Submitted: 07.01.2024

COMP 205

Systems Programming

Employee Management System

Sinem Özbey 042101122
Muhammed Baha Karadağlı 042101106
Semih Yılmaz 042101116
Yağmur Kaykaç 041801112

EXECUTIVE SUMMARY

In this project, the assignment was to create a project of our choosing, and we chose an employee system. We worked as a team to finish the project and create the documents while developing the program.

In addition to that, this report includes a project proposal, code description, outputs, timeline, challenges, explanation of the project and the resources we have used.

TABLE OF CONTENTS

PROJECT PROPOSAL

The employee management system enables effective management of employee and manager information within the organization, employee additions and removals, annual leave requests and salary management. The system provides secure access and streamlined operations by providing separate functions for both administrator and employee logins.

Our goal for this project is to create an Employee Management System that is easy to use and helps organize information for both managers and employees. One of the most important functions is secure login, which allows anyone to access their features and perform actions on their behalf.

The main purpose of the system is for managers to manage employee information, control additions and deletions, meet annual leave requests and manage salary-related activities, while providing an effective and safe platform for employees by offering the opportunity to request leave. With the usage of structures such as EmployeeCard, Employee, IDCards and EmployeeSurvey, we have ensured that essential details are organized systematically. It is very important to use file manipulation to keep all this information safe and persistent. User interfaces are designed to provide ease of use for both administrators and regular employees. Basically, our project is set up to help with managing people at work in a complete way.

CODE DESCRIPTIONS

1. DEFINES

```
6
7  #define MAX_EMPLOYEES 1000
8
9  #define FILENAME "employees.txt"
10 #define TEMPFILE "temp.txt"
11 #define ADMINFILE "adminaccount.txt"
12 #define USERFILE "useraccount.txt"
13 #define ADMINANNUALFILE "annualdayrequest.txt"
14
```

Figure 1. Define functions

FILENAME: Represents the employees.txt file, which contains the employee's ID, worker title, name, surname, age, gender, country, city, total leave duration, salary and leave request.

TEMPFILE: Represents the temp.txt, which isa file where temporary data is stored

ADMINFILE: Represents the adminaccount.txt file, which contains the admin's username and password.

USERFILE: Represents the useraccount.txt file, which contains the employee's username and password.

ADMINANNUALFILE: Represents the annualdayrequest.txt, which contains the start and end dates of the requested leave and the number of days the leave is valid.

Also the maximum number of employees is defined.

2. DATA STRUCTURES

Structures are written to be useful and effective. For example, variables that are included in the EmployeeCard structure already exist in other structures, but requesting only these three helps us write the function more efficiently.

```
15 struct EmployeeCard {
16     char name[50];
17     char surname[50];
18     char id[10];
19 };
```

Figure 2. Structure

```
21 struct IDCards {
22 
23     char username[20];
24     char password[20];
25 };
```

Figure 3. Structure

```
27 struct Employee{
28 
29     char name[50];
30     char surname[50];
31     char age[50];
32     char title[50];
33     char gender[50];
34     char call[3];
35     char country[50];
36     char city[50];
37     char id[20];
38     int annual;
39     float salary;
40     char AnnualRequest[10];
41 };
```

Figure 4. Structure

```
43 struct EmployeeAnnual {
44 
45     char name[50];
46     char surname[50];
47     char startDay[10];
48     char startMonth[10];
49     char startYear[10];
50     char finishDay[10];
51     char finishMonth[10];
52     char finishYear[10];
53     char id[20];
54     int requestedDay;
55     int totalDay;
56 
57 };
```

Figure 5. Structure

```
struct EmployeeSurvey {

    char name[50];
    char surname[50];
    char id[10];
    char date_day[2];
    char date_month[2];
    char date_year[4];
    char title[20];
    char subject[20];
    char message[100];
    int requestedDay;
    int totalDay;

};
```

Figure 6. Structure

3. FUNCTIONS

All the functions have been declared from the beginning before it is used.

```
void loginpage();
void userpage(char username[20]);
void adminpage(char username[20]);
int LineSelector(char username[20], FILE *file);
void Username_to_ID(char username[20],int LineNumber, FILE *file, struct
    EmployeeCard IDCard[1]);
void ID_to_Username(char ID[10], int LineNumber, FILE *file, struct IDCards
    IDCard[1]);
void ID_to_EmployeeInformation(char username[20],float updated_salary, int
    LineNumber);
void SalaryChangeEmployee();
void FileCopier();
void FileCopier_toAnnual();
void FileCopier_toUser();
void ID_to_EmployeeAnnualLeave(char username[20], char decision[10], int
    LineNumber);
```

Figure 7. Declaring functions

3.1. FUNCTIONS FOR MANAGERS

3.1.1. addEmployee:

This function is designed to include a new employee in the existing employee array. The function makes it easier for the administrator to add new employees to the system by allowing the administrator to enter important identification information such as the new employee's ID, name, surname, employee title, username and password.

```
void addEmployee(struct Employee employees[], int *numEmployees) {
    if (*numEmployees < MAX_EMPLOYEES) {
        struct Employee newEmployee;
        struct IDCards ID;

        printf("Enter employee ID: ");
        scanf("%s", newEmployee.id);

        printf("Enter employee name: ");
        scanf("%s", newEmployee.name);

        printf("Enter employee surname: ");
        scanf("%s", newEmployee.surname);

        bool IsOkey = true;
        int choicefortitle;
        do {
        printf("Enter employee title: \n");
        printf("1. CEO\n");
        printf("2. CFO\n");
        printf("3. Project Manager\n");
        printf("4. Team Leader\n");
        printf("5. Engineer\n");
        printf("6. Intern\n");
        scanf("%d", &choicefortitle);
```

