**INDUSTRY Management System**



Session: 2021 – 2024

**Submitted by:**

M Abdullah

2021-CS-104

**Supervised by:**

Dr. Awais Hassan

**For Fulfillment of**

CS 161 Programming Fundamentals

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

**Project Description:**

**INDUSTRY Management System** is basically helpful for admin/manager in many different ways. It help in managing employees and also their work and pay. We can add or remove employee. We also made new admin or we can also change admin or delete admin. It is also helpful to check the stations or book the seat. All information is available on it. It is much helpful for us.

**Users of Application:**

­This is two user application.

1. **Admin**

* Admin can manage every feature of management system and can create login accounts for new admin and add staff information.

1. **Applicant**

* Applicant can be able to check the stations and their prices or book their seats.

**Functional Requirements:**

1. **Admin**

* Edit Director.
* Edit Writer.
* Edit Producer.
* Edit Actor.
* Edit Employee.
* View Dramas.
* Exit

1. **Applicant**

* View Profile.
* View Pay.
* View Dramas.
* Exit

**Wireframes:**

* **SIGN IN**

Graphical user interface, application, email

Description automatically generated

* **ADMIN**

A picture containing diagram

Description automatically generated

* **ADD CHARACTER**

A picture containing table

Description automatically generated

* **UPDATE EMPLOYEE INFO**

Graphical user interface, text, application

Description automatically generated

* **DELETE EMPLOYEE**
* **VIEW ALL EMPLOYEE**

Graphical user interface, text, application

Description automatically generated

* **VIEW ALL**

Graphical user interface, text, application

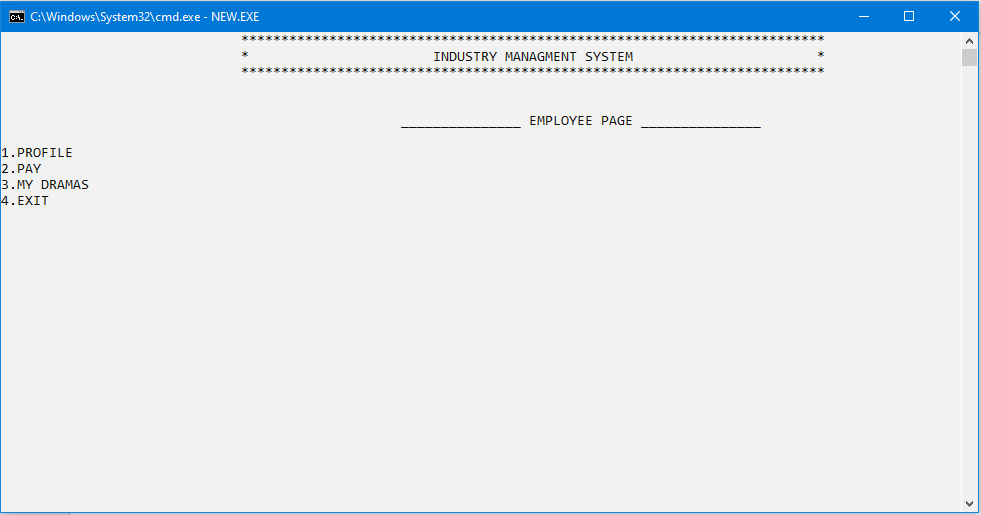
Description automatically generated

* **VIEW DRAMA**

A picture containing text

Description automatically generated

* **USER MENU**



* **VIEW DRAMAS**

A picture containing text

Description automatically generated

**Data Structure**

//----------------------ARRAYS

string user\_name[min], user\_pass[min], role[min], cnic[min], contact[min], address[min];

int pay[min];

string dramas[min], date[min];

string serial[min], sdate[min];

//----------------------ARRAYS END

**Functions Prototypes**

//----------------------FUNCCTIONS PROTOTYPE END

void header(HANDLE h);

void clear\_screen();

void gotoxy(int x, int y);

string main\_page(HANDLE h, string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], const int min, int user\_count, string u);

bool login();

string Admin(HANDLE h);

string director\_1(HANDLE h, string rol);

int add\_user(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], const int min, string dramas[], string date[], int user\_count, HANDLE h, string rol);

int remove\_user(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], const int min, string dramas[], string date[], int user\_count, HANDLE h, string rol);

int update\_user(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], const int min, string dramas[], string date[], int user\_count, HANDLE h, string rol);

int view\_user(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], const int min, string dramas[], string date[], int user\_count, HANDLE h, string rol);

void sort(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], string dramas[], string date[], int user\_count, HANDLE h);

int func(int pay[], int position, int user\_count);

void store(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], string dramas[], string date[], int user\_count);

int load(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], string dramas[], string date[]);

string parse(string record, int field);

int drama\_check(string drama[], string date[], int user\_count, string serial[], string sdate[]);

int show\_drama(string serial[], string sdate[], int dcount, HANDLE h);

void view\_drama(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], string dramas[], string date[], int user\_count, HANDLE h, string serial[], string sdate[], int idx);

