EMPLOYEE MANAGEMENT

A PROJECT REPORT

Submitted by

VETRIVEL M(2303811710621121)

in partial fulfillment of requirements for the award of the course

EGB1122 – DATA STRUCTURES

in

ELECTRONICS AND COMMUNICATION ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

May, 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report titled **“EMPLOYEE MANAGEMENT”** is the bonafide work of **VETRIVEL M (2303811710621121),** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

[illegible] [illegible]

SIGNATURE	**SIGNATURE**
Mrs.A.Delphin Carolina Rani, M.E.,Ph.D.,	Mr.P.Devan.B.E.,M.E.,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR	ASSISTANT PROFESSOR
Department of CSE	Department of CSE
K. Ramakrishnan College of Technology (Autonomous)	K. Ramakrishnan College of Technology (Autonomous)
Samayapuram–621112.	Samayapuram–621112.

Submitted for the viva-voce examination held on 18.06.2024

[illegible]

INTERNAL EXAMINER

DECLARATION

I declare that the project report on **"EMPLOYEE MANAGEMENT"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfillment of the requirement of the award of the course **CGA1121- DATA STRUCTURES.**

Signature

M. Vetrivel

VETRIVEL M

Place: Samayapuram

Date: 18.06.2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and indebtedness to our institution, "**K. Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

I extend our sincere acknowledgment and appreciation to the esteemed and honorable Chairman, **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director, **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding our project and offering an adequate duration to complete it.

I would like to thank **Dr. N. VASUDEVAN, M.TECH., Ph.D.,** Principal, who gave the opportunity to frame the project to full satisfaction.

I thank **Dr. A. DELPHIN CAROLINA RANI M.E., Ph.D.,** Head of the Department of COMPUTER SCIENCE AND ENGINEERING, for providing her encouragement in pursuing this project.

I wish to convey our profound and heartfelt gratitude to our esteemed project guide **Ms.Sowmiya.M.E.,**Department of **COMPUTER SCIENCE AND ENGINEERING,** for her incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project.

I render our sincere thanks to the Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To emerge as a leader among the top institutions in the field of technical education.

MISSION OF THE INSTITUTION

- ➢ Produce smart technocrats with empirical knowledge who can surmount the global challenges.
- ➢ Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students.
- ➢ Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations.

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems ofsociety.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities

with an understanding of the limitations

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

Managing employee data manually is complex, error-prone, inefficient, and time-consuming. An automated solution is needed to handle tasks such as adding, updating, and removing employee records, setting salaries, and searching by specific criteria efficiently. The Employee Management System (EMS) automates the management of employee data, providing a user-friendly interface for administrators to perform operations like adding new employees, updating details, removing employees, setting salaries, displaying employees by designation, and searching by ID. The system is implemented using the C programming language, chosen for its suitability in system-level programming. Structures are used to define complex data types for employees and the management system. Robust input validation ensures accurate data entry using functions like 'scanf', ‘fgets', and sscanf, while buffer management techniques clear unwanted characters from the input stream. String handling functions such as 'strcpy’, 'strcspn', and 'strcmp' effectively manage and manipulate string data. Iterative control structures, including loops and conditionals, are employed to navigate and manipulate the array of employee structures based on user inputs. The implementation of the EMS successfully automates the management of employee records, significantly reducing errors and improving efficiency. The system provides a seamless experience for administrators, ensuring that employee data is managed accurately and kept up-to-date. This makes the EMS a valuable tool for maintaining organized and efficient employee data management within an organization.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS

UI	-	User Interface
EM	-	Employee management
ID	-	Identification
GUI	-	Graphical User Interface
APIs	-	Application Programming Interface

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO PROJECT

The Employee Management System (EMS) automates and streamlines the management of employee records, addressing the inefficiencies and errors of manual processes. the EMS provides a user-friendly interface for administrators to add, update, remove, and search for employees, as well as set salaries and display employees by designation. By using structured arrays and robust input validation, the system ensures accurate data entry, organized storage, and efficient manipulation of employee information. This tool significantly enhances the accuracy and efficiency of employee data management, making it a valuable asset for organizations.

1.2 PURPOSE AND IMPORTANCE OF THE PROJECT

The Employee Management System (EMS) project serves a crucial purpose in modern organizations by automating and optimizing the management of employee records. Its significance lies in its ability to enhance accuracy, efficiency, and data integrity while facilitating quick access to employee information. By replacing manual methods with a reliable software solution, the EMS streamlines operations such as adding, updating, removing, and searching for employee records, ultimately saving time and resources for administrators.

Moreover, the system's structured storage of employee data ensures organized records, which are essential for effective human resource management. Overall, the EMS plays a pivotal role in increasing organizational productivity by handling routine administrative tasks efficiently and allowing personnel to focus on more strategic activities. Its implementation not only improves the efficiency of day-to-day operations but also contributes to a more organized and well-managed workforce, ensuring the smooth functioning of the organization.