Figure 8. Part of the addEmployees function

```
void addEmployee(struct Employee employees[], int *numEmployees) {

    printf("We have completed entering information about the %s and %s username is
        %s\n", newEmployee.name, newEmployee.call, ID.username);

        employees[*numEmployees] = newEmployee;
        (*numEmployees)++;

        // Open the file in append mode
        FILE *file = fopen(FILENAME, "a");
        if (file != NULL) {
            // Write the new employee information to the file
            fprintf(file, "%s %s %s %s %s %s %s %s %d %.2f %s\n", newEmployee.id,
                newEmployee.title, newEmployee.name, newEmployee.surname,
                newEmployee.age, newEmployee.gender, newEmployee.country,
                newEmployee.city, newEmployee.annual, newEmployee.salary,
                newEmployee.AnnualRequest);

            // Close the file
            fclose(file);
        } else {
            printf("Error opening file for writing!\n");
        }


            // Open the file in append mode
        FILE *userfile = fopen(USERFILE, "a");
        if (userfile != NULL) {
            // Write the new employee information to the file
            fprintf(userfile, "%s %s\n", ID.username, ID.password);

            // Close the file
            fclose(userfile);
        } else {
            printf("Error opening file for writing!\n");
        }
```

Figure 9. Part of the addEmployees function

OUTPUT:

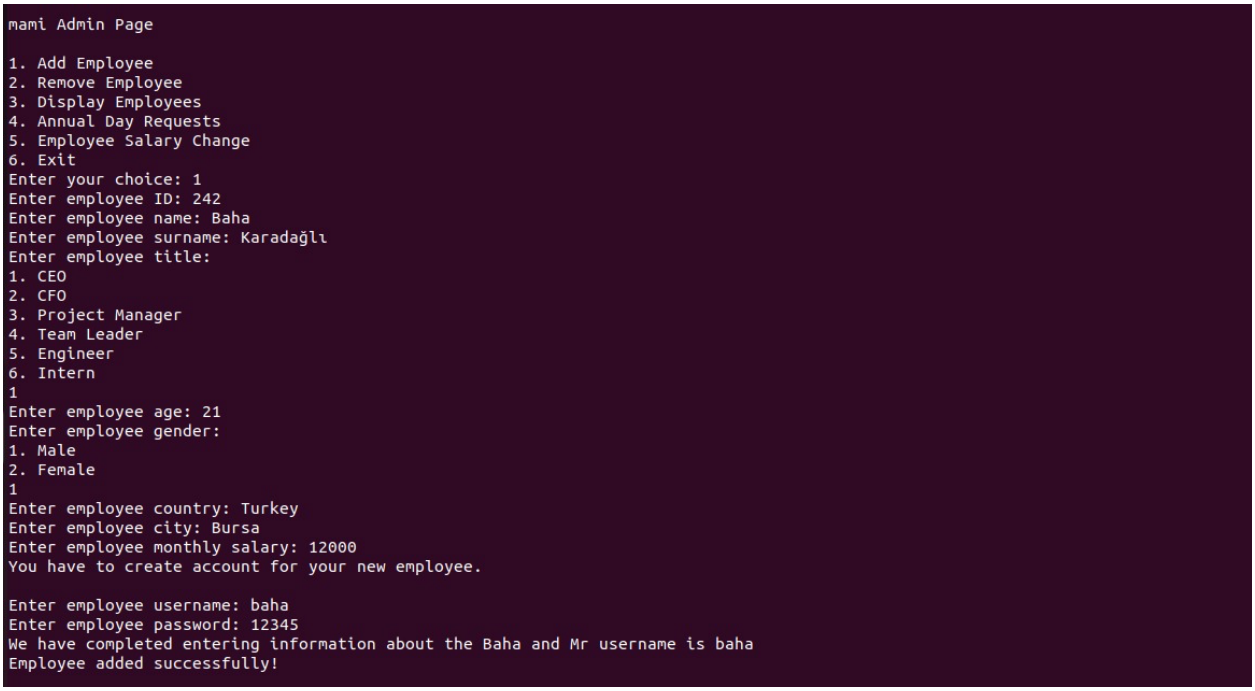

Figure 10. Output layout

3.1.2.removeEmployee:

This function allows the deregistration of an employee. The function finds the user with the specified ID in the employee file, excluding the relevant line, and then copies the remaining lines to a temporary file. It then removes the specified employee from the system by transferring the remaining lines from the temporary file back to the employee file.

```c
void removeEmployee() {
    char line[256];
    char line2[256];
    int lineNumber = 0;
    int currentLine = 0;
      char employee_id[20];

        printf("Enter employee ID who you gonna remove: ");
        scanf("%s", employee_id);

        FILE *employeefile_r = fopen(FILENAME, "r");
       FILE *tempFile = fopen(TEMPFILE,"w");

       lineNumber = LineSelector(employee_id,employeefile_r);

       printf("ID is in %d. line",lineNumber);

    while (fgets(line, 256, employeefile_r) != NULL) {

            currentLine++;

        if (currentLine != lineNumber) {

            fputs(line, tempFile);

        }
    }
    fclose(employeefile_r);
    fclose(tempFile);
    FileCopier();

    FILE *userfile_r = fopen(USERFILE, "r");
    FILE *tempFile_2 = fopen(TEMPFILE,"w");

         while (fgets(line2, 256, userfile_r) != NULL) {

        currentLine++;

        if (currentLine != lineNumber) {

            fputs(line2, tempFile_2);

        }
    }
    fclose(userfile_r);
    fclose(tempFile_2);
    FileCopier_toUser();
}
```

Figure 11. removeEmployee Function

3.1.3. SalaryChangeEmployee:

This function allows updating an employee's salary by asking the manager for the employee ID and new salary. The function identifies the relevant line number in the employee file and then uses the ID_to_EmployeeInformation function to modify and save the salary information.

```c
void SalaryChangeEmployee() {

    char line[256];
      int lineNumber = 0;
      int currentLine = 0;

      char employee_id[20];
      float updated_salary;

      char name[20];
      char surname[20];
      char id[10];

        printf("Enter employee ID who you gonna remove: ");
        scanf("%s", employee_id);

        printf("Enter employee`s new salary: ");
        scanf("%f", &updated_salary);


        //FILE *tempfile_w = fopen(TEMPFILE, "a");
        FILE *employeefile_r = fopen(FILENAME, "r");

    lineNumber = LineSelector(employee_id,employeefile_r);

    //printf("ID is in %d \n", LineSelector(employee_id,employeefile_r));
    fclose(employeefile_r);

    ID_to_EmployeeInformation(employee_id , updated_salary,lineNumber);

}
```

