**Blood Donation Management System**



Session: 2022 – 2026

**Submitted by:**

Muhammad Wali Ahmad 2022-CS-65

**Supervised by:**

Maida Shahid

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

Table of Contents

[**1.** **Short Description of your project:** 3](#_Toc128668643)

[**2.** **Users of Application** 3](#_Toc128668644)

[**3.** **Functional Requirements:** 3](#_Toc128668645)

[**3.1 Admin Functionalities:** 3](#_Toc128668646)

[**3.1 Employee Functionalities:** 4](#_Toc128668647)

[**4.** **Wireframes:** 5](#_Toc128668648)

[**4.1 Login screen:** 5](#_Toc128668649)

[**4.2 Admin screen and Options:** 5](#_Toc128668650)

[**5.** **Data Structures:** 16](#_Toc128668651)

[**6.** **Function Prototypes:** 16](#_Toc128668652)

[**7.** **Functions Working Flow:** 18](#_Toc128668653)

[**8.** **Complete Code of the Business Application:** 20](#_Toc128668654)

[**9.** **Weakness in the Application:** 59](#_Toc128668655)

[**10.** **Future Directions:** 59](#_Toc128668656)

# **Short Description of your project:**

During the emergency situation or surgery people need blood for the survival of their lives and to search their desirable blood they find difficulty. My management system provides them facility to find their blood group easily and in short time. A blood donation management system can make significant contributions in the field of computer science by utilizing technology to improve various aspects of the blood donation process. Overall, the development and implementation of a blood donation management system can have a positive impact on the field of computer science by applying technology to address real-world problems and improve healthcare outcomes. The importance of a blood donation management system lies in its ability to ensure the availability of safe and adequate blood supplies for transfusions. This can have significant benefits for both patients and healthcare systems. Person can find their desirable blood group by simple searching on my management system in short time and save their love one’s lives.

# **Users of Application**

* **Admin:**

An Admin control the management system for example add, delete or update the record of the employee and also see the clients (donors and recipients) information in the management system.

* **Employee:**

An employee uses the management system to add, delete or update record of clients (donors and recipients) and also search their clients by their blood group according to the need of other clients.

# **Functional Requirements:**

Following table contains all functionalities of users (Admin and Employee) which they can perform in the management system.

|  |  |  |  |
| --- | --- | --- | --- |
| **Admin Functionalities:** | | | |
| ***User Story ID*** | ***As a*** | ***I want to perform*** | ***So that I can*** |
| ***1.*** | Admin | Add employee | So that, the new employees are added |
| ***1.*** | Admin | Delete employee | So that, the old employees are removed |
| ***1.*** | Admin | Edit details of employee | So that the details of the employee are updated |
| ***1.*** | Admin | Search employee | So that details of the employee are displayed |
| ***1.*** | Admin | View all employee | So that details of all employees are displayed |
| ***1.*** | Admin | View all donors | So that details of all donors are  displayed |
| ***1.*** | Admin | Search donors | So that details of the donor are  displayed |
| ***1.*** | Admin | View all recipient | So that details of all recipients are displayed |
| ***1.*** | Admin | Search recipient | So that details of the recipient are  displayed |
| ***1.*** | Admin | Log out | So that application is closed |
| **Employee Functionalities:** | | | |
| ***User Story ID*** | ***As a*** | ***I want to perform*** | ***So that I can*** |
| ***2.*** | Employee | Add donor | So that, the new donors are added |
| ***2.*** | Employee | Delete donor | So that, the old donors are removed |
| ***2..*** | Employee | Edit details of donor | So that the details of the donor are updated |
| ***2.*** | Employee | Search donors | So that details of the donor are  displayed |
| ***2.*** | Employee | View all donors | So that details of all donors are  displayed |
| ***2.*** | Employee | Add recipient | So that, the new recipients are added |
| ***2.*** | Employee | Delete recipient | So that, the old recipients are removed |
| ***2.*** | Employee | Edit details of recipient | So that the details of the recipient are updated |
| ***2.*** | Employee | Search recipient | So that details of the recipient are  displayed |
| ***2.*** | Employee | View all recipient | So that details of all recipients are displayed |
| ***2.*** | Employee | Log out | So that application is closed |

