INTRODUCTION

This project depicts simple **Employee Management System** software, which contains various functions that allow the user/company to keep track of its employees and store the details of its employees in a database.

This project has been made using C language, and uses some important concepts, features and algorithms to accomplish the task. This software is entirely text based and could be run on any machine having C compiler (GCC, Turbo-C, Borland, etc.) through a Terminal window (LINUX), or Command Prompt (Windows).

HEADER FILES USED:

1. `stdio.h`
2. `stdlib.h`
3. `string.h`
4. `ctype.h`
5. `termios.h`

FUNCTIONS USED:

void head_disp( ) – This function displays the masthead of the Employee Management Software program and is used as a header for displaying the Software name to all subparts of the program. As indicated by its return type, it returns no value, and accepts no parameter.

void initTermios(int) – This function is used to set the terminal properties to its default stage, and also it is used to implement getch() and getche() functions which are by default not present in the standard C library, but are required for usage in some parts of the program.

void resetTermios( ) – This function resets the default state of the terminal which might have been changed during implementation of getch() and getche() functions. The default values to be used are stored in a structure variable named **old**.

char getch_(int) - This function takes an integer argument which is passed to it by, the getch() and getche() functions, and according to the input passed to it, it instructs the initTermios(int) function to implement those functions, and whether to echo the output on the screen or not.

char getch( ) – This function takes a character from the standard input i.e. keyboard and returns that character to a char type variable, but the value which is typed by the user on the screen is not displayed. Also, getch() function does not wait for the user to press Enter key after entering a character. The entered character is automatically used.

char getche( ) – This function takes a character from the standard input i.e. keyboard and returns that character to a char type variable, and the value which is typed by the user is displayed on the screen. Like getch() function, getche() function also does not wait for the user to press Enter key after entering a character. The entered character is automatically used.

NOTE- The employee database where information about all the employees will be stored is a binary file, with name "**employeeRecord.txt**". The program has been designed in such a manner that it uses this file to work with all its functions. The software also analyzes and checks if the file is present or not, and prompts the user for creating Employee database, if it does not exists already.

void emp_append( ) – This function is used to add an Employee's record in the existing database.

void emp_delete( ) – This function is used for deleting the records of an employee present in the database of the software.

void emp_modify( ) – This function is used to modify the records of the employees which are present in the software's database.

void emp_display( ) – This function takes Employee ID from the user, and displays all information about that employee, from the database.

void emp_displayAll( ) – This function displays information about all the employees present in the current software's database.

void emp_search( ) – This function allows a user to search through the employee records present in the database. User can search for employee either by Name or by Employee ID. If found, the function displays the information about that employee to the user.

int main(void) – This function is the driving function of the program. The execution of the software begins from this function, and it contains all the information about the database, and it manages the control of execution of the program, by displaying user to choose what operation which he wants to perform on the given database.

EMPLOYEE INFORMATION:

An employee has usually these details on which this software works on-

Name of Employee, Position of Employee in Company, Qualifications of Employee, Certifications(if any) of Employees, Employee ID, Programming Languages known.

Personal Information of Employee including his/her Age, Phone number.

These values are stores in a structure **personal**.

ALGORITHM

int main(void)-

1. START

2. Create File Pointer FILE *data

3. Call function- head_disp()

4. OPEN File employeeRecord.txt in Read/Edit mode(r+).

5. IF file Open Operation is Unsuccessful THEN

6. PRINT “Employee Database Not Found. Press N key to create new database. To EXIT press any other key.”

7. READ create

8. IF create = ‘n’ THEN create employeeRecord.txt

9. ELSE Exit the Application

10. WHILE(1)

11. OPEN head_disp() function

12. PRINT All Choices Related to Employee management for user to choose.

13. READ choice

14. SWITCH(choice) and go to respective functions as per user’s choice

15.END

void emp_append()-

1. OPEN File employeeRecord.txt in Append mode

2.OPEN head_disp() function to display the title of software

3.READ all Employee Details present in **personal** structure

4. APPEND All Employee Record data entered in the file, using **fwrite** function.

5.IF Successful, Display Appropriate Messages.