Figure 12. SalaryChangeEmployee function

```
Employee List from File:
ID      Title           Name            Surname         Age     Gender          Country         City            Annual  Salary
-------------------------------------------------------------------------------------------------------------------------
242     CEO             Baha            Karadağlı       21      Male            Turkey          Bursa           30      12000.00

Annual Request: False
-------------------------------------------------------------------------------------------------------------------------
243     CEO             Yağmur          Kaykaç          21      Female          Turkey          Istanbul        30      15000.00

Annual Request: False
-------------------------------------------------------------------------------------------------------------------------
244     CEO             Sinem           Özbey           21      Female          Turkey          Istanbul        30      16000.00

Annual Request: False
-------------------------------------------------------------------------------------------------------------------------
245     CEO             Semih           Yılmaz          21      Male            Turkey          Istanbul        30      18000.00

Annual Request: False
-------------------------------------------------------------------------------------------------------------------------

mami Admin Page
```

Figure 13. SalaryChangeEmployee output

```
mami Admin Page

1. Add Employee
2. Remove Employee
3. Display Employees
4. Annual Day Requests
5. Employee Salary Change
6. Exit
Enter your choice: 5
Enter employee ID who you gonna change salary: 243
Enter employee`s new salary: 25000
File copied successfully.
```

```
mami Admin Page

1. Add Employee
2. Remove Employee
3. Display Employees
4. Annual Day Requests
5. Employee Salary Change
6. Exit
Enter your choice: 3

Employee List from File:
ID      Title           Name            Surname         Age     Gender          Country         City            Annual  Salary
-------------------------------------------------------------------------------------------------------------------------
242     CEO             Baha            Karadağlı       21      Male            Turkey          Bursa           30      12000.00

Annual Request: False
-------------------------------------------------------------------------------------------------------------------------
243     CEO             Yağmur          Kaykaç          21      Female          Turkey          Istanbul        30      25000.00

Annual Request: False
-------------------------------------------------------------------------------------------------------------------------
244     CEO             Sinem           Özbey           21      Female          Turkey          Istanbul        30      16000.00

Annual Request: False
-------------------------------------------------------------------------------------------------------------------------
245     CEO             Semih           Yılmaz          21      Male            Turkey          Istanbul        30      18000.00

Annual Request: False
-------------------------------------------------------------------------------------------------------------------------
```

Figure 14. SalaryChangeEmployee output

3.1.4. DisplayAnnualDayRequests:

This function presents annual leave requests stored in the ADMINANNUALFILE file to the administrator; thus allowing the administrator to accept or reject requests. Following user decisions, the function updates the FILENAME file according to the acceptance or rejection of annual leave requests.

```c
void DisplayAnnualDayRequests() {

    int currentLine = 0;
    int lineNumber = 0;
    char line[256];
    char line2[256];
    char line3[256];
    struct EmployeeAnnual employeeAnnual[1];
    struct Employee employee[1];
    FILE *annualfile = fopen(ADMINANNUALFILE, "r");
    FILE *employeefile = fopen(FILENAME, "r");
    if (annualfile != NULL) {

    //Display the elements in the function
        printf("\nEmployee List from File:\n");
        printf("ID\t\tName\t\tSurname\t\tStart Date\tFinish Date\tRequested Days\n");
        while (fscanf(annualfile, "%s %s %s %s %s %s %s %s %s %d", employeeAnnual[0].id, employeeAnnual[0].name, employeeAnnual[0].surname,
            employeeAnnual[0].startDay, employeeAnnual[0].startMonth, employeeAnnual[0].startYear, employeeAnnual[0].finishDay, employeeAnnual[0].finishMonth,
            employeeAnnual[0].finishYear, &employeeAnnual[0].requestedDay) == 10) {



            printf("%s\t\t%-15s\t%-15s\t%-2s.%-2s.%-4s\t%-2s.%-2s.%-4s\t%d\t\t%d\t%s\n", employeeAnnual[0].id, employeeAnnual[0].name, employeeAnnual[0].surname,
                employeeAnnual[0].startDay, employeeAnnual[0].startMonth, employeeAnnual[0].startYear, employeeAnnual[0].finishDay, employeeAnnual[0].finishMonth,
                employeeAnnual[0].finishYear, employeeAnnual[0].requestedDay, employee[0].annual, employee[0].AnnualRequest);

         }



    } else {
        printf("Error opening file for reading!\n");
    }


       fclose(annualfile);
        fclose(employeefile);
}
```

Figure 15. Part of a DisplayAnnualDayRequests function

```c
switch (choiceforselect) {
            case 1:
                strcpy(employee[0].AnnualRequest, "True");
                IsValid = false;
                break;
            case 2:
                strcpy(employee[0].AnnualRequest, "False");
                IsValid = false;
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }


                } while (IsValid);
          printf("\n---------------------------------- ");
```

Figure 16. Part of the DisplayAnnualDayRequests function

OUTPUT:

```
mami Admin Page

1. Add Employee
2. Remove Employee
3. Display Employees
4. Annual Day Requests
5. Employee Salary Change
6. Exit
Enter your choice: 3

Employee List from File:
ID      Title           Name            Surname         Age     Gender          Country         City            Annual  Salary
-----------------------------------------------------------------------------------------------------------------------------
242     CEO             Baha            Karadağlı       21      Male            Turkey          Bursa           30      12000.00

Annual Request: False
-----------------------------------------------------------------------------------------------------------------------------
```