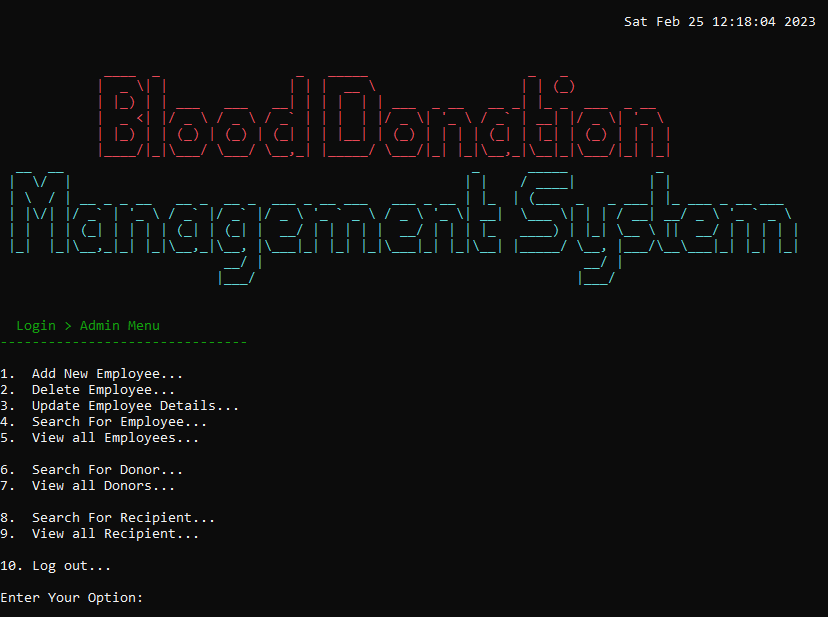
# **Wireframes:**

# **Login screen:**



**Figure 1: Login Screen**

# **Admin screen and Options:**



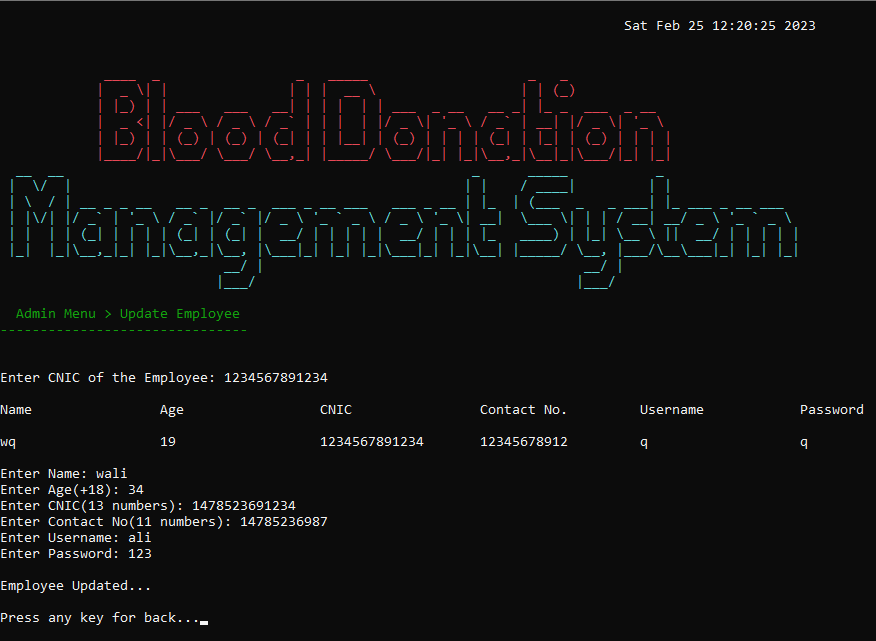
**Figure 2: Admin Main Menu Screen**



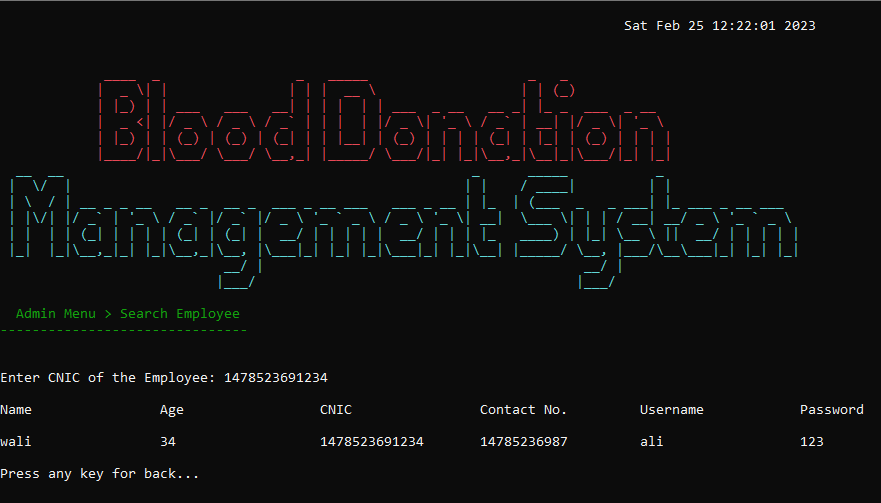
**Figure 2.1: Add Employee Screen**

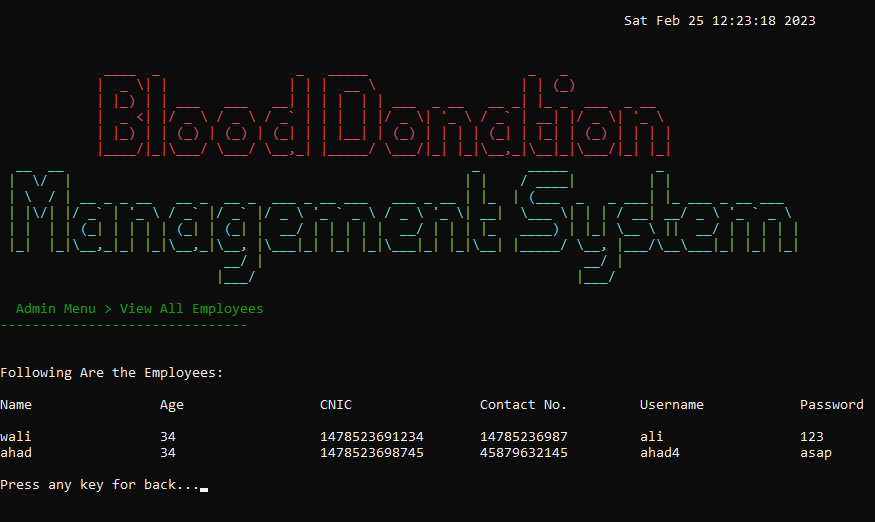
****

**Figure 2.2: Delete Employee Screen**

****

**Figure 2.3: Update Employee Screen**

**Figure 2.4: Search Employee Screen**

****

**Figure 2.5: View All Employee Screen**

****

**Figure 2.6: Search Donor Screen**

****

**Figure 2.7: View All Donor Screen**

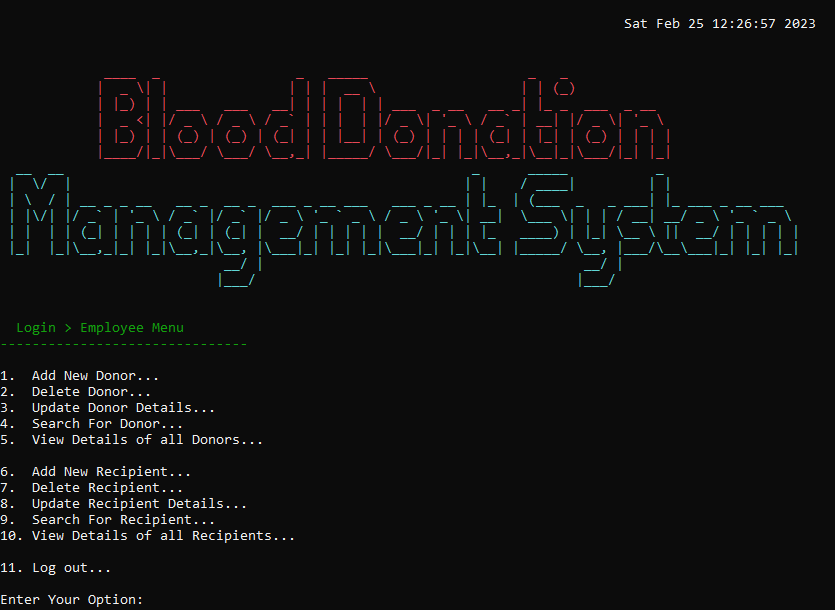