1.3 OBJECTIVES

1. Automate employee data management tasks
2. Enhance accuracy and efficiency
3. Streamline HR record-keeping processes
4. Facilitate quick access to data
5. Improve organizational productivity
6. Ensure data integrity and organization

1.4 PROJECT SUMMARIZATION

The Employee Management System (EMS) is a software solution designed to automate and streamline the management of employee records within organizations. This project aims to replace manual methods of employee data handling with a reliable and efficient software system. The EMS offers several advantages:

a) **Efficiency**: By automating tasks such as adding, updating, and removing employee records, the EMS significantly reduces the time and effort required for administrative tasks.

b) **Accuracy**: With robust input validation and structured data storage, the system ensures accurate and consistent employee information, minimizing errors in record-keeping.

c) **Organization**: The structured storage of employee data enables easy access and navigation, contributing to a more organized and efficient workflow within the organization.

d) Data Integrity: The system maintains data integrity by enforcing validation rules and ensuring that only valid and relevant information is entered into the database.

e) **Accessibility**: The EMS facilitates quick access to employee information, enabling administrators to retrieve relevant data efficiently, thereby supporting informed decision-making processes.

CHAPTER 2
PROJECT METHODOLOGY

2.1 INTRODUCTION TO SYSTEM ARCHITECTURE

System architecture refers to the conceptual model that defines the structure, behavior, and more views of a system. It serves as a blueprint for both the system and the project developing it. In the context of employee management, the system architecture outlines how various components interact and work together to achieve the desired functionality.

2.1.1 High-Level System Architecture

The high-level system architecture of the Employee Management System (EMS) comprises several key components:

i. User Interface (UI)
ii. Application Logic
iii. Data Management Layer

2.1.2 Components of the System Architecture

a. User Interface (UI)

The User Interface in this context is a text-based menu interface. It presents a list of options to the user, captures their inputs through a command-line interface, and displays the results of their actions

b. Software Logic

The software Logic is the core of the EMS, handling the business rules and operations required to manage employee data. This includes functions for adding, updating, deleting, and searching for employee records, as well as managing employee salaries and ensuring data integrity through validation

c. Data Management Layer

The Data Management Layer is responsible for storing and retrieving employee information efficiently. It uses structured arrays to store employee records and implements operations while ensuring data integrity. This architecture ensures the system can manage employee data accurately and efficiently.

2.2 DETAILED SYSTEM ARCHITECTURE DIAGRAM

The Employee Management System employs a Command-Line Interface (CLI) for managing employee data, facilitating operations like adding, updating, and removing employees. The system is menu-driven, ensuring efficient data handling and reducing errors compared to manual processes. This straightforward interface simplifies maintaining accurate and current employee records.

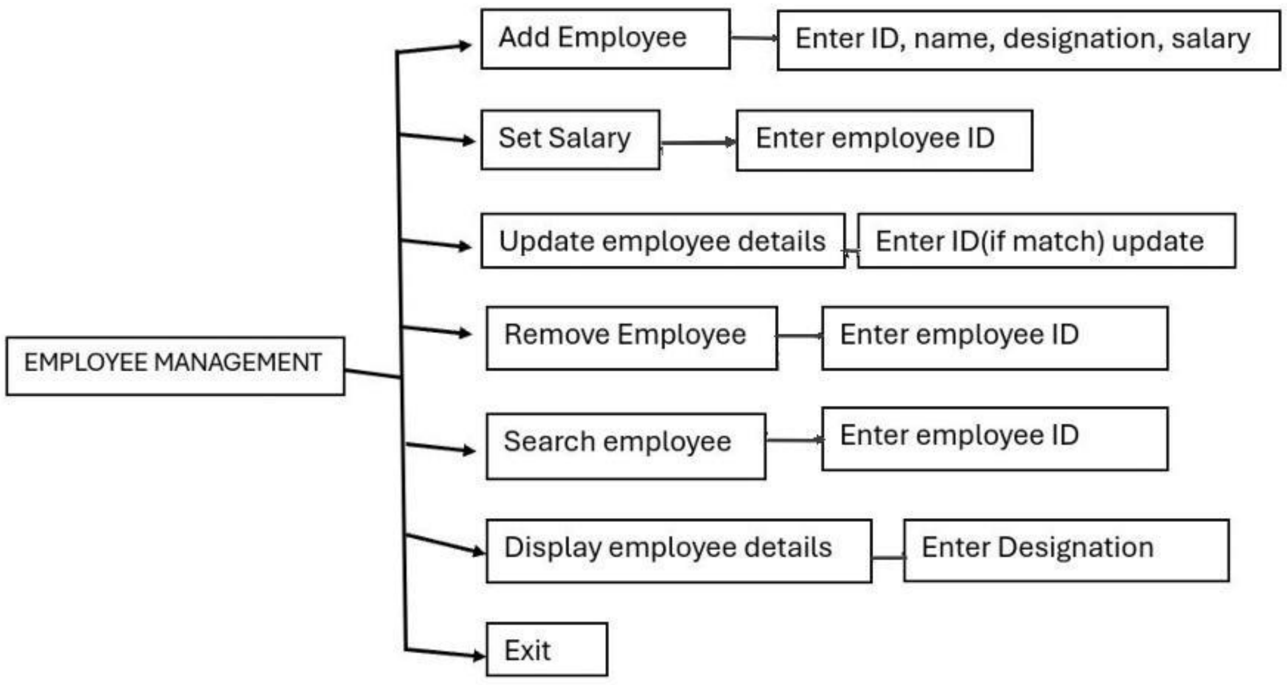


Fig 2.1 : Architecture Diagram

CHAPTER 3
DATA STRUCTURE PREFERANCE

3.1 EXPLANATION OF WHY A ARRAY WAS CHOSEN

In the employee management system, an array was chosen as the primary data structure for several compelling reasons:

