# SHRI VAISHNAV VIDYAPEETH VISHWAVIDYALAYA

**Session – 2018-22**

**Computer Science Engineering (Artificial Intelligence)**
**Semester- II**

## Project Report

## EMPLOYEE MANAGEMENT SYSTEM

**Submitted to :**

**Prof. Gurpreet Singh**

**Submitted by :**

| | | |
|---|---|---|
| **Khushboo Solanki** | **–** | **18100BTCSAII02842** |
| Abhijith Nair | - | 18100BTCSAII02827 |
| Akshata Gupta | - | 18100BTCSAII02828 |
| Amarpreet Singh | - | 18100BTCSAII02831 |
| Anshul Ranade | - | 18100BTCSAII02832 |
| Deep Shrivastav | - | 18100BTCSAII02838 |
| Elma Anwar | - | 18100BTCSAII04465 |
| Harsh Porwal | - | 18100BTCSAII02840 |

## CONTENTS

## Introduction –

This Project Report document is a guideline. The document details all the high level requirements. The document should be used as a guideline by the students to design the Solution Architecture for the project. The document also describes the broad scope of the project and high level DB requirements are captured in the DB specification. But while developing the solution if the developer has a valid point to **add more details being within the scope specified** then it can be accommodated after consultation.

## Scope -

This document describes the scope of the requirements for the Employee Management System (EMS) for **ABC PRIVATE LIMITED**. The document details all the high level requirements with intent to validate **ABC PRIVATE LIMITED** requirements. This document should be used by the Authority and the developers to design the Solution Architecture for the project. In addition to this, the document also describes the broad scope of the project. The scope of the project involves the integration of a subset of all the components of current IT environment. The Employee Management System ( EMS ) should interface with the existing Employee Management and the company' authority.

The system must maintain central data for all the employees, their personal information, salary, years served by them to the company, shifts, departments, attendance, cut off in their salary according to the leaves taken. It should also validate the credentials of the employee with the one with the authority. It should acts as an interface between the authority and the employee.

## Assumptions -

The following are the assumptions for this document :-

**1.** The Authority is responsible for unique ID given to each employee.

**2.** Only Registered employees can browse / search for details and only authority have an access to credentials.

## Management Summary -

**ABC PRIVATE LIMITED** is a leading company in India. **ABC PRIVATE LIMITED** is trying to expand their business and would like to provide various services online. **One of the most in demand service is EMPLOYEE MANAGEMENT**. Employees are provided with the facility of viewing their details online. Currently, the employees have to go through the long process of formal interaction with the company authority to get the information that they require about themselves. Many employees have expressed wastage of time writing the application to the superior authority and then seeking permission. Some of the employees have also questioned the transparency of their salary details. To maintain the trust, the company has decided to add the service of delivering employee’s details to them on their web portal.

## Existing System –

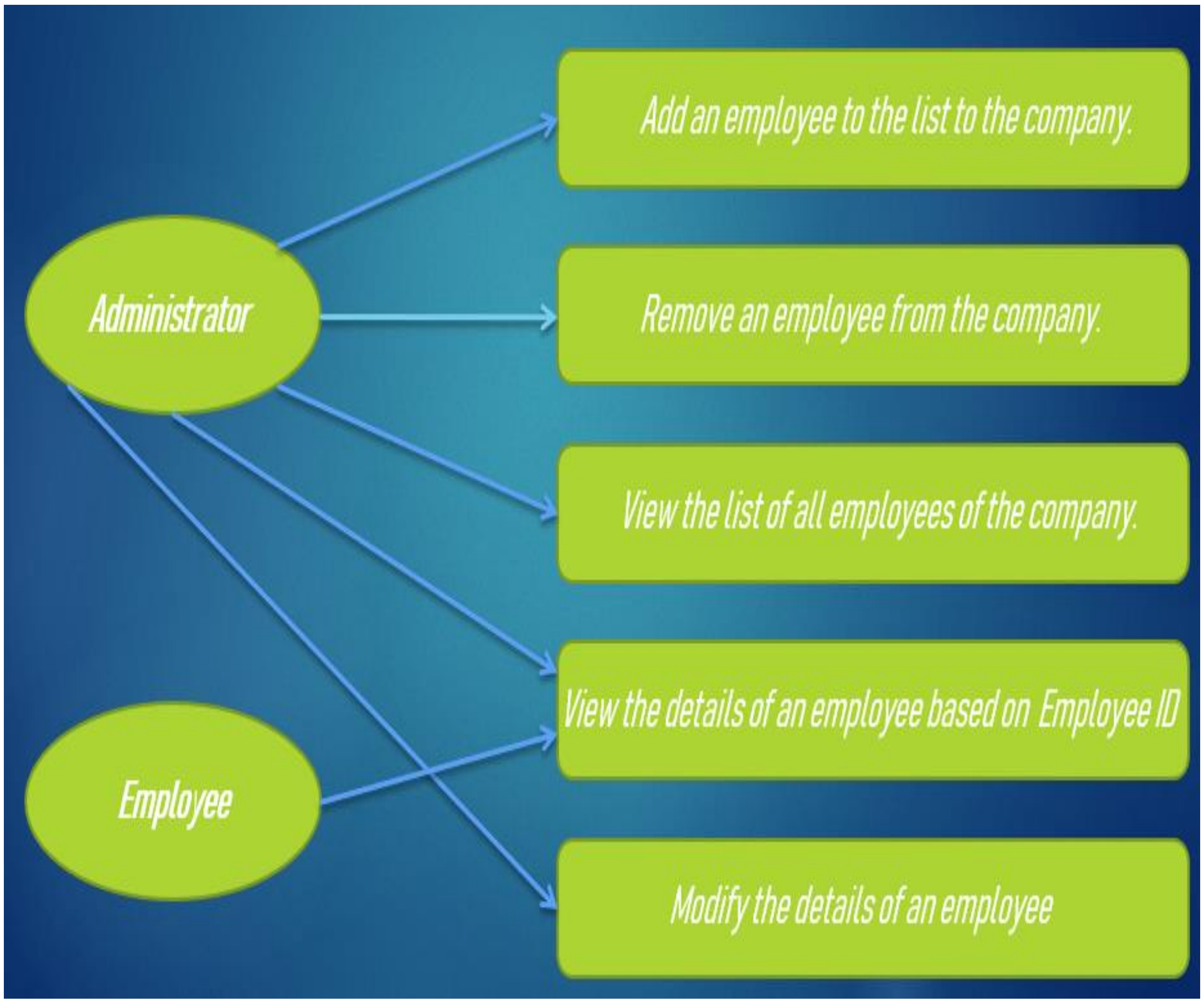
Presently, the employees of any organization need to consult their Human Resource (HR) Department to get their details or in case of any sorts of updation in their information registered with the company. This all requires extra time consumptions as well as resource consumption, as the people involve in the process can perform other official tasks at the same time if they have got a means to do all these tasks at a glance anytime.

## Proposed System –

The Employee Management System (EMS) will work as virtual bridge between the employees and their higher authorities. Each employee will be provided with a unique ‘Employee ID’ that they will use for logging into the application. They will be able to see their registered information with the company and will be able to keep their attendance under the check. On the other hand, the administrator will be able to  perform tasks like listing all employees, adding or removing an employee, update their information as per the request, and many more.

## Block Diagram –

The users of the EMS are classified into two major categories, based on the kind of operations they can perform:


Administrator
Employee
Add an employee to the list to the company.
Remove an employee from the company.
View the list of all employees of the company.
View the details of an employee based on Employee ID
Modify the details of an employee

## System Objectives / Overview –

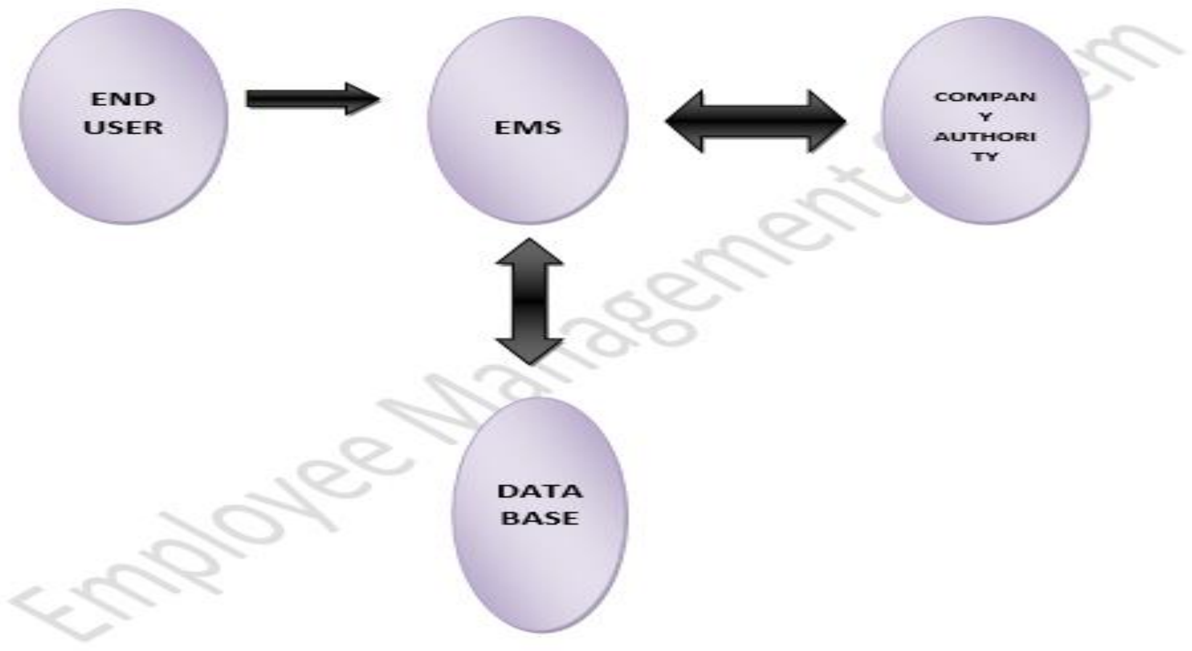


**Diagram showing the relation between the end user(employee), EMS, database, and the company-**

| FIELD | DESCRIPTION |
|---|---|
| **END USER** | Certainly, this field represents all the employees. Here, the employees gets an interface with the database through EMS. |

| | |
|---|---|
| **DATABASE** | This field represents all the information about every employee stored in the database. |
| **EMS** | This field is quite clear by its name that it suggest the management of the information of the employee. |
| **COMPANY AUTHORITY** | It represents the company authority under which the several employees work and serve their services to it. It is responsible to manage all the data. |