6.CLOSE File employeeRecord.txt

7.PRINT Press 1 to Continue and 0 to Exit

8. READ exit_status

9. IF exit_status is not equal to 1 THEN

10. EXIT

void emp_modify( )-

1. OPEN head_disp() function

2. OPEN File employeeRecord.txt in r+ mode

3. PRINT “Enter Employee ID to edit Record”

4. READ id

5. WHILE End of File is not Reached

6. IF personal.emp_id equals id THEN

7. PRINT Existing Record Details

8. READ New Employee Information and Write it to the file employeeRecord.txt

9. IF Record Not Found THEN

10. PRINT Record Not Found

11. CLOSE File employeeRecord.txt

12. Ask User to Continue or not and perform necessary actions

13. END

void emp_search( )-

1. OPEN head_disp()

2. OPEN File employeeRecord.txt in r mode

3.PRINT User to enter either Employee ID or Employee name to search

4. READ search_choice

5. READ Employee Search ID or Name as per previous choice using SWITCH construct

6. WHILE End of File is not reached

7. IF Employee Id or Name matches one in file THEN

8. THEN PRINT All Employee Record

9. IF found=0 i.e. Employee Not Found THEN

10. PRINT “No Such Employee Record Found”

11. Ask user to Continue, and take appropriate actions as required.

12. END

void emp_delete( )-

1. OPEN head_disp()

2. Open File employeeRecord.txt, and also create new file temp_data.txt

3. PRINT “Enter Employee ID to delete it’s Record”

4. READ del_emp_id

5. WHILE till End of File has been reached

6. IF del_emp_id is NOT equal to personal.emp_id THEN

7. Write all records to temp_data.txt file

8. CLOSE employeeRecord.txt and temp_data.txt

9. REMOVE employeeRecord.txt from the current directory

10. RENAME temp_data.txt to employeeRecord.txt

11. PRINT “Employee Successfully Deleted”

12. Ask User to Continue or not and take appropriate actions required.

13. END

void emp_displayAll( )-

1. OPEN head_disp() function

2. OPEN File employeeRecord.txt in Read (r) mode.

3. WHILE End of File is Not Reached

4. PRINT All Employees Individual Information

5. PRINT Total Employees in File

6. Ask User to Continue or not, and take appropriate actions as required.

7. END

void emp_display( )-

1. OPEN function head_disp()

2. OPEN File employeeRecord.txt

3. PRINT “Enter Employee ID to display it’s Record”

4. READ search_emp_id

5. WHILE End of File has not been Reached

6. IF personal.emp_id equals search_emp_id

7. found =1

8. PRINT All Details of that Employee

9. IF found equals 0 THEN

10. PRINT “No Such Employee Record Found”

11. Ask User to Continue or not, and take appropriate actions required.

12. END

CODE

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <termios.h>

static struct termios old, new;

static struct employee_data
{
      char name[100],position[100],qualification[100];
      int age,emp_id,work_exp;
      float salary;
      long int phone;
      char certifications[200];
      char prog_lang[100];
}personal;

void head_disp();
void emp_append();
void emp_delete();
void emp_modify();
void emp_displayAll();
void emp_display();
void emp_search();
void initTermios(int);
void resetTermios();
char getch_(int);
char getch();
char getche();

#define size 1

/* Initialize new terminal i/o settings */
void initTermios(int echo)
{
  tcgetattr(0, &old); /* grab old terminal i/o settings */
  new = old; /* make new settings same as old settings */
  new.c_lflag &= ~ICANON; /* disable buffered i/o */
  new.c_lflag &= echo ? ECHO : ~ECHO; /* set echo mode */
  tcsetattr(0, TCSANOW, &new); /* use these new terminal i/o
settings now */
}

/* Restore old terminal i/o settings */
void resetTermios(void)
{
  tcsetattr(0, TCSANOW, &old);
}

/* Read 1 character - echo defines echo mode */
```

```
char getch_(int echo)
{
  char ch;
  initTermios(echo);
  ch = getchar();
  resetTermios();
  return ch;
}

/* Read 1 character without echo */
char getch(void)
{
  return getch_(0);
}

/* Read 1 character with echo */
char getche(void)
{
  return getch_(1);
}

int main(void)
{
      FILE *data;
      int menu_choice;
      char create;
      head_disp();
      data = fopen("employeeRecord.txt","r+");
      if(data==NULL)
      {
            printf("\n\t Employee Database Not Found!\n Press N key
to create new Database.");
            printf("\n\tTo EXIT, press Enter key\n");
            create = getche();
            if(tolower(create)=='n')
            {
                  data = fopen("employeeRecord.txt","w+");
                  printf("\n\tEmployee Record Database has been
created in the current directory.\n");
                  printf("\n\tYou must Restart the Application\nPress
Enter key to continue.\n");
                  char x;
                  x = getch();
                  exit(0);
            }
            else
            {
                  printf("\n\n\tThank You for using this
Application.\n");
                  printf("\n\n\tApplication will EXIT Now.");
                  exit(0);
            }
      }
      while(1)
      {
            head_disp();;
```

