**Game studio management system**



**Session 2023 – 2027**

**Submitted by:**

Muhammad Omer (2023 – CS - 68)

**Supervised by:**

Miss Maida

Course: CSC-102 Programming Fundamentals

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

Short Description of your project

I want to create a game studio management system. It can be used to manage a game studio. You can use it to give tasks to your employees and the employees can use it to keep track of their deadlines and workload

Users of Application:

The program will have three users:

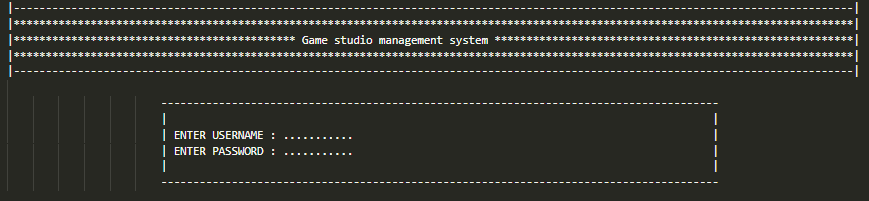
1. Admin
2. Programmer
3. Game Artist

Functional Requirements:

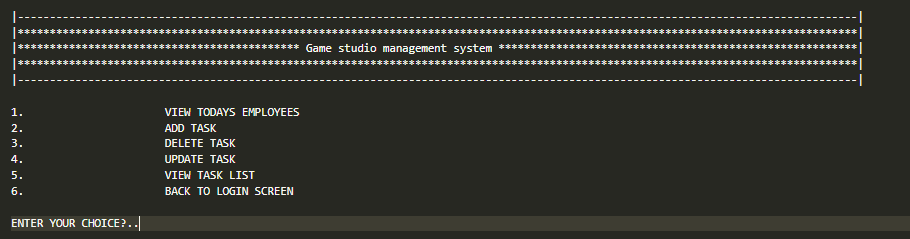
|  |  |
| --- | --- |
| **User** | **task** |
| Admin | Add tasks |
| Admin | Remove tasks |
| Admin | Preview task completion |
| Admin | View employee attendance |
| Artist | Clock in |
| Artist | Clock out |
| Artist | View given tasks |
| Artist | Mark task as done |
| Programmer | Clock in |
| Programmer | Clock out |
| Programmer | View given tasks |
| Programmer | Mark task as done |

WireFrames:

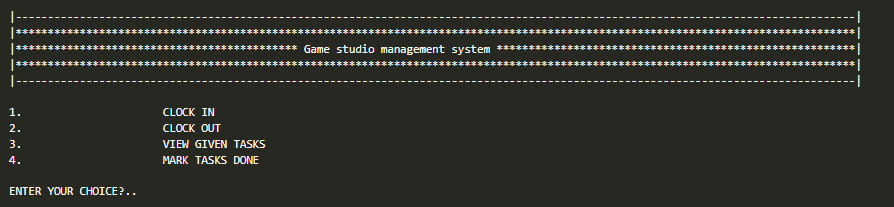
Login screen:



Admin screen:



Employee screen



**Function Prototypes**

//general use functions

*int* ascii\_to\_int(string *x*);

string get\_string\_input();

*void* gotoxy(*int* *x*, *int* *y*);

//program functions

*void* ready(string *x*[],string *y*[],string *z*[],string *employee\_id*,*bool* *taskdone\_pr*[], string *tasklist\_pr*[]);

*void* printHeader();

//login functions

*void* loginMenu(string *username\_database*[],string *password\_database*[],string *user\_perm*[],string *employee\_id*);

string loginInput(string *username\_database*[],string *password\_database*[],string *user\_perm*[],string *employee\_id*);

string sign\_in(string *x*[],string *y*[],string *z*[],string *employee\_id*);

*void* sign\_up(string *x*[],string *y*[],string *z*[],string *employee\_id*);

*void*invalid\_creditionals(string*username\_database*[],string *password\_database*[],string *user\_perm*[],string *employee\_id*);

//user front page

*void* admin\_page(*int* *task\_num*,*bool* *taskdone\_pr*[20],string *tasklist\_pr*[20],*int* *taskid\_pr*);