**<u>The above table is the description of all the elements shown in the block diagram-</u>**

Here, the end user, i.e., the employee interacts with the employee management system through the portal for seeking whatever information he/she wants. First of all, the end user enters the unique user ID password , which eventually gets validated with the one in database and if the validation succeeds the user can have a look at its profile. Similarly, the company authority also has the same credentials for all the users but the password is different than the one possessed by the user. Also, the company authority has certain privileges like modification, deletion and including the new profile in the database.

## Functional Requirements –

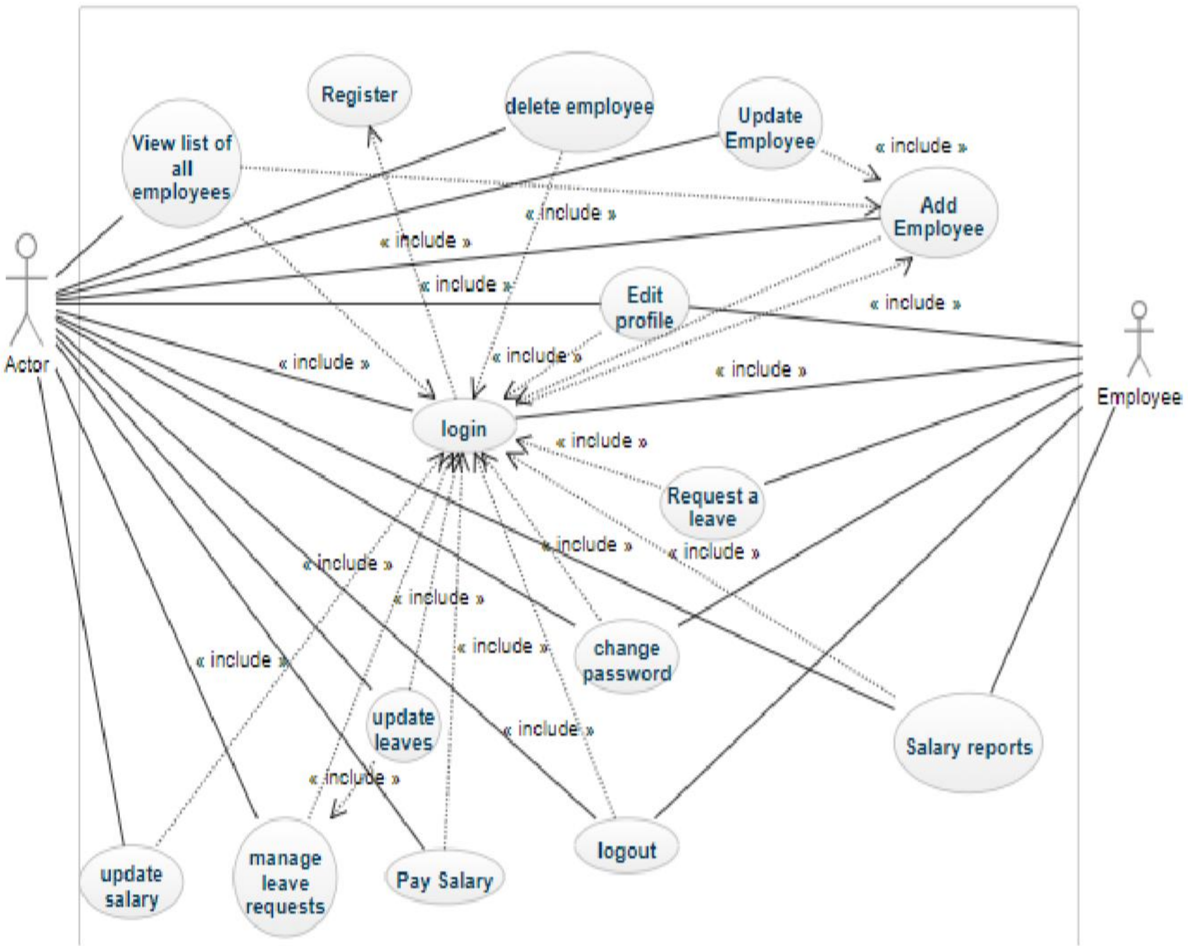
Following are the identified functional requirements of the EMS System. It provides role based access to the system (Authentication and Authorization).

1. **Authorized Company Employee –** To login and maintain profile details. To approve profile requests, discontinuation requests, get various analytical reports.

2. **Registered Employee –** To Login and perform various request for renewal of profile, application of discontinuation. Also to go through the details of the profile and his/her attendance.

## Use Case Diagram –

The below is given a use case diagram which explains the various features (or we can say that the various cases) for the proposed EMS (Employee Management System) :



## Detail Description of Use Cases –

### 1. To Login functional requirement :

#### 1.1 Input

Authorized Company Employee, Registered employee feed in unique User ID and Password.

#### 1.2 Process

The system on receiving the details validates it with an interface with the employee and Employee Management System and provides role based access to the DBMS.

#### 1.3 Output

System provides role based access to the EMS.

a. Authorized Company Employee – To maintain profile details. To approve profile requests, discontinuation requests, get various analytical reports.

b. Registered Customer – To Login and perform various request for renewal of profile, application of discontinuation. Also to go through the details of the profile and his/her attendance.

### 2. Request / Apply for profile edit : functional requirement :

#### 2.1 Input

Valid login by registered employee.

#### 2.2 Process

Here, the user selects the edit option and the request is send to the authority user for the modification in the profile. Hence the profile gets edited.

### 2.3 Output

The system saves and updates the profile request, generates auto-mailer for approval to the Authorized Company Employee.

## 3. Discontinue profile allocation functional requirement:

### 3.1 Input

Registered Employee valid login and request for locker discontinuation.

### 3.2 Process

The Registered customer submits an application / request for profile discontinuation through the EMS system online. System sends an auto-mailer to the Authorized company Employee approves the discontinuation of the facility on cross checking against balance payment for the locker, if any. System updates the profile Allocation status as "Discontinued" and saves the details of discontinuation in the database.

### 3.3 Output

The profile gets deleted.

## Database Schema -

The proposed database schema is as follows:

### 1. Profile Details:

a) User ID (uniquely assigned by the system)

b) User Department

c) User Salary

d) Working Days for User

e) User's attendance

f) Cut Off in Salary

g) Age

h) Years served in the company

i) User's email id

j) User's designation

k) Residential address

l) User's account no.

## 2. Profile requests:

a) Edit

b) Deletion

## Test Cases -

Sample Test cases for the login by Registered employee and authorized Employee are as under based on the following inputs:

**User ID** - 6 digit (mandatory )

**Password** - 6-10 characters (mandatory)

After accepting these inputs, the user will be provided with access role based access to the system (Authentication and Authorization).

**a. Administrators** – To add item(s) to sell and get various sales Analysis Reports

**b. Registered Customer** –To add item to shopping bag and checkout.

**Some of the test cases for the above Test Scenario:**

| **Test Case No.** | **Reference for Traceability** | **Test Case Description and Test Data** | **Expected Result** |
|---|---|---|---|
| **1.** | Login Screen | User ID < 6 Digits | Should not proceed ahead, ”please enter a 6 digit no”. |
| **2.** | Login Screen | User ID = 6 Digits | Should proceed to the next filed. |
| **3.** | Login Screen | User ID > 6 Digits | Should not permit entry of more than 6 digits. |
| **4.** | Login Screen | User ID is Char Data | Should permit entry only of character data. |

|  |  |  |  |
|---|---|---|---|
| **5.** | Login Screen | User ID is Alphanumeric | Should not permit entry of alphanumeric data. |
| **6.** | Login Screen | User ID is Blank | Should not permit. |
| **7.** | Login Screen | Password < 6 Char | Should not be allowed. |
| **8.** | Login Screen | Password > 10 Char | Should not be allowed. |
| **9.** | Login Screen | Password is between 6 to 10 Char | Should be allowed. |
| **10.** | Login Screen | Password is Numeric Data | Should permit entry of Numeric data. |
| **11.** | Login Screen | Password is Blank | Should not permit. |

## Merits of EMS –

Following listed are the merits of the Employee Management System:

- It will enable the employees as well as the administrators to view the details of employees anytime, and from anywhere.
- It will save the time required for the updation, addition , and modification of information of the existing as well as the former employees.
- The information and details of the employees like their attendance, salary, etc, will be in double-check by the employee himself as well as the administrators.
- Each employee will be able to view only their information, thereby, avoiding the conflict of comparisons.
- The information of employees will be in their as well as the administrator’s reach and they will be able to access it on their fingertips.