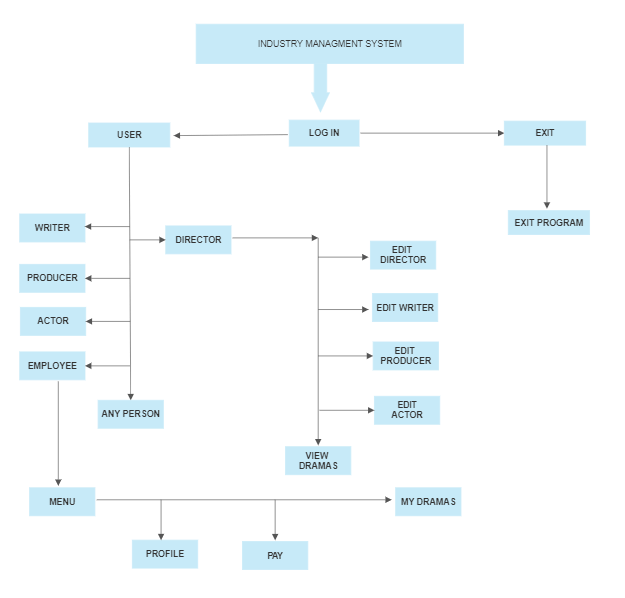
string user\_page(HANDLE h, string rol, string user\_name[], string role[], int pay[], int user\_count, string u);

void user\_role(HANDLE h, string rol, string user\_name[], string role[], string dramas[], int pay[], int user\_count, string u);

void user\_profile(HANDLE h, string rol, string user\_name[], string user\_pass[], string cnic[], string contact[], string address[], string dramas[], string date[], string role[], int pay[], int user\_count, string u);

void user\_pay(HANDLE h, string rol, string user\_name[], string role[], int pay[], int user\_count, string u);

**FLOW CHART**



**COMPLETE CODE**

#include <iostream>

#include <fstream>

#include <conio.h>

#include <string>

#include <windows.h>

using namespace std;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_PROTOTYPES\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

void header(HANDLE h);

void clear\_screen();

void gotoxy(int x, int y);

string main\_page(HANDLE h, string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], const int min, int user\_count, string u);

bool login();

string Admin(HANDLE h);

string director\_1(HANDLE h, string rol);

int add\_user(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], const int min, string dramas[], string date[], int user\_count, HANDLE h, string rol);

int remove\_user(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], const int min, string dramas[], string date[], int user\_count, HANDLE h, string rol);

int update\_user(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], const int min, string dramas[], string date[], int user\_count, HANDLE h, string rol);

int view\_user(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], const int min, string dramas[], string date[], int user\_count, HANDLE h, string rol);

void sort(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], string dramas[], string date[], int user\_count, HANDLE h);

int func(int pay[], int position, int user\_count);

void store(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], string dramas[], string date[], int user\_count);

int load(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], string dramas[], string date[]);

string parse(string record, int field);

int drama\_check(string drama[], string date[], int user\_count, string serial[], string sdate[]);

int show\_drama(string serial[], string sdate[], int dcount, HANDLE h);

void view\_drama(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], string dramas[], string date[], int user\_count, HANDLE h, string serial[], string sdate[], int idx);

string user\_page(HANDLE h, string rol, string user\_name[], string role[], int pay[], int user\_count, string u);

void user\_role(HANDLE h, string rol, string user\_name[], string role[], string dramas[], int pay[], int user\_count, string u);

void user\_profile(HANDLE h, string rol, string user\_name[], string user\_pass[], string cnic[], string contact[], string address[], string dramas[], string date[], string role[], int pay[], int user\_count, string u);

void user\_pay(HANDLE h, string rol, string user\_name[], string role[], int pay[], int user\_count, string u);

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_MAIN

main()

{

    HANDLE h = GetStdHandle(STD\_OUTPUT\_HANDLE);

    int user\_count;

    const int min = 100;

    string user\_name[min], user\_pass[min], role[min], cnic[min], contact[min], address[min];

    int pay[min];

    string dramas[min], date[min];

    string main;

    string serial[min], sdate[min];

    user\_count = load(user\_name, user\_pass, role, cnic, contact, address, pay, dramas, date);

    sort(user\_name, user\_pass, role, cnic, contact, address, pay, dramas, date, user\_count, h);

    // for (int i = 0; i < user\_count; i++)

    // {

    //     cout << user\_name[i] << " " << user\_pass[i] << " " << role[i] << " " << cnic[i] << " " << contact[i] << " " << address[i] << " " << pay[i] << " " << dramas[i] << " " << date[i] << endl;

    // }

    // clear\_screen();

    int dcount = drama\_check(dramas, date, user\_count, serial, sdate);

    string u;

    main = main\_page(h, user\_name, user\_pass, role, cnic, contact, address, pay, min, user\_count, u);

    if (main == "DIRECTOR")