```
            printf("\n \t 1.Add New Employee");
            printf("\n \t 2.Modify an Employee's Record");
            printf("\n \t 3.Search for an Employee");
            printf("\n \t 4.Delete an Employee Record");
            printf("\n \t 5.Display All Employee's Record");
            printf("\n \t 6.Display an Employee Record (Employee
ID/Name Required)");
            printf("\n \t Enter any number to EXIT");
            printf("\n\n \tINPUT:\t");
            scanf("%d",&menu_choice);
            switch(menu_choice)
            {
                  case 1:
                              emp_append();
                              break;
                  case 2:
                              emp_modify();
                              break;
                  case 3:
                              emp_search();
                              break;
                  case 4:
                              emp_delete();
                              break;
                  case 5:
                              emp_displayAll();
                              break;
                  case 6:
                              emp_display();
                              break;
                  default:
                              head_disp();
                              printf("\n\n\tThank You for using this
Application.\n");
                              exit(0);
            }
      }
      return 0;
}

void head_disp()
{
      system("clear");
      printf("\n\n\t\t\t*********************************************
****");
      printf("\n\t  \t\t**
**");
      printf("\n\t  \t\t**      EMPLOYEE MANAGEMENT SOFTWARE
**");
      printf("\n\t  \t\t**
**");
      printf("\n\t\t\t*********************************************
**\n");
}

void emp_append()
```

```
{
	FILE *data;
	data = fopen("employeeRecord.txt","a");
	head_disp();;
	printf("\n \tEmployee ID:\t");
	scanf("%d",&personal.emp_id);
	printf("\n \tEmployee Name:\t");
	scanf("%s",personal.name);
	printf("\n \tEmployee Position:\t");
	scanf("%s",personal.position);
	printf("\n \tEmployee Age:\t");
	scanf("%d",&personal.age);
	printf("\n \tEmployee Work Experience:\t");
	scanf("%d",&personal.work_exp);
	printf("\n \tEmployee Salary:\t");
	scanf("%f",&personal.salary);
	printf("\n \tEmployee Phone Number:\t");
	scanf("%ld",&personal.phone);
	printf("\n \tEmployee Qualifications:\t");
	scanf("%s",personal.qualification);
	printf("\n \tEmployee Certifications:\t");
	scanf("%s",personal.certifications);
	printf("\n \tProgramming languages Known:\t");
	scanf("%s",personal.prog_lang);
	if(fwrite(&personal,sizeof(personal),size,data)!=size)
	{
		printf("\n\tError in Writing to File.");
		exit(0);
	}
	else
	{
		printf("\n\tEmployee Record Successfully Written.");
	}
	fclose(data);
	printf("\n\n\tPress 1 to Continue and 0 to EXIT");
	printf("\n\n\tINPUT:\t");
	int exit_status;
	scanf("%d",&exit_status);
	if(exit_status!=1)
	{
		head_disp();
		printf("\n\n\tThank You for Using this Application.\n");
		exit(0);
	}
}

void emp_modify()
{
	head_disp();;
	FILE *data;
	int id,found=0;
	data = fopen("employeeRecord.txt","r+");
	printf("\n\tEnter Employee ID to Edit Record:\t");
	scanf("%d",&id);
	while(fread(&personal,sizeof(personal),size,data)==size)
	{
```

```
            if(personal.emp_id == id)
            {
                  found = 1;
                  printf("\n\tEXISTING RECORD IS......");
                  printf("\n\n\tEMPLOYEE ID:\t%d",personal.emp_id);
                  printf("\n\tEMPLOYEE NAME:\t%s",personal.name);
                  printf("\n\tEMPLOYEE
POSITION:\t%s",personal.position);
                  printf("\n\tEMPLOYEE AGE:\t%d",personal.age);
                  printf("\n\tEMPLOYEE
QUALIFICATIONS:\t%s",personal.qualification);
                  printf("\n\tEMPLOYEE
CERTIFICATIONS:\t%s",personal.certifications);
                  printf("\n\tEMPLOYEE WORK
EXPERIENCE:\t%d",personal.work_exp);
                  printf("\n\tEMLOYEE SALARY:\t%.2f",personal.salary);
                  printf("\n\tEMPLOYEE PHONE
NUMBER:\t%ld",personal.phone);
                  printf("\n\tPROGRAMMING LANGUAGES
KNOWN:\t%s",personal.prog_lang);
                  printf("\n\n************MODIFYING EMPLOYEE
RECORD**************\n");
                  printf("\n\tEnter New Employee Name:\t");
                  scanf("%s",personal.name);
                  printf("\n\tEnter New Employee Position:\t");
                  scanf("%s",personal.position);
                  printf("\n\tEnter New Employee Age:\t");
                  scanf("%d",&personal.age);
                  printf("\n\tEnter New Employee Qualifications:\t");
                  scanf("%s",personal.qualification);
                  printf("\n\tEnter New Employee Certifications:\t");
                  scanf("%s",personal.certifications);
                  printf("\n\tEnter New Employee Work Experience:\t");
                  scanf("%d",&personal.work_exp);
                  printf("\n\tEnter new Employee Salary:\t");
                  scanf("%f",&personal.salary);
                  printf("\n\tEnter new Employee Phone Number:\t");
                  scanf("%ld",&personal.phone);
                  printf("\n\tEnter new Employee Programming languages
Known:\t");
                  scanf("%s",personal.prog_lang);
                  fseek(data,-sizeof(personal),SEEK_CUR);
            fwrite(&personal,sizeof(personal),size,data);
            head_disp();;
            printf("\n\n\tEmployee Record Successfuly Written.");
            }
            break;
      }
      if(found == 0)
      {
            printf("\n\tRECORD NOT FOUND...");
      }
      fclose(data);
      printf("\n\n\tPress 1 to Continue and 0 to EXIT");
      printf("\n\n\tINPUT:\t");
      int exit_status;
```