## Program/Code-

```
#include<iostream.h>

#include<fstream.h>

#include<stdlib.h>

#include<conio.h>

#include<iomanip.h>

#include<string.h>

#include<stdio.h>

#include<dos.h>
```

```
#include<graphics.h>
const char* fileName = "Employee.txt";


class Employee
{
private:
	int EmpID;
	char EmpName[50], Post[50], Department[10];
	float Salary;
public:
	void ReadData();
	int GetID();
	void DisplayRecord();
	char* GetDepartment();
};
void Employee::ReadData()
{
	cout << endl << "Employee ID:";
	cin >> EmpID;
	cout << "Employee Name:";
	cin >> EmpName;
	cout << "Employee's Post:";
	cin >> Post;
	cout << "Employee's Department:";
	cin >> Department;
```

```
        cout << "Salary:";

        cin >> Salary;

}

void Employee::DisplayRecord()

{

        cout << endl << "________________________________";

        cout << endl << setw(5) << EmpID << setw(15) << EmpName << setw(15) << Post <<
setw(15) << Department << setw(8) << Salary;

}


int Employee::GetID()

{

        return EmpID;

}


char* Employee::GetDepartment()

{

        return Department;

}


int main()

{

clrscr();

        int choice;

        void asd();
```

```
            void qwe();
            int gdriver = DETECT, gmode;
            initgraph(&gdriver,&gmode, "c:\\TURBOC3\\BGI");
            setbkcolor(CYAN);
line(15,10,622,10);
line(15,10,15,466);
line(15,466,622,466);
line(622,466,622,10);
line(25,20,611,20);
line(25,20,25,455);
line(25,455,611,455);
line(611,455,611,20);
settextstyle(5,0,3);
gotoxy(50,200);
outtextxy(60,100,"<<<<<<<<employee management system>>>>>>>>");
        gotoxy(10,15);
            cout<<"\tENTER 1: TO GO TO THE EMPLOYEE SECTION\n\n";
            gotoxy(10,17) ;
            cout<<"\tENTER 2: TO GO TO THE ADMINISTRATOR SECTION\n\n";
            gotoxy(10,19);
            cout<<"         ENTER YOUR CHOICE";
            gotoxy(35,19);
             cin>>choice;
            switch(choice)
            {
```

```
            case 1:

            qwe();

            break;

            case 2:

            asd();

            break;

            default:

            cout<<"can't understand that";

            }


  getch();

}


void qwe()

{

clrscr();

            Employee emp, e;

            char ch, Dept[50];

            int option,ID, isFound;

            system("cls");

                int login();

                        system("cls");

                setbkcolor(CYAN);

                                fstream file;

            file.open(fileName, ios::ate | ios::in | ios::out | ios::binary);
		isFound = 0;

		cout << endl << "Enter ID of an employee to be searched:";

		cin >> ID;

		login();

		setbkcolor(CYAN);

		file.seekg(0, ios::beg);

		file.read((char*)&e, sizeof(e));

		while (!file.eof())

		{

			if (e.GetID() == ID)

			{

				cout << endl << "The record found...." << endl;

				cout << endl << setw(5) << "ID" << setw(15) << "Name" <<
setw(15) << "Post" << setw(15) << "Department" << setw(8) << "Salary";

				e.DisplayRecord();

				isFound = 1;

				break;

			}

			file.read((char*)&e, sizeof(e));

		}

		file.clear();

		if (isFound == 0)

			cout << endl << "Data not found for employee ID#" << ID;
```

```
}
void asd()
{
 clrscr();
        Employee emp, e;
        char ch, Dept[50];
        int option,ID, isFound;
        int login();
        login();
        setbkcolor(CYAN);
        fstream file;
        file.open(fileName, ios::ate | ios::in | ios::out | ios::binary);
        do
        {
                cout << "\n\n\n\t\t*******Menu********\n";
                cout << endl << "\t\tEnter your option\n";
                cout << endl << "\t\t1 => Add a new record\n";
                cout << endl << "\t\t2 => Search record from employee id\n";
                cout << endl << "\t\t3 => List Employee of particular department\n";
                cout << endl << "\t\t4 => Display all employee\n";
                cout << endl << "\t\t5 => Update record of an employee\n";
                cout << endl << "\t\t6 => Delete record of particular employee\n";
                cout << endl << "\t\t7 => Exit from the program\n" << endl;
                cout << "\t\t*******************\n" << endl;
```

```
cin >> option;
switch (option)
{
case 1:
{

system("cls");
setbkcolor(CYAN);
emp.ReadData();
file.seekg(0, ios::beg);
isFound = 0;
file.read((char*)&e, sizeof(e));
while (!file.eof())
{
        if (e.GetID() == emp.GetID())
        {
                cout << "This ID already exist...Try for another ID";
                isFound = 1;
                break;
        }
        file.read((char*)&e, sizeof(e));
}
if (isFound == 1)
        break;
file.clear();
```

```
            file.seekp(0, ios::end);

            file.write((char*)&emp, sizeof(emp));

            cout << endl << "New record has been added successfully...";

            break;

            }

            case 2:

            {


            system("cls");

            setbkcolor(CYAN);

                isFound = 0;

            cout << endl << "Enter ID of an employee to be searched:";

            cin >> ID;

            file.seekg(0, ios::beg);

            file.read((char*)&e, sizeof(e));

            while (!file.eof())

            {

                if (e.GetID() == ID)

                {

                    cout << endl << "The record found...." << endl;

                    cout << endl << setw(5) << "ID" << setw(15) << "Name" <<
setw(15) << "Post" << setw(15) << "Department" << setw(8) << "Salary";

                    e.DisplayRecord();

                    isFound = 1;

                    break;
                }
                file.read((char*)&e, sizeof(e));
            }
            file.clear();
            if (isFound == 0)
                cout << endl << "Data not found for employee ID#" << ID;
            break; }
            case 3:
            system("cls");
            setbkcolor(CYAN);
            {       isFound = 0;
            cout << "Enter department name to list employee within it:";
            cin >> Dept;
            file.seekg(0, ios::beg);
            file.read((char*)&e, sizeof(e));
            while (!file.eof())
            {
                if (strcmp(e.GetDepartment(), Dept) == 0)
                {
                    cout << endl << "The record found for this department" << endl;


                    cout << endl << setw(5) << "ID" << setw(15) << "Name" <<
setw(15) << "Post" << setw(15) << "Department" << setw(8) << "Salary";
                    e.DisplayRecord();
                    isFound = 1;
                    break;
                }
                file.read((char*)&e, sizeof(e));
            }
            file.clear();
            if (isFound == 0)
                cout << endl << "Data not found for department" << Dept;
            break;
            }
            case 4:
            system("cls");
            setbkcolor(CYAN);
            {   cout << endl << "Record for employee";
            file.clear();
            file.seekg(0, ios::beg);
            int counter = 0;
            file.read((char*)&e, sizeof(e));
            while (!file.eof())
            {
                counter++;
                if (counter == 1)
                {
                    cout << endl << setw(5) << "ID" << setw(15) << "Name" <<
setw(15) << "Post" << setw(15) << "Department" << setw(8) << "Salary";
                }
                e.DisplayRecord();
                file.read((char*)&e, sizeof(e));
            }
            cout << endl << counter << "records found......";
            file.clear();
            break;
            }
            case 5:
            {

            system("cls");
            setbkcolor(CYAN);
                int recordNo = 0;
            cout << endl << "File is being modified....";
            cout << endl << "Enter employee ID to be updated:";
            cin >> ID;
            isFound = 0;
            file.seekg(0, ios::beg);
            file.read((char*)&e, sizeof(e));
            while (!file.eof())
            {
                recordNo++;
                if (e.GetID() == ID)
                {
                    cout << "The old record of employee having ID" << ID << "is:";
                    e.DisplayRecord();
                    isFound = 1;
                    break;
                }
                file.read((char*)&e, sizeof(e));
            }

            if (isFound == 0)
            {
                cout << endl << "Data not found for employee ID#" << ID;
                break;
            }
            file.clear();
            int location = (recordNo - 1) * sizeof(e);
            file.seekp(location, ios::beg);
            cout << endl << "Enter new record for employee having ID" << ID;
            e.ReadData();
            file.write((char*)&e, sizeof(e));
            break;
            }
            case 6:
            {
            system("cls");
            setbkcolor(CYAN);
                int recordNo = 0;//Fix Needed?
```

```
cout << endl << "Enter employment ID to be deleted:";
cin >> ID;
isFound = 0;
file.seekg(0, ios::beg);
file.read((char*)&e, sizeof(e));
while (!file.eof())
{
        recordNo++;//Fix Needed?
        if (e.GetID() == ID)
        {
                cout << " The old record of employee having ID " << ID << " is: ";
                e.DisplayRecord();
                isFound = 1;
                break;
        }
        file.read((char*)&e, sizeof(e));
}
char tempFile[] = "temp.txt";
fstream temp(tempFile, ios::out | ios::binary);
if (isFound == 0)
{
        cout << endl << "Data not found for employee ID#" << ID;
        break;
}
else
```

```
{
        file.clear();
        file.seekg(0, ios::beg);
        file.read((char*)&e, sizeof(e));
        while (!file.eof())
        {
                if (e.GetID() != ID)
                        temp.write((char*)&e, sizeof(e));
                file.read((char*)&e, sizeof(e));
        }
        file.close();
        temp.close();
        temp.open(tempFile, ios::in | ios::binary);
        file.open(fileName, ios::out | ios::binary);
        temp.read((char*)&e, sizeof(e));
        while (!temp.eof())
        {
                file.write((char*)&e, sizeof(e));
                temp.read((char*)&e, sizeof(e));
        }
}
temp.close();
file.close();
remove(tempFile);
file.open(fileName, ios::ate | ios::in | ios::out | ios::binary);
```