    {

        string op = "1";

        while (op != "7")

        {

            string rol;

            op = Admin(h);

            if (op == "1")

            {

                string op1 = "2";

                while (op1 != "5")

                {

                    rol = "DIRECTOR";

                    op1 = director\_1(h, rol);

                    if (op1 == "1")

                    {

                        user\_count = add\_user(user\_name, user\_pass, role, cnic, contact, address, pay, min, dramas, date, user\_count, h, rol);

                    }

                    else if (op1 == "2")

                    {

                        user\_count = remove\_user(user\_name, user\_pass, role, cnic, contact, address, pay, min, dramas, date, user\_count, h, rol);

                    }

                    else if (op1 == "3")

                    {

                        user\_count = update\_user(user\_name, user\_pass, role, cnic, contact, address, pay, min, dramas, date, user\_count, h, rol);

                    }

                    else if (op1 == "4")

                    {

                        sort(user\_name, user\_pass, role, cnic, contact, address, pay, dramas, date, user\_count, h);

                        user\_count = view\_user(user\_name, user\_pass, role, cnic, contact, address, pay, min, dramas, date, user\_count, h, rol);

                    }

                    else if (op1 != "5")

                    {

                        cout << "ENTER THE VALID OPTION" << endl;

                        clear\_screen();

                    }

                    store(user\_name, user\_pass, role, cnic, contact, address, pay, dramas, date, user\_count);

                }

            }

            else if (op == "2")

            {

                string op1 = "2";

                while (op1 != "5")

                {

                    rol = "WRITER";

                    op1 = director\_1(h, rol);

                    if (op1 == "1")

                    {

                        user\_count = add\_user(user\_name, user\_pass, role, cnic, contact, address, pay, min, dramas, date, user\_count, h, rol);

                    }

                    else if (op1 == "2")

                    {

                        user\_count = remove\_user(user\_name, user\_pass, role, cnic, contact, address, pay, min, dramas, date, user\_count, h, rol);

                    }

                    else if (op1 == "3")

                    {

                        user\_count = update\_user(user\_name, user\_pass, role, cnic, contact, address, pay, min, dramas, date, user\_count, h, rol);

                    }

                    else if (op1 == "4")

                    {

                        sort(user\_name, user\_pass, role, cnic, contact, address, pay, dramas, date, user\_count, h);

                        user\_count = view\_user(user\_name, user\_pass, role, cnic, contact, address, pay, min, dramas, date, user\_count, h, rol);

                    }

                    else if (op1 != "5")

                    {

                        cout << "ENTER THE VALID OPTION" << endl;

                        clear\_screen();

                    }

                    store(user\_name, user\_pass, role, cnic, contact, address, pay, dramas, date, user\_count);

                }

            }

            else if (op == "3")

            {

                string op1 = "2";

                while (op1 != "5")

                {

                    rol = "PRODUCER";

                    op1 = director\_1(h, rol);

                    if (op1 == "1")

                    {

                        user\_count = add\_user(user\_name, user\_pass, role, cnic, contact, address, pay, min, dramas, date, user\_count, h, rol);

                    }

                    else if (op1 == "2")

                    {

                        user\_count = remove\_user(user\_name, user\_pass, role, cnic, contact, address, pay, min, dramas, date, user\_count, h, rol);

                    }

                    else if (op1 == "3")

                    {

                        user\_count = update\_user(user\_name, user\_pass, role, cnic, contact, address, pay, min, dramas, date, user\_count, h, rol);

                    }

                    else if (op1 == "4")

                    {

                        sort(user\_name, user\_pass, role, cnic, contact, address, pay, dramas, date, user\_count, h);

                        user\_count = view\_user(user\_name, user\_pass, role, cnic, contact, address, pay, min, dramas, date, user\_count, h, rol);

                    }

                    else if (op1 != "5")

                    {

                        cout << "ENTER THE VALID OPTION" << endl;

                        clear\_screen();

                    }

                    store(user\_name, user\_pass, role, cnic, contact, address, pay, dramas, date, user\_count);

                }

            }

            else if (op == "4")

            {

                string op1 = "2";

                while (op1 != "5")

                {

                    rol = "ACTOR";

                    op1 = director\_1(h, rol);

                    if (op1 == "1")

                    {

                        user\_count = add\_user(user\_name, user\_pass, role, cnic, contact, address, pay, min, dramas, date, user\_count, h, rol);

                    }

                    else if (op1 == "2")

                    {

                        user\_count = remove\_user(user\_name, user\_pass, role, cnic, contact, address, pay, min, dramas, date, user\_count, h, rol);

                    }

                    else if (op1 == "3")

                    {

                        user\_count = update\_user(user\_name, user\_pass, role, cnic, contact, address, pay, min, dramas, date, user\_count, h, rol);

                    }

                    else if (op1 == "4")

                    {

                        sort(user\_name, user\_pass, role, cnic, contact, address, pay, dramas, date, user\_count, h);