```
	scanf("%d",&exit_status);
	if(exit_status!=1)
	{
		head_disp();
		printf("\n\n\tThank You for Using this Application.\n");
		exit(0);
	}
}

void emp_search()
{
	head_disp();;
	FILE *data;
	data = fopen("employeeRecord.txt","r");
	int found = 0,search_choice,search_emp_id;
	char search_name[100],ch='Y';
	printf("\n\tWhat Do You Know about the Employee?");
	printf("\n\tEmployee ID: Enter 1");
	printf("\n\tEmployee Name: Enter 2");
	printf("\n\tINPUT:\t");
	scanf("%d",&search_choice);
	switch(search_choice)
	{
		case 1:
				printf("\n\tEnter Employee ID:\t");
				scanf("%d",&search_emp_id);
				break;
		case 2:
				printf("\n\tEnter Employee Name:\t");
				scanf("%s",search_name);
				break;
		default:
				ch='N';
				printf("\n\tWRONG CHOICE.\n");
	}
	while(fread(&personal,sizeof(personal),size,data)==size)
	{
		if(personal.emp_id == search_emp_id ||
strcmp(personal.name,search_name)==0)
			{
				found = 1;
				printf("\n\t*****************EMPLOYEE RECORD
IS......");
				printf("\n\n\tEMPLOYEE
ID:\t%d",personal.emp_id);
				printf("\n\tEMPLOYEE
NAME:\t%s",personal.name);
				printf("\n\tEMPLOYEE
POSITION:\t%s",personal.position);
				printf("\n\tEMPLOYEE AGE:\t%d",personal.age);
				printf("\n\tEMPLOYEE
QUALIFICATIONS:\t%s",personal.qualification);
				printf("\n\tEMPLOYEE
CERTIFICATIONS:\t%s",personal.certifications);
				printf("\n\tEMPLOYEE WORK
EXPERIENCE:\t%d",personal.work_exp);
```

```
                        printf("\n\tEMLOYEE
SALARY:\t%.2f",personal.salary);
                        printf("\n\tEMPLOYEE PHONE
NUMBER:\t%ld",personal.phone);
                        printf("\n\tPROGRAMMING LANGUAGES
KNOWN:\t%s",personal.prog_lang);
                  }
      }
      if(found == 0 && ch!='N')
      {
            printf("\n\tNO SUCH EMPLOYEE RECORD FOUND.");
      }
      fclose(data);
      printf("\n\n\tPress 1 to Continue and 0 to EXIT");
      printf("\n\n\tINPUT:\t");
      int exit_status;
      scanf("%d",&exit_status);
      if(exit_status!=1)
      {
            head_disp();
            printf("\n\n\tThank You for Using this Application.\n");
            exit(0);
      }
}

void emp_delete()
{
      head_disp();;
      int found = 0,del_emp_id;
      FILE *data,*temp;
      data = fopen("employeeRecord.txt","r");
      temp = fopen("temp_data.txt","w");
      if(data == NULL)
      {
            printf("\n\tError in Opening File");
            exit(0);
      }
      printf("\n\tEnter Employee ID to delete it's Record:\t");
      scanf("%d",&del_emp_id);
      while((fread(&personal,sizeof(personal),size,data)==size))
      {
            if(personal.emp_id!=del_emp_id)
                  fwrite(&personal,sizeof(personal),size,temp);
      }
      fclose(data);
    fclose(temp);
    remove("employeeRecord.txt");
    rename("temp_data.txt","employeeRecord.txt");
    data=fopen("employeeRecord.txt","r");
    printf("\n\tRecord successfully deleted.\n\n");
    printf("\n\n\tPress 1 to Continue and 0 to EXIT");
      printf("\n\n\tINPUT:\t");
      int exit_status;
      scanf("%d",&exit_status);
      if(exit_status!=1)
      {
```

```
            head_disp();
            printf("\n\n\tThank You for Using this Application.\n");
            exit(0);
      }
}