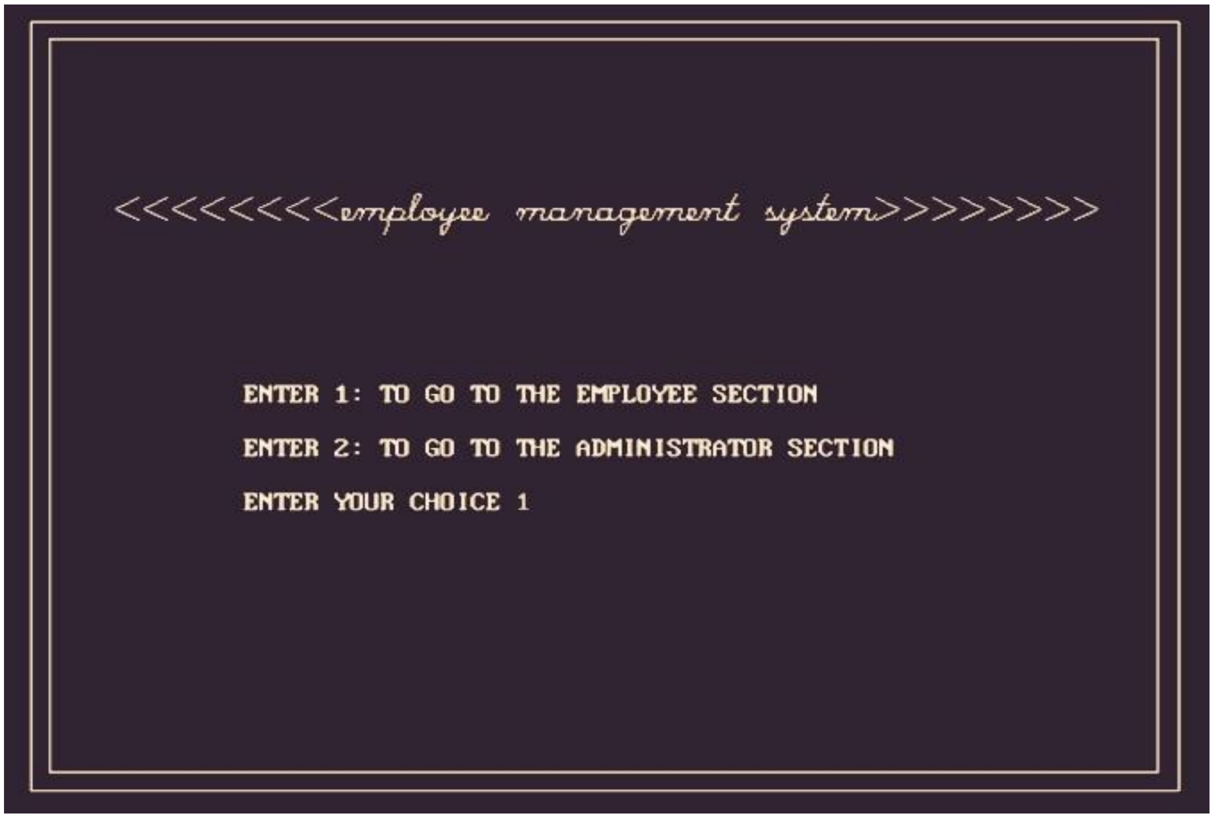
```
                break; }

                case 7:
                {
                cout<<"thank u for visitng!!!!!\n" ;
                cout<<"press any key to exit......";
                exit(0);
                break;
                }
                default:
                        cout << "Invalid Options";
                }
                cout << "\nDo you want to continue.....?y/n";
                cin >> ch;
        clrscr();
        } while (ch != 'n');
}
int login()
{
  void pass();
  char ch;
  system("cls") ;
  setbkcolor(CYAN);
 gotoxy(70,50);
  cout <<"\n\n\n\n\t\tEMPLOYEE MANAGEMENT SYSTEM";
```

```
   cout <<"\n\n\t\tEnter Your Password :";
   pass();
return 1;
}
void pass()
  {
    int i,x;
    char ch='/0',password[]="password",match[20];
    for(i=0;i>=0;)
     {
       ch=getch();

      if(ch!=8&&ch!=13)
            {
             cout<<"*";
             match[i]=ch;
             i++;
            }
      else if (ch==8)
       {
             cout<<"\b \b";
             i--;
       }
      else if(ch==13)
    {
```

```
             match[i]='\0';
            break;
    }
    else
    {
             break;
    }
  }
  if(strcmp(match,password)==0)
  {
cout<<"\n\n\n\t\tLOADING \n\t\t";
            for(int a=1;a<8;a++)
            {
                        delay(500);
                        cout << "...";
            }
      cout << "\n\n\n\t\tAccess Granted!! \n\n\n";
      system("PAUSE");
      system("CLS");


 }else{
      cout << "\n\t\tAccess Aborted...\n";
      login();
 }
}
```

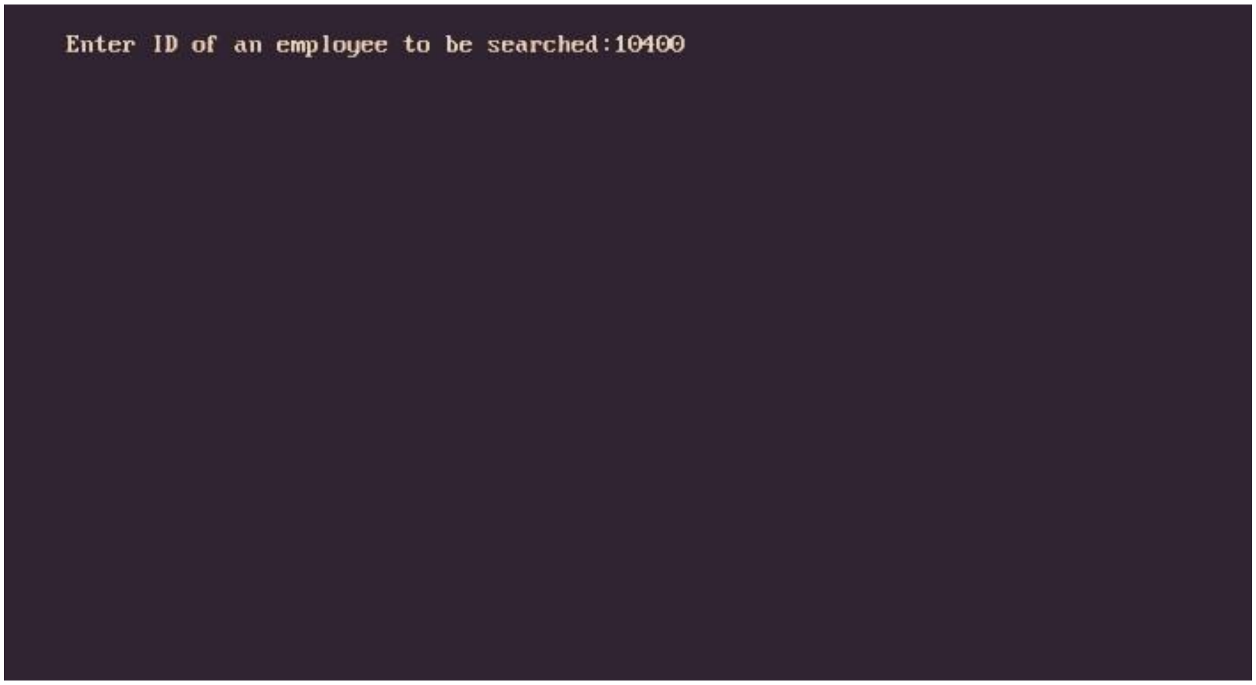
## Output of the above code for Employee Management System-