*void* employee\_page(*int* *task\_num*,*bool* *taskdone\_pr*[20],string *tasklist\_pr*[20],*int* *taskid\_pr*);

//tasklist

*void* add\_tasks(*int* *task\_num*,*bool* *taskdone\_pr*[20],string *tasklist\_pr*[20],*int* *taskid\_pr*);

*void* view\_task(*int* *task\_num*,*bool* *taskdone\_pr*[20],string *tasklist\_pr*[20],*int* *taskid\_pr*);

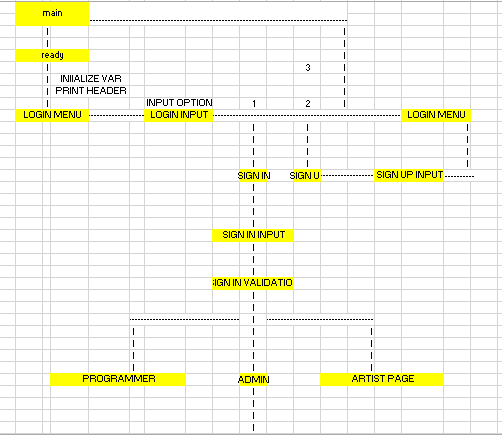
//validation functions

*void* validation\_fail();

*bool* login\_Input\_validation(string *option*);

*bool* sign\_up\_validation(string *option*);

**Functions Working Flow**



**Complete Code of the Business Application**

#include <iostream>

#include <conio.h>

#include <windows.h>

#include <string>

using *namespace* std;

//gobal variables

string state;

//general use functions

*int* ascii\_to\_int(string *x*);

string get\_string\_input();

*void* gotoxy(*int* *x*, *int* *y*);

//program functions

*void* ready(string *x*[],string *y*[],string *z*[],string *employee\_id*,*bool* *taskdone\_pr*[], string *tasklist\_pr*[]);

*void* printHeader();

//login functions

*void* loginMenu(string *username\_database*[],string *password\_database*[],string *user\_perm*[],string *employee\_id*);

string loginInput(string *username\_database*[],string *password\_database*[],string *user\_perm*[],string *employee\_id*);

string sign\_in(string *x*[],string *y*[],string *z*[],string *employee\_id*);

*void* sign\_up(string *x*[],string *y*[],string *z*[],string *employee\_id*);

*void* invalid\_creditionals(string *username\_database*[],string *password\_database*[],string *user\_perm*[],string *employee\_id*);

//user front page

*void* admin\_page(*int* *task\_num*,*bool* *taskdone\_pr*[20],string *tasklist\_pr*[20],*int* *taskid\_pr*);

*void* employee\_page(*int* *task\_num*,*bool* *taskdone\_pr*[20],string *tasklist\_pr*[20],*int* *taskid\_pr*);

//tasklist

*void* add\_tasks(*int* *task\_num*,*bool* *taskdone\_pr*[20],string *tasklist\_pr*[20],*int* *taskid\_pr*);

*void* view\_task(*int* *task\_num*,*bool* *taskdone\_pr*[20],string *tasklist\_pr*[20],*int* *taskid\_pr*);

*void* mark\_task\_done(*int* *task\_num*, *bool* *taskdone\_pr*[20], string *tasklist\_pr*[20], *int* *taskid\_pr*);

//validation functions

*void* validation\_fail();

*bool* login\_Input\_validation(string *option*);

*bool* sign\_up\_validation(string *option*);

*int* main()

{

    system ( "cls" );

    printHeader();

    cout << endl;

    cout <<"press any key to continue :-";

    getch ();

    //sign in creditionals

    string employee\_id;

    string username\_database[10];

    string password\_database[10];

    string user\_perm[10];

    //tasklist var

*int* task\_num = 0 ;

*bool* taskdone\_pr[20];

    string tasklist\_pr[20];

*int* taskid\_pr = 0;

    ready(username\_database,password\_database,user\_perm,employee\_id, taskdone\_pr, tasklist\_pr);     //calling ready to initialize variables

    loginMenu(username\_database,password\_database,user\_perm,employee\_id);             //login menu redirects to required login menu

    if (state == "admin")

    {

        admin\_page(task\_num,taskdone\_pr,tasklist\_pr,taskid\_pr);

