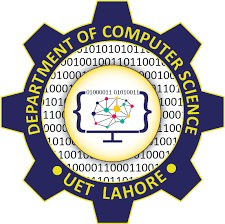
**Pharmaceutical Record keeper**

****

**2022-2026**

**Submitted by**

Usman kibria 2022-CS-90

**Supervised by**

Miss Maida Shahid

Department of computer science

**University of Engineering and Technology**

**Lahore Pakistan**

**Description:**

Pharmacies are an important part of our daily lives and due to large number of customers it can be difficult to keep record of everything, hence I want to make this software to help them keep record of everything so that they can provide better customer service. In the field of computer science this program will help support user computer interaction and help explain how the use of computer can make our work easier in various fields. The output I am expecting from this software is to give a sum of the total sales of each day, the amount of each medicine remaining and the amount required to be brought in, it will mark the presence of each of the employees in the shop etc.

**Users:**

The main users of this project will be the employees of the pharmacy who will need this software to keep record of their sales and purchases. This software can also be used by the company owners or any other higher authority to keep check of the attendance of the employees and also keep check of what medicine are being sold at the pharmacy. Hence we can say that this application can be used by both the employees of and the admins of a pharmaceutical company.

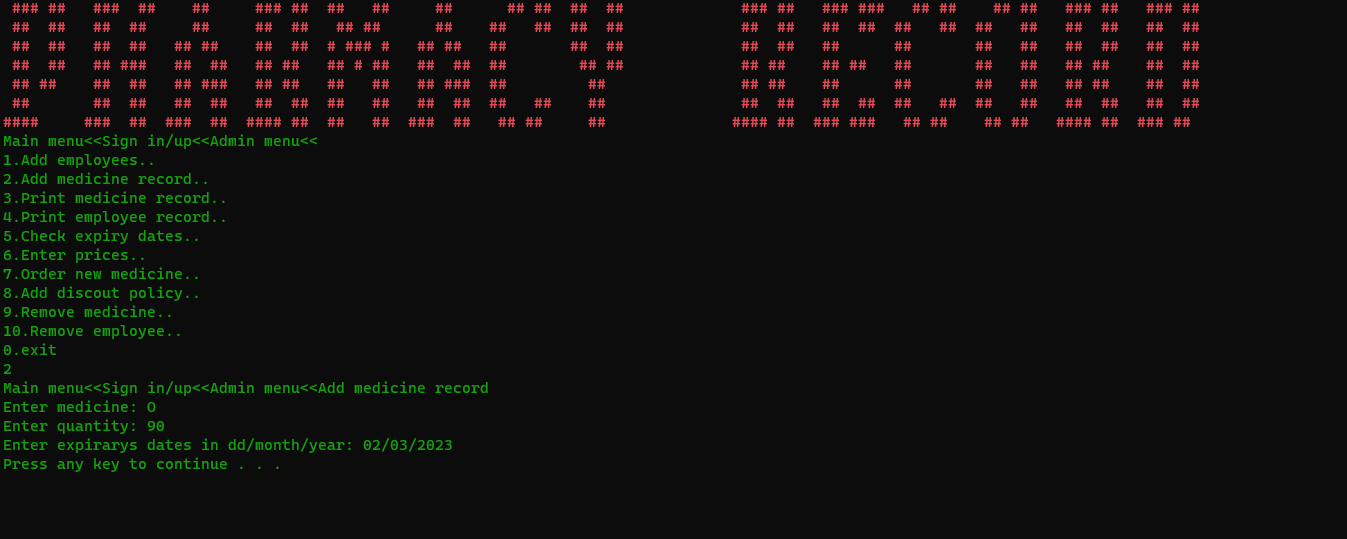
**Functional requirements:**

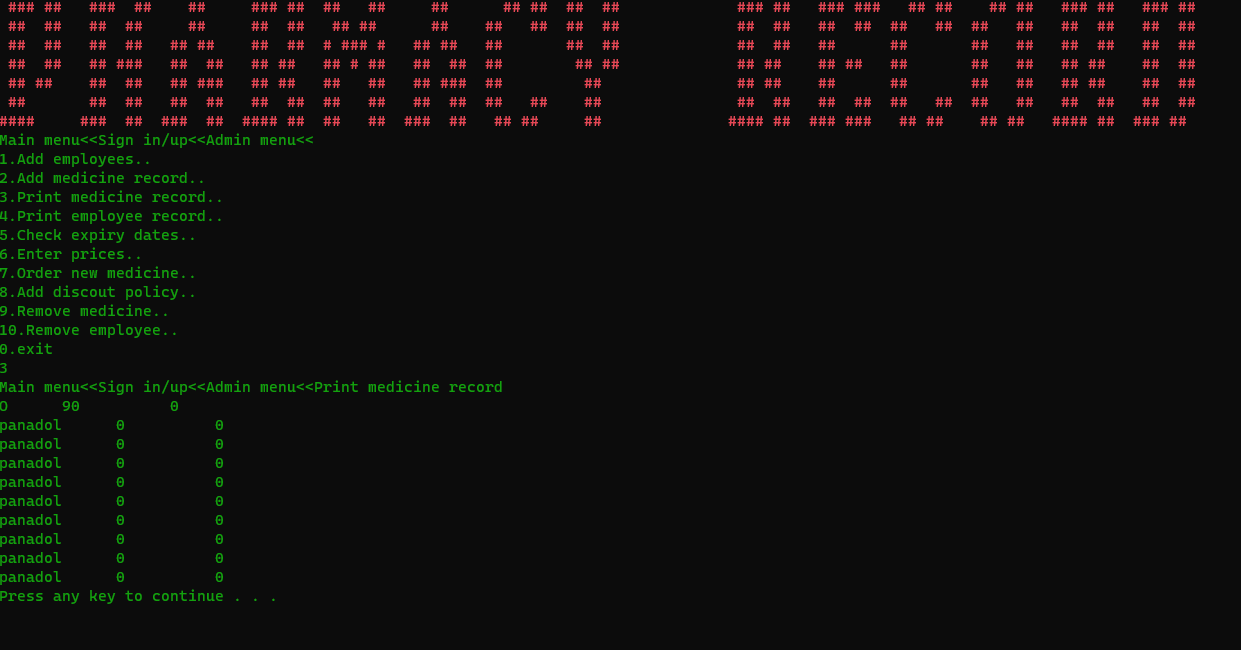
|  |  |  |
| --- | --- | --- |
| Admin | Check attendance & medicine, register employees, check expiry dates of medicine, order new medicine, make discount policy, check medicine amounts, remove medicine, remove employee and edit prices. | The admin can check attendance of the employees and keep check of what medicine is being sold |
| Employees | Record new medicine, mark attendance, register medicine, prepare bill, check expiry, offer discount and add medicine amounts. | The employees can record of the medicine being sold and mark their attendance. |