### Home Page:



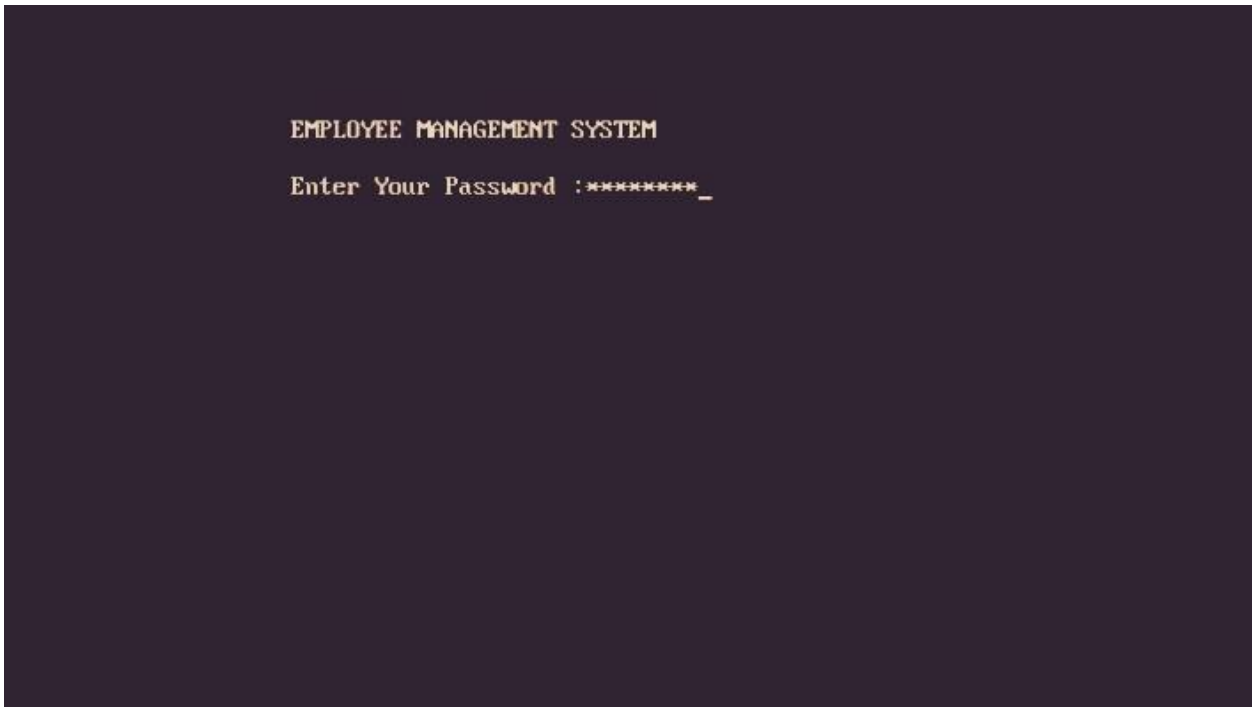
```
<<<<<<<<<employee management system>>>>>>>>>

          ENTER 1: TO GO TO THE EMPLOYEE SECTION
          ENTER 2: TO GO TO THE ADMINISTRATOR SECTION
          ENTER YOUR CHOICE 1
```

### Employee Section:



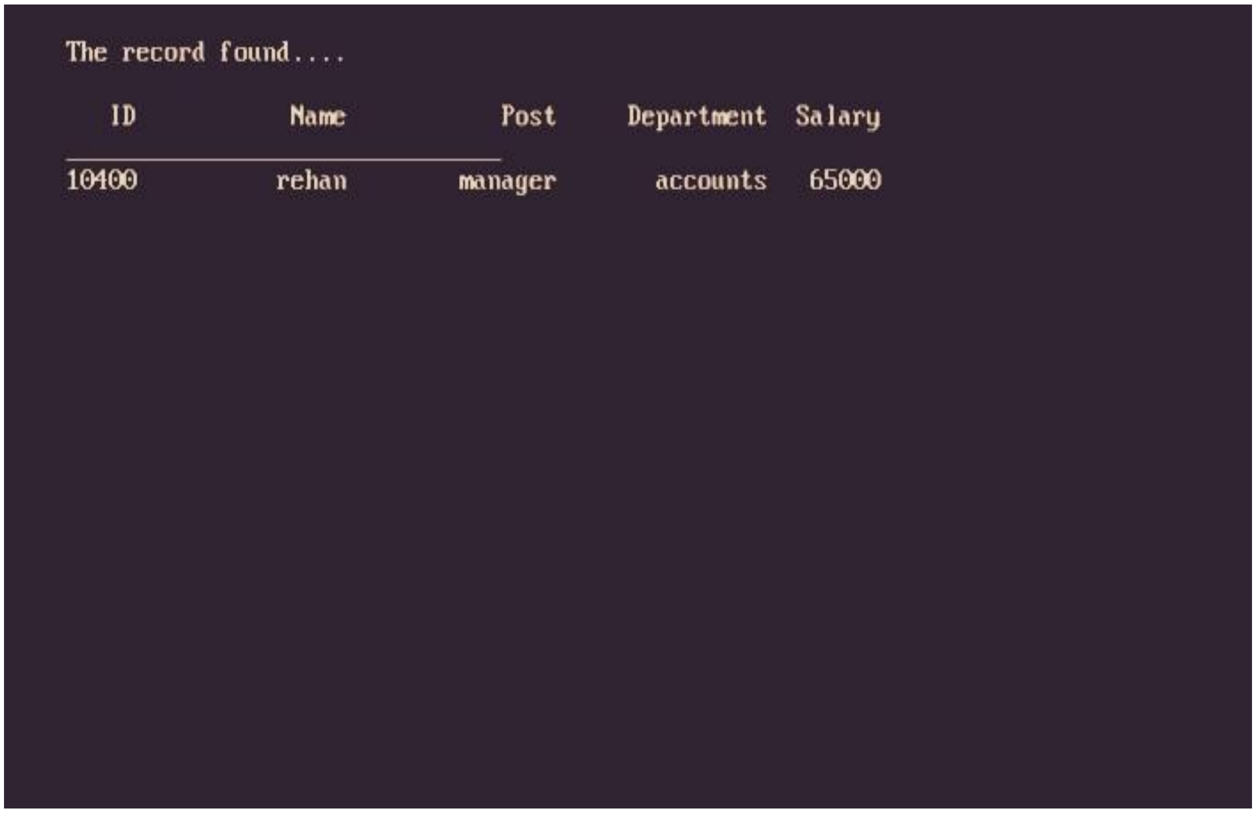
```
  Enter ID of an employee to be searched:10400
```

**Password Entry for the Employee:**



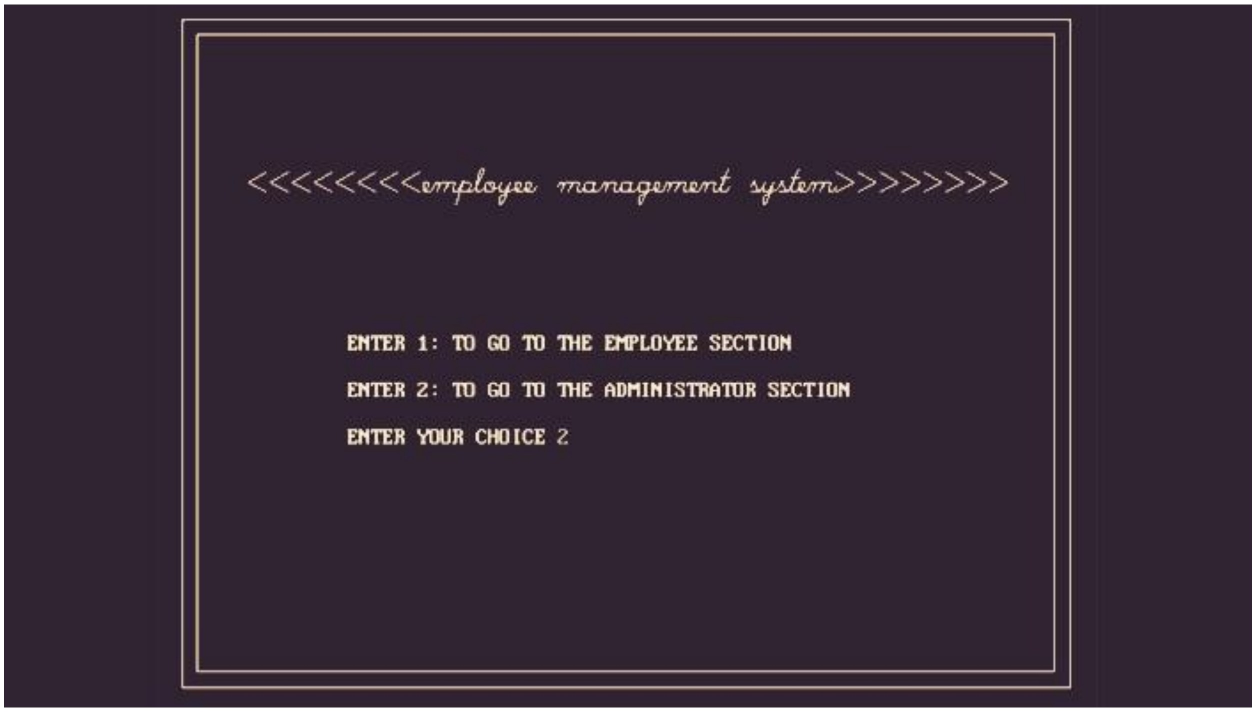
```
EMPLOYEE MANAGEMENT SYSTEM

Enter Your Password :*********_
```

