially time-consuming because even small errors affected the overall workflow. Despite these issues, each challenge helped us communicate better, understand the system more deeply, and improve our teamwork.

7.SLIP OF HOLD (EMPLOYEE RECORD SLIP):

In an Employee Management System, whenever we add a new employee or view an existing employee's full details, we can generate a small **Employee Information Slip**.

-------------------- EMPLOYEE INFORMATION SLIP --------------------

Company Name : [Your Company Name]

Employee Name	: ________________________
Employee ID	: ________________________
Department	: ________________________
Designation	: ________________________
Age	: ________________________
Salary	: ________________________

Date of Issue : ___ / ___ / ___

HR Signature : ________ Employee Signature : ________

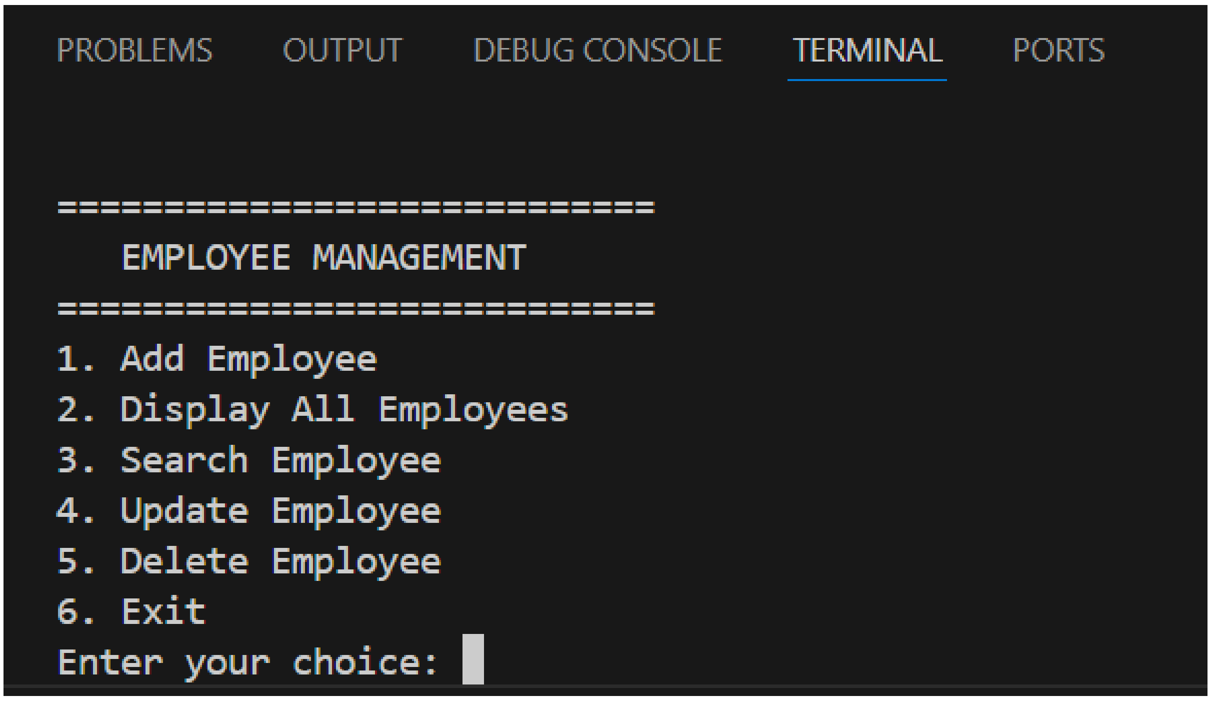
SAMPLE OUTPUT :

“This slip can be printed or written on paper and attached to the employee's HR file.”

```
-------------------- EMPLOYEE INFORMATION SLIP --------------------


            Company Name : ABC Technologies Pvt. Ltd.


                Employee Name : Rahul Mehta

                  Employee ID   : 1023

             Department    : Software Development

               Designation   : Junior Developer

                    Age          : 24

                  Salary       : 32,000 INR


                Date of Issue : 04-12-2025


     HR Signature : ________     Employee Signature : ________


     ----------------------------------------------------------------
```

8. OUTPUT :

The main menu, display output ! where we can see list /slip of hold on our terminal.