Figure 16. DisplayAnnualDayRequests Output

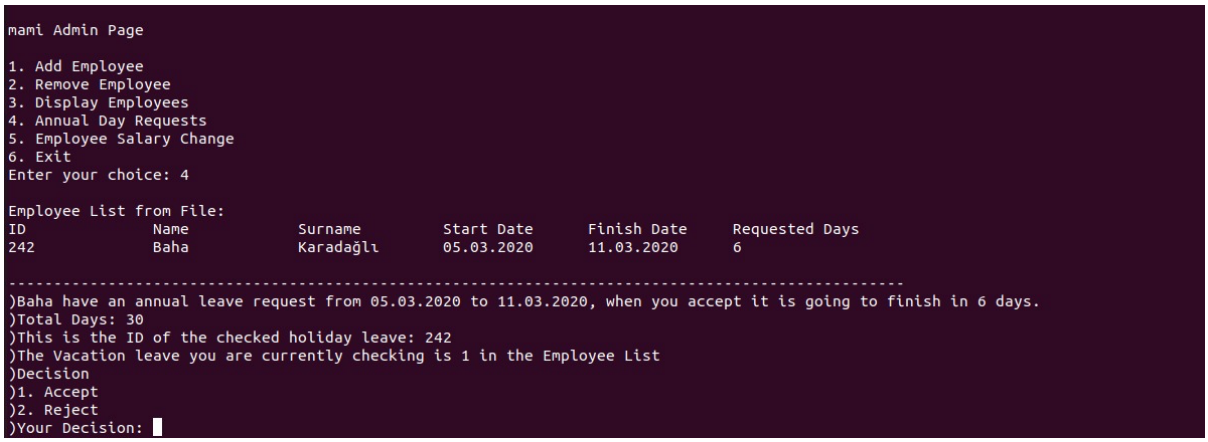

Figure 17. DisplayAnnualDayRequests Output

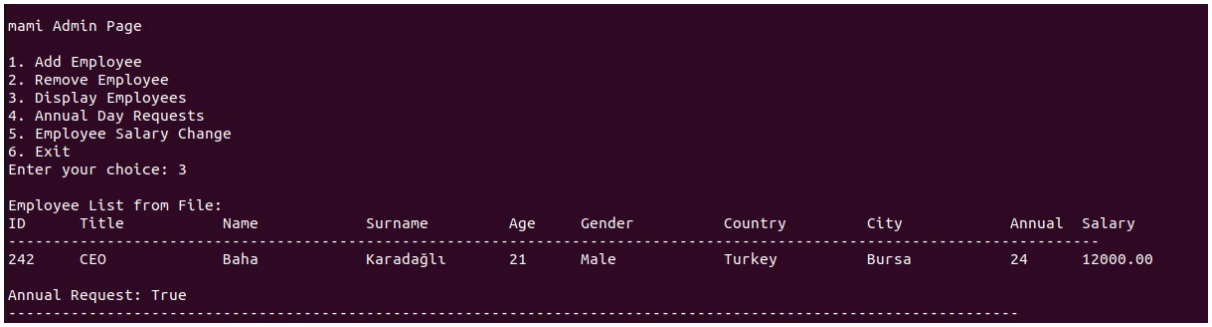

Figure 18. DisplayAnnualDayRequests Output

3.1.5. DisplayEmployeesFromFile:

This function reads employee information from the FILENAME file and displays it in a table. It prints out details such as ID, title, name, surname, age, gender, country, city, annual leave status, salary, and annual leave request status for each employee. This function helps the manager to easily review employee data.

```c
void DisplayEmployeesFromFile() {

    FILE *file = fopen(FILENAME, "r");
    if (file != NULL) {
        printf("\nEmployee List from File:\n");
        printf
            ("ID\tTitle\t\tName\t\tSurname\t\tAge\tGender\t\tCountry\t\tCity\t\tAnnu
            al\tSalary\n");
        printf
            ("-------------------------------------------------------------------------
            ------------------------------------------------\n");

        struct Employee employee;

        while (fscanf(file, "%s %s %s %s %s %s %s %s %d %f %s",  employee.id,
            employee.title, employee.name, employee.surname, employee.age,
            employee.gender, employee.country, employee.city, &employee.annual,
            &employee.salary, employee.AnnualRequest) == 11) {
        printf("%s\t%-15s\t%-15s\t%-15s\t%-3s\t%-6s\t\t%-15s\t%-15s\t%d\t%.2f\n\n",
                   employee.id, employee.title, employee.name, employee.surname,
                       employee.age, employee.gender, employee.country,
                       employee.city, employee.annual, employee.salary);

        printf("Annual Request: %s\n",employee.AnnualRequest);
    printf
        ("-------------------------------------------------------------------------
        ------------------------------------------------\n");
                    }
        } else {
        printf("Error opening file for reading!\n");
        }

        fclose(file);
}
```

Figure 19. Part of displayEmployeesFromFile Function

- **OUTPUT:**

```
mami Admin Page

1. Add Employee
2. Remove Employee
3. Display Employees
4. Annual Day Requests
5. Employee Salary Change
6. Exit
Enter your choice: 3

Employee List from File:
ID      Title           Name            Surname         Age     Gender          Country         City            Annual  Salary
-------------------------------------------------------------------------------------------------------------------------
242     CEO             Baha            Karadağlı       21      Male            Turkey          Bursa           30      12000.00

Annual Request: False
-------------------------------------------------------------------------------------------------------------------------
243     CEO             Yağmur          Kaykaç          21      Female          Turkey          Istanbul        30      15000.00

Annual Request: False
-------------------------------------------------------------------------------------------------------------------------
244     CEO             Sinem           Özbey           21      Female          Turkey          Istanbul        30      16000.00

Annual Request: False
-------------------------------------------------------------------------------------------------------------------------
245     CEO             Semih           Yılmaz          21      Male            Turkey          Istanbul        30      18000.00

Annual Request: False
-------------------------------------------------------------------------------------------------------------------------
```

Figure 20. Display Employee Output

3.2. FUNCTIONS FOR EMPLOYEES

3.2.1. EmployeeAnnualDayRequest:

The EmployeeAnnualDayRequest function first determines the correct employee by reading from a file containing the user's information (USERFILE) and then retrieves the employee's detailed information from a separate file (FILENAME). It receives the start and end dates, year, month and day information for the annual leave request from the user.

After receiving the date information, it calculates the day difference between the start and end dates with the calculateTimeDifference function. If the requested leave days are more than 30, it automatically adjusts this number to 30 and informs the user. It then updates the number of remaining leave days by subtracting the number of requested days from the total leave entitlement.