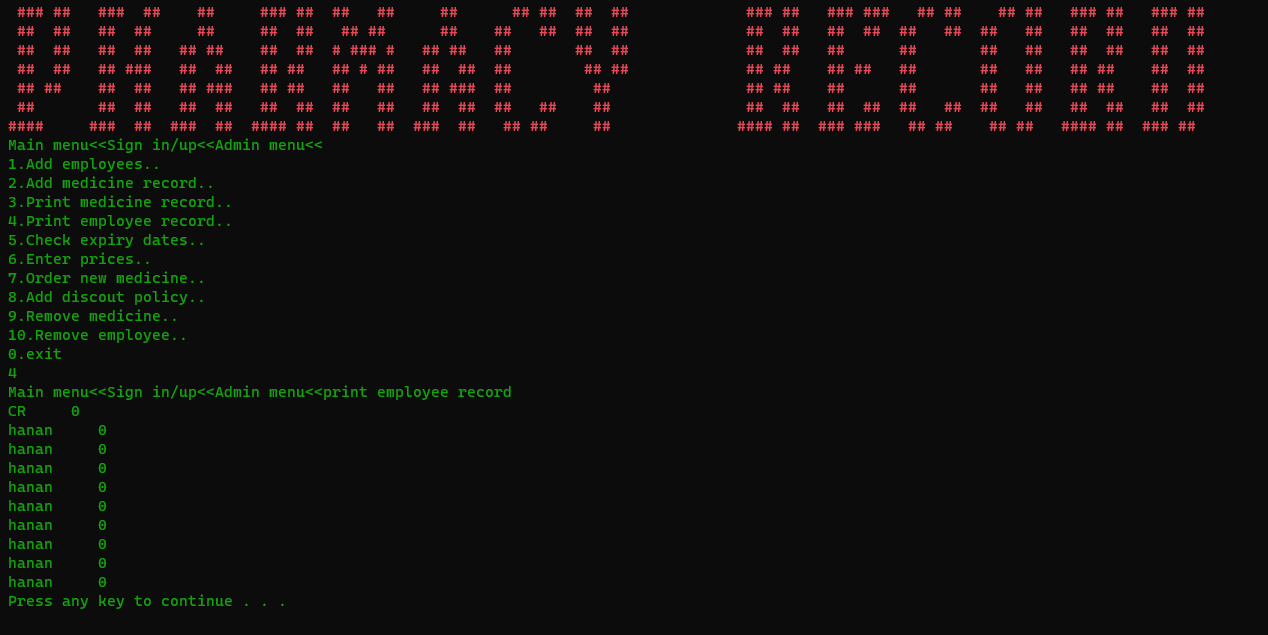
**Wireframe:**

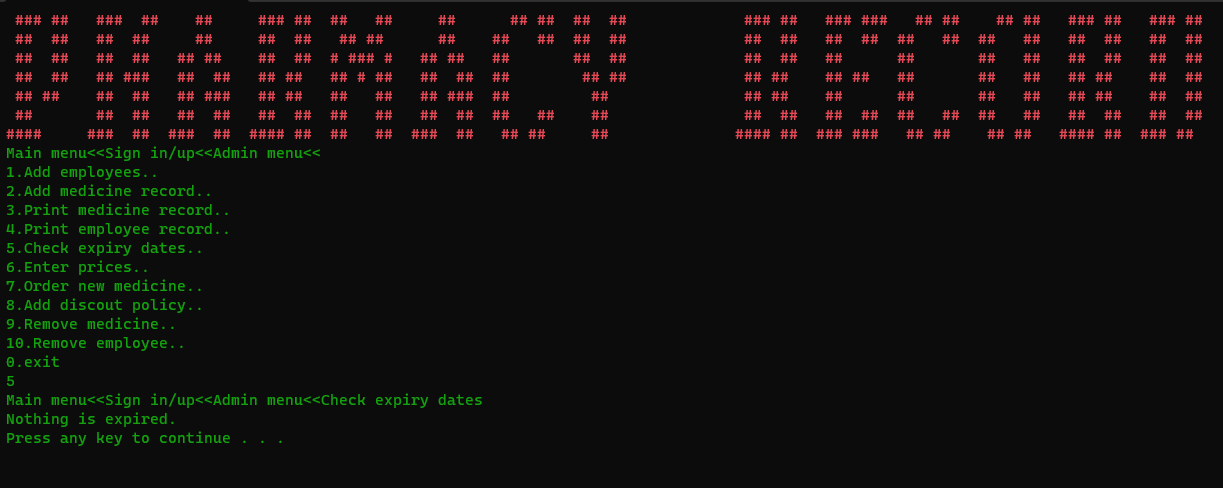
****



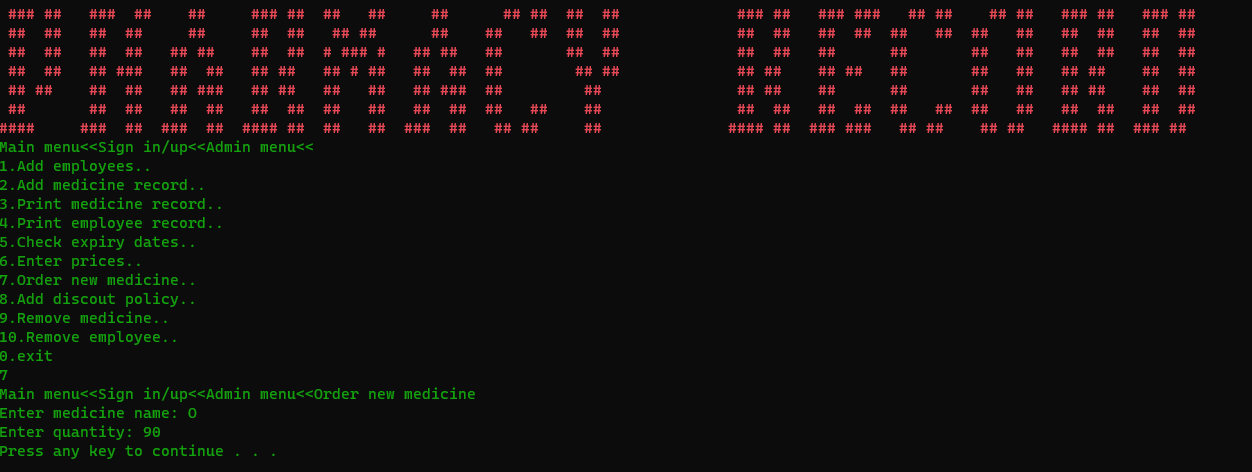






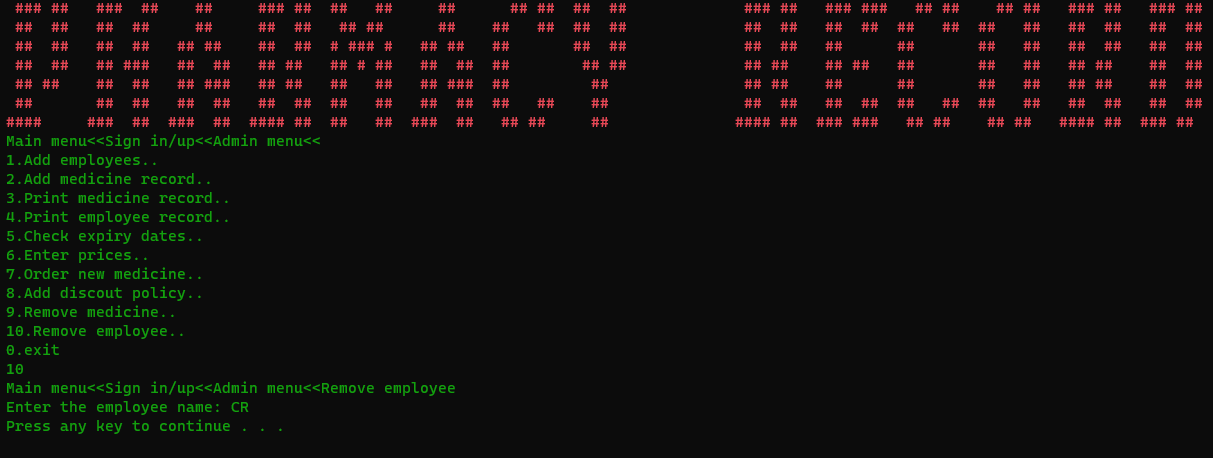


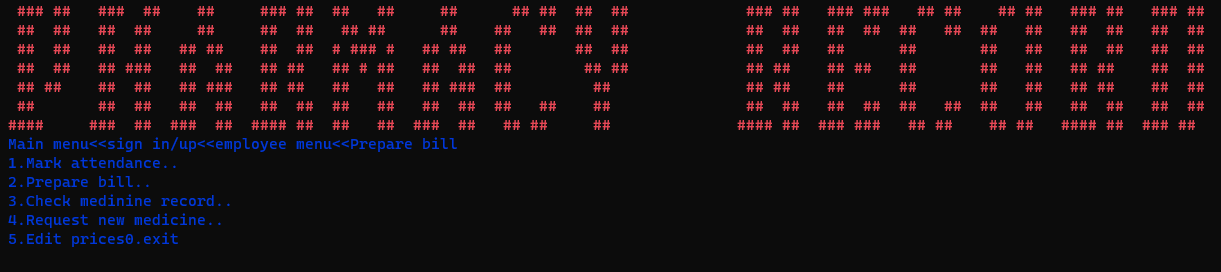




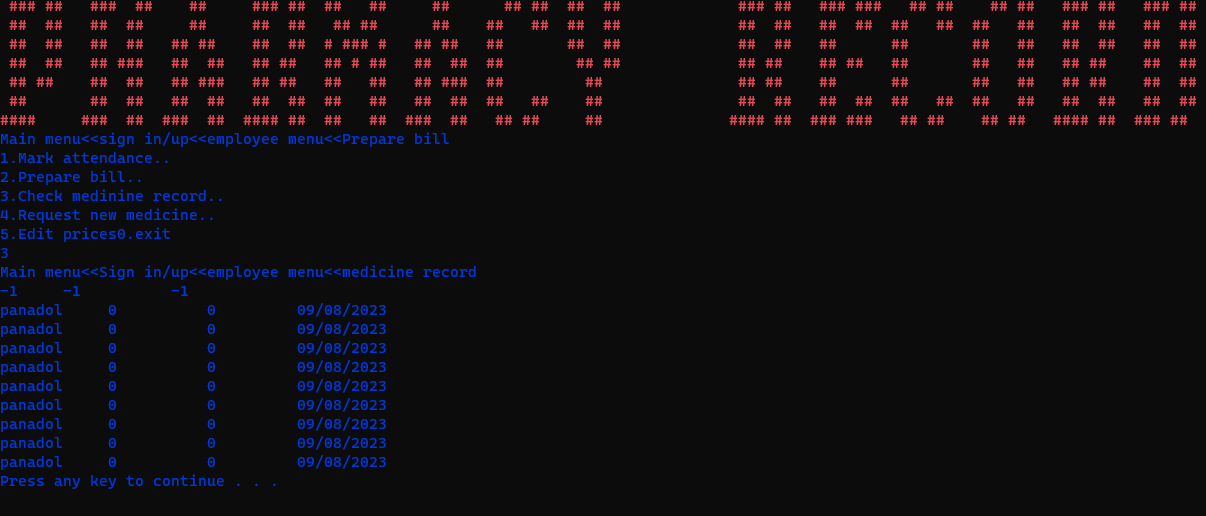


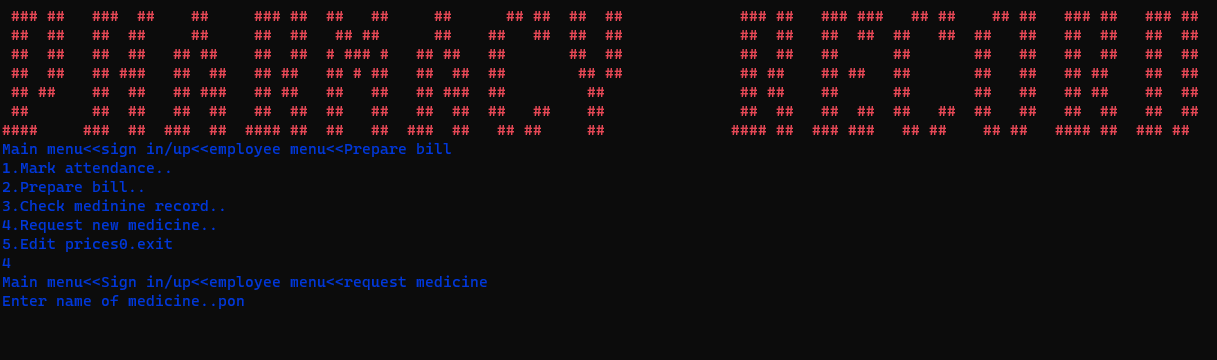


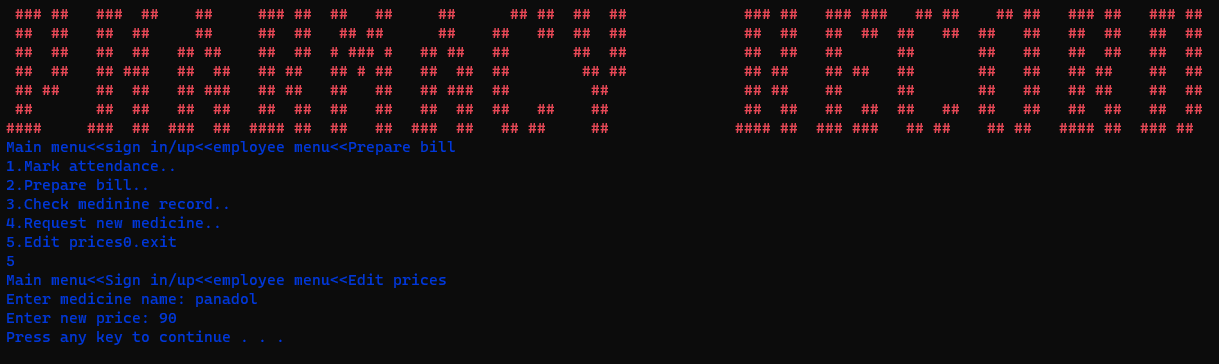












**Data structures:**

string medicine[10],int quantity[10],string expirary[10]

string username[10],string password[10],string role[10]

string employee\_name[10],float attendance[10]

int x,int y,int z:

**Function prototypes:**

#include "updatefiles.h"

#include "get\_character.h"

#include "storedata.h"

#include "admin.h"

#include "signupin.h"

#include"upload\_sign.h"

#include "employee.h"

#include "readfile.h"

#include"signup\_count.h"

#include"time.h"

#include"compare\_date.h"

**Weakness:**

The weakness of that it only deals with 10 medicine and employees at a time.