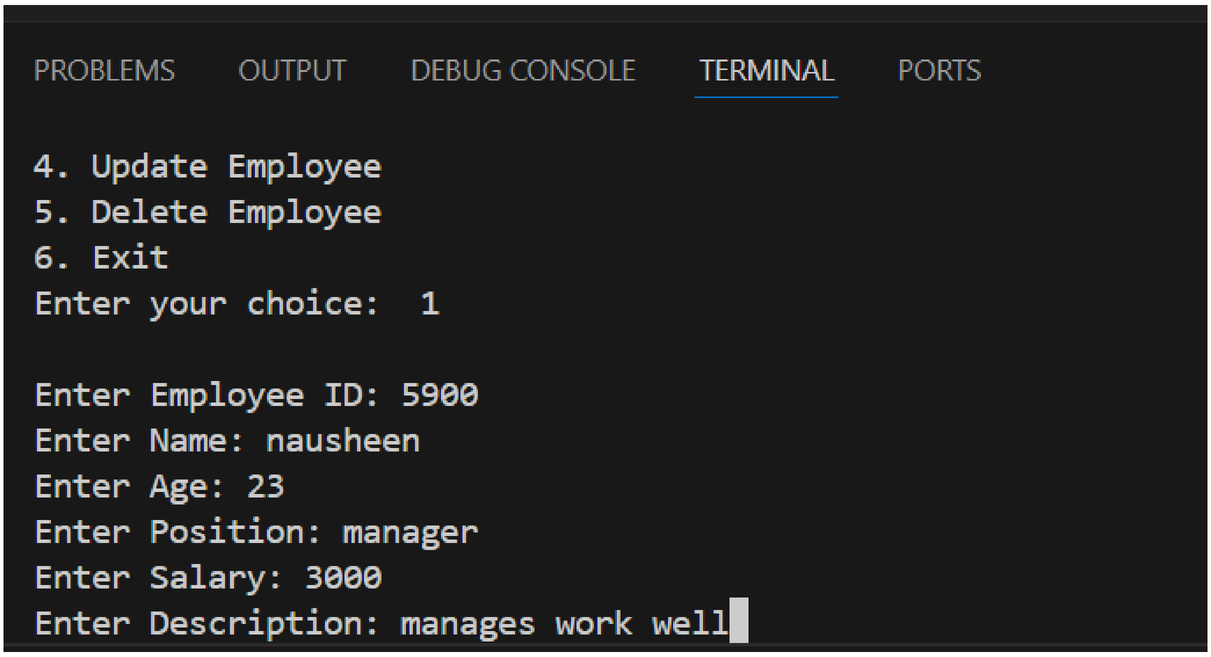


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
============================
   EMPLOYEE MANAGEMENT
============================
1. Add Employee
2. Display All Employees
3. Search Employee
4. Update Employee