After adding the employee's information and permission details to a specific file (ADMINANNUALFILE), it provides information that the transactions have been completed successfully. However, if an error occurs during file operations, a file write error message is delivered to the user. This function is used to receive and process employees' annual leave requests and save the results in a file; It also gives feedback to the user about whether the operation was successful or not.

```c
void EmployeeAnnualDayRequest(char username[20]) {
    char line[256];
    int lineNumber = 0;
    int currentLine = 0;
    int requestedDays, remainingAnnualLeave;
    int startYear, finishYear;
    int startMonth, finishMonth;


    FILE *accountfile = fopen(USERFILE, "r");

    if (accountfile == NULL) {
        perror("No Records Found");
        return;
    } else {
        while (fgets(line, 256, accountfile) != NULL) {
            lineNumber++;
            if (strstr(line, username) != NULL) {
                break;
            }
        }
        fclose(accountfile);
    }

    FILE *employeefile = fopen(FILENAME, "r");
    if (employeefile == NULL) {
        perror("No Records Found");
        return;
    }

    while (fgets(line, sizeof(line), employeefile) != NULL) {
        currentLine++;
        if (currentLine == lineNumber) {
            sscanf(line, "%d %s %s", &employeeAnnual.id, employeeAnnual.name,
                employeeAnnual.surname);
        }
    }
```

Figure 21. Part of a EmployeeAnnualRequest function

- **DaysInMonth Function**
  This function calculates the time difference between the entered values and prints the annual leave request on the screen. It also prints the remaining annual leave request on the screen.

```c
int daysInMonth(int month, int year) {

    switch (month) {
        case 1: case 3: case 5: case 7: case 8: case 10: case 12:
            return 31;
        case 4: case 6: case 9: case 11:
            return 30;
        case 2:

            if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))
                return 29;
            else
                return 28;
        default:
            return -1;
    }
}
```

Figure 22. DaysInMonth function

- **Calculate Time Difference Function**

This function calculates the time difference between the entered values and prints the annual leave request on the screen. It also prints the remaining annual leave request on the screen.

```c
int calculateTimeDifference(const char *startDay, const char *startMonth, const char
    *startYear,
                            const char *finishDay, const char *finishMonth, const char
                                *finishYear) {
    struct tm tmStart = {0}, tmEnd = {0};

    sscanf(startYear, "%d", &tmStart.tm_year);
    sscanf(startMonth, "%d", &tmStart.tm_mon);
    sscanf(startDay, "%d", &tmStart.tm_mday);


    sscanf(finishYear, "%d", &tmEnd.tm_year);
    sscanf(finishMonth, "%d", &tmEnd.tm_mon);
    sscanf(finishDay, "%d", &tmEnd.tm_mday);


    time_t timeStart = mktime(&tmStart);
    time_t timeEnd = mktime(&tmEnd);

    return (int)difftime(timeEnd, timeStart) / (60 * 60 * 24);
}
```

Figure 23. calculateTimeDifference function

OUTPUT:

```
baha User Page

1. Annual Day Request
2. Exit
Enter your choice: 1
Enter start year (2000 or later): 1999
Invalid input. Please enter a valid 4-digit year of 2000 or later.
Enter start year (2000 or later): -1
Invalid input. Please enter a valid 4-digit year of 2000 or later.
Enter start year (2000 or later): 571
Invalid input. Please enter a valid 4-digit year of 2000 or later.
Enter start year (2000 or later): 2016
Enter finish year (equal to or greater than start year): 2015
Invalid input. Please enter a valid 4-digit positive year equal to or greater than start year.
Enter finish year (equal to or greater than start year): 2016
Enter start day (01-31): 1
Invalid input. Please enter a valid two-digit day between 01 and 31.
Enter start day (01-31): 2
Invalid input. Please enter a valid two-digit day between 01 and 31.
Enter start day (01-31): 3
Invalid input. Please enter a valid two-digit day between 01 and 31.
Enter start day (01-31): 4
Invalid input. Please enter a valid two-digit day between 01 and 31.
Enter start day (01-31): 5
Invalid input. Please enter a valid two-digit day between 01 and 31.
Enter start day (01-31): 6
Invalid input. Please enter a valid two-digit day between 01 and 31.
Enter start day (01-31): 01
Enter finish day (01-31, not included in the annual leave): 06
Enter start month (01-12): 13
Invalid input. Please enter a valid two-digit month between 01 and 12.
Enter start month (01-12): 1
Invalid input. Please enter a valid two-digit month between 01 and 12.
Enter start month (01-12): 02
Enter finish month (01-12): 02
Requested Annual Leave: 5 days
Remaining Annual Leave: 25 days
Employee annual leave request added successfully!
```

Figure 24. Output of the Annual Day Request.

3.2.2. EmployeeSearchRequests:

This function displays information such as ID, title, name, surname, age, gender, country, city, annual leave status and salary of each employee in the FILENAME file. The purpose of the function is to facilitate access and tracking of important data in the file.

```
void EmployeeSearchRequests(char username[20]) {

    FILE *file = fopen(FILENAME, "r");
    if (file != NULL) {
        printf("\nEmployee List from File:\n");
        printf("ID\tTitle\t\tName\t\tSurname\t\tAge\tGender\t\tCountry\t\tCity\t\tAnnual\tSalary\n");

        // Read and display employee information from the file
        struct Employee employee;


        while (fscanf(file, "%s %s %s %s %s %s %s %s %d %f ",  employee.id, employee.title, employee.name,
            employee.surname, employee.age, employee.gender, employee.country, employee.city, &employee.annual,
            &employee.salary) == 10) {
            printf("%s\t%-15s\t%-15s\t%-15s\t%-3s\t%-6s\t\t%-15s\t%-15s\t%d\t%.2f\n",
                   employee.id, employee.title, employee.name, employee.surname, employee.age, employee.gender,
                       employee.country, employee.city, employee.annual, employee.salary);
        }

        // Close the file
        fclose(file);
        } else {
        printf("Error opening file for reading!\n");
        }
```

Figure 25. Part of the EmployeesearchRequests function

3.3. USEFUL FUNCTIONS

3.3.1. LineSelector:

This function takes a username and a file as input, and it searches for the specified username within the lines of the given file. If the file is not successfully opened, it returns an error. It then iterates through each line of the file, incrementing a line counter. If the username is found in any line, it immediately returns the corresponding line number. If the username is not found in the entire file the function returns -1.

```
int LineSelector(char username[20], FILE *file)
{
    int LineNumber = 0;
    char line[256];
    if (file == NULL) {
        perror("File opening error");
        return -1;

        }
            while (fgets(line, sizeof(line), file) != NULL) {

            LineNumber++;

            if (strstr(line, username) != NULL) {
                return LineNumber;
            }
            }

    return -1;
}
```

Figure 26. Part of the LineSelector function

3.3.2. Username_to_ID:

This function allows us to retrieve and organize employee details based on the username in the specified file. It reads the lines in the file, looks for the line corresponding to the provided LineNumber, and extracts the employee ID, first name, and last name associated with the given username. The information is stored in the EmployeeCard structure array.

```
void Username_to_ID(char username[20],int LineNumber, FILE *file, struct EmployeeCard IDCard[1])
{
    char line[256];
    int currentLine = 0;
        if (file == NULL) {
        perror("File opening error");
        return;
        }

        while (fgets(line, sizeof(line), file) != NULL) {
        currentLine++;
        if (currentLine == LineNumber) {
            sscanf(line, "%s %s %s", IDCard[0].id, IDCard[0].name, IDCard[0].surname);
        }
        }
        fclose(file);
}
```