**Future applications:**

I intend to run the code in GUI and make it capable of working for any number of employees and medicine.

**Flow chart:**

Main menuenu

Sign Up/In

Employee menu

Admin menu

Request new medicine

Mark attendance

Add employee

Enter prices

Prepare bill

Edit prices

Order new medicine

Add medicine

Check medicine record

Print employee record

Add discount policy

Print medicine record

Remove medicine

Check expiry dates

Remove employee

**Code:**

**Main:**

#include <iostream>

#include <windows.h>

#include <fstream>

#include <string>

#include <ctime>

#include <conio.h>

#include "updatefiles.h"

#include "get\_character.h"

#include "storedata.h"

#include "admin.h"

#include "signupin.h"

#include"upload\_sign.h"

#include "employee.h"

#include "readfile.h"

#include"signup\_count.h"

using namespace std;

main()

{

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

    int signup\_count = 0;

  signup\_count=read\_signup\_count(signup\_count);

  readfile();

  upload\_sign(signup\_count);

    while (true)

    {

        system("cls");

           SetConsoleTextAttribute(hConsole, 12);

cout<<" ### ##   ###  ##    ##     ### ##  ##   ##     ##      ## ##  ##  ##             ### ##   ### ###   ## ##    ## ##   ### ##   ### ##"<<endl;

cout<<" ##  ##   ##  ##     ##     ##  ##   ## ##      ##    ##   ##  ##  ##             ##  ##   ##  ##  ##   ##  ##   ##   ##  ##   ##  ##"<<endl;

cout<<" ##  ##   ##  ##   ## ##    ##  ##  # ### #   ## ##   ##       ##  ##             ##  ##   ##      ##       ##   ##   ##  ##   ##  ##"<<endl;

cout<<" ##  ##   ## ###   ##  ##   ## ##   ## # ##   ##  ##  ##        ## ##             ## ##    ## ##   ##       ##   ##   ## ##    ##  ##"<<endl;

cout<<" ## ##    ##  ##   ## ###   ## ##   ##   ##   ## ###  ##         ##               ## ##    ##      ##       ##   ##   ## ##    ##  ##"<<endl;

cout<<" ##       ##  ##   ##  ##   ##  ##  ##   ##   ##  ##  ##   ##    ##               ##  ##   ##  ##  ##   ##  ##   ##   ##  ##   ##  ##"<<endl;

cout<<"####     ###  ##  ###  ##  #### ##  ##   ##  ###  ##   ## ##     ##              #### ##  ### ###   ## ##    ## ##   #### ##  ### ## "<<endl;

   SetConsoleTextAttribute(hConsole, 15);

    jump:

        int option;

        cout <<"1.Sign up " << endl

             <<"2.sign in" << endl

             <<"0.exit" << endl;

        cin >> option;

        if (option == 1)

        {

            sign\_up(signup\_count);

            signup\_count++;

            system("pause");

        }

        else if (option == 2)

        {

            string Role = sign\_in(signup\_count);

            if (Role == "admin")

            {

                admin(hConsole);

            }

            if (Role == "employee")

            {

                employee\_(hConsole);

            }

        }

        else if (option == 0)

        {

            signup\_countstore(signup\_count);

            store\_a\_b\_c\_d();

            return 0;

        }

        else if (option != 1 && option != 2 &&option!=0)

        {

            cout << "Wrong input!!!";

            system("pause");

            goto jump;

        }

    }

}

**Admin:**

#include "time.h"

using namespace std;

string mydate[5];

char const \*A;//variables to compare dates

char const \*B;//same

const int y = 10;//array indexes

const int z = 10;//array indexes

bool result;

int discount\_c;

int payable\_amount=0; //'y' shows the number of medicine and 'z' shows the number of employees

int option, quantity[y], prices[y], amount1, dis;

float attendance[z], discount;

 int employee\_count = 0;

int medicine\_count = 0;

string mname, ename, medicine[y], employee[z], name, price\_name;

string expirary[y];

int admin(HANDLE hConsole)

{

        int u = 0;

    while (true)

    {

        jump:

         system("cls");

         SetConsoleTextAttribute(hConsole, 12);

cout<<" ### ##   ###  ##    ##     ### ##  ##   ##     ##      ## ##  ##  ##             ### ##   ### ###   ## ##    ## ##   ### ##   ### ##"<<endl;

cout<<" ##  ##   ##  ##     ##     ##  ##   ## ##      ##    ##   ##  ##  ##             ##  ##   ##  ##  ##   ##  ##   ##   ##  ##   ##  ##"<<endl;

cout<<" ##  ##   ##  ##   ## ##    ##  ##  # ### #   ## ##   ##       ##  ##             ##  ##   ##      ##       ##   ##   ##  ##   ##  ##"<<endl;

cout<<" ##  ##   ## ###   ##  ##   ## ##   ## # ##   ##  ##  ##        ## ##             ## ##    ## ##   ##       ##   ##   ## ##    ##  ##"<<endl;

cout<<" ## ##    ##  ##   ## ###   ## ##   ##   ##   ## ###  ##         ##               ## ##    ##      ##       ##   ##   ## ##    ##  ##"<<endl;

cout<<" ##       ##  ##   ##  ##   ##  ##  ##   ##   ##  ##  ##   ##    ##               ##  ##   ##  ##  ##   ##  ##   ##   ##  ##   ##  ##"<<endl;

cout<<"####     ###  ##  ###  ##  #### ##  ##   ##  ###  ##   ## ##     ##              #### ##  ### ###   ## ##    ## ##   #### ##  ### ## "<<endl;

   SetConsoleTextAttribute(hConsole, 2);

        cout << "Main menu<<Sign in/up<<Admin menu<<" << endl;

        cout << "1.Add employees.." << endl;

        cout << "2.Add medicine record.." << endl;

        cout << "3.Print medicine record.." << endl;

        cout << "4.Print employee record.." << endl;

        cout << "5.Check expiry dates.." << endl;

        cout << "6.Enter prices.." << endl;

        cout << "7.Order new medicine.." << endl;

        cout << "8.Add discout policy.." << endl;

        cout << "9.Remove medicine.." << endl;

        cout << "10.Remove employee.." << endl;

        cout<<"0.exit"<<endl;

        cin>>option;

        if (option == 1)

        {

            cout << "Main menu<<Sign in/up<<Admin menu<<Add employee " << endl;

            for (; employee\_count < z;)