void emp_displayAll()
{
      head_disp();;
      FILE *data;
    data=fopen("employeeRecord.txt","r");
    int totalemployee=0;
    int i=0;
    printf("\n\n\t**********************ALL EMPLOYEES
INFORMATION**********************\n\n");
    while((fread(&personal,sizeof(personal),size,data)==size))
    {
      i++;
      printf("\n\n\tEMPLOYEE ID:\t%d",personal.emp_id);
            printf("\n\tEMPLOYEE NAME:\t%s",personal.name);
            printf("\n\tEMPLOYEE POSITION:\t%s",personal.position);
            printf("\n\tEMPLOYEE AGE:\t%d",personal.age);
            printf("\n\tEMPLOYEE
QUALIFICATIONS:\t%s",personal.qualification);
            printf("\n\tEMPLOYEE
CERTIFICATIONS:\t%s",personal.certifications);
            printf("\n\tEMPLOYEE WORK
EXPERIENCE:\t%d",personal.work_exp);
            printf("\n\tEMLOYEE SALARY:\t%.2f",personal.salary);
            printf("\n\tEMPLOYEE PHONE NUMBER:\t%ld",personal.phone);
            printf("\n\tPROGRAMMING LANGUAGES
KNOWN:\t%s\n\n\n",personal.prog_lang);
      }
      totalemployee+=i;
    printf("\n\n\tTOTAL EMPLOYEES: %d\n\n",totalemployee);
    fclose(data);
    printf("\n\n\tPress 1 to Continue and 0 to EXIT");
      printf("\n\n\tINPUT:\t");
      int exit_status;
      scanf("%d",&exit_status);
      if(exit_status!=1)
      {
            head_disp();
            printf("\n\n\tThank You for Using this Application.\n");
            exit(0);
      }