3.1.1 Simplicity and Ease of Use: Arrays provide a straightforward way to store and access employee records using indices. This simplicity makes the implementation of basic operations like addition, deletion, and updates relatively easy to understand and manage.

3.1.2 Direct Access to Elements: Arrays allow direct access to any employee record using its index, enabling fast retrieval and modification operations. This direct access is particularly useful for operations that involve frequent reads and writes, such as updating employee details or setting salaries.

3.1.3 Memory Contiguity: Arrays store elements in contiguous memory locations, which improves cache performance and overall access speed. This memory contiguity can lead to better performance, especially when the dataset is not excessively large and fits well within the available memory.

3.1.4 Predictable Iteration: Iterating over an array is straightforward and predictable, which simplifies the implementation of operations like displaying all employees, searching for employees by ID or designation, and other iterative tasks.

3.1.5 Static Size Management: In many scenarios, the maximum number of employees can be estimated beforehand. Using a fixed-size array helps to manage memory allocation statically, avoiding the overhead of dynamic memory allocation and potential memory leaks.

3.1.6 Reduced Overhead: Unlike more complex data structures such as linked lists or trees, arrays have minimal overhead. This lack of additional pointers or structural complexity makes arrays more efficient in terms of both memory and processing time for the operations that are commonly performed in an employee management system.

3.2 COMPARISON WITH OTHER DATA STRUCTURES

Choosing a data structure for a employee management involves considering trade-offs. Arrays offer constant-time access, facilitating swift retrieval of employee information, yet they necessitate resizing to accommodate dynamic changes in the workforce, potentially impacting performance. Singly linked lists, though memory-efficient, lack efficient backward traversal, limiting their utility in certain employee management scenarios. Hash tables present constant-time operations, ideal for speedy data retrieval, but the occurrence of collisions introduces overhead, which might affect performance. For instance, a doubly linked list boasts efficient insertion, deletion, and bidirectional traversal capabilities, rendering it apt for dynamic employee management where updates are frequent.

3.3 ADVANTAGES AND DISADVANTAGES OF USING A ARRAY

3.3.1 Advantages of Using a Array:

Employing arrays in an Employee Management System (EMS) offers certain advantages, notably their constant-time access to employee records and straightforward implementation. This efficiency in retrieval operations is particularly beneficial for tasks requiring quick access to employee data, such as retrieving contact information or salary details.

3.3.2 Disadvantages of Using a Array:

Employing arrays in an EMS also presents challenges, especially when the size of the workforce changes dynamically. Resizing arrays to accommodate additions or deletions of employees can be inefficient, leading to performance overhead, particularly for larger datasets. Additionally, arrays occupy contiguous memory blocks, which can result in memory fragmentation, potentially limiting the scalability of the EMS as the dataset grows.

CHAPTER -4

DATA STRUCTURE METHODOLOGY

4.1 STRUCTURED ARRAY

A structured array methodology, data is organized in a linear structure where each element holds a specific type of data. This approach offers a systematic arrangement of elements, enabling efficient access and manipulation. Structured arrays are commonly used in scenarios where elements have a uniform data type or format, facilitating streamlined processing and retrieval operations.

4.1.1 Key features of a circular doubly linked list:

1.Linear Structure: Structured arrays organize data in a linear sequence, where each element is accessed sequentially based on its position or index within the array.

2.Homogeneous Data Type: All elements in a structured array typically hold data of the same type, ensuring uniformity and facilitating streamlined processing.

3.Fixed Size: Structured arrays have a predetermined size that is fixed during initialization. This fixed size allows for efficient memory allocation and access but may require resizing operations if the array needs to accommodate more elements than initially allocated.

4.Random Access: Elements in a structured array can be accessed directly using their indices, allowing for constant-time access to any element.

Efficient Memory Usage: Structured arrays use contiguous memory blocks to store elements, enabling efficient memory utilization and fast access times.

5.Simple Implementation: Implementing structured arrays is straightforward, making them suitable for a wide range of applications where data organization and retrieval efficiency are crucial.

6.Limited Dynamic Behavior: Unlike dynamic data structures like linked lists, structured arrays have limited dynamic behavior. They cannot easily accommodate changes in size or accommodate elements of varying sizes without resizing operations.

1.2. ARRAY STRUCTURE

The Employee Array Structure represents a format for organizing employee data within a system. an individual employee record , with detailing specific attributes such as Employee ID , Name , and Position. For instance , represent an employee with his ID , name and position . This structured format enables efficient storage and retrieval of employee information , facilitating tasks such as accessing individual records or performing operations based on specific criteria. It allows for easy expansion by adding more rows for additional employees , making it adaptable to varying workforce sizes.