            {

                cout << "Enter employee name: " << endl;

                cin.ignore();

                getline(cin, employee[employee\_count]);

                employee\_count++;

                system("pause");

                break;

            }

            Employeedata(employee,attendance);

        }

        if (option == 2)

        {

            cout << "Main menu<<Sign in/up<<Admin menu<<Add medicine record " << endl;

            for (; medicine\_count < y; )

            {

                cout << "Enter medicine: ";

                cin >> medicine[medicine\_count];

                cout << "Enter quantity: ";

                cin >> quantity[medicine\_count];

                cout << "Enter expirarys dates in dd/month/year: ";

                cin.ignore();

                getline(cin, expirary[medicine\_count]);

                medicine\_count++;

                system("pause");

                break;

            }

            Medicinedata(medicine,quantity,expirary);

        }

        if (option == 3)

        {

            cout << "Main menu<<Sign in/up<<Admin menu<<Print medicine record " << endl;

            for (int r = 0; r < y; r++)

            {

                if (medicine[r] != "-1")

                    cout << medicine[r] << "      " << quantity[r] <<"          "<<prices[r] << endl;

            }

            system("pause");

        }

        if (option == 4)

        {

            cout << "Main menu<<Sign in/up<<Admin menu<<print employee record " << endl;

            for (int e = 0; e < y; e++)

            {

                if (employee[e] != "-1")

                {

                    cout << employee[e] << "     " << attendance[e] << endl;

                }

            }

            system("pause");

        }

        if (option == 5)

        {

            cout << "Main menu<<Sign in/up<<Admin menu<<Check expiry dates " << endl;

            for (int t = 0; t < medicine\_count; t++)

            {

                string expiry=expirary[t];

               bool expired=time2(expiry);

               if(expired==true)

               {

                cout<<medicine[t]<<"        "<<"expired"<<endl;

                medicine[t]="-1";

               }

               else

               cout<<"Nothing is expired."<<endl;

            }

            system("pause");

        }

        if (option == 6)

        {

            cout << "Main menu<<Sign in/up<<Admin menu<<Enter prices " << endl;

            for (; u <=medicine\_count ;)

            {

                cout << medicine[u] << "  Enter price: ";

                cin >> prices[u];

                u++;

                system("pause");

                break;

            }

            price(prices);

        }

        if (option == 7)

        {

            cout << "Main menu<<Sign in/up<<Admin menu<<Order new medicine " << endl;

            cout << "Enter medicine name: ";

            cin >> name;

            for (int i = 0; i < y; i++)

            {

                if (name != medicine[i])

                {

                    cout << "Register medicine first" << endl;

                    system("pause");

                    break;

                }

                else if (name == medicine[i])

                {

                    cout << "Enter quantity: ";

                    cin >> amount1;

                    quantity[i] = quantity[i] + amount1;

                    system("pause");

                    break;

                }

            }

            updatamedicinedata(medicine,quantity,expirary,medicine\_count);

        }

        if (option == 8)

        {

            cout << "Main menu<<Sign in/up<<Admin menu<<Add dicount policy " << endl;

            cout<<"Enter discount condition: ";

            cin>>discount\_c;

            cout << "Add discount..";

            cin >> dis;

            if (payable\_amount > discount\_c)

            {

                discount = (dis / 100) \* payable\_amount;

                system("pause");

            }

            discount\_store(discount\_c,dis);

        }

        if (option == 9)

        {

            bool exists\_or\_not=false;

            cout << "Main menu<<Sign in/up<<Admin menu<<Remove medicine " << endl;

            cout << "Enter medicine name: ";

            cin >> mname;

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

            {

                if (mname == medicine[x])

                {

                    medicine[x] = "-1";

                    quantity[x] =-1;

                    prices[x] =-1;

                    expirary[x]=" ";

                    system("pause");

                    exists\_or\_not=true;

                    break;

                }

            }

                if(exists\_or\_not==false)

                {

                    cout<<"register medicine first:"<<endl;

                    system("pause");

                    break;

                }

            updatamedicinedata(medicine,quantity,expirary,medicine\_count);

        }

        if (option == 10)

        {

            bool exists\_or\_not=false;

            cout << "Main menu<<Sign in/up<<Admin menu<<Remove employee " << endl;

            cout << "Enter the employee name: ";

            cin >> ename;

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

            {

                if (ename == employee[x])

                {

                    employee[x] = "-1";

                    attendance[x]=-1;

                    system("pause");

                    exists\_or\_not=true;

                    break;

                }

            }

                if(exists\_or\_not==false)

                {

                    cout<<"register employee first:"<<endl;

                    system("pause");

                    break;

                }

            updateemployeedata(employee,attendance,employee\_count);

        }

        if(option==0)

        {

            return 0;

        }

        else if(option!=1 &&option!=2 &&option!=3 &&option!=4 &&option!=5 &&option!=6 &&option!=7 &&option!=8 &&option!=9 &&option!=10 &&option!=0 )

        {

            cout<<"Wrong input!!!";

            system("pause");

            goto jump;

        }

    }

    }

**Employee:**

using namespace std;

int employee\_(HANDLE hConsole)

{

    string name\_of\_medicine;

    while(true)

    {

        jump:

    system("cls");

        SetConsoleTextAttribute(hConsole, 12);

cout<<" ### ##   ###  ##    ##     ### ##  ##   ##     ##      ## ##  ##  ##             ### ##   ### ###   ## ##    ## ##   ### ##   ### ##"<<endl;

cout<<" ##  ##   ##  ##     ##     ##  ##   ## ##      ##    ##   ##  ##  ##             ##  ##   ##  ##  ##   ##  ##   ##   ##  ##   ##  ##"<<endl;

cout<<" ##  ##   ##  ##   ## ##    ##  ##  # ### #   ## ##   ##       ##  ##             ##  ##   ##      ##       ##   ##   ##  ##   ##  ##"<<endl;

cout<<" ##  ##   ## ###   ##  ##   ## ##   ## # ##   ##  ##  ##        ## ##             ## ##    ## ##   ##       ##   ##   ## ##    ##  ##"<<endl;

cout<<" ## ##    ##  ##   ## ###   ## ##   ##   ##   ## ###  ##         ##               ## ##    ##      ##       ##   ##   ## ##    ##  ##"<<endl;

cout<<" ##       ##  ##   ##  ##   ##  ##  ##   ##   ##  ##  ##   ##    ##               ##  ##   ##  ##  ##   ##  ##   ##   ##  ##   ##  ##"<<endl;

cout<<"####     ###  ##  ###  ##  #### ##  ##   ##  ###  ##   ## ##     ##              #### ##  ### ###   ## ##    ## ##   #### ##  ### ## "<<endl;

  SetConsoleTextAttribute(hConsole, 1);

    int option,quantity\_,no,price\_;

    string name\_att,medicine\_;

    cout<<"Main menu<<sign in/up<<employee menu<<Prepare bill"<<endl;

    cout<<"1.Mark attendance.."<<endl;

    cout<<"2.Prepare bill.."<<endl;

    cout<<"3.Check medinine record.."<<endl;

    cout<<"4.Request new medicine.."<<endl;

    cout<<"5.Edit prices"<<endl;

    cout<<"0.exit"<<endl;

    cin>>option;

    if(option==1)