**Record of the searched Employee by Employee:**



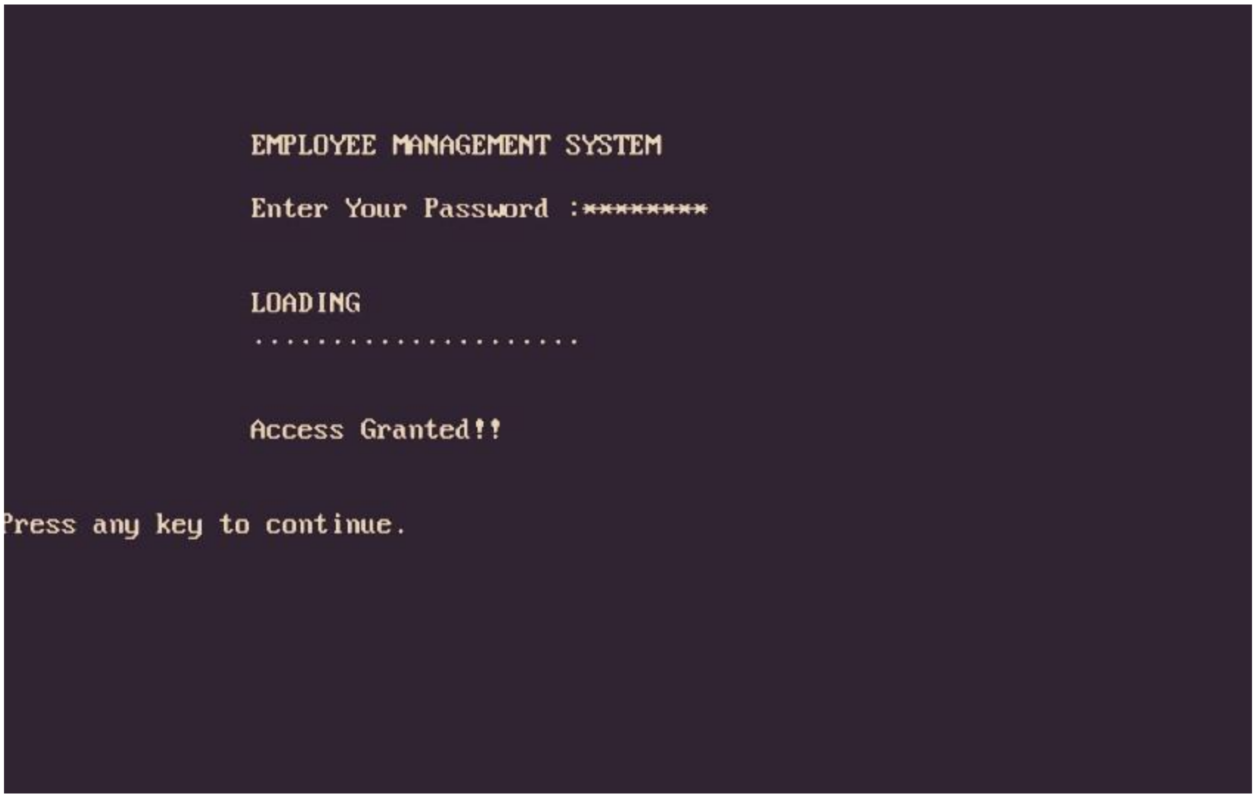
```
The record found....

   ID          Name          Post     Department  Salary
______________________________________
10400         rehan        manager       accounts   65000
```

## Home Page to choose another option:



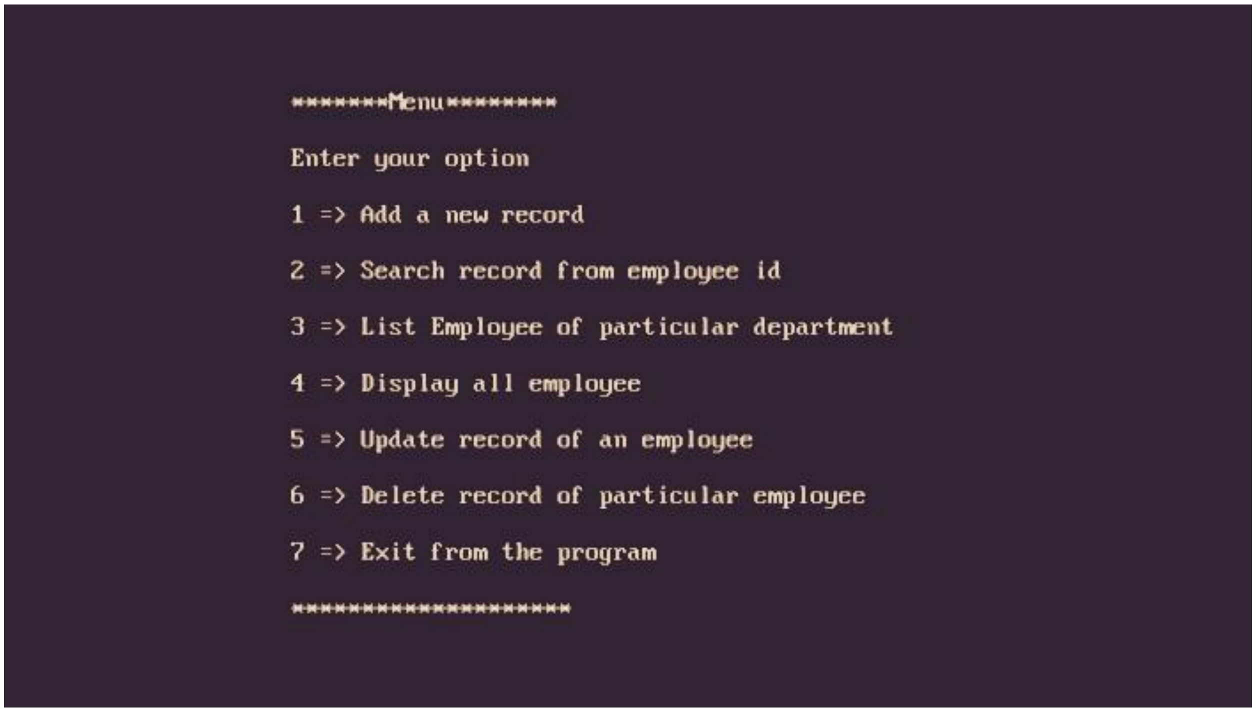
```
<<<<<<<<<employee management system>>>>>>>>

ENTER 1: TO GO TO THE EMPLOYEE SECTION
ENTER 2: TO GO TO THE ADMINISTRATOR SECTION
ENTER YOUR CHOICE 2
```

## Administrator Section:



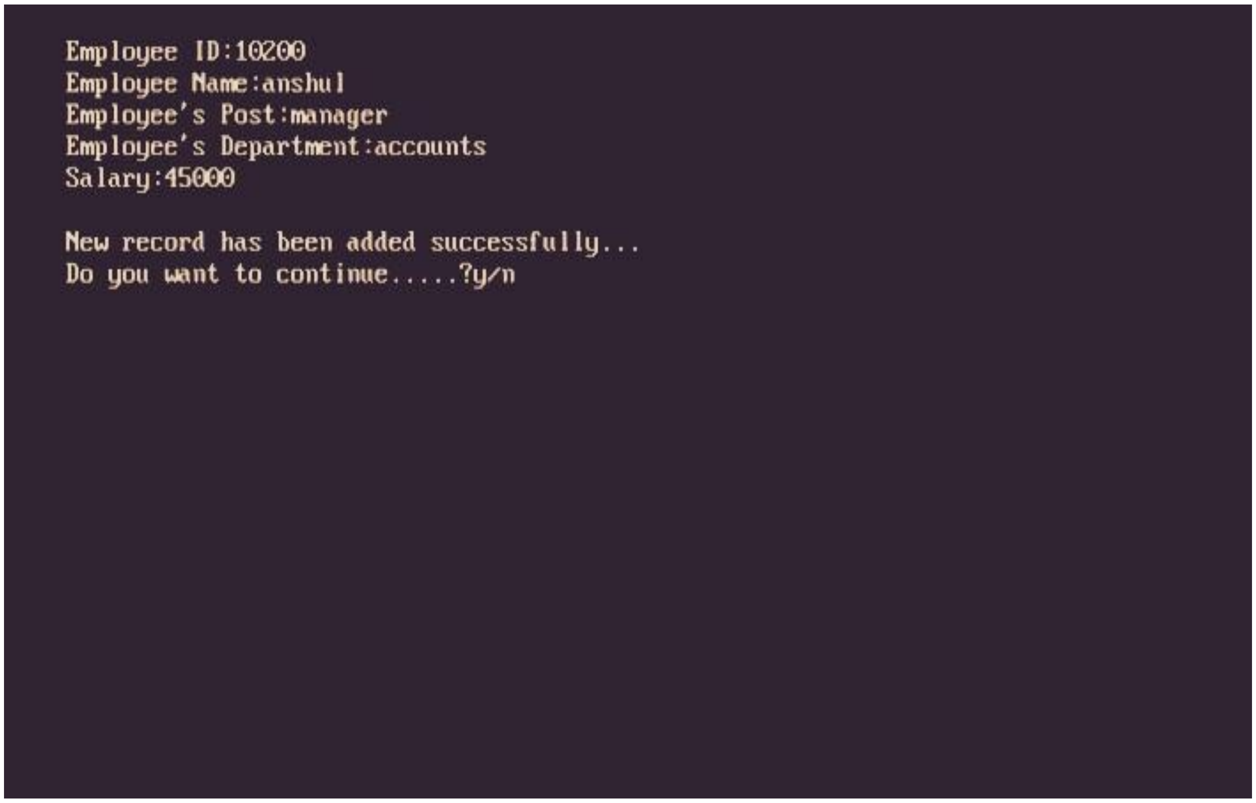
```
          EMPLOYEE MANAGEMENT SYSTEM

          Enter Your Password :*********


          LOADING
          .....................


          Access Granted!!

Press any key to continue.
```