                        user\_count = view\_user(user\_name, user\_pass, role, cnic, contact, address, pay, min, dramas, date, user\_count, h, rol);

                    }

                    else if (op1 != "5")

                    {

                        cout << "ENTER THE VALID OPTION" << endl;

                        clear\_screen();

                    }

                    store(user\_name, user\_pass, role, cnic, contact, address, pay, dramas, date, user\_count);

                }

            }

            else if (op == "5")

            {

                string op1 = "2";

                while (op1 != "5")

                {

                    rol = "EMPLOYEE";

                    op1 = director\_1(h, rol);

                    if (op1 == "1")

                    {

                        user\_count = add\_user(user\_name, user\_pass, role, cnic, contact, address, pay, min, dramas, date, user\_count, h, rol);

                    }

                    else if (op1 == "2")

                    {

                        user\_count = remove\_user(user\_name, user\_pass, role, cnic, contact, address, pay, min, dramas, date, user\_count, h, rol);

                    }

                    else if (op1 == "3")

                    {

                        user\_count = update\_user(user\_name, user\_pass, role, cnic, contact, address, pay, min, dramas, date, user\_count, h, rol);

                    }

                    else if (op1 == "4")

                    {

                        sort(user\_name, user\_pass, role, cnic, contact, address, pay, dramas, date, user\_count, h);

                        user\_count = view\_user(user\_name, user\_pass, role, cnic, contact, address, pay, min, dramas, date, user\_count, h, rol);

                    }

                    else if (op1 != "5")

                    {

                        cout << "ENTER THE VALID OPTION" << endl;

                        clear\_screen();

                    }

                    store(user\_name, user\_pass, role, cnic, contact, address, pay, dramas, date, user\_count);

                }

            }

            else if (op == "6")

            {

                sort(user\_name, user\_pass, role, cnic, contact, address, pay, dramas, date, user\_count, h);

                int opt = show\_drama(serial, sdate, dcount, h);

                for (int x = 0; x < dcount; x++)

                {

                    if (serial[opt - 1] == dramas[x] && sdate[opt - 1] == date[x])

                    {

                        view\_drama(user\_name, user\_pass, role, cnic, contact, address, pay, dramas, date, user\_count, h, serial, sdate, x);

                    }

                }

            }

        }

    }

    if (main == "WRITER" || main == "PRODUCER" || main == "ACTOR" || main == "EMPLOYEE")

    {

        sort(user\_name, user\_pass, role, cnic, contact, address, pay, dramas, date, user\_count, h);

        string rol = main;

        string op2 = "1";

        while (op2 != "4")

        {

            op2 = user\_page(h, rol, user\_name, role, pay, user\_count, u);

            if (op2 == "1")

            {

                user\_profile(h, rol, user\_name, user\_pass, cnic, contact, address, dramas, date, role, pay, user\_count, u);

            }

            else if (op2 == "2")

            {

                user\_pay(h, rol, user\_name, role, pay, user\_count, u);

            }

            else if (op2 == "3")

            {

                user\_role(h, rol, user\_name, role, dramas, pay, user\_count, u);

            }

            else if (op2 != "4")

            {

                cout << "ENTER THE VALID OPTION" << endl;

                clear\_screen();

            }

        }

    }

    if (main == "USER")

    {

        sort(user\_name, user\_pass, role, cnic, contact, address, pay, dramas, date, user\_count, h);

        int opt = show\_drama(serial, sdate, dcount, h);

        for (int x = 0; x < dcount; x++)

        {

            if (serial[opt - 1] == dramas[x] && sdate[opt - 1] == date[x])

            {

                view\_drama(user\_name, user\_pass, role, cnic, contact, address, pay, dramas, date, user\_count, h, serial, sdate, x);

            }

        }

    }

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_MAIN PAGE

string main\_page(HANDLE h, string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], const int min, int user\_count, string u)

{

    string name;

    string pass;

    bool flag = false;

    while (flag == false)

    {

        header(h);

        gotoxy(50, 5);

        cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ SIGN IN \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl

             << endl;

        gotoxy(30, 7);

        cout << "USER NAME ";

        gotoxy(30, 9);

        cout << "PASSWARD";

        gotoxy(50, 7);

        getline(cin, name);

        u = name;

        gotoxy(50, 9);

        getline(cin, pass);

        for (int x = 0; x < user\_count; x++)

        {

            if (name == user\_name[x] && pass == user\_pass[x])

            {

                flag = true;

                return role[x];

            }

        }

        header(h);

        gotoxy(40, 5);

        cout << "ENTER THE VALID USERNAME OR PASSWARD" << endl;

        clear\_screen();

    }

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_DIRECTOR PAGE

string Admin(HANDLE h)

{

    string op = "0";

    header(h);

    gotoxy(40, 6);

    cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ADMIN MENU \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl

         << endl;

    cout << "1. DIRECTOR    " << endl;

    cout << "2. WRITER      " << endl;

    cout << "3. PRODUCER    " << endl;

    cout << "4. ACTOR       " << endl;

    cout << "5. EMPLOYEE    " << endl;

    cout << "6. DRAMAS    " << endl;

    cout << "7. EXIT    " << endl;

    cin >> op;

    return op;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ADMIN=====>>DIRECTOR

string director\_1(HANDLE h, string rol)

{

    header(h);

    string op1 = "0";

    gotoxy(40, 6);

    cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << rol << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl

         << endl;

    cout << "1. ADD " << rol << endl;

    cout << "2. REMOVE  " << rol << endl;

    cout << "3. UPDATE " << rol << endl;

    cout << "4. VIEW ALL " << rol << endl;

    cout << "5. EXIT " << endl;

    cin >> op1;

    return op1;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ADD USER

int add\_user(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], const int min, string dramas[], string date[], int user\_count, HANDLE h, string rol)

{

    header(h);

    gotoxy(50, 6);

    cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ADD " << rol << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl

         << endl;

    gotoxy(0, 7);

    cout << "USERNAME :";

    gotoxy(0, 8);

    cout << "PASSWARD :";

    gotoxy(0, 9);

    cout << "ROLE :";

    role[user\_count] = rol;

    gotoxy(0, 10);

    cout << "CNIC :";

    gotoxy(0, 11);

    cout << "CONTACT NO / EMAIL :";

    gotoxy(0, 12);

    cout << "ADRESS :";

    gotoxy(0, 13);

    cout << "PAY :";

    gotoxy(0, 14);

    cout << "DRAMA :";

    gotoxy(0, 15);

    cout << "RELEASING DATE :";

    gotoxy(30, 7);

    cin.ignore();

    getline(cin, user\_name[user\_count]);

    gotoxy(30, 8);

    getline(cin, user\_pass[user\_count]);

    role[user\_count] = rol;

    gotoxy(30, 9);

    cout << role[user\_count];

    gotoxy(30, 10);

    getline(cin, cnic[user\_count]);

    gotoxy(30, 11);

    getline(cin, contact[user\_count]);

    gotoxy(30, 12);

    getline(cin, address[user\_count]);

    gotoxy(30, 13);

    cin >> pay[user\_count];

    cin.ignore();

    gotoxy(30, 14);

    getline(cin, dramas[user\_count]);

    gotoxy(30, 15);

    getline(cin, date[user\_count]);

    gotoxy(20, 18);

    cout << role[user\_count] << " IS ADDED" << endl;

    user\_count++;

    clear\_screen();

    return user\_count;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_REMOVE DIRECTOR

int remove\_user(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], const int min, string dramas[], string date[], int user\_count, HANDLE h, string rol)

{

    string nam;

    header(h);

    gotoxy(40, 5);

    cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ REMOVE " << rol << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl

         << endl;

    cin.ignore();

    gotoxy(0, 7);

    cout << "USERNAME :";

    gotoxy(30, 7);

    getline(cin, nam);

    char y = ' ';

    for (int x = 0; x < min; x++)

    {

        if (nam == user\_name[x] && role[x] == rol)

        {

            gotoxy(15, 10);

            cout << "PRESS Y TO DELETE THIS USER ";

            cin >> y;

            if (y == 'y')

            {

                for (int i = x; i < user\_count; i++)

                {

                    user\_name[i] = user\_name[i + 1];

                    user\_pass[i] = user\_pass[i + 1];

                    role[i] = role[i + 1];

                    cnic[i] = cnic[i + 1];

                    contact[i] = contact[i + 1];

                    address[i] = address[i + 1];

                    pay[i] = pay[i + 1];

                    dramas[i] = dramas[i + 1];

                    date[i] = date[i + 1];

                }

                user\_count--;

            }

        }

        else

        {

            gotoxy(0, 10);

            cout << nam << " IS NOT A " << rol << endl;

            cout << "ENTER THE VALID USERNAME.......... ";

        }

    }

    if (y == 'y')

    {

        gotoxy(0, 10);

        cout << nam << " IS REMOVED.......";

    }

    clear\_screen();

    return user\_count;

}

int update\_user(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], const int min, string dramas[], string date[], int user\_count, HANDLE h, string rol)

{

    string nam;

    header(h);

    gotoxy(40, 5);

    cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ UPDATE " << rol << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl

         << endl;

    cin.ignore();

    gotoxy(0, 7);

    cout << "USERNAME :";

    gotoxy(30, 7);

    getline(cin, nam);

    char y = ' ';

    for (int x = 0; x < user\_count; x++)

    {

        if (nam == user\_name[x] && role[x] == rol)

        {

            gotoxy(15, 10);

            cout << "PRESS Y TO DELETE THIS USER ";

            cin >> y;

            if (y == 'y')