    {

        bool exists\_or\_not=false;

        cout << "Main menu<<Sign in/up<<employee menu<<Mark attendance " << endl;

        cout<<"Enter name: ";

        cin>>name\_att;

        for(int x=0;x<z;x++)

        {

            if(name\_att==employee[x])

            {

                attendance[x]++;

                exists\_or\_not=true;

                break;

            }

        }

            if(exists\_or\_not==false)

            {

                cout<<"Register employee first"<<endl;

                break;

            }

        system("pause");

    }

    if(option==2)

    {

        bool exists\_or\_not=false;

        go:

        hope:

        cout << "Main menu<<Sign in/up<<employee menu<<prepare bill " << endl;

        cout<<"Enter number of medicine: ";

        cin>>no;

        for(int x=0;x<no;x++)

        {

            cout<<"Enter the name of medicine: ";

        cin>>medicine\_;

        cout<<"quantity: ";

        cin>>quantity\_;

        for(int h=0;h<10;h++)

        {

        if(medicine\_==medicine[h])

        {

            if(quantity\_>quantity[h])

            {

                cout<<"quantity too low";

                system("pause");

                goto go;

            }

            price\_=prices[h]\*quantity\_;

            quantity[h]=quantity[h]-quantity\_;

               payable\_amount=payable\_amount+price\_;

               exists\_or\_not=true;

        h++;

        break;

        }

        }

        if(exists\_or\_not==false)

        {

        cout<<"Invalid input";

        system("pause");

        goto hope;

        }

        }

        if(payable\_amount>3000)

        {

            cout<<"Original amount: "<<payable\_amount;

            payable\_amount=payable\_amount-discount;

        }

    cout<<"Payable amount"<<payable\_amount<<endl;

    system("pause");

    }

    if(option==3)

    {

        cout << "Main menu<<Sign in/up<<employee menu<<medicine record " << endl;

        for(int x=0;x<y;x++)

        {

            if(medicine[x]!="-1")

            cout<<medicine[x] <<"     "<<quantity[x]<<"          "<<prices[x]<<"         "<<expirary[x]<<endl;

        }

        system("pause");

    }

    if(option==4)

    {

        cout << "Main menu<<Sign in/up<<employee menu<<request medicine " << endl;

        bool if\_it\_exists=false;

        cout<<"Enter name of medicine..";

        cin>>name\_of\_medicine;

        for(int j=0;j<medicine\_count;j++)

        {

            if(name\_of\_medicine==medicine[j])

            {

                cout<<"Medicine already exists";

                system("pause");

                if\_it\_exists=true;

                break;

            }

        }

        if(if\_it\_exists==false && medicine\_count<10)

        {

            medicine[medicine\_count+1]=name\_of\_medicine;

        }

        else

        cout<<"Medicine already exists or the store is full";

    }

     if (option == 5)

        {

            cout << "Main menu<<Sign in/up<<employee menu<<Edit prices " << endl;

            cout << "Enter medicine name: ";

            cin >> price\_name;

            bool check\_x=false;

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

            {

                if (price\_name == medicine[x])

                {

                    cout << "Enter new price: ";

                    cin >> prices[x];

                    check\_x=true;

                    system("pause");

                    break;

                }

            }

                if(check\_x==false)

                {

                cout<<"Register medicine first:"<<endl;

                system("pause");

                }

            updatapricedata(prices,medicine\_count);

        }

    if(option==0)

    {

        return 0;

    }

    else if(option!=1 && option!=2 && option!=3 &&option!=4 &&option!=5 && option!=0)

    {

        cout<<"Wrong input!!!";

        system("pause");

        goto jump;

    }

}

}

**Time:**

#include<cstdlib>

#include "compare\_date.h"

using namespace std;

int index=0;

 bool time2(string expiry)

 {

   string digit;

   int year=0,month=0,day=0;

      for(int z=0;z<3;z++)

      {

         if(z==0)

         {

            while(expiry[index]!='/')

            {

               day=expiry[index]-'0';

      index++;

            }

         }

      if(z==1)

      {

         while(expiry[index]!='/')

         {

            month=expiry[index]-'0';

         index++;

      }

      }

      if(z==2)

      {

         while(expiry[index]!='\0')

         {

              digit=digit+(expiry[index]);

         index++;

      }

      }

      index++;

   }

   bool expired= compare\_time(year,month,day);

   return expired;

   }

**Compare date:**

#include<cstdlib>

using namespace std;

int year\_system;

int month\_system;

int day\_system;

bool compare\_time(int year,int month,int day);

void current\_time()

{

     time\_t now = time(NULL);

   tm \*ltm = localtime(&now);

   year\_system=  1900 + ltm->tm\_year ;

   month\_system= 1 + ltm->tm\_mon;

   day\_system=ltm->tm\_mday;

  }

  bool compare\_time(int year,int month,int day)

  {

    if(year\_system<year)

    {

        return false;

    }

    else if(year\_system==year &&month\_system<month)

    {

        return false;

    }

    else if(year\_system==year &&month\_system==month && (day\_system<day))

    {

        return false;

    }

    else

    {

        return true;

    }

  }

**Store data:**

#include<cstdlib>

using namespace std;

int a=0,b=0,c=0,d=0;

string line\_\_,s\_line\_\_;

void store\_a\_b\_c\_d();

void read\_a\_b\_c\_d();

void discount\_store(int discount\_condition, int discountd);

void price(int prices[10]);

void Medicinedata(string medicine\_\_[10], int quantity\_\_[10], string expirary\_\_[10]);

void signupindata(string username\_\_[10], string password\_\_[10], string role\_\_[10],int signup\_count);

void Employeedata(string epmloyee\_\_\_name[10], float attendance[10])

{

    read\_a\_b\_c\_d();

    fstream data;

    data.open("data.txt", ios::app);

        if(a>0)

        {

    data<< "\n";

        }

    data << epmloyee\_\_\_name[a]<<',';

    data << attendance[a];

    data.close();

    a++;

}

void Medicinedata(string medicine\_\_[10], int quantity\_\_[10], string expirary\_\_[10])

{

   read\_a\_b\_c\_d();

    fstream data;

    data.open("data1.txt", ios::app);

        if(b>0)

        {

    data<<"\n";

        }

    data << medicine\_\_[b]<<",";

    data << quantity\_\_[b]<<",";

    data << expirary\_\_[b] ;

    data.close();

    b++;

}

void price(int prices[10])

{

    read\_a\_b\_c\_d();

    fstream data;

    data.open("prices.txt", ios::app);

    if(c>0)

    data<< "\n";

    data << prices[c];

    data.close();

    c++;

}

void discount\_store(int discount\_condition, int discountd)

{

    read\_a\_b\_c\_d();

    fstream data;

    data.open("discount.txt", ios::out);

    data << discount\_condition <<",";

    data << discountd;

    data.close();

}

void signupindata(string username\_\_[10], string password\_\_[10], string role\_\_[10],int signup\_count)