}

void emp_display()
{
      head_disp();;
      FILE *data;
      data = fopen("employeeRecord.txt","r");
      int found = 0,search_emp_id;
      printf("\n\tEnter Employee ID to display it's Record:\t");
```

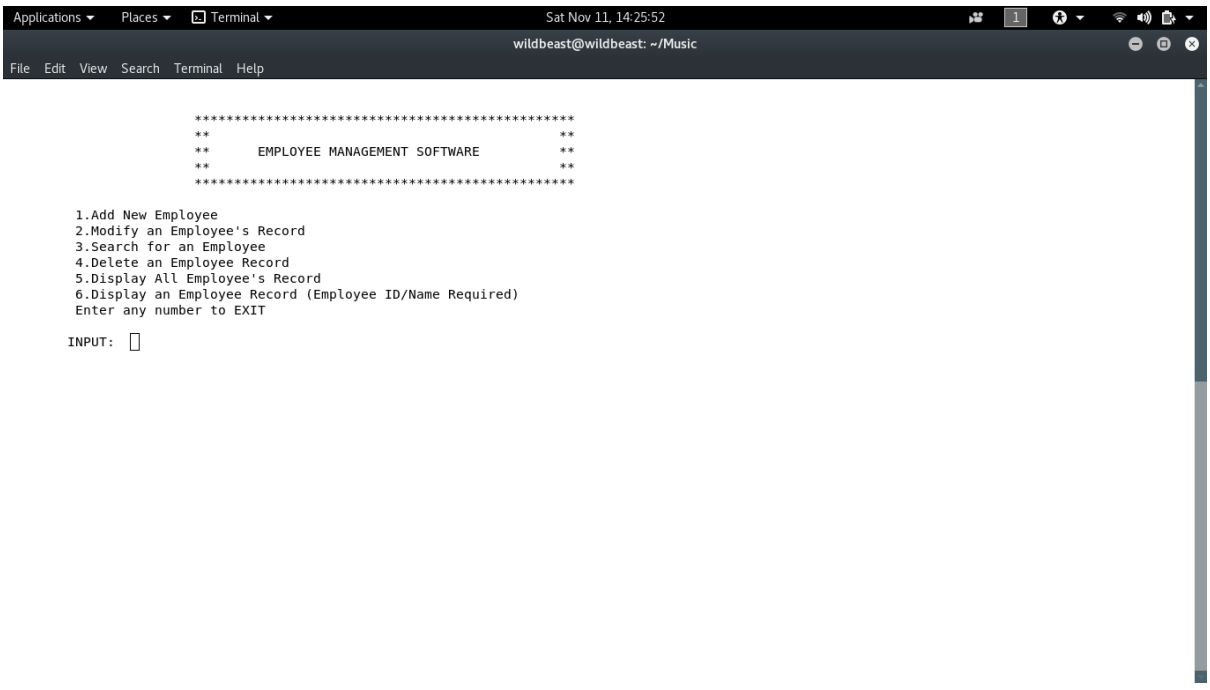
```
	scanf("%d",&search_emp_id);
	while(fread(&personal,sizeof(personal),size,data)==size)
	{
		if(personal.emp_id == search_emp_id)
			{
				found = 1;
				printf("\n\tEMPLOYEE RECORD IS......");
				printf("\n\tEMPLOYEE
ID:\t%d",personal.emp_id);
				printf("\n\tEMPLOYEE
NAME:\t%s",personal.name);
				printf("\n\tEMPLOYEE
POSITION:\t%s",personal.position);
				printf("\n\tEMPLOYEE AGE:\t%d",personal.age);