            {

                header(h);

                gotoxy(40, 5);

                cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ UPDATE " << rol << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

                gotoxy(0, 7);

                cout << "UPDATE USERNAME :";

                gotoxy(0, 8);

                cout << "UPDATE PASSWARD :";

                gotoxy(0, 9);

                cout << "UPDATE ROLE :";

                gotoxy(0, 10);

                cout << "UPDATE CNIC :";

                gotoxy(0, 11);

                cout << "UPDATE CONTACT NO / EMAIL :";

                gotoxy(0, 12);

                cout << "UPDATE ADRESS :";

                gotoxy(0, 13);

                cout << "UPDATE PAY :";

                gotoxy(0, 14);

                cout << "UPDATE DRAMA :";

                gotoxy(0, 15);

                cout << "UPDATE RELEASING DATE :";

                cin.ignore();

                gotoxy(30, 7);

                cin.ignore();

                getline(cin, user\_name[x]);

                gotoxy(30, 8);

                getline(cin, user\_pass[x]);

                gotoxy(30, 9);

                getline(cin, role[x]);

                gotoxy(30, 10);

                getline(cin, cnic[x]);

                gotoxy(30, 11);

                getline(cin, contact[x]);

                gotoxy(30, 12);

                getline(cin, address[x]);

                gotoxy(30, 13);

                cin >> pay[x];

                cin.ignore();

                gotoxy(30, 14);

                getline(cin, dramas[x]);

                gotoxy(30, 15);

                getline(cin, date[x]);

            }

        }

        else

        {

            gotoxy(0, 10);

            cout << nam << " IS NOT A " << rol << endl;

            cout << "ENTER THE VALID USERNAME.......... ";

        }

    }

    if (y == 'y')

    {

        gotoxy(0, 10);

        cout << nam << " IS UPDATED.......";

    }

    clear\_screen();

    return user\_count;

}

int view\_user(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], const int min, string dramas[1000], string date[1000], int user\_count, HANDLE h, string rol)

{

    header(h);

    gotoxy(40, 5);

    cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ VIEW ALL " << rol << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

    gotoxy(0, 7);

    cout << "NAME ";

    gotoxy(30, 7);

    cout << "ROLE ";

    gotoxy(40, 7);

    cout << "CNIC ";

    gotoxy(60, 7);

    cout << "CONTACT ";

    gotoxy(80, 7);

    cout << "ADDRESS ";

    gotoxy(100, 7);

    cout << "PAY ";

    gotoxy(110, 7);

    cout << "DRAMA ";

    gotoxy(120, 7);

    cout << "RELEASE DATE ";

    int y = 9;

    for (int x = 0; x < user\_count; x++)

    {

        if (role[x] == rol)

        {

            gotoxy(0, y);

            cout << user\_name[x];

            gotoxy(30, y);

            cout << role[x];

            gotoxy(40, y);

            cout << cnic[x];

            gotoxy(60, y);

            cout << contact[x];

            gotoxy(80, y);

            cout << address[x];

            gotoxy(100, y);

            cout << pay[x];

            gotoxy(110, y);

            cout << dramas[x];

            gotoxy(120, y);

            cout << date[x];

            y++;

        }

    }

    clear\_screen();

    return user\_count;

}

//-------------------------WRITER

string writer\_1(HANDLE h, string rol)

{

    header(h);

    string op1 = "0";

    gotoxy(40, 6);

    cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << rol << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl

         << endl;

    cout << "1. ADD " << rol << endl;

    cout << "2. REMOVE  " << rol << endl;

    cout << "3. UPDATE " << rol << endl;

    cout << "4. VIEW ALL " << rol << endl;

    cout << "5. EXIT " << endl;

    cin >> op1;

    return op1;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_HEADER

void header(HANDLE h)

{

    system("cls");

    gotoxy(30, 0);

    cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

    gotoxy(30, 1);

    cout << "\*                       INDUSTRY MANAGMENT SYSTEM                       \*" << endl;

    gotoxy(30, 2);

    cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_GOTOXY FUNCTION

void gotoxy(int x, int y)

{

    COORD coordinates; // coordinates is declared as COORD

    coordinates.X = x; // defining x-axis

    coordinates.Y = y; // defining  y-axis

    SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), coordinates);

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_CLEAR SCREEN

void clear\_screen()

{

    cout << endl

         << " PRESS ANY KET TO CONTINUE ..... " << endl;

    getch();

    system("cls");

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_SORTING

void sort(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], string dramas[], string date[], int user\_count, HANDLE h)

{

    int position = 0;

    int largest = pay[position];

    for (int i = 0; i < user\_count; i++)

    {

        position=i;

        int large = func(pay, position, user\_count);

        int temp = pay[i];

        pay[i] = pay[large];

        pay[large] = temp;

        string temp\_1;

        temp\_1= user\_name[i];

        user\_name[i] = user\_name[large];

        user\_name[large] = temp\_1;

        temp\_1 = role[i];

        role[i] = role[large];

        role[large] = temp\_1;

        temp\_1 = cnic[i];

        cnic[i] = cnic[large];

        cnic[large] = temp\_1;

        temp\_1 = contact[i];

        contact[i] = contact[large];

        contact[large] = temp\_1;

        temp\_1 = address[i];

        address[i] = address[large];

        address[large] = temp\_1;

        temp\_1 = dramas[i];

        dramas[i] = dramas[large];

        dramas[large] = temp\_1;

        temp\_1 = date[i];

        date[i] = date[large];

        date[large] = temp\_1;