{

    read\_a\_b\_c\_d();

    fstream data;

    data.open("signupindata.txt", ios::app);

    if(signup\_count>0)

    {

        data<<"\n";

    }

        data << username\_\_[d] << ",";

        data << password\_\_[d] << ",";

        data << role\_\_[d];

        d++;

    data.close();

}

void store\_a\_b\_c\_d()

{

    fstream data;

    data.open("a\_b\_c\_d.txt",ios::out);

    data<< a<<","<<b<<","<<c<<","<<d;

    data.close();

}

void read\_a\_b\_c\_d()

{

    int y=0;

    fstream data4;

    data4.open("a\_b\_c\_d.txt",ios::in);

    getline(data4,line\_\_);

    while(line\_\_[y]!=',')

    {

        a=int(line\_\_[y])-'0';

        y++;

    }

    y++;

    while(line\_\_[y]!=',')

    {

        b=int(line\_\_[y])-'0';

        y++;

    }

    y++;

    while(line\_\_[y]!=',')

    {

        c=int(line\_\_[y])-'0';

        y++;

    }

    y++;

    while(line\_\_[y]!='\0')

    {

        d=int(line\_\_[y])-'0';

        y++;

    }

    data4.close();

}

**Read file:**

#include<cstdlib>

using namespace std;

void readfile()

{

    string line, s\_line;

    int y = 0;

    fstream data;

    data.open("data.txt", ios::in);

    for (int x = 0; x < 10; x++)

    {

        getline(data, line);

        y = 0;

        while (line[y] != ',')

        {

            employee[x] = employee[x] + line[y];

            y++;

        }

        y++;

        while (line[y] != '\0')

        {

            s\_line = s\_line + line[y];

            y++;

        }

        attendance[x] = stoi(s\_line);

        y++;

    }

    data.close();

    y = 0;

    fstream data1;

    data1.open("data1.txt", ios::in);

    for (int x = 0; x < 10; x++)

    {

        getline(data1, line);

        y = 0;

        while (line[y] != ',')

        {

            medicine[x] = medicine[x] + line[y];

            y++;

        }

        y++;

        while (line[y] != ',')

        {

            s\_line = s\_line + line[y];

            y++;

        }

        quantity[x] = stoi(s\_line);

        y++;

        while (line[y] != '\0')

        {

            expirary[x] = expirary[x] + line[y];

            y++;

        }

        y++;

    }

    data1.close();

     y = 0;

    fstream data2;

    data2.open("prices.txt", ios::in);

    for (int x = 0; x < 10; x++)

    {

        getline(data2, line);

        y = 0;

        while (line[y] != '\0')

        {

            s\_line = s\_line + line[y];

            y++;

        }

        prices[x] = stoi(s\_line);

    }

    data2.close();

     y = 0;

    fstream data3;

    data3.open("discount.txt", ios::in);

    getline(data3, line);

    while (line[y] != ',')

    {

        s\_line = s\_line + line[y];

        y++;

    }

    discount\_c = stoi(s\_line);

    y++;

    while (line[y] != '\0')

    {

        s\_line = s\_line + line[y];

        y++;

    }

    dis = stoi(s\_line);

    data3.close();

 }

**Sign up in count:**

using namespace std;

int read\_signup\_count(int singup\_count);

void signup\_countstore(int singup\_count)

{

    fstream data;

    data.open("signup\_count.txt", ios::out);

    data << singup\_count << "\n";

    data.close();

}

int read\_signup\_count(int singup\_count)

{

    int y = 0;

    string line, s\_line;

    fstream data;

    data.open("signup\_count.txt", ios::in);

    getline(data, line);

    while (line[y] != ' ')

    {

        s\_line = s\_line + line[y];

        y++;

    }

    singup\_count = stoi(s\_line);

    data.close();

    return singup\_count;

}

**Signup/in:**

using namespace std;

string username[10], role[10], password[10];

string sign\_in(int signup\_count);

string sign\_up(int signup\_count)

{

    string user;

    cout << "Enter the username: ";

    cin >> user;

    bool flag = true;

    for (int c = 0; c < 10; c++)

    {

        if ((username[c] == user))

        {

            flag = false;

            break;

        }

    }

    if (flag == true)

    {

        username[signup\_count] = user;

    }

    cout << "Enter password: ";

    cin >> password[signup\_count];

    cout << "Enter role: ";

    cin >> role[signup\_count];

    signupindata(username, password, role,signup\_count);

}

string sign\_in(int signup\_count)

{

    if (signup\_count == 0)

    {

        cout << "You need to sign up first." << endl;

        system("pause");

        return "0";

    }

    bool exists = false;

    string user\_name, pass\_word, Role;

    cout << "Enter username: ";

    cin >> user\_name;

    cout << "Enter password:";

    cin >> pass\_word;

    int d = 0;

    for (; d < 10; d++)

    {

        if (user\_name == username[d] && pass\_word==password[d])

        {

            exists = true;

            break;

        }

    }

        if (exists == false)

        {

            cout << "Invalid input" << endl;

            system("pause");

            return "0";

        }

    return role[d];

}

**Get character:**

using namespace std;

char getCharAtxy(short int x, short int y)

{

CHAR\_INFO ci;

COORD xy = {0, 0};

SMALL\_RECT rect = {x, y, x, y};

COORD coordBufSize;

coordBufSize.X = 1;

coordBufSize.Y = 1;

return ReadConsoleOutput(GetStdHandle(STD\_OUTPUT\_HANDLE), &ci, coordBufSize, xy, &rect) ? ci.Char.AsciiChar

: ' ';

}

**Update files:**

using namespace std;

void updateemployeedata(string array[10],float array1[10],int employee\_count)

{

    fstream data;

    data.open("data.txt", ios::out);

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

        data << array[x]<<",";

        data<<array1[x]<<" ";

            data << "\n";

    }

    data.close();

}

void updatamedicinedata(string array1[10],int array2[10],string array3[10],int medicine\_count)

{

    fstream data;

    data.open("data1.txt", ios::out);

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

    {

        data << array1[x]<<",";

         data<<array2[x]<<",";

        data<<array3[x]<<" "<<"\n";

    }

    data.close();

}

void updatapricedata(int prices\_[10],int medicine\_count)

{

    fstream data;

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

    {

    data.open("prices.txt",ios::out);

    data<<prices\_[x]<<" "<<"\n";

    }

    data.close();

}

**Upload sign:**

using namespace std;

void upload\_sign(int a\_count)

{

    string line;

    int y = 0;

    fstream data;

    data.open("signupindata.txt", ios::in);

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

    {

        getline(data, line);

        y = 0;

        while (line[y] != ',')

        {

            username[x] = username[x] + line[y];

            y++;

        }

        y++;

        while (line[y] != ',')

        {

            password[x] = password[x] + line[y];

            y++;

        }

        y++;

        while (line[y] != ' ')

        {

            role[x] = role[x] + line[y];

            y++;