**Options for the Administrator to operate:**



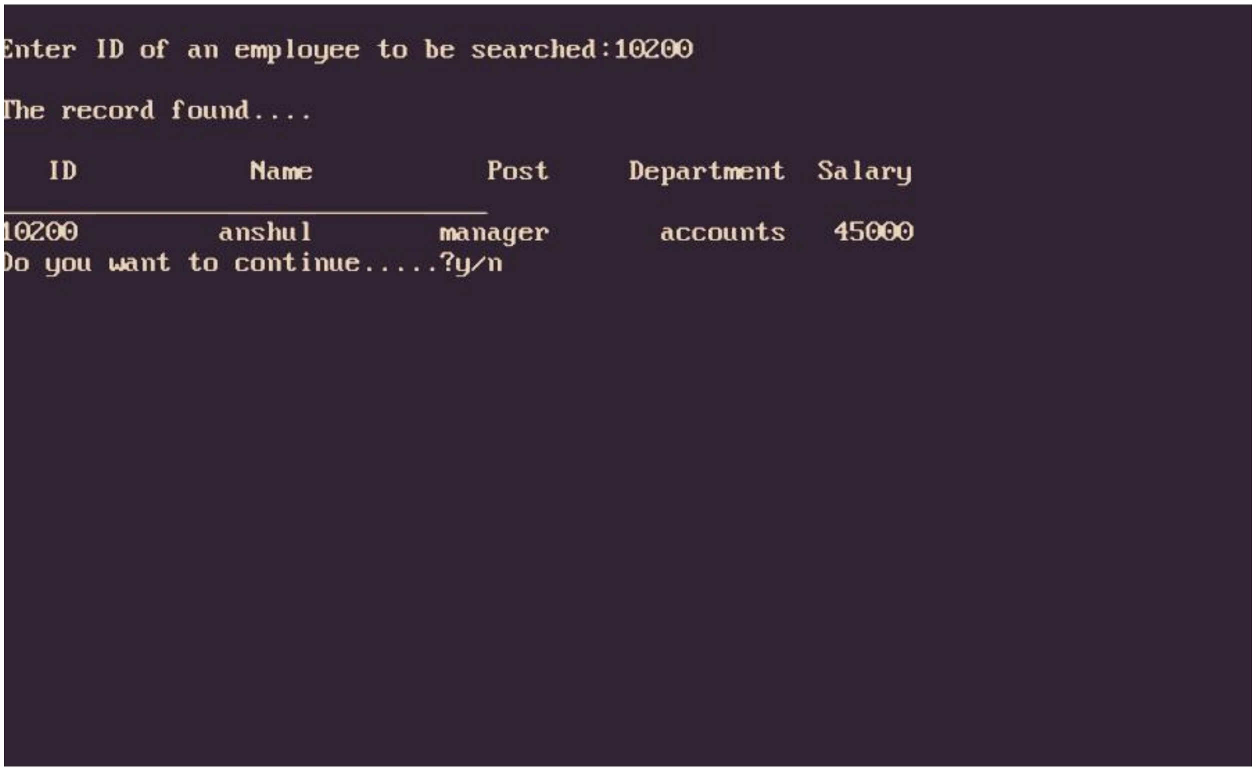
```
*******Menu********

Enter your option

1 => Add a new record

2 => Search record from employee id

3 => List Employee of particular department

4 => Display all employee

5 => Update record of an employee

6 => Delete record of particular employee

7 => Exit from the program

********************
```

**Addition of new Employee:**



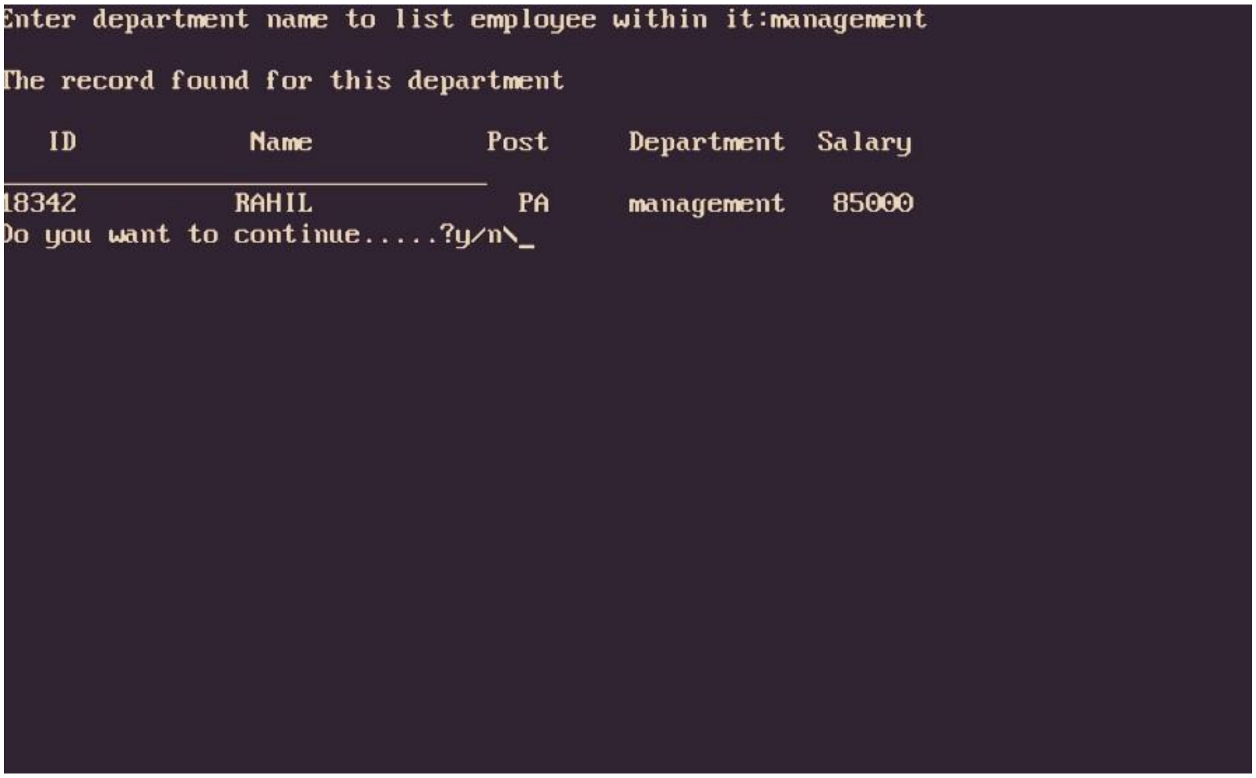
```
Employee ID:10200
Employee Name:anshul
Employee's Post:manager
Employee's Department:accounts
Salary:45000

New record has been added successfully...
Do you want to continue.....?y/n
```

## Searching for Employee record by Administrator:



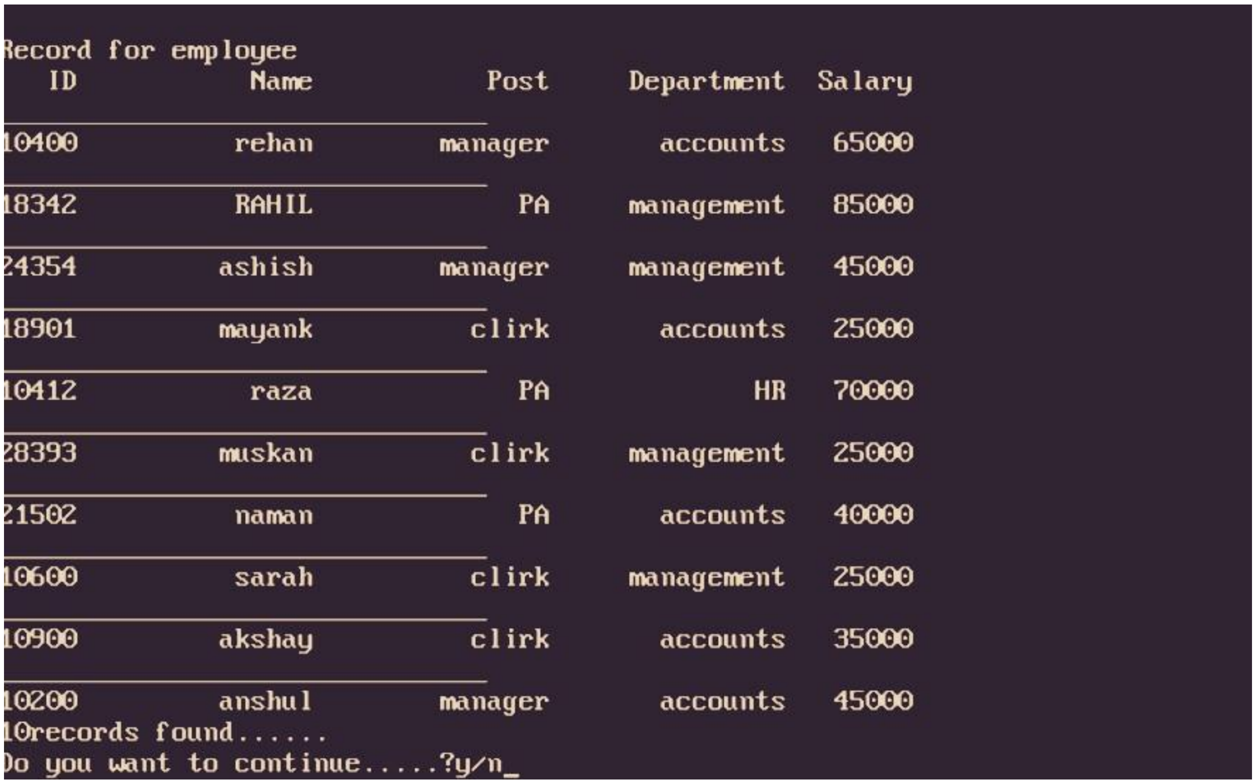
```
Enter ID of an employee to be searched:10200

The record found....

   ID           Name           Post     Department  Salary
_______________________________
10200         anshul        manager       accounts   45000
Do you want to continue.....?y/n
```