****

**Figure 2.8: Search Recipient Screen**

****

**Figure 2.9: View All Recipients Screen**

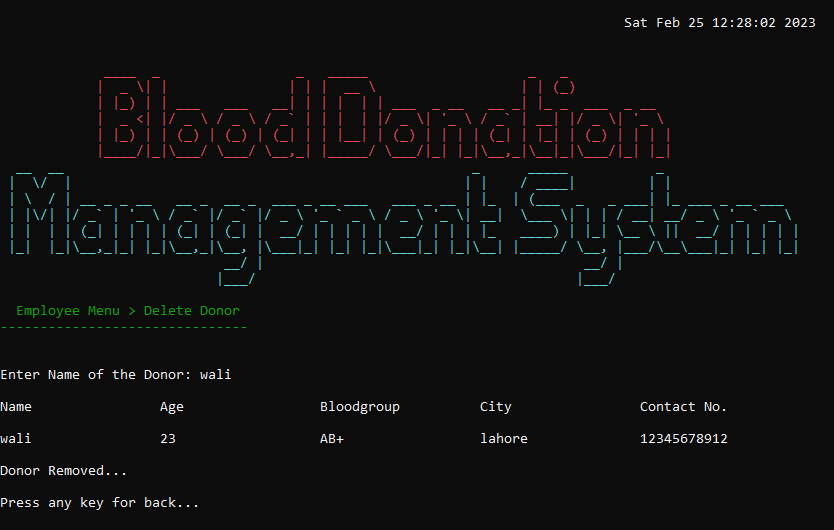
* 1. **Employee screen and Option:**



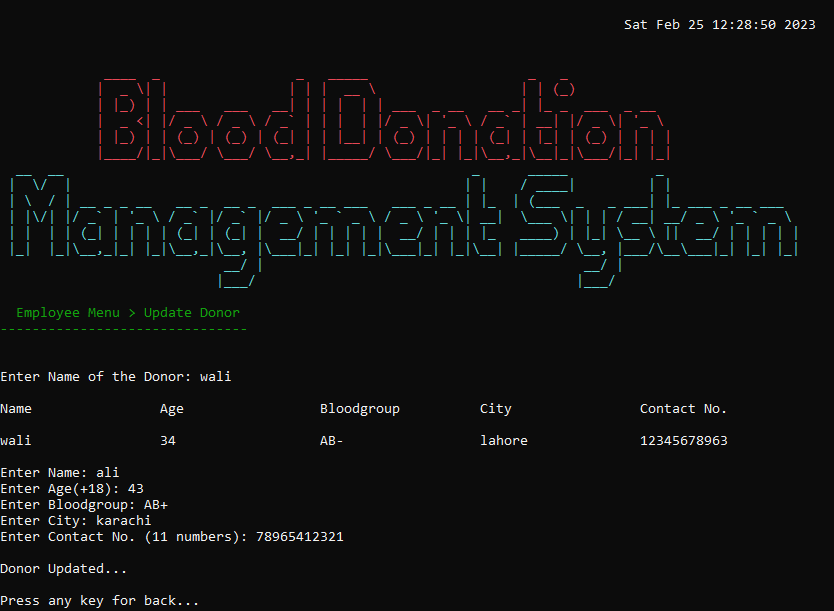
**Figure 3: Employee Main Menu Screen**

****

**Figure 3.1: Add Donor Screen**

****

**Figure 3.2: Delete Donor Screen**

****

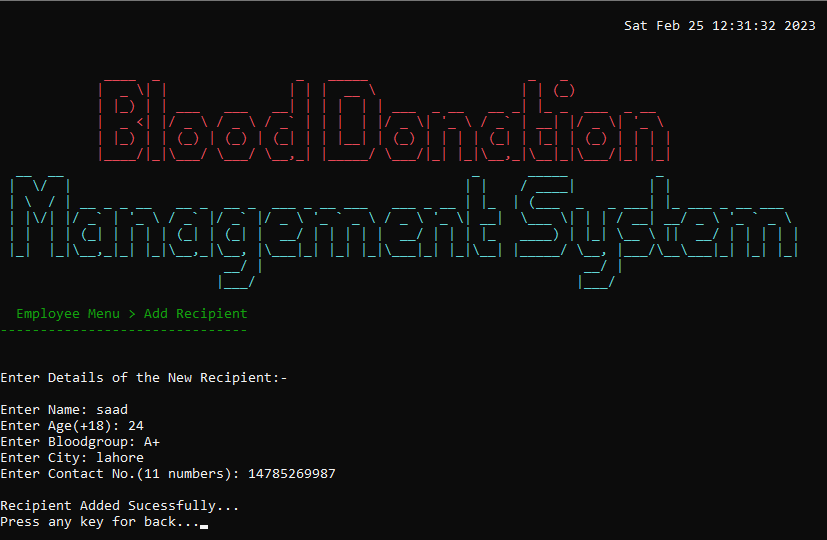
**Figure 3.3: Update Donor Screen**

****

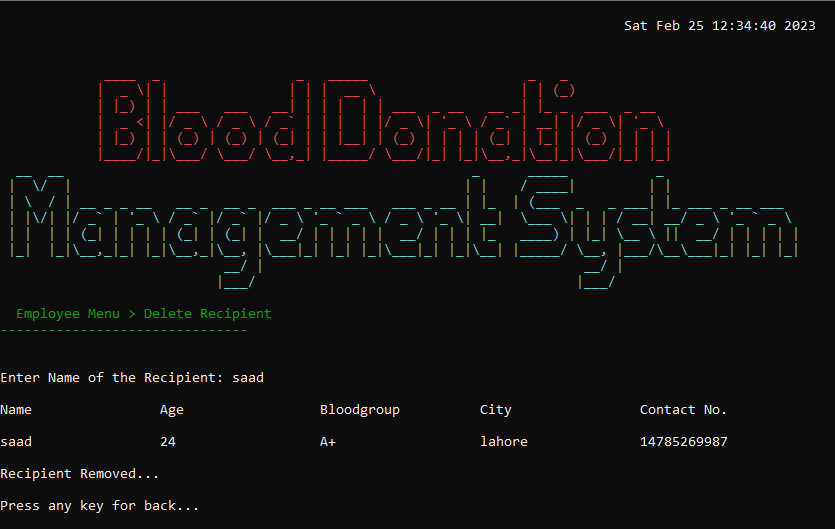
**Figure 3.4: Search Donor Screen**

****

**Figure 3.5: View All Donor Screen**

****

**Figure 3.6: Add Recipient Screen**

****

**Figure 3.7: Delete Recipient Screen**

****

**Figure 3.8: Update Recipient Screen**

****

**Figure 3.9: Search Recipient Screen**

****

**Figure 3.10: View All Recipients Screen**

# **Data Structures:**

// employee data

string nameE[100];

string ageE[100];

string usernameE[100];

string passwordE[100];

string cnicE[100];

string contactE[100];

string contributer;

int indexE = 0; // index for employees arrays

// donor data

string nameD[100];

string ageD[100];

string bloodgroupD[100];

string cityD[100];

string statusD[100];

string contactD[100];

string contributerD[100];

int indexD = 0; // index for donors arrays

// recipient data

string nameR[100];

string ageR[100];

string bloodgroupR[100];

string cityR[100];

string contactR[100];

string contributerR[100];

int indexR = 0; // index for recipients arrays

# **Function Prototypes:**

// prototypes

void printHeader();

int login();

void adminMenu();

void printAMenu();

void employeeMenu();

void printEMenu();

void addDonor(); // Donor Functions

void deleteDonor();

void updateDonor();

void searchDonor();

void viewDonor();

void addRecipient(); // Recipient Functions

void deleteRecipient();

void updateRecipient();

void searchRecipient();

void viewRecipient();

void addEmployee(); // Employee Functions

void deleteEmployee();

void updateEmployee();

void searchEmployee();

void viewEmployee();

void searchDonorbyAdmin(); // Functions for admin

void viewDonorbyAdmin();