5. Delete Employee
6. Exit
Enter your choice: 
```

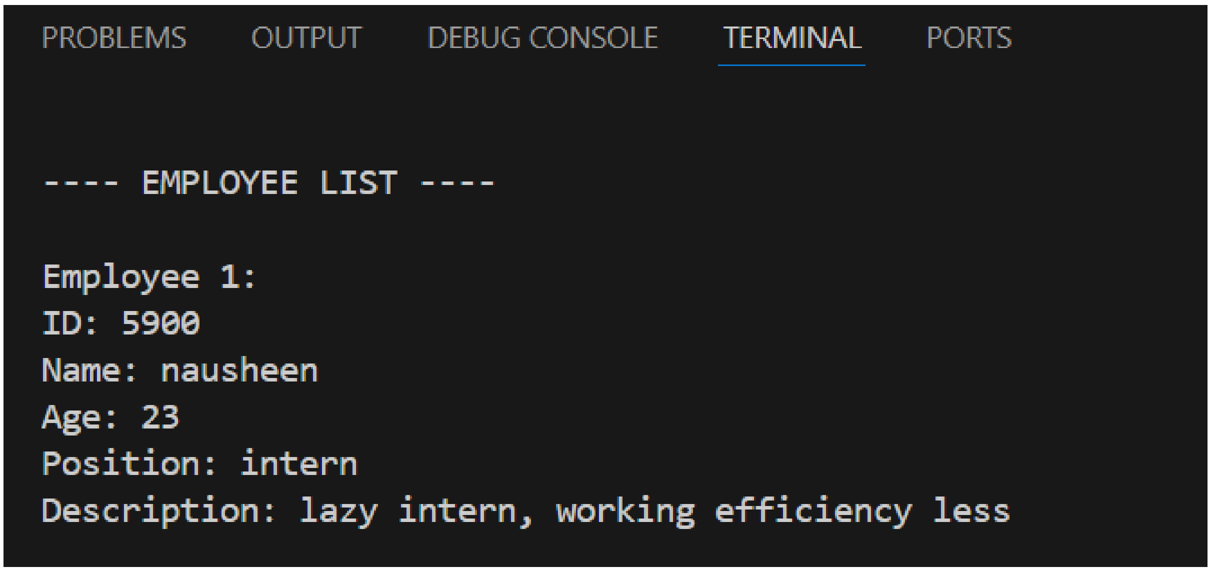
9. TESTING AND RESULTS :

TEST CASE 1: TO ADD AN EMPLOYEE



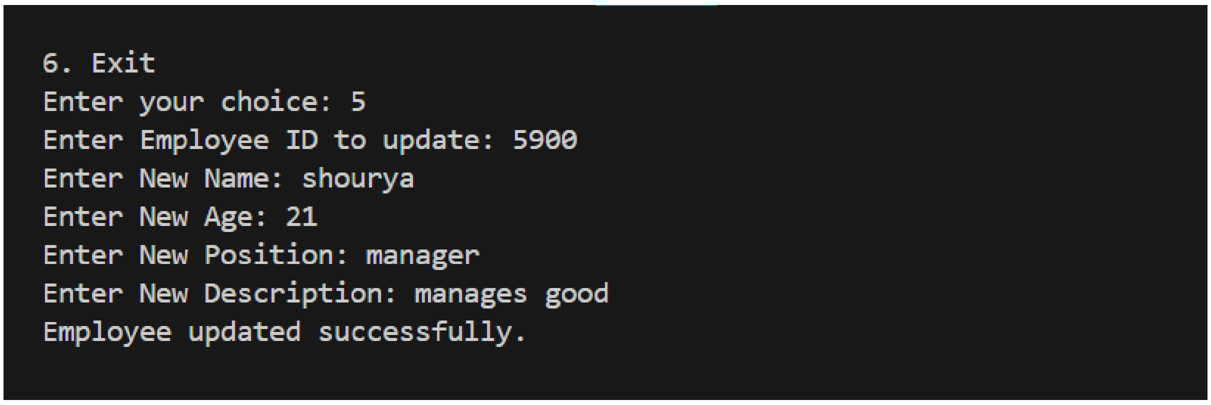
```
4. Update Employee
5. Delete Employee
6. Exit
Enter your choice:  1