7.3.INITIALIZATION, INSERTION & DELETION

Initialization of a structured array involves allocating memory for a predefined number of elements and setting initial values if required. Insertion of new elements typically involves finding an empty slot in the array and placing the new data item there. However, if the array is full, resizing might be necessary, which can involve creating a new, larger array and copying existing elements. Deletion from a structured array usually entails marking the element as empty or nullifying its value, making the slot available for reuse. However, if maintaining order is crucial, shifting elements might be necessary to close the gap left by the deleted item. Overall, while structured arrays offer efficient access and manipulation, proper initialization, insertion, and deletion strategies are vital for effective data management.

CHAPTER-5
MODULES

5.1 ADD NEW EMPLOYEE

5.1.1 Function Name: `addEmployee()`

Description: Adds a new employee to the system by prompting for ID, name, designation, and salary.

5.2 SET SALARY

5.2.1 Function name: 'setSalary()'

Description: It will Sets or updates the salary of an existing employee by entering their ID and the new salary value.

5.3 SEARCH EMPLOYEE

5.3.1 Function name: `searchEmployeeById()`

Description: It enables the user to enter the ID of the employee they want and it will show the employee details such as name, designation and salary.

5.4 UPDATE EMPLOYEE DETAILS

5.4.1 Function name:`updateEmployee() `

Description: Updates the details (name, designation, salary) of an existing employee by providing their ID and new values.

5.5 REMOVE AN EMPLOYEE

5.5.1 Function name:`removeEmployee()`

Description: Removes an employee from the system by entering their ID.

5.6 DISPLAY EMPLOYEE DETAILS BY DESIGNATION

5.6.1 Function name:`displayEmployeesByDesignation()`

Description: Displays employees based on their designation by entering the desired designation.

CHAPTER-6
CONCLUSION & FUTURE SCOPE

6.1 CONCLUSION

The Employee Management System (EMS) presented here offers essential functionalities for managing employee data efficiently. Users can add, update, remove, and search for employees based on various criteria such as ID, name, and designation. The system provides a user-friendly interface with input validation to ensure data integrity. Overall, the EMS serves as a valuable tool for organizations to streamline their employee management processes and maintain accurate records.

6.2 FUTURE SCOPE

- The future scope of the Employee Management System includes several Potential enhancements and expansions . One area for improvement is the implementation of additional features such as employee performance tracking attendance management , and task assignment . Integrating with external databases or cloud services could enhance scalability and accessibility.

- Furthermore, incorporating data visualization tools could provide insights into employee trends and performance metrics. Additionally, implementing security measures such as user authentication and data encryption would enhance the system’s robustness and compliance with privacy regulations . Overall , by continuously evolving and incorporating new functionalities , the EMS can better meet the meet volving needs of organizations in managing their workforce effectively.

APPENDICES

APPENDIX A-SOURCE CODE

```
#include <stdio.h>

#include <string.h>

#include <stdlib.h>

// Assuming these structures are defined somewhere in your code

Struct Employee {

   Int id;

   Char name[100];

   Char designation[100];

   Float salary;

};

Struct EmployeeManagementSystem {

   Struct Employee employees[100];

   Int employeeCount;

};

// Function to flush the input buffer

Void flushInputBuffer() {

   Int c;

   While ((c = getchar()) != ‘\n’ && c != EOF);

}

Void addEmployee(struct EmployeeManagementSystem *ems) {

   Struct Employee newEmployee;
```

```
    Printf("Enter ID of the new employee: ");
    While (scanf("%d", &newEmployee.id) != 1) {
        Printf("Invalid input. Please enter a valid integer for the ID: ");
        flushInputBuffer();
    }
    flushInputBuffer(); // Consume newline character left by scanf
    printf("Enter name of the new employee: ");
    fgets(newEmployee.name, sizeof(newEmployee.name), stdin);
    newEmployee.name[strcspn(newEmployee.name, "\n")] = '\0';
    printf("Enter designation of the new employee: ");
    fgets(newEmployee.designation, sizeof(newEmployee.designation), stdin);
    newEmployee.designation[strcspn(newEmployee.designation, "\n")] = '\0';
    printf("Enter salary of the new employee: ");
    while (scanf("%f", &newEmployee.salary) != 1) {
        printf("Invalid input. Please enter a valid number for the salary: ");
        flushInputBuffer();
    }
    flushInputBuffer(); // Consume newline character left by scanf
    ems->employees[ems->employeeCount] = newEmployee;
    ems->employeeCount++;
    printf("New employee added with ID %d\n", newEmployee.id);
}
```

```
Void setSalary(struct EmployeeManagementSystem *ems) {
   Int id;
   Float newSalary;

   Printf("Enter ID of the employee to set salary: ");
   While (scanf("%d", &id) != 1) {
      Printf("Invalid input. Please enter a valid integer for the ID: ");
      flushInputBuffer();
   }
   flushInputBuffer(); // Consume newline character left by scanf

   printf("Enter new salary of the employee: ");
   while (scanf("%f", &newSalary) != 1) {
      printf("Invalid input. Please enter a valid number for the salary: ");
      flushInputBuffer();
   }
   flushInputBuffer(); // Consume newline character left by scanf

   for (int I = 0; I < ems->employeeCount; i++) {
      if (ems->employees[i].id == id) {
         ems->employees[i].salary = newSalary;
         printf("Salary updated for employee with ID %d\n", id);
         return;
      }
   }
   Printf("Employee with ID %d not found in the system.\n", id);
}
```