void searchRecipientbyAdmin();

void viewRecipientbyAdmin();

void menuName(string menu, string subMenu); // print submenu

string setcolor(unsigned short color); // color set

string isAlpha(string input); // functions for input

string isNum(string input); // functions for input

string isBG(string input); // functions for input

string contactCheck(string contact); // functions for input

string cnicCheck(string cnic); // functions for input

string usercheck(string username); // functions for input

int choiceCheck(int choice); // functions for input

void DonorToFile(string name, string age, string bloodgroup, string city, string contact, string contributer); // store data to file

void RecipientToFile(string name, string age, string bloodgroup, string city, string contact, string contributer); // store data to file

void EmployeeToFile(string name, string age, string cnic, string contact, string username, string password); // store data to file

void LoadDonor(); // load file to arrays

void LoadRecipient(); // load file to arrays

void LoadEmployee();// load file to arrays

void updateDonorFile(); // update file

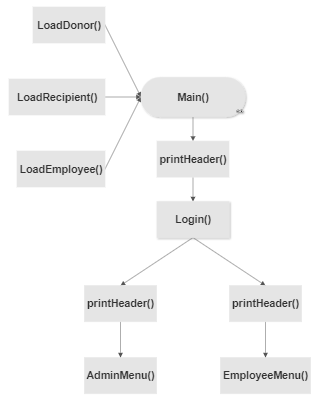
void updateEmployeeFile();// update file

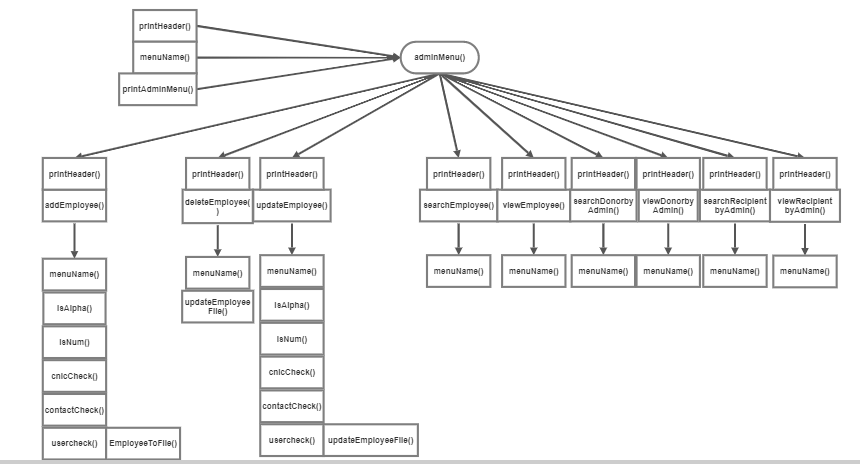
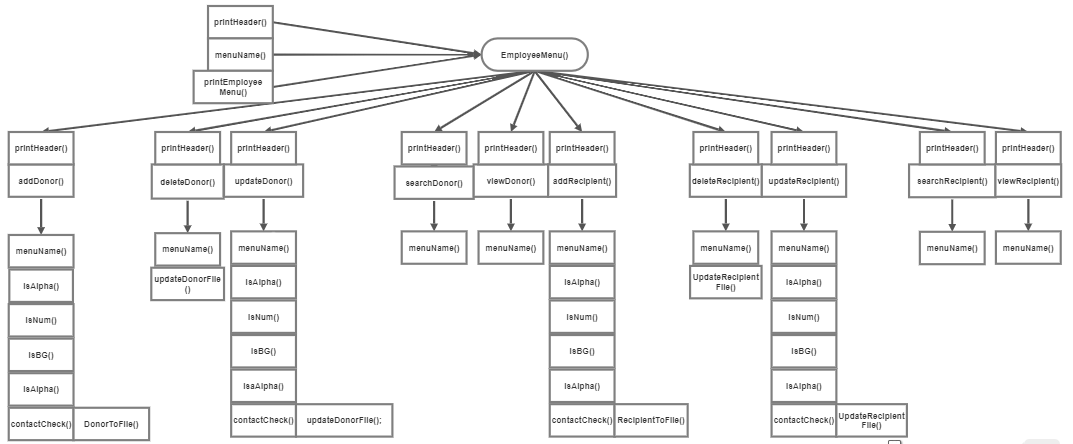
void updateRecipientFile(); // update file

string Dataparse(string line, int field);

void gotoxy(int x, int y);