				printf("\n\tEMPLOYEE
QUALIFICATIONS:\t%s",personal.qualification);
				printf("\n\tEMPLOYEE
CERTIFICATIONS:\t%s",personal.certifications);
				printf("\n\tEMPLOYEE WORK
EXPERIENCE:\t%d",personal.work_exp);
				printf("\n\tEMLOYEE
SALARY:\t%.2f",personal.salary);
				printf("\n\tEMPLOYEE PHONE
NUMBER:\t%ld",personal.phone);
				printf("\n\tPROGRAMMING LANGUAGES
KNOWN:\t%s",personal.prog_lang);
				break;
			}
	}
	if(found == 0)
	{
		printf("\n\tNO SUCH EMPLOYEE RECORD FOUND.");
	}
	printf("\n\n\tPress 1 to Continue and 0 to EXIT");
	printf("\n\n\tINPUT:\t");
	int exit_status;
	scanf("%d",&exit_status);
	if(exit_status!=1)
	{
		head_disp();
		printf("\n\n\tThank You for Using this Application.\n");
		exit(0);
	}
	fclose(data);
}
```

OUTPUT

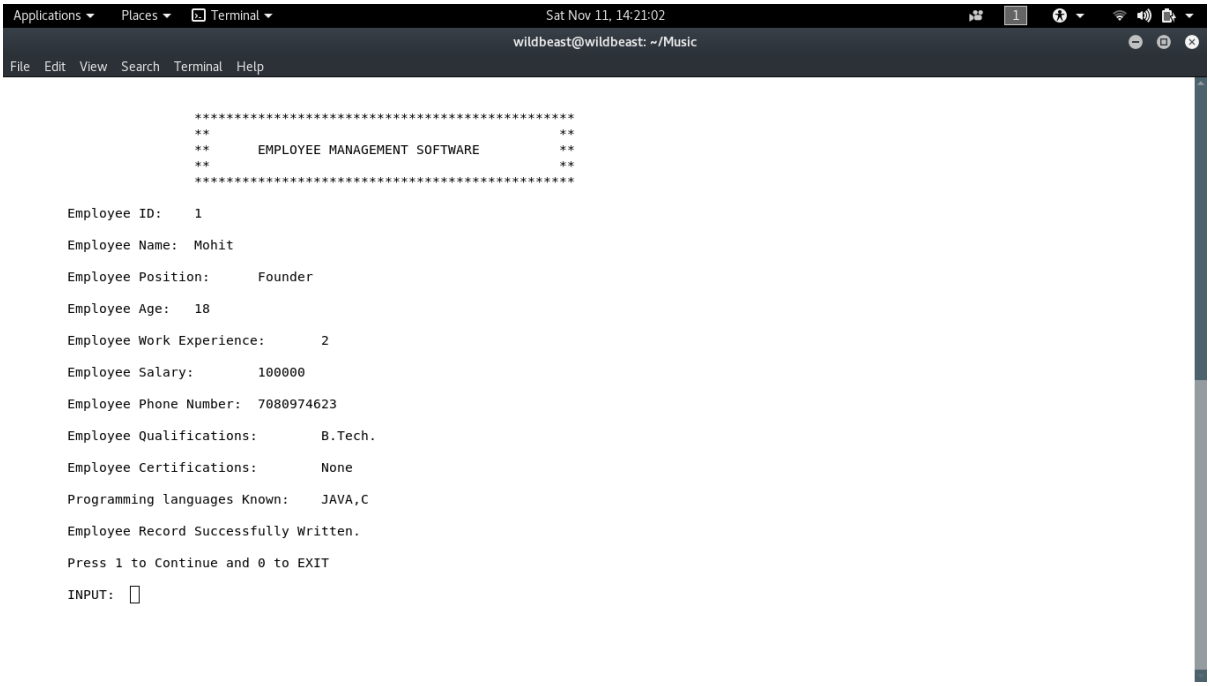
1.Default Application Screen (Database Not Present)

---

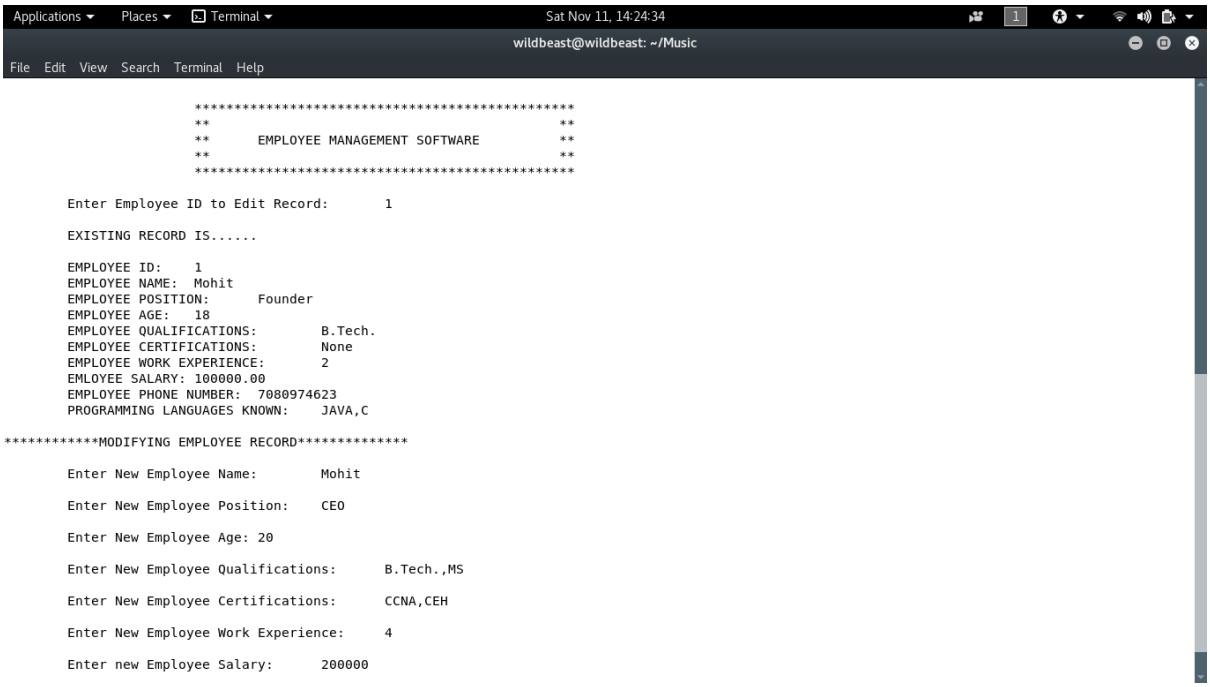
2. Default Application Screen (with Database)



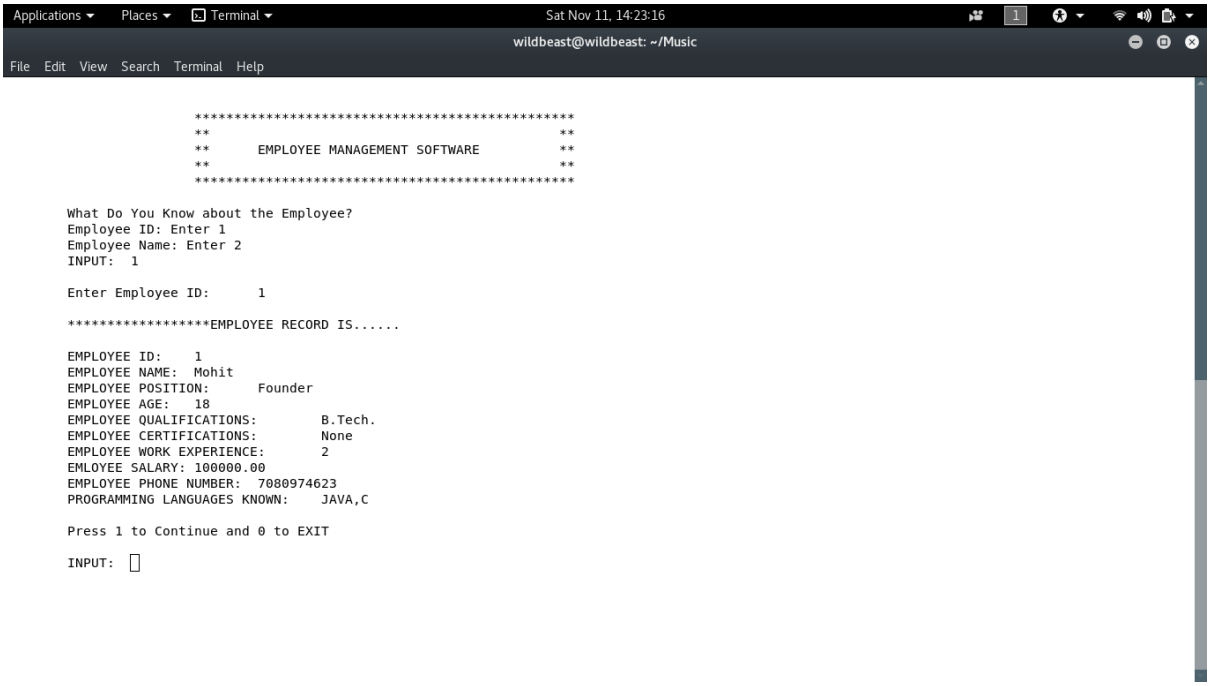
3.Add New Employee Function



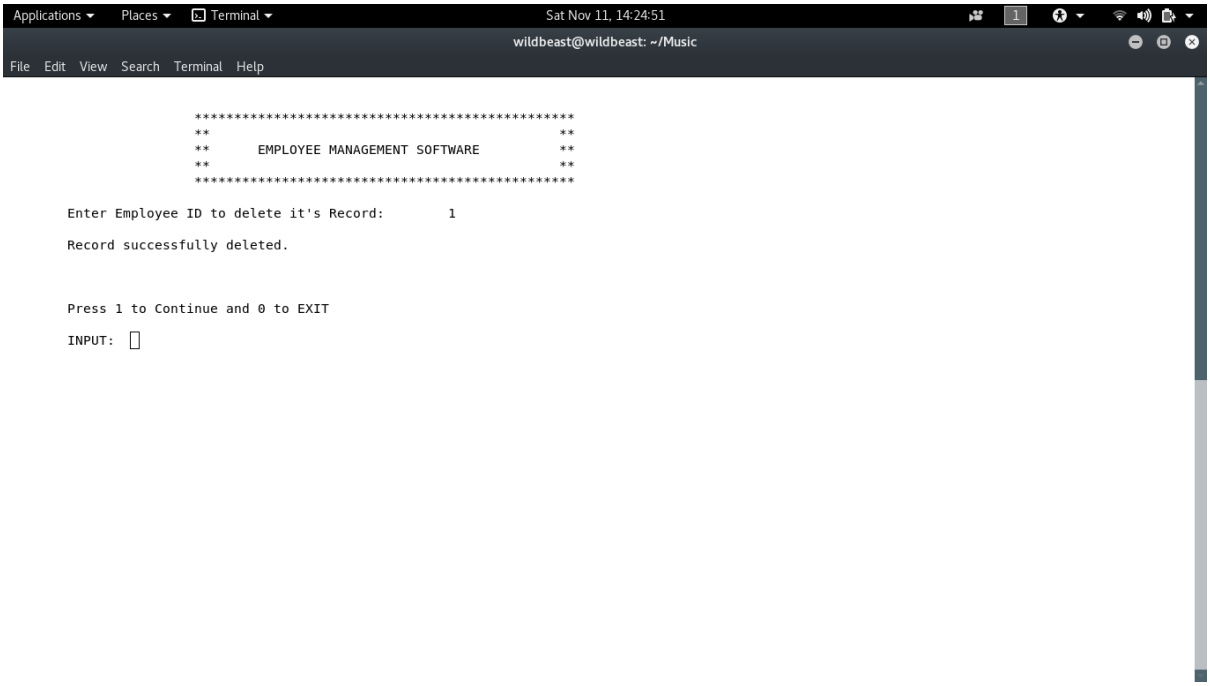
4. Modify Employee's Record Function



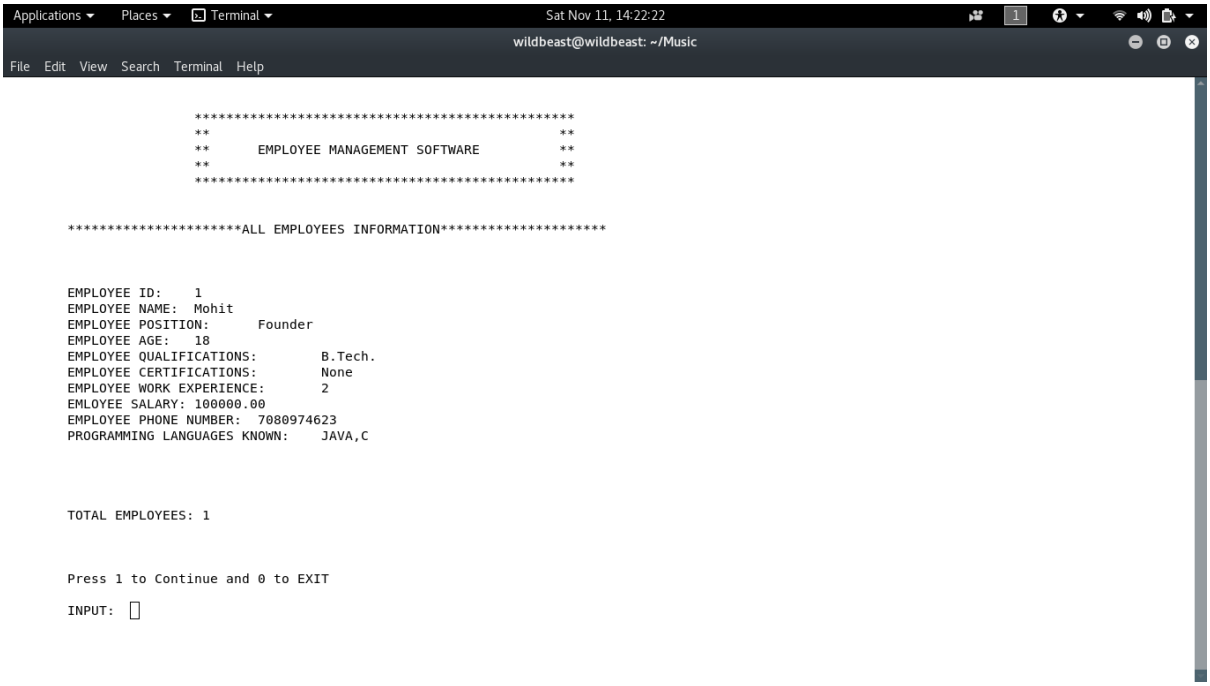
5. Search for an Employee Function



6. Delete an Employee's Record Function



7. Display All Employee's Information Function



8. Display an Employee's Record Function