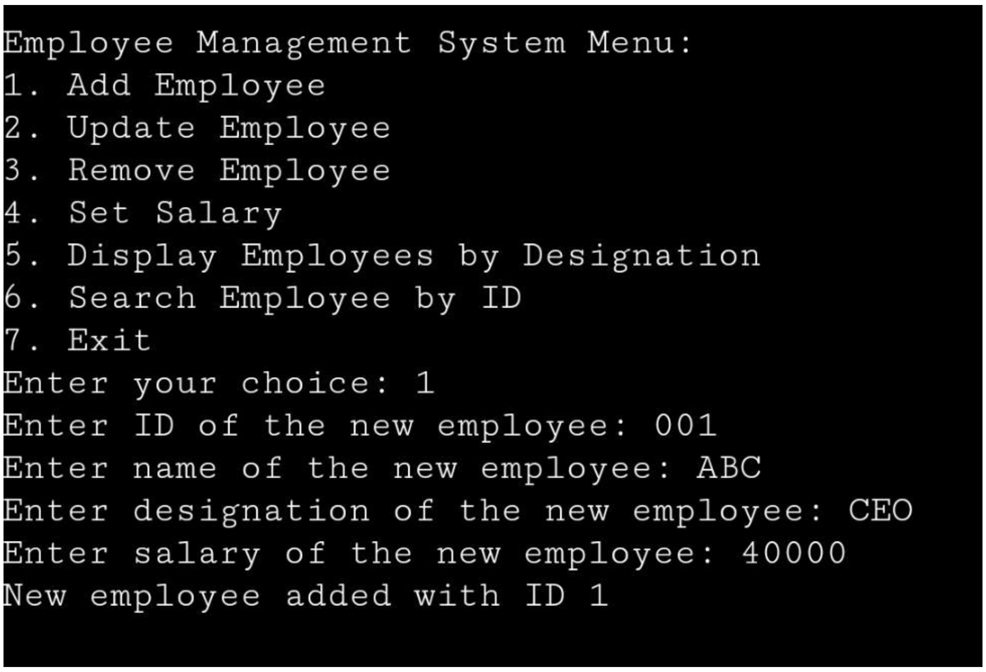
```
Void updateEmployee(struct EmployeeManagementSystem *ems) {
  Int id;
  Printf("Enter ID of the employee to update: ");
  While (scanf("%d", &id) != 1) {
    Printf("Invalid input. Please enter a valid integer for the ID: ");
    flushInputBuffer();
  }
  flushInputBuffer(); // Consume newline character left by scanf

  int I;
  for (I = 0; I < ems->employeeCount; i++) {
    if (ems->employees[i].id == id) {
      char newName[100] = "";
      char newDesignation[100] = "";
      char salaryInput[20] = "";
      float newSalary = -1.0f; // Initialize to a value that indicates no change

      printf("Enter new name of the employee (leave blank to keep current): ");
      fgets(newName, sizeof(newName), stdin);
      newName[strcspn(newName, "\n")] = '\0'; // Remove newline character

      printf("Enter new designation of the employee (leave blank to keep current): ");
      fgets(newDesignation, sizeof(newDesignation), stdin);
      newDesignation[strcspn(newDesignation, "\n")] = '\0'; // Remove newline character

      printf("Enter new salary of the employee (leave blank to keep current): ");
      fgets(salaryInput, sizeof(salaryInput), stdin);
      if (strlen(salaryInput) > 1) { // Check if something other than Enter was entered
        sscanf(salaryInput, "%f", &newSalary);
      }

      If (strlen(newName) > 0) {
        Strcpy(ems->employees[i].name, newName);
      }
      If (strlen(newDesignation) > 0) {
        Strcpy(ems->employees[i].designation, newDesignation);
      }
      If (newSalary > 0) {
        Ems->employees[i].salary = newSalary;
      }
      Printf("Details updated for employee with ID %d\n", id);
      Return;
    }
  }
  Printf("Employee with ID %d not found in the system.\n", id);
}

Void removeEmployee(struct EmployeeManagementSystem *ems) {
  Int id;
  Printf("Enter ID of the employee to remove: ");
  While (scanf("%d", &id) != 1) {
    Printf("Invalid input. Please enter a valid integer for the ID: ");
    flushInputBuffer();
  }
  flushInputBuffer(); // Consume newline character left by scanf
  int I, j;
  for (I = 0; I < ems->employeeCount; i++) {
    if (ems->employees[i].id == id) {
      for (j = I; j < ems->employeeCount – 1; j++) {
            ems->employees[j] = ems->employees[j + 1];
         }
         Ems->employeeCount--;
         Printf("Employee with ID %d has been removed from the system.\n", id);
         Return;
      }
   }
   Printf("Employee with ID %d not found in the system.\n", id);
}

Void displayEmployeesByDesignation(struct EmployeeManagementSystem *ems) {
   Char designation[100];
   Printf("Enter designation to search: ");
   Fgets(designation, sizeof(designation), stdin);
   Designation[strcspn(designation, "\n")] = '\0'; // Remove newline character
   Int found = 0;
   For (int I = 0; I < ems->employeeCount; i++) {
      If (strcmp(ems->employees[i].designation, designation) == 0) {
         Printf("ID: %d, Name: %s, Designation: %s, Salary: %.2f\n",
               Ems->employees[i].id, ems->employees[i].name,
               Ems->employees[i].designation, ems->employees[i].salary);
         Found = 1;
      }
   }
   If (!found) {
      Printf("No employees found with designation %s\n", designation);
   }
}
Void searchEmployeeById(struct EmployeeManagementSystem *ems) {
   Int id;
```

```
    Printf("Enter ID of the employee to search: ");
    While (scanf("%d", &id) != 1) {
        Printf("Invalid input. Please enter a valid integer for the ID: ");
        flushInputBuffer();
    }
    flushInputBuffer(); // Consume newline character left by scanf
    for (int I = 0; I < ems->employeeCount; i++) {
        if (ems->employees[i].id == id) {
            printf("Employee found:\n");
            printf("ID: %d, Name: %s, Designation: %s, Salary: %.2f\n",
                   ems->employees[i].id, ems->employees[i].name,
                   ems->employees[i].designation, ems->employees[i].salary);
            return;
        }
    }
    Printf("Employee with ID %d not found in the system.\n", id);
}
int main() {
    Struct EmployeeManagementSystem ems;
    Ems.employeeCount = 0;
    Int choice;
    While (1) {
        Printf("\nEmployee Management System Menu:\n");
        Printf("1. Add Employee\n");
        Printf("2. Update Employee\n");
        Printf("3. Remove Employee\n");
        Printf("4. Set Salary\n");
        Printf("5. Display Employees by Designation\n");
        Printf("6. Search Employee by ID\n");
        Printf("7. Exit\n");
Printf(“Enter your choice: “);
While (scanf(“%d”, &choice) != 1) {
   Printf(“Invalid input. Please enter a valid integer for the choice: “);
   flushInputBuffer();
}
flushInputBuffer(); // Consume newline character left by scanf
switch (choice) {
   case 1:
      addEmployee(&ems);
      break;
   case 2:
      updateEmployee(&ems);
      break;
   case 3:
      removeEmployee(&ems);
      break;
   case 4:
      setSalary(&ems);
      break;
   case 5:
      displayEmployeesByDesignation(&ems);
      break;
   case 6:
      searchEmployeeById(&ems);
      break;
   case 7:
      printf(“Exiting the program.\n”);
      return 0;
   default:
      printf(“Invalid choice. Please try again.\n”);
```

```
        }
    }
    Return 0;
}
```