## Searching based on specified type by Administrator:



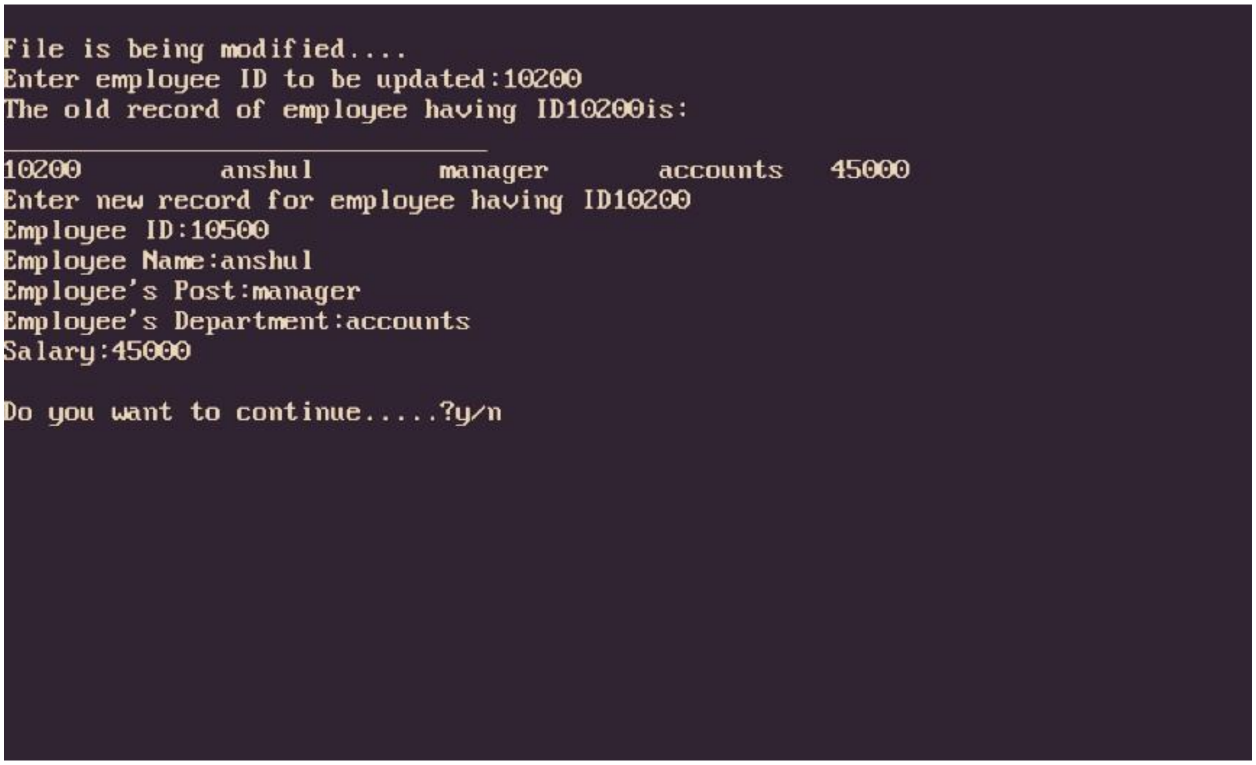
```
Enter department name to list employee within it:management

The record found for this department

   ID           Name           Post     Department  Salary
_______________________________
18342          RAHIL             PA     management   85000
Do you want to continue.....?y/n\_
```

**Whole record of employees:**



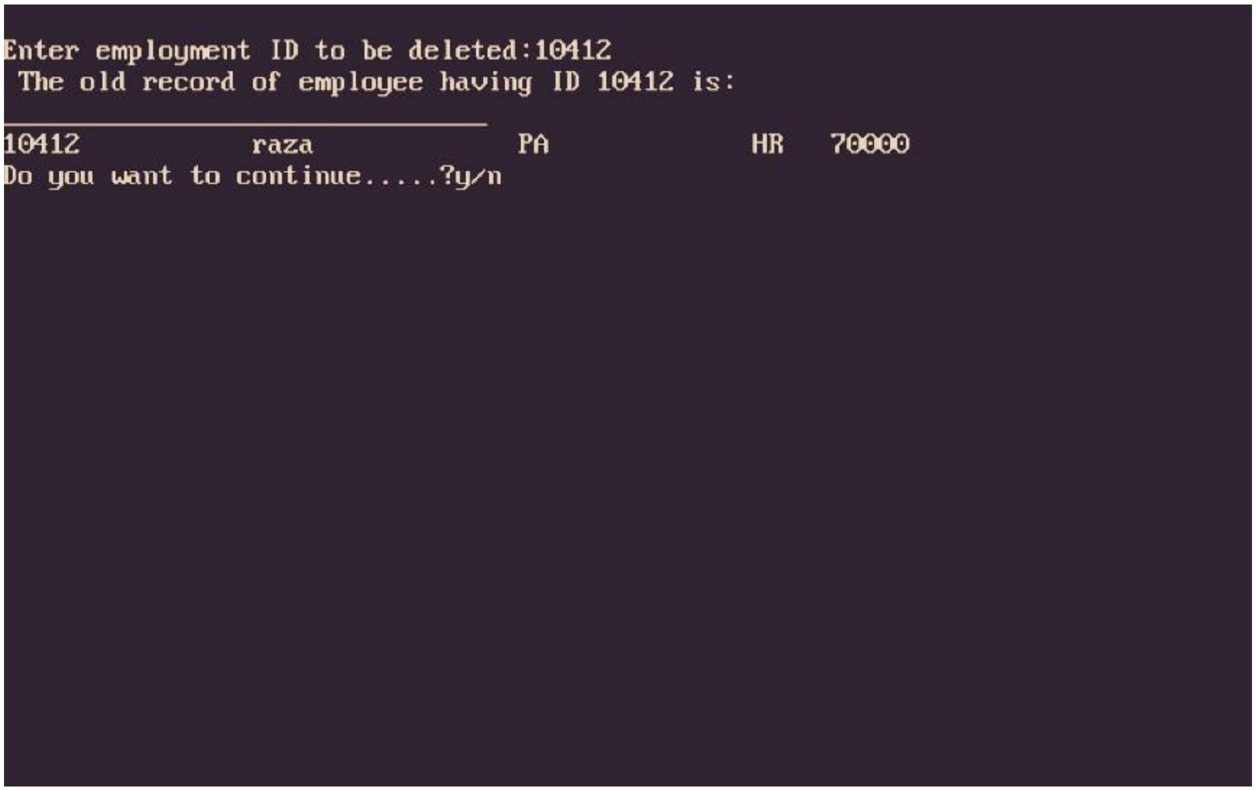
```
Record for employee
   ID           Name           Post     Department  Salary
_______________________________
10400          rehan        manager       accounts   65000
_______________________________
18342          RAHIL             PA     management   85000
_______________________________
24354         ashish        manager     management   45000
_______________________________
18901         mayank          clirk       accounts   25000
_______________________________
10412           raza             PA             HR   70000
_______________________________
28393         muskan          clirk     management   25000
_______________________________
21502          naman             PA       accounts   40000
_______________________________
10600          sarah          clirk     management   25000
_______________________________
10900         akshay          clirk       accounts   35000
_______________________________
10200         anshul        manager       accounts   45000
10records found......
Do you want to continue.....?y/n_
```

**Updating record of an Employee-**



```
File is being modified....
Enter employee ID to be updated:10200
The old record of employee having ID10200is:
_______________________________
10200         anshul        manager       accounts   45000
Enter new record for employee having ID10200
Employee ID:10500
Employee Name:anshul
Employee's Post:manager
Employee's Department:accounts
Salary:45000

Do you want to continue.....?y/n
```

## Deletion of the record of an Employee:



```
Enter employment ID to be deleted:10412
 The old record of employee having ID 10412 is:
______________________________
10412          raza           PA            HR   70000
Do you want to continue.....?y/n
```

## Conclusion:

This Project Report gives the details of all the functions of the proposed Employee Management System. The employee can see details; their salary and all other information. It has reduced the time to a large level, means it is very time efficient. Now, the employees do not need to go through all the formal interaction processes like writing letters to the superior authority for seeking permissions to get access to their credentials. This application also provides each employee with an unique Id and password ,so that their details can be kept confidential and safe. All the data is managed perfectly by this application. There is a lot more scope of improvement in this application ,we can add a lot more features to it and can make it better for the management.