    }

    else if (state == "employee")

    {

        employee\_page(task\_num,taskdone\_pr,tasklist\_pr,taskid\_pr);

    }

}

//general use functions

*int* ascii\_to\_int(string *convert\_to\_int*)

{

    string temp\_string;

*int* temp\_int;

    temp\_string = *convert\_to\_int*;

    temp\_int = stoi(temp\_string);

    return temp\_int;

}

string get\_string\_input()

{

    string temp\_string;

    cin.ignore();

    getline(cin,temp\_string);

    return temp\_string;

}

*void* gotoxy(*int* *x*, *int* *y*)

{

  static HANDLE h = NULL;

  if(!h)

    h = GetStdHandle(STD\_OUTPUT\_HANDLE);

  COORD c = { *x*, *y* };

  SetConsoleCursorPosition(h,c);

}

//headers

*void* printHeader()

{

cout << "|-----------------------------------------------------------------------------------------------------------------------------------|" <<endl;

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

cout << "|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Game studio management system \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|" <<endl;

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

cout << "|-----------------------------------------------------------------------------------------------------------------------------------|" <<endl;

}

//ready

*void* ready(string *username\_database*[],string *password\_database*[],string *user\_perm*[],string *employee\_id*,*bool* *taskdone\_pr*[],string *tasklist\_pr*[])

{

    for (*int* x =0 ;x < 10 ;x++)

    {

*username\_database*[x] = "0";

*password\_database*[x] = "0";

    }

    for (*int* x =0 ;x < 20 ;x++)

    {

*taskdone\_pr*[x] = false;

    }

*int* x=0;

    for(*int* x =1;x<=20;x++)

    {

*taskdone\_pr*[x] = false;

    }

*tasklist\_pr*[1] = "program";

*tasklist\_pr*[2] = "draw";

*tasklist\_pr*[3] = "manage";

*username\_database*[0] = "admin";

*password\_database*[0] = "admin";

*user\_perm*[0] = "admin";

*employee\_id* = "0";

    system ( "cls" );

}

//login functions

*void* loginMenu(string *username\_database*[],string *password\_database*[],string *user\_perm*[],string *employee\_id*)

{

    string option;

    option = loginInput(*username\_database*,*password\_database*,*user\_perm*,*employee\_id*);

    if (option == "1")

    {

        sign\_in(*username\_database*,*password\_database*,*user\_perm*,*employee\_id*);

    }

    else if (option =="2")

    {

        sign\_up(*username\_database*,*password\_database*,*user\_perm*,*employee\_id*);

    }

    else if (option =="3")

    {

        main();

    }

    else

    {

        validation\_fail();

        loginMenu(*username\_database*,*password\_database*,*user\_perm*,*employee\_id*);

    }

}

string loginInput(string *username\_database*[],string *password\_database*[],string *user\_perm*[],string *employee\_id*)

{

    string option;

    system ( "cls" );

    printHeader();

    std::cout << endl;

    std::cout << endl;

    std::cout << "1:) sign in"<<endl;

    std::cout << "2:) sign up"<<endl;

    std::cout << "3:) exit"<<endl;

    std::cout << endl;

    std::cout<< "Select option :-";

    cin >> option;

    if (login\_Input\_validation(option))

    {

        return option ;

    }

    else

    {

        validation\_fail();

        loginMenu(*username\_database*,*password\_database*,*user\_perm*,*employee\_id*);

    }

}

string sign\_in(string *username\_database*[],string *password\_database*[],string *user\_perm*[],string *employee\_id*)

{

    string entered\_username,entered\_password;

*int* entered\_employee\_id;

    string option;

    system ( "cls" );

    printHeader();

    cout << endl;

    cout << "ENTER EMPLOYEE ID :- ";

    cin >> *employee\_id*;

    entered\_employee\_id = ascii\_to\_int(*employee\_id*);

    cout << "ENTER USERNAME :- ";

    cin >> entered\_username;

    cout << "ENTER PASSWORD :- ";

    cin >> entered\_password;

    cout <<endl;

    cout << "press any key to continue";

    getch();

    if ((entered\_username == *username\_database*[entered\_employee\_id]) && (entered\_password == *password\_database*[entered\_employee\_id]))