Figure 27. Part of the Username_tı_ID function

3.3.3. ID_to_EmployeeInformation:

This function updates the employee's salary identified by the provided line number. The function reads each line in the FILENAME file and, if the line matches the specified line number, extracts the existing information, replaces the salary with the updated value, and temporarily writes the replaced data to the TEMPFILE file. The function then copies the contents of the temporary file to the original file, effectively updating the employee's salary information. Manages file errors and prints informative messages.

```
void ID_to_EmployeeInformation(char username[20], float updated_salary, int LineNumber)
{
        char line[256];
        int currentLine = 0;
        struct Employee Temp[1];
        FILE *infofile = fopen(FILENAME, "r");
        FILE *tempfile = fopen(TEMPFILE, "a");
        if (infofile == NULL) {
        perror("File opening error");
        return;
    }
    if (tempfile == NULL) {
        perror("File opening error");
           return;
    }
    while (fgets(line, sizeof(line), infofile) != NULL) {
        currentLine++;
        if (currentLine == LineNumber) {
                         sscanf(line, "%s %s %s %s %s %s %s %s %d %f %s", Temp[0].id, Temp[0].title, Temp[0].name,
                             Temp[0].surname,  Temp[0].age, Temp[0].gender, Temp[0].country, Temp[0].city, &Temp[0].annual,
                             &Temp[0].salary, Temp[0].AnnualRequest);

                    fprintf(tempfile,"%s %s %s %s %s %s %s %s %d %f %s\n", Temp[0].id, Temp[0].title, Temp[0].name,
                        Temp[0].surname, Temp[0].age, Temp[0].gender, Temp[0].country, Temp[0].city, Temp[0].annual,
                        updated_salary, Temp[0].AnnualRequest);

        printf("It was updated and put because LineNumber: %d CurrentLine: %d \n",LineNumber,currentLine);
        } else {
        printf("It was put because LineNumber: %d CurrentLine: %d \n",LineNumber,currentLine);
        fputs(line, tempfile);
        }
    }
        fclose(tempfile);
    fclose(infofile);
    FileCopier();
}
```

Figure 28. Part of the ID_to_Employee function

3.3.4. ID_to_EmployeeAnnualLeave:

This function updates the employee's annual leave amount specified by the given line number. It reads each line in the FILENAME file and extracts the existing information if the line matches the given line number, replaces the annual leave amount with the specified amount and writes the replaced data to the temporary TEMPFILE file. The function then updates the employee's annual leave status by copying the contents of the temporary file to the main file. It handles file errors and uses a file copy function to complete the update process.

```c
void ID_to_EmployeeAnnualLeave(char username[20], char decision[10], int LineNumber)
{
    char line[256];
    int currentLine = 0;
    struct Employee Temp[1];

        FILE *infofile = fopen(FILENAME, "r");
        FILE *tempfile = fopen(TEMPFILE, "a");
        if (infofile == NULL) {
        perror("File opening error");
        return;
    }
    if (tempfile == NULL) {
        perror("File opening error");
           return;
    }

    while (fgets(line, sizeof(line), infofile) != NULL) {

        currentLine++;

        if (currentLine == LineNumber) {

                        sscanf(line, "%s %s %s %s %s %s %s %s %d %f %s", Temp[0].id, Temp[0].title, Temp[0].name,
                            Temp[0].surname,  Temp[0].age, Temp[0].gender, Temp[0].country, Temp[0].city, &Temp[0].annual,
                            &Temp[0].salary, Temp[0].AnnualRequest);

                fprintf(tempfile,"%s %s %s %s %s %s %s %s %d %f %s\n", Temp[0].id, Temp[0].title, Temp[0].name,
                    Temp[0].surname, Temp[0].age, Temp[0].gender, Temp[0].country, Temp[0].city, Temp[0].annual,
                    Temp[0].salary, decision);

        //printf("Updated and put because LineNumber: %d CurrentLine: %d \n",LineNumber,currentLine);
        } else {
        //printf("Put because LineNumber: %d CurrentLine: %d \n",LineNumber,currentLine);
        fputs(line, tempfile);
        }
    }
        fclose(tempfile);
    fclose(infofile);
    FileCopier();
}
```

Figure 29. Part of the ID_to_EmployeeAnnualLeave function

3.3.5. ID_to_Username:

This function makes it easy to retrieve employee ID-based username and password information from a file. The function takes the employee ID and line number as input, searches for the corresponding ID in the provided file, and outputs the corresponding username and password. It then stores them in a structure called IDCard. The function closes the file after extraction.

```
void ID_to_Username(char ID[10], int LineNumber, FILE *file, struct IDCards IDCard[1])
{
    char line[256];
        if (file == NULL) {
        perror("File opening error");
        return;
        }

        while (fgets(line, sizeof(line), file) != NULL) {
        if (strstr(line, ID) != NULL) {

        sscanf(line, "%s %s", IDCard[0].username, IDCard[0].password);
        }
        }
        fclose(file);
}
```

Figure 30. Part of a ID_to_Username

3.3.6. FileCopier:

This function copies the contents of the temporary TEMPFILE file to the FILENAME file. After copying, it closes both files, clears the contents of the temporary file, and prints a success message.

```
void FileCopier()
{

    char ch;

    // Open the source file for reading
    FILE *sourceFile = fopen(FILENAME, "w");
    if (sourceFile == NULL) {
        perror("Error opening source file");
        return;
    }

    // Open the destination file for writing
    FILE *destinationFile = fopen(TEMPFILE, "r");
    if (destinationFile == NULL) {
        perror("Error opening destination file");
        fclose(sourceFile);

    }

    // Read from the source file and write to the destination file
    while ((ch = fgetc(destinationFile)) != EOF) {
        fputc(ch, sourceFile);
    }

    // Close both files
    fclose(sourceFile);
    fclose(destinationFile);

    printf("File copied successfully.\n");

    fclose(fopen(TEMPFILE,"w"));
    return;

}
```