        // for (int i = 0; i < user\_count; i++)

        // {

        //     cout << user\_name[i] << " " << user\_pass[i] << " " << role[i] << " " << cnic[i] << " " << contact[i] << " " << address[i] << " " << pay[i] << " " << dramas[i] << " " << date[i] << endl;

        // }

        // clear\_screen();

    }

}

int func(int pay[], int position, int user\_count)

{

    int largest = pay[position];

    int idx=position;

    for (int i = position; i < user\_count; i++)

    {

        if (largest <= pay[i])

        {

            largest = pay[i];

            idx = i;

        }

    }

    return idx;

}

void store(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], string dramas[], string date[], int user\_count)

{

    fstream file;

    file.open("industry.txt", ios::out);

    for (int i = 0; i < user\_count - 1; i++)

    {

        file << user\_name[i] << "," << user\_pass[i] << "," << role[i] << "," << cnic[i] << "," << contact[i] << "," << address[i] << "," << pay[i] << "," << dramas[i] << "," << date[i] << endl;

    }

    file << user\_name[user\_count - 1] << "," << user\_pass[user\_count - 1] << "," << role[user\_count - 1] << "," << cnic[user\_count - 1] << "," << contact[user\_count - 1] << "," << address[user\_count - 1] << "," << pay[user\_count - 1] << "," << dramas[user\_count - 1] << "," << date[user\_count - 1];

    file.close();

}

int load(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], string dramas[], string date[])

{

    fstream file;

    string record;

    int idx = 0;

    file.open("industry.txt", ios::in);

    while (!(file.eof()))

    {

        getline(file, record);

        user\_name[idx] = parse(record, 1);

        user\_pass[idx] = parse(record, 2);

        role[idx] = parse(record, 3);

        cnic[idx] = parse(record, 4);

        contact[idx] = parse(record, 5);

        address[idx] = parse(record, 6);

        pay[idx] = stoi(parse(record, 7));

        dramas[idx] = parse(record, 8);

        date[idx] = parse(record, 9);

        idx++;

    }

    return idx;

}

string parse(string record, int field)

{

    string term;

    int comma = 1;

    for (int i = 0; i < record.length(); i++)

    {

        if (record[i] == ',')

        {

            comma++;

        }

        else if (comma == field)

        {

            term = term + record[i];

        }

    }

    return term;

}

int drama\_check(string drama[], string date[], int user\_count, string serial[], string sdate[])

{

    int idx = 0;

    for (int i = 0; i < user\_count; i++)

    {

        int count = 0;

        for (int x = 0; x < user\_count; x++)

        {

            if (drama[i] == serial[x])

            {

                count++;

                if (count >= 1)

                {

                    break;

                }

            }

        }

        if (count <= 0)

        {

            serial[idx] = drama[i];

            sdate[idx] = date[i];

            idx++;

        }

    }

    return idx;

}

int show\_drama(string serial[], string sdate[], int dcount, HANDLE h)

{

    header(h);

    gotoxy(50, 5);

    cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ DRAMAS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl

         << endl;

    gotoxy(30, 7);

    cout << "DRAMA ";

    gotoxy(50, 7);

    cout << "RELEASE DATE";

    int y = 9;

    int x = 1;

    for (int i = 0; i < dcount; i++)

    {

        gotoxy(20, y);

        cout << x;

        gotoxy(30, y);

        cout << serial[i];

        gotoxy(50, y);

        cout << sdate[i];

        y++;

        x++;

    }

    int point;

    y++;

    gotoxy(30, y);

    cout << "ENTER THE NUMBER OF DRAMA : ";

    gotoxy(60, y);

    cin >> point;

    return point;

}

void view\_drama(string user\_name[], string user\_pass[], string role[], string cnic[], string contact[], string address[], int pay[], string dramas[], string date[], int user\_count, HANDLE h, string serial[], string sdate[], int idx)

{

    for (int i = 0; i < user\_count; i++)

    {

        header(h);

        gotoxy(50, 5);

        cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ DRAMA DETAILS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl

             << endl;

        gotoxy(30, 7);

        cout << "DRAMA :";

        gotoxy(30, 8);

        cout << "RELEASE DATE :";

        gotoxy(60, 7);

        cout << serial[idx];

        gotoxy(60, 8);

        cout << sdate[idx];

        int d = 0, p = 0, w = 0, a = 0;

        for (int x = 0; x < user\_count; x++)

        {

            if (dramas[x] == serial[idx] && date[x] == sdate[idx])

            {

                gotoxy(35, 9);

                cout << "DIRECTOR";

                if (role[x] == "DIRECTOR")