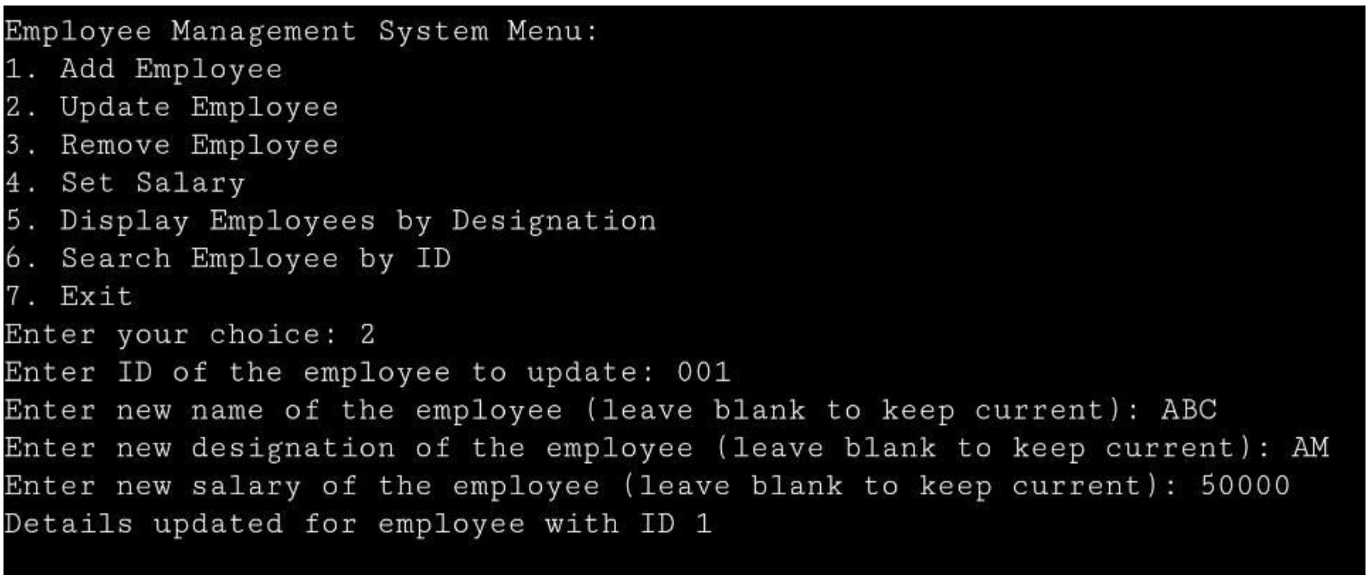
APPENDIX B -
SCREENSHOTS RESULT
AND DISCUSSION