Figure 31. File Copier function

3.3.7. FileCopier_toAnnual:

This function copies the contents of the temporary TEMPFILE file to the ADMINANNUALFILE file. After copying, it closes both files, clears the contents of the temporary file, and prints a success message.

```
void FileCopier_toAnnual()
{
    char ch;

    // Open the source file for reading
    FILE *sourceFile = fopen(ADMINANNUALFILE, "w");
    if (sourceFile == NULL) {
        perror("Error opening source file");
        return;
    }

    // Open the destination file for writing
    FILE *destinationFile = fopen(TEMPFILE, "r");
    if (destinationFile == NULL) {
        perror("Error opening destination file");
        fclose(sourceFile);

    }

    // Read from the source file and write to the destination file
    while ((ch = fgetc(destinationFile)) != EOF) {
        fputc(ch, sourceFile);
    }

    // Close both files
    fclose(sourceFile);
    fclose(destinationFile);

    printf("File copied successfully.\n");

    fclose(fopen(TEMPFILE,"w"));
    return;

}
```

Figure 32. FileCopier_toAnnual function

3.3.8 FileCopier_toUser:

This function copies the contents of the temporary TEMPFILE file to the USERFILE file. After copying, it closes both files, clears the contents of the temporary file, and prints a success message.

```c
void FileCopier_toUser()
{
    char ch;

    // Open the source file for reading
    FILE *sourceFile = fopen(USERFILE, "w");
    if (sourceFile == NULL) {
        perror("Error opening source file");
        return;
    }

    // Open the destination file for writing
    FILE *destinationFile = fopen(TEMPFILE, "r");
    if (destinationFile == NULL) {
        perror("Error opening destination file");
        fclose(sourceFile);

    }

    // Read from the source file and write to the destination file
    while ((ch = fgetc(destinationFile)) != EOF) {
        fputc(ch, sourceFile);
    }

    // Close both files
    fclose(sourceFile);
    fclose(destinationFile);

    printf("File copied successfully.\n");

    fclose(fopen(TEMPFILE,"w"));
    return;

}
```

Figure 33. FileCopier_toUserl function

3.4. FUNCTIONS FOR PAGES

3.4.1. adminpage:

This function provides an interface for administrators, offering options to add or remove employees, display employee information, handle annual day requests, change employee salaries, and exit the admin session. It uses a simple menu system and continues to prompt for choices until the user decides to exit.

```c
void adminpage(char username[20]) {

    struct Employee employees[MAX_EMPLOYEES];
    int numEmployees = 0;
    int choice;

   do {
        printf("\n%s Admin Page \n\n", username);
        printf("1. Add Employee\n");
        printf("2. Remove Employee\n");
        printf("3. Display Employees\n");
        printf("4. Annual Day Requests\n");
        printf("5. Employee Salary Change\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addEmployee(employees, &numEmployees);
                break;
            case 2:
                removeEmployee();
                break;
            case 3:
                DisplayEmployeesFromFile();
                break;
            case 4:
                DisplayAnnualDayRequests();
                break;
            case 5:
                SalaryChangeEmployee();
                break;
            case 6:
                loginpage();
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }

    } while (choice != 6);

}
```

Figure 34. adminpage function

```
baha@baha-VirtualBox:~/Desktop/205Project$ ./employeeinterview

Employee Detect System
1. Admin Login
2. User Login
3. Exit
Enter your choice: 1
Enter your username: mami
Enter your password: asd

mami Admin Page

1. Add Employee
2. Remove Employee
3. Display Employees
4. Annual Day Requests
5. Employee Salary Change
6. Exit
Enter your choice:
```

Figure 35. adminpage output

3.4.2. userpage:

This function is a simple user interface that allows users to request annual leave or log out. Users enter their selections through a basic menu and the function performs the corresponding action. The cycle continues until the user decides to exit**.**

```
void userpage(char username[20]) {


    struct Employee employees[MAX_EMPLOYEES];
    int numEmployees = 0;
    int choice;


   do {
        printf("\n%s User Page \n\n", username);
        printf("1. Annual Day Request\n");
        printf("2. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                EmployeeAnnualDayRequest(username);
                break;
            case 2:
                loginpage;
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }

    } while (choice != 2);

}
```

Figure 36. userpage function

```
Employee Detect System
1. Admin Login
2. User Login
3. Exit
Enter your choice: 2
Enter your username: baha
Enter your password: 12345

baha User Page

1. Annual Day Request
2. Exit
Enter your choice:
```

Figure 37. Userpage layout

3.4.3. loginpage:

This function provides a simple login experience with basic functions. The function is a basic user authentication system where users can choose to log in as an administrator or user, register as an administrator, or exit the program. Users enter their selections and credentials, which are then checked against the usernames and passwords found in the ADMINFILE and USERFILE files. Successful authentication redirects users to the admin or user interface, while invalid

credentials return an error message. The cycle continues until the user decides to exit the program or register as an administrator.

```
void loginpage() {
    struct IDCards ID;
    int choice;

   do {
        printf("\nEmployee Detect System\n");
        printf("1. Admin Login\n");
        printf("2. User Login\n");
        printf("3. Admin Register\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
            char username[20];
            char password[20];

        switch (choice) {
            case 1:
            printf("Enter your username: ");
            scanf("%s", username);
            printf("Enter your password: ");
            scanf("%s", password);

            FILE *adminfile = fopen(ADMINFILE, "r");
            if (adminfile != NULL) {
                while (fscanf(adminfile, "%s %s", ID.username, ID.password) == 2) {
                    if (strcmp(username, ID.username) == 0 && strcmp(password, ID.password) == 0) {
                        fclose(adminfile);
                        adminpage(ID.username);
                    }
                }
                printf("Invalid username or password!\n");
                fclose(adminfile);
            } else {
                printf("Error opening file for reading!\n");
            }

            break;
                break;
            case 2:

            printf("Enter your username: ");
            scanf("%s", username);
            printf("Enter your password: ");
            scanf("%s", password);
```

Figure 38. loginpage function

```
            FILE *userfile = fopen(USERFILE, "r");
            if (userfile != NULL) {
                while (fscanf(userfile, "%s %s", ID.username, ID.password) == 2) {
                    if (strcmp(username, ID.username) == 0 && strcmp(password, ID.password) == 0) {
                        fclose(userfile);
                        userpage(ID.username);
                    }
                }

                printf("Invalid username or password!\n");
                fclose(userfile);
            } else {
                printf("Error opening file for reading!\n");
            }
                break;
            case 3:
                printf("Exiting the program. Goodbye!\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }

    } while (choice != 3);


}
```