        }

    }

}

Student Reg. No. : 2022-CS-90 Student Name. Usman kibria

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A-Extensive Evidence** | **B-Convincing Evidence** | **C-Limited Evidence** | **D-No Evidence** |
| Documentation Formatting  **Grade:** | All the documentation meets all the criteria. | Documentation is well formatted but some of the criteria is not fulfilled. | Documentation is required a lot of improvement. | Documentation is not Available |
| **Documentation Formatting Criteria:** In **Binder**, **Title** Page, **Header**-Footers, Font **Style**, Font **Size** all are all consistence and according to given **guidelines**. Project **Poster** is professionally design and well presented | | | | |
| Documentation Contents  **Grade:** | Documentation includes all of the criteria. | Documentation meet more than 80% of the criteria given. | Documentation meet more than 50% of the criteria. | When the documentation meet less than 50% of the criteria. |
| **Documentation Contents Criteria:** **Title** Page - **Table** of Contents - Project **Abstract** - **Functional** Requirements - **Wire** Frames –**Data Flow** Diagram-**Data** Structure (Arrays)-**Function** Headers and Description -Project **Code.** - **Weakness** in the Project and **Future** Directions. - **Conclusion** and What your **Learn** from the Project and Course and What is your **Future** Planning. | | | | |
| Project Complexity  **Grade:** | Project has at least 2 user’s types and each user has at least 5 functionalities. | Project complexity meet 80% criteria given in extensive evidence | Project complexity meet 50% criteria given in extensive evidence | Project complexity meet less than 50% criteria given in extensive evidence |
| Code Style  **Grade:** | All Code style criteria is followed | All code style criteria followed but some improvements required | lot of improvements required in coding style. | **Did not follow** code style, |
| **Code Style Criteria:**  Consistent code style. Code is well indented. Variable and Function names are well defined.  White Spaces are well used. Comments are added. | | | | |
| Code Documentation Mapping  **Grade:** | Code and documentation is synchronized. | Code and documentation does not synchronized at **some** places | Code and documentation does not synchronized at **many** places | Code and documentation **does not** synchronized. |
| Data Structure (Arrays)  **Grade:** | Data structure is sufficient for the project requirements | Data Structure is sufficient but require improvement to meet project requirements. | Data structure is not sufficient and need a lot of improvement | Data Structure is not properly identified and declared. |
| Modularity  **Grade:** | Meet all Modularity criteria | Meet all Modularity criteria but at some places it is missing | Do not sufficiently meet the modularity criteria. | No modularity or very minimum modularity. |
| **Modularity criteria:** Functions are defined for each major feature. Functions are independent (identify from parameter list and return types). | | | | |
| Validations  **Grade:** | Validations on all number type inputs are applied | Validations are applied but at some places it is missing. | Validations are missing at lot of places | No Validations are used |
| File Handling  **Grade:** | Separate files for separate data. Data in csv format | File handing require some improvements | File handing require a lot of improvements | Not implemented |
| Aesthetics of the User Interface  **Grade:** | UI is presentable. Proper coloring, Headers and clear screen is done | UI require some improvements | UI require a lot of improvements | Not implemented |
| Presentation and Demo  **Grade:** | Presentation and Demo was 100% working | Presentation and Demo require some improvements | Presentation and Demo require a lot of improvements | Presentation was not ok and Demo was not working |
| Student Understanding with the Code.  **Grade:** | Student has complete understanding how the code is working and knows the concept. | Student has good understand but some place he does not know the concepts | Student has a very little understand and lack the major concepts. | Student does not have any level of understanding of the code. |

|  |  |
| --- | --- |
| **Checked by:** |  |
| **Comments:** |  |

Student Reg. No. : 2022-CS-90 Student Name. Usman kibria

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A-Extensive Evidence** | **B-Convincing Evidence** | **C-Limited Evidence** | **D-No Evidence** |
| Documentation Formatting  **Grade:** | All the documentation meets all the criteria. | Documentation is well formatted but some of the criteria is not fulfilled. | Documentation is required a lot of improvement. | Documentation is not Available |
| **Documentation Formatting Criteria:** In **Binder**, **Title** Page, **Header**-Footers, Font **Style**, Font **Size** all are all consistence and according to given **guidelines**. Project **Poster** is professionally design and well presented | | | | |
| Documentation Contents  **Grade:** | Documentation includes all of the criteria. | Documentation meet more than 80% of the criteria given. | Documentation meet more than 50% of the criteria. | When the documentation meet less than 50% of the criteria. |
| **Documentation Contents Criteria:** **Title** Page - **Table** of Contents - Project **Abstract** - **Functional** Requirements - **Wire** Frames –**Data Flow** Diagram-**Data** Structure (Arrays)-**Function** Headers and Description -Project **Code.** - **Weakness** in the Project and **Future** Directions. - **Conclusion** and What your **Learn** from the Project and Course and What is your **Future** Planning. | | | | |
| Project Complexity  **Grade:** | Project has at least 2 user’s types and each user has at least 5 functionalities. | Project complexity meet 80% criteria given in extensive evidence | Project complexity meet 50% criteria given in extensive evidence | Project complexity meet less than 50% criteria given in extensive evidence |
| Code Style  **Grade:** | All Code style criteria is followed | All code style criteria followed but some improvements required | lot of improvements required in coding style. | **Did not follow** code style, |
| **Code Style Criteria:**  Consistent code style. Code is well indented. Variable and Function names are well defined.  White Spaces are well used. Comments are added. | | | | |
| Code Documentation Mapping  **Grade:** | Code and documentation is synchronized. | Code and documentation does not synchronized at **some** places | Code and documentation does not synchronized at **many** places | Code and documentation **does not** synchronized. |
| Data Structure (Arrays)  **Grade:** | Data structure is sufficient for the project requirements | Data Structure is sufficient but require improvement to meet project requirements. | Data structure is not sufficient and need a lot of improvement | Data Structure is not properly identified and declared. |
| Modularity  **Grade:** | Meet all Modularity criteria | Meet all Modularity criteria but at some places it is missing | Do not sufficiently meet the modularity criteria. | No modularity or very minimum modularity. |
| **Modularity criteria:** Functions are defined for each major feature. Functions are independent (identify from parameter list and return types). | | | | |
| Validations  **Grade:** | Validations on all number type inputs are applied | Validations are applied but at some places it is missing. | Validations are missing at lot of places | No Validations are used |
| File Handling  **Grade:** | Separate files for separate data. Data in csv format | File handing require some improvements | File handing require a lot of improvements | Not implemented |
| Aesthetics of the User Interface  **Grade:** | UI is presentable. Proper coloring, Headers and clear screen is done | UI require some improvements | UI require a lot of improvements | Not implemented |
| Presentation and Demo  **Grade:** | Presentation and Demo was 100% working | Presentation and Demo require some improvements | Presentation and Demo require a lot of improvements | Presentation was not ok and Demo was not working |
| Student Understanding with the Code.  **Grade:** | Student has complete understanding how the code is working and knows the concept. | Student has good understand but some place he does not know the concepts | Student has a very little understand and lack the major concepts. | Student does not have any level of understanding of the code. |

|  |  |
| --- | --- |
| **Checked by:** |  |
| **Comments:** |  |