    {

        if (*user\_perm*[entered\_employee\_id]=="admin")

            state ="admin";

        if ((*user\_perm*[entered\_employee\_id]=="programmar") || (*user\_perm*[entered\_employee\_id]=="artist"))

            state="employee";

    }

    else

    {

        invalid\_creditionals(*username\_database*,*password\_database*,*user\_perm*,*employee\_id*);

    }

}

*void* sign\_up(string *username\_database*[],string *password\_database*[],string *user\_perm*[],string *employee\_id*)

{

*int* temp\_int;                                                               //used to store conversion from ascii to int

    system ( "cls" );

    printHeader();

    cout << endl;

    cout << "ENTER EMPLOYEE ID :- ";

    cin >> *employee\_id*;

    temp\_int = ascii\_to\_int(*employee\_id*);

    cout << "ENTER USERNAME :- ";

    cin >> *username\_database*[temp\_int];

    cout << "ENTER PASSWORD :- ";

    cin >> *password\_database*[temp\_int];

    cout << "admin , programmar or artist ? ";

    cin >> *user\_perm*[temp\_int];

    string option = *user\_perm*[temp\_int];

    if(sign\_up\_validation(option))

    {

        loginMenu(*username\_database*,*password\_database*,*user\_perm*,*employee\_id*);

    }

    else

    {

        validation\_fail();

        sign\_up(*username\_database*,*password\_database*,*user\_perm*,*employee\_id*);

    }

}

*void* invalid\_creditionals(string *username\_database*[],string *password\_database*[],string *user\_perm*[],string *employee\_id*)

{

    string option;

    cout <<endl;

    cout <<"INVALID CREDITIONALS (press y to retry or press n to go back to login screen)";

    cin >> option;

    if (option == "y")

        sign\_in(*username\_database*,*password\_database*,*user\_perm*,*employee\_id*);

    else if (option == "n")

        loginMenu(*username\_database*,*password\_database*,*user\_perm*,*employee\_id*);

    else

    {

        validation\_fail();

        invalid\_creditionals(*username\_database*,*password\_database*,*user\_perm*,*employee\_id*);

    }

}

*void* admin\_page(*int* *task\_num*,*bool* *taskdone\_pr*[20],string *tasklist\_pr*[20],*int* *taskid\_pr*)

{

    system ( "cls" );

*char* option;

    cout<<"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx||  A  D  M  I  N   P   A   G    E ||xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"<<endl;

    cout<<"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|| For Adding tasks ------>Enter 1.||xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"<<endl;

    cout<<"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|| For Task List    ------>Enter 2.||xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"<<endl;

    cout<<"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|| Mark Task as Done ----->Enter 3.||xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"<<endl;

    cout<<"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|| CHANGE USERS     ------>Enter 0.||xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"<<endl;

    cin>> option;

    if (option =='0'||option =='1' ||option =='2'||option=='3')

    {

        if (option=='0')

        {

            main();

        }

        if (option=='1')

        {

            add\_tasks(*task\_num*,*taskdone\_pr*,*tasklist\_pr*,*taskid\_pr*);

        }

        if(option=='2')

        {

            view\_task(*task\_num*,*taskdone\_pr*,*tasklist\_pr*,*taskid\_pr*);

        }

        if(option=='3')

        {

            mark\_task\_done(*task\_num*,*taskdone\_pr*,*tasklist\_pr*,*taskid\_pr*);

        }

    }

    else

    {

        validation\_fail();

        admin\_page(*task\_num*,*taskdone\_pr*,*tasklist\_pr*,*taskid\_pr*);

    }

}

*void* employee\_page(*int* *task\_num*,*bool* *taskdone\_pr*[20],string *tasklist\_pr*[20],*int* *taskid\_pr*)

{

 system ( "cls" );

*char* option;

    cout<<"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx||  EMPLOYEE               PAGE    ||xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"<<endl;

    cout<<"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|| For MARKING tasks ----->Enter 1.||xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"<<endl;

    cout<<"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|| For Task List    ------>Enter 2.||xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"<<endl;

    cout<<"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|| CHANGE USERS     ------>Enter 0.||xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"<<endl;

    cin>> option;

    if (option =='0'||option =='1' ||option =='2')