                {

                    gotoxy(45, 9);

                    cout << user\_name[x];

                    d++;

                }

                gotoxy(35, 10);

                cout << "WRITER";

                if (role[x] == "WRITER")

                {

                    gotoxy(45, 10);

                    cout << user\_name[x];

                    w++;

                }

                gotoxy(35, 11);

                cout << "PRODUCER";

                if (role[x] == "PRODUCER")

                {

                    gotoxy(45, 11);

                    cout << user\_name[x];

                    p++;

                }

                gotoxy(35, 12);

                cout << "ACTOR";

                if (role[x] == "ACTOR")

                {

                    gotoxy(45, 12);

                    cout << user\_name[x];

                    a++;

                }

            }

        }

        if (d == 0)

        {

            gotoxy(45, 9);

            cout << "NEED DIRECTOR";

        }

        if (w == 0)

        {

            gotoxy(45, 10);

            cout << "NEED WRITER";

        }

        if (p == 0)

        {

            gotoxy(45, 11);

            cout << "NEED PRODUCER";

        }

        if (a == 0)

        {

            gotoxy(45, 12);

            cout << "NEED ACTORS";

        }

    }

    gotoxy(35, 14);

    clear\_screen();

}

string user\_page(HANDLE h, string rol, string user\_name[], string role[], int pay[], int user\_count, string u)

{

    header(h);

    string op = "0";

    gotoxy(50, 5);

    cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ " << rol << " PAGE \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl

         << endl;

    gotoxy(0, 7);

    cout << "1.PROFILE ";

    gotoxy(0, 8);

    cout << "2.PAY";

    gotoxy(0, 9);

    cout << "3.MY DRAMAS";

    gotoxy(0, 10);

    cout << "4.EXIT";

    gotoxy(0, 11);

    cin >> op;

    return op;

}

void user\_profile(HANDLE h, string rol, string user\_name[], string user\_pass[], string cnic[], string contact[], string address[], string dramas[], string date[], string role[], int pay[], int user\_count, string u)

{

    header(h);

    gotoxy(40, 5);

    cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ " << rol << " PROFILE \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

    gotoxy(0, 7);

    cout << "NAME ";

    gotoxy(20, 7);

    cout << "PASSWARD ";

    gotoxy(30, 7);

    cout << "ROLE ";

    gotoxy(40, 7);

    cout << "CNIC ";

    gotoxy(60, 7);

    cout << "CONTACT ";

    gotoxy(80, 7);

    cout << "ADDRESS ";

    gotoxy(100, 7);

    cout << "PAY ";

    gotoxy(110, 7);

    cout << "DRAMA ";

    gotoxy(120, 7);

    cout << "RELEASE DATE ";

    int y = 9;

    for (int x = 0; x < user\_count; x++)

    {

        if (role[x] == rol)

        {

            gotoxy(0, y);

            cout << user\_name[x];

            gotoxy(20, y);

            cout << user\_pass[x];

            gotoxy(30, y);

            cout << role[x];

            gotoxy(40, y);

            cout << cnic[x];

            gotoxy(60, y);

            cout << contact[x];

            gotoxy(80, y);

            cout << address[x];

            gotoxy(100, y);

            cout << pay[x];

            gotoxy(110, y);

            cout << dramas[x];

            gotoxy(120, y);

            cout << date[x];

            y++;

        }

    }

    clear\_screen();

}

void user\_pay(HANDLE h, string rol, string user\_name[], string role[], int pay[], int user\_count, string u)

{

    header(h);

    gotoxy(40, 5);

    cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ " << rol << " PAY \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

    gotoxy(0, 7);

    cout << "NAME ";

    gotoxy(20, 7);

    cout << "ROLE ";

    gotoxy(40, 7);

    cout << "PAY ";

    gotoxy(60, 7);

    for (int x = 0; x < user\_count; x++)

    {

        if (rol == role[x])

        {

            gotoxy(0, 9);

            cout << user\_name[x];

            gotoxy(20, 9);

            cout << role[x];

            gotoxy(40, 9);

            cout << pay[x];

            gotoxy(60, 9);

        }

    }

    clear\_screen();

}

void user\_role(HANDLE h, string rol, string user\_name[], string role[], string dramas[], int pay[], int user\_count, string u)

{

    header(h);

    gotoxy(40, 5);

    cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ " << rol << " ROLE \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

    gotoxy(0, 7);

    cout << "NAME ";

    gotoxy(20, 7);

    cout << "ROLE ";

    gotoxy(40, 7);

    cout << "DRAMA ";

    gotoxy(60, 7);

    for (int x = 0; x < user\_count; x++)

    {

        if (rol == role[x])

        {

            gotoxy(0, 9);

            cout << user\_name[x];

            gotoxy(20, 9);

            cout << role[x];

            gotoxy(40, 9);

            cout << dramas[x];

            gotoxy(60, 9);

        }

    }

    clear\_screen();

}