Add employee



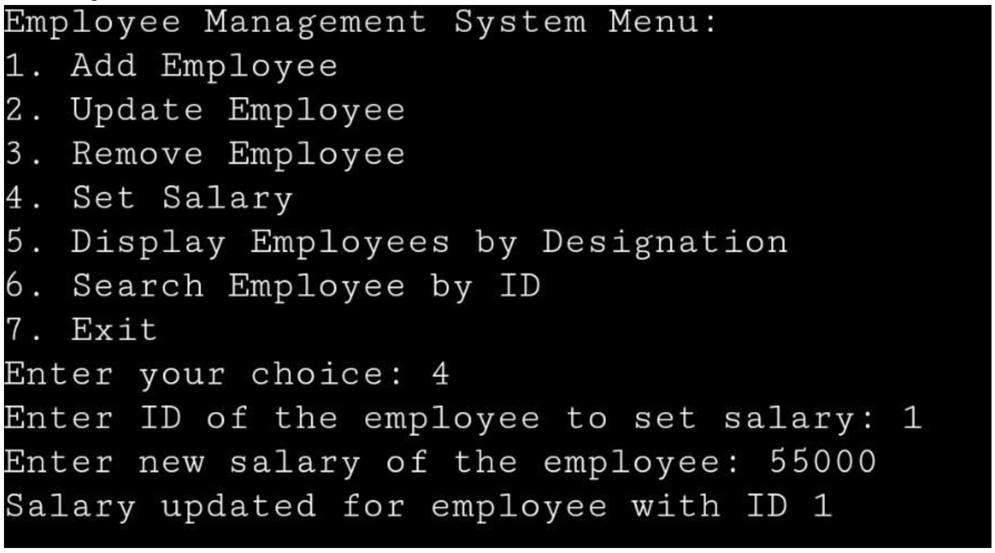
```
Employee Management System Menu:
1. Add Employee
2. Update Employee
3. Remove Employee
4. Set Salary
5. Display Employees by Designation
6. Search Employee by ID
7. Exit
Enter your choice: 1
Enter ID of the new employee: 001
Enter name of the new employee: ABC
Enter designation of the new employee: CEO
Enter salary of the new employee: 40000
New employee added with ID 1
```

Update employee details



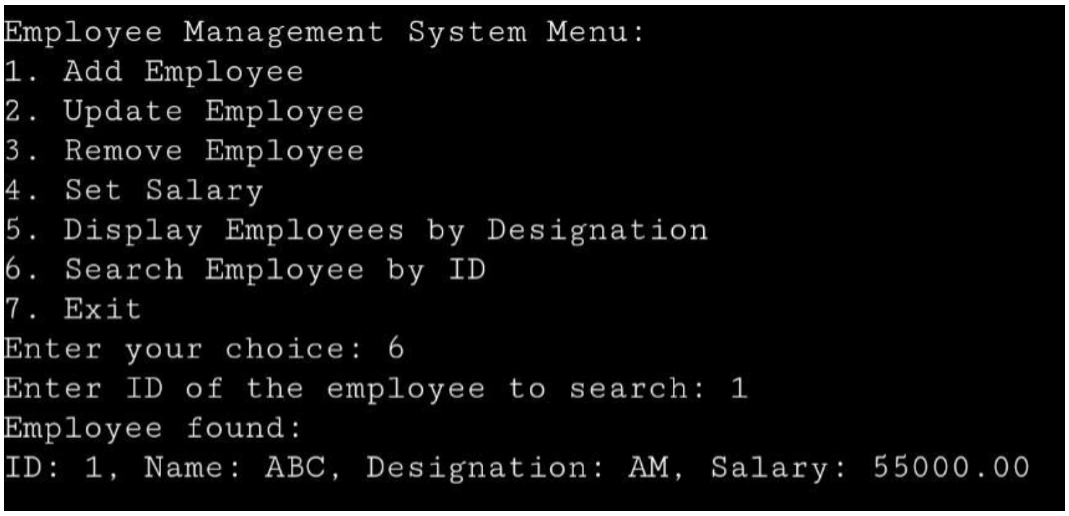
```
Employee Management System Menu:
1. Add Employee
2. Update Employee
3. Remove Employee
4. Set Salary
5. Display Employees by Designation
6. Search Employee by ID
7. Exit
Enter your choice: 2
Enter ID of the employee to update: 001
Enter new name of the employee (leave blank to keep current): ABC
Enter new designation of the employee (leave blank to keep current): AM
Enter new salary of the employee (leave blank to keep current): 50000
Details updated for employee with ID 1
```