    {

        if (option=='0')

        {

            main();

        }

        if (option=='1')

        {

            mark\_task\_done(*task\_num*,*taskdone\_pr*,*tasklist\_pr*,*taskid\_pr*);

        }

        if(option=='2')

        {

            view\_task(*task\_num*,*taskdone\_pr*,*tasklist\_pr*,*taskid\_pr*);

        }

    }

    else

    {

        validation\_fail();

        admin\_page(*task\_num*,*taskdone\_pr*,*tasklist\_pr*,*taskid\_pr*);

    }

}

//tasklist functions

*void* add\_tasks(*int* *task\_num*,*bool* *taskdone\_pr*[20],string *tasklist\_pr*[20],*int* *taskid\_pr*)

{

*char* option;

*bool* loop\_cont = true;

    while(loop\_cont)

    {

        cout << "ENTER TASK :- ";

*tasklist\_pr*[*taskid\_pr*]=get\_string\_input();

        cout << "ENTER ANOTHER TASK (y or n) ?";

        cin >> option;

        if (option =='y')

        {

            loop\_cont = true;

*taskid\_pr* = *taskid\_pr* + 1 ;

        }

        else if (option =='n')

        {

            loop\_cont = false;

*taskid\_pr* = *taskid\_pr* + 1 ;

        }

    }

    admin\_page(*task\_num*,*taskdone\_pr*,*tasklist\_pr*,*taskid\_pr*);

}

*void* view\_task(*int* *task\_num*, *bool* *taskdone\_pr*[20], string *tasklist\_pr*[20], *int* *taskid\_pr*)

{

*int* hei = 5;

    for (*int* i =0; i <*taskid\_pr*; i++)

    {

        gotoxy(0, hei);

        cout << i+1 << ")- ";

        gotoxy(5, hei);

        cout << *tasklist\_pr*[i];

        gotoxy(55, hei);

        if (*taskdone\_pr*[i])

        {

            cout << "done";

        }

        else

        {

            cout << "not done";

        }

        cout << endl;

        hei = hei + 1;

    }

    getch();

    admin\_page(*task\_num*,*taskdone\_pr*,*tasklist\_pr*,*taskid\_pr*);

}

*void* mark\_task\_done(*int* *task\_num*, *bool* *taskdone\_pr*[20], string *tasklist\_pr*[20], *int* *taskid\_pr*)

{

*int* hei = 5;

    for (*int* i =0; i <*taskid\_pr*; i++)

    {

        gotoxy(0, hei);

        cout << i+1 << ")- ";

        gotoxy(5, hei);

        cout << *tasklist\_pr*[i];

        gotoxy(55, hei);

        if (*taskdone\_pr*[i])

        {

            cout << "done";

        }

        else

        {

            cout << "not done";

        }

        cout << endl;

        hei = hei + 1;

    }

    hei = hei + 1;

    gotoxy(0, hei);

*int* temp\_int;

    cout <<"Enter task number to mark as done): ";

    cin>>temp\_int;

    if (temp\_int<= *taskid\_pr* && temp\_int > 0)

    {

*taskdone\_pr*[temp\_int-1] = true;

        cout << "Task " << *taskid\_pr* << " marked as done." << endl;

    }

    else

    {

        validation\_fail();

    }

        if (state == "admin")

    {

        admin\_page(*task\_num*,*taskdone\_pr*,*tasklist\_pr*,*taskid\_pr*);

    }

    else if (state == "employee")

    {

        employee\_page(*task\_num*,*taskdone\_pr*,*tasklist\_pr*,*taskid\_pr*);

    }

}

//validation functions

*void* validation\_fail()

{

    cout <<"invalid input (press any key to continue)";

    getch();

}

*bool* login\_Input\_validation(string *option*)

{

    if (*option* == "1" || *option* =="2" || *option* =="3")

    {

        return true;

    }

    else

    {

        return false;

    }

}

*bool* sign\_up\_validation(string *option*)

{

    if (*option* == "admin" || *option* =="programmar" || *option* =="artist")

    {

        return true;

    }

    else

    {

        return false;

    }

}

**Weakness in the Business Application**

* + Need to add more validations and polish.

**Future Directions**

* + Add more polish to the program.