Enter Employee ID: 5900
Enter Name: nausheen
Enter Age: 23
Enter Position: manager
Enter Salary: 3000
Enter Description: manages work well
```

TEST CASE 2: TO DISPLAY ALL EMPLOYEES



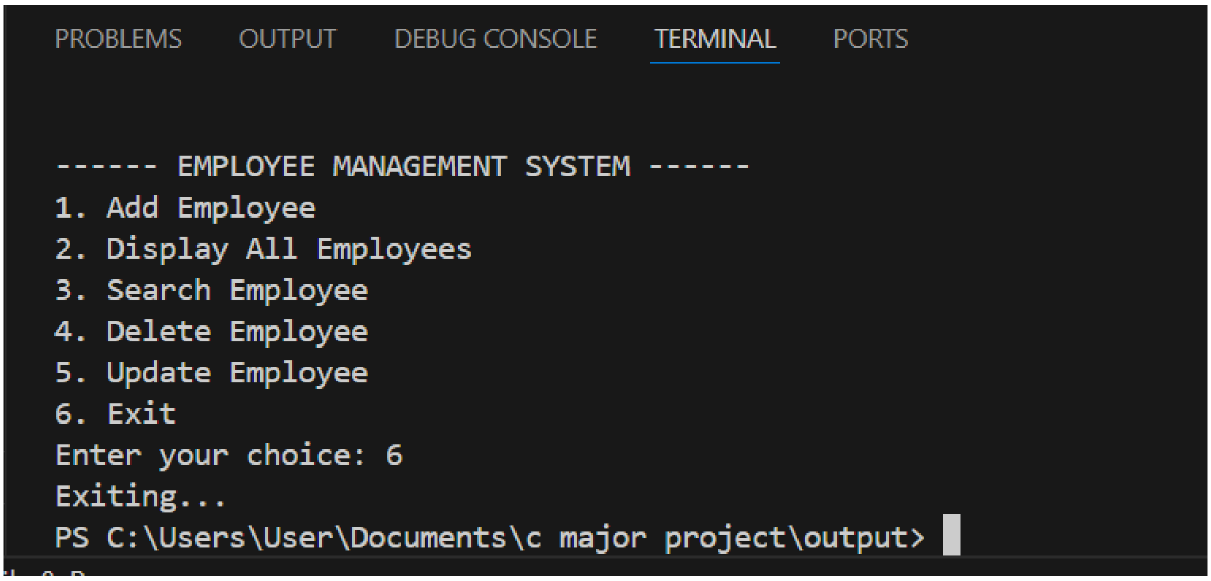
```
---- EMPLOYEE LIST ----

Employee 1:
ID: 5900
Name: nausheen
Age: 23
Position: intern
Description: lazy intern, working efficiency less
```

TEST CASE 3: TO UPDATE EMPLOYEE



```
6. Exit
Enter your choice: 5
Enter Employee ID to update: 5900
Enter New Name: shourya
Enter New Age: 21
Enter New Position: manager
Enter New Description: manages good
Employee updated successfully.
```

TEST CASE 4: HOW TO EXIT



```
------ EMPLOYEE MANAGEMENT SYSTEM ------
1. Add Employee
2. Display All Employees
3. Search Employee
4. Delete Employee
5. Update Employee
6. Exit
Enter your choice: 6
Exiting...
PS C:\Users\User\Documents\c major project\output>
```

10.Future Scope :

- Allow users to edit or remove specific employee fields (name, department, salary, etc.) directly from the interface with improved validation.
- Maintain a separate log file to track changes such as promotions, salary updates, and new hires for better record-keeping.
- Add an Admin Mode with password protection to secure sensitive operations like deleting records or modifying employee data.
- Provide options to export employee reports in various formats such as text, CSV, or Excel for easier sharing and documentation.
- Implement sorting and filtering features, allowing users to organize employees by ID, name, department, designation, or salary range.
- Introduce advanced search filters (e.g., search by salary range, joining date, or department) to improve data accessibility.

11. Conclusion :

The Employee Management System developed in C successfully provides a simple and effective way to manage employee information through a console-based interface. The system allows users to **add new employees, view all records, search for specific employees, update existing details, and delete records**, ensuring smooth and organized data handling. All employee information is stored permanently using file handling, making it reliable for repeated use without data loss.

This project demonstrates the practical application of core C programming concepts such as **structures, functions, arrays, modular programming, and file operations**. Implementing CRUD operations strengthened the understanding of persistent storage and record manipulation, while designing a structured codebase improved debugging and problem-solving skills.

Overall, the system meets its objective of providing a lightweight yet functional employee record-keeping tool. It forms a strong foundation for future enhancements such as sorting records, adding authentication, integrating a database, or creating a graphical user interface for improved usability.

12. References :

- TutorialsPoint. (n.d.). *Structures in C*.
  from
  https://www.tutorialspoint.com/cprogramming/c_structures.htm
  — Referred for understanding how to store employee attributes using struct in C

- GeeksforGeeks. (n.d.). *File Handling in C*.
  Retrieved from https://www.geeksforgeeks.org/file-handling-c/
  — Helped in implementing file operations such as reading, writing, appending and updating employee records.

- Shelly, G. B., & Vermaat, M. E. (2015). *Introduction to Computers* (8th ed.). Cengage Learning.— Provides information systems and data organization used during the planning of the Employee Management System.