Set salary



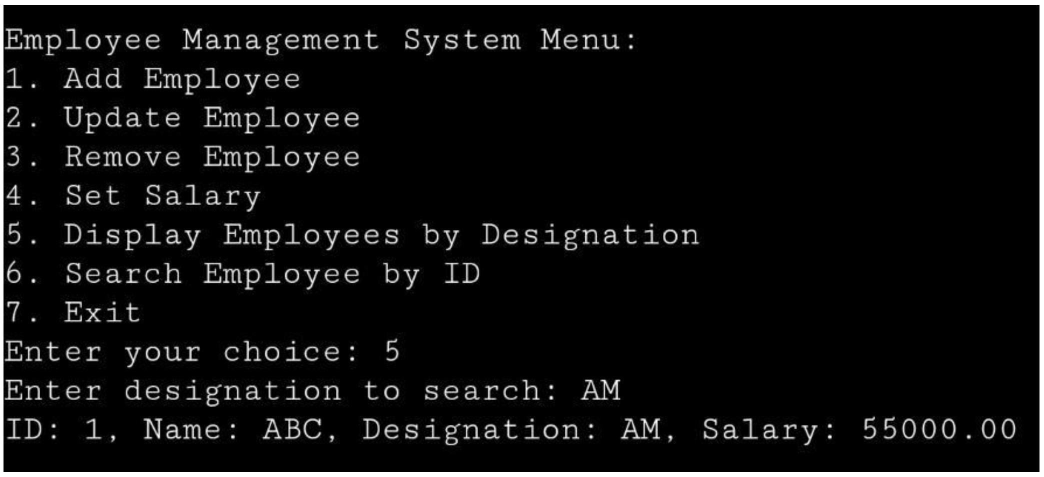
```
Employee Management System Menu:
1. Add Employee
2. Update Employee
3. Remove Employee
4. Set Salary
5. Display Employees by Designation
6. Search Employee by ID
7. Exit
Enter your choice: 4
Enter ID of the employee to set salary: 1
Enter new salary of the employee: 55000
Salary updated for employee with ID 1
```

Search employee



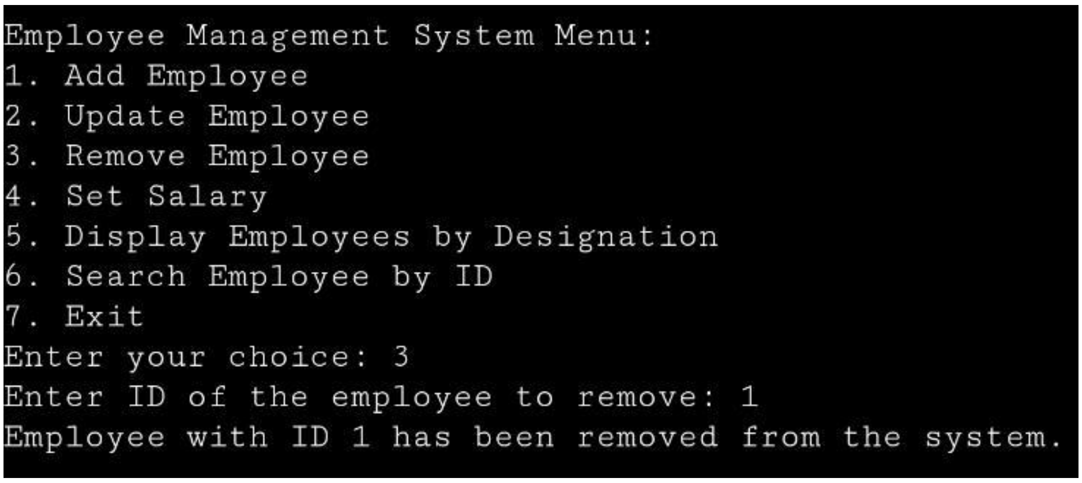
```
Employee Management System Menu:
1. Add Employee
2. Update Employee
3. Remove Employee
4. Set Salary
5. Display Employees by Designation
6. Search Employee by ID
7. Exit
Enter your choice: 6
Enter ID of the employee to search: 1
Employee found:
ID: 1, Name: ABC, Designation: AM, Salary: 55000.00
```

Search employees on same designation



```
Employee Management System Menu:
1. Add Employee
2. Update Employee
3. Remove Employee
4. Set Salary
5. Display Employees by Designation
6. Search Employee by ID
7. Exit
Enter your choice: 5
Enter designation to search: AM
ID: 1, Name: ABC, Designation: AM, Salary: 55000.00
```

Remove an employee



```
Employee Management System Menu:
1. Add Employee
2. Update Employee
3. Remove Employee
4. Set Salary
5. Display Employees by Designation
6. Search Employee by ID
7. Exit
Enter your choice: 3
Enter ID of the employee to remove: 1
Employee with ID 1 has been removed from the system.
```