Figure 39. loginpage function

3.5. MAIN FUNCTION

The main function initiates the program by calling the loginpage function, which serves as a basic user authentication system. Users can log in as admins or regular users, register as admins, or choose to exit the program.

```
int main() {

    loginpage();

return 0;
}
```

Figure 33. Main function

ACTIVITY TABLE

Date	Activity	Summary
21.12.2023	Discussion	The project group met to decide how to proceed with our project. The project group discussed what kind of project the group wanted and the group agreed on a project. The project group decided to meet again later for the project.
21.12.2023	Planning	The project group planned on how to proceed and decided that our aim should be to create an algorithm and structure.
29.12.2023	Skelton code	The project group met on Discord to help each other.
30.12.2023	Distribution of the tasks	The project group met on Discord again and talked about how to proceed with the code. After a bit of brainstorming, the group decided how they wanted to go (what kind of variables to use and change, what are the redundant parts etc.) and the coding part was shared among the group.
02.01.2024	Meeting	The project group met to see if everyone had done their part so far. After seeing what was missing, the group discussed a bit more about what else could be added to the code.
05.01.2024	Meeting	The project group met to brainstorm about more functionalities in the system. After finding the additional scenarios, the code was distributed among the group again and the group proceeded with the coding.
07.01.2024	Finishing Meeting	The project group met for the final time and checked if everything was working. After checking everything was as it should be they decided to pile everything up for the report. A doc was created so every group member can upload their part and check how the report is going.

Table 1. Activity table

PROJECT EXPLANATION

1- INTRODUCTION

This project aims to develop a system with an efficient and user-friendly interface using C programming language.

2- PURPOSE

The main purpose of implementing an employee management system is to increase workforce productivity, explore strategies for talent engagement and retention, and reduce the administrative workload of HR professionals. Leveraging technology to increase efficiency not only helps control costs but also reduces compliance risks[1]. This project aims to create a system that provides the specified features.

3- GOALS

- Implement a secure user authentication system for both administrators and employees.
- Establish proper authorization mechanisms to control access levels and privileges based on user roles.
- Develop functionality to add new employees to the system, including capturing essential information such as name, surname, title, age, gender, contact details, and employment details.
- Provide the ability to remove employees from the system when necessary.
- Create a user-friendly interface for administrators to view and manage employee information.
- Implement features to display the list of employees, including their IDs, titles, names, surnames, ages, genders, locations, annual leave entitlements, and salaries.
- Enable employees to request annual leave through the system.
- Implement an approval mechanism for administrators to review and respond to annual leave requests.
- Develop a functionality allowing administrators to adjust the salary of employees.
- Integrate a survey module for employees to submit feedback or respond to surveys.
- Provide administrators with tools to create and manage surveys within the system.
- Implement file-handling mechanisms to store and retrieve employee data, annual leave requests, and survey responses.
- Ensure data persistence and reliability to prevent data loss.
- Design a user-friendly interface for both administrators and employees.

4- FEATURES

- **ADMIN FEATURES**

1. Add Employee:

- Collects information about a new employee, including ID, name, surname, title, age, gender, country, city, total annual leave days, and monthly salary, and creates a user account for the employee.
- Validates and categorizes the employee's title (CEO, CFO, Project Manager, Team Leader, Engineer, Intern).

2. Remove Employee:

- Removes an employee based on their ID.

3. Display Employees:

- Lists all employees along with their details, including ID, title, name, surname, age, gender, country, city, total annual leave days, and monthly salary.
- Highlights if an employee has an outstanding annual leave request.

4. Annual Day Requests:

- Displays a list of employees who have requested annual leave, along with their start and finish dates.
- Allows the admin to accept or reject leave requests.

5. Employee Salary Change:

- Allows the admin to change the salary of an employee.

- **User Features**

1. Annual Day Request:

- Allows employees to request annual leave by providing start and finish dates.

2. Survey:

- Enables employees to submit surveys by providing a title, subject, and message.

- **General Features**

1. Login System:

- Allows both admins and employees to log in using a username and password.
- Authenticates users based on the information stored in separate admin and user account files.

2. Data Storage:

- Stores employee information, user accounts, annual leave requests, and surveys in separate text files.
- Utilizes structures to organize and manage employee data efficiently.

3. **File Operations:**

- Performs file operations for reading, writing, and updating employee and user data.
- Implements a file copying function to update the main employee information file.

6- Limitations and Future Enhancements

Limitations

User Interface:

- The system currently relies on a command-line interface, which may not be user-friendly for non-technical users. Implementing a graphical user interface (GUI) could enhance the overall user experience.

Limited User Roles:

- The system currently supports only two user roles: admin and regular user. A more comprehensive system might include different levels of access and roles to cater to various organizational hierarchies.

Future Enhancements

Database Integration:

- Consider transitioning from file-based storage to a database management system (DBMS) for better data organization, retrieval, and security.

Web or Desktop GUI:

- Develop a graphical user interface (GUI) to make the system more user-friendly, especially for non-technical users.

CHALLENGES

Our biggest challenge was our lack of knowledge regarding our lack of coding practice in the C language. We have used our previous knowledge and understanding of algorithms from coding languages like Python or Java and applied it to this project as much as we could.

In the code section, there were functions that we could do very easily or that required us to struggle for days. We especially had a hard time with—-. Even though we pulled through in both of them at the end of the project, they took a lot more time than we had anticipated.

RESOURCES

1- "Employee Management and Employee Management Systems | ADP." Www.adp.com, 15 Jan. 2021, www.adp.com/resources/articles-and-insights/articles/e/employee-management.aspx#:~:text=The%20purpose%20of%20an%20employee%20management%20system%20is%20to%20help.

2- "C Programming Language." GeeksforGeeks, www.geeksforgeeks.org/c-programming-language/.

3- "C Tutorial." Www.w3schools.com, www.w3schools.com/c/.

DEMOLINK:
https://drive.google.com/drive/folders/12qXX1u0wJY-IG5o9ZB1O_kqWIr9Os8Zr?usp=sharing