# **Functions Working Flow:**





# **Complete Code of the Business Application:**

main()

{

    LoadDonor();

    LoadRecipient();

    LoadEmployee();

    while (true)

    {

        system("cls");

        printHeader();

        int user = login();

        if (user == 1)

        {

            system("cls");

            printHeader();

            adminMenu();

        }

        else if (user == 2)

        {

            system("cls");

            printHeader();

            employeeMenu();

        }

        else if (user == 3)

        {

            cout << endl;

            cout << "Wrong Credentials!! Try again!!";

            Sleep(300);

        }

    }

}

void printHeader()

{

    time\_t now = time(0); // date display

    char \*date\_time = ctime(&now);

    gotoxy(78, 1);

    cout << date\_time << endl;

    cout << endl;

    setcolor(12);

  cout << " \_\_\_\_   \_               \_   \_\_\_\_\_                    \_   \_              " << endl;

    cout << "|  \_ \| |               | | |  \_\_ \                  | | (\_)                  " << endl;

    cout << "| |\_) | | \_\_\_   \_\_\_   \_\_| | | |  | | \_\_\_  \_ \_\_   \_\_ \_| |\_ \_  \_\_\_  \_ \_\_         " << endl;

    cout << "|  \_ <| |/ \_ \ / \_ \ / \_` | | |  | |/ \_ \| '\_ \ / \_` | \_\_| |/ \_ \| '\_ \         " << endl;

    cout << "| |\_) | | (\_) | (\_) | (\_| | | |\_\_| | (\_) | | | | (\_| | |\_| | (\_) | | | |         " << endl;

    cout << "|\_\_\_\_/|\_|\\_\_\_/ \\_\_\_/ \\_\_,\_| |\_\_\_\_\_/ \\_\_\_/|\_| |\_|\\_\_,\_|\\_\_|\_|\\_\_\_/|\_| |\_|          " << endl;

setcolor(11);

    cout << "  \_\_  \_\_                                                   \_      \_\_\_\_\_           \_               " << endl;

    cout << " |  \/  |                                                 | |    / \_\_\_\_|         | |               " << endl;

    cout << " | \  / | \_\_ \_ \_ \_\_   \_\_ \_  \_\_ \_  \_\_\_ \_ \_\_ \_\_\_   \_\_\_ \_ \_\_ | |\_  | (\_\_\_  \_   \_ \_\_\_| |\_ \_\_\_ \_ \_\_ \_\_\_  " << endl;

    cout << " | |\/| |/ \_` | '\_ \ / \_` |/ \_` |/ \_ \ '\_ ` \_ \ / \_ \ '\_ \| \_\_|  \\_\_\_ \| | | / \_\_| \_\_/ \_ \ '\_ ` \_ \  " << endl;

    cout << " | |  | | (\_| | | | | (\_| | (\_| |  \_\_/ | | | | |  \_\_/ | | | |\_   \_\_\_\_) | |\_| \\_\_ \ ||  \_\_/ | | | | | " << endl;

    cout << " |\_|  |\_|\\_\_,\_|\_| |\_|\\_\_,\_|\\_\_, |\\_\_\_|\_| |\_| |\_|\\_\_\_|\_| |\_|\\_\_| |\_\_\_\_\_/ \\_\_, |\_\_\_/\\_\_\\_\_\_|\_| |\_| |\_| " << endl;

    cout << "                            \_\_/ |                                        \_\_/ |                       " << endl;

    cout << "                           |\_\_\_/                                        |\_\_\_/                      " << endl;

    setcolor(15);

}

int login()

{

    string username, password;

    int choice;

    cout << endl;

    cout << endl;

    cout << endl;

    cout << "Welcome to Blood Donation Management System";

    cout << endl;

    cout << endl;

    cout << endl;

    cout << "Enter Username: ";

    cin >> username;

    cout << "Enter Password: ";

    cin >> password;

    if (username == "admin" && password == "admin")

    {

        choice = 1;

    }

    else

    {

        for (int i = 0; i < 100; i++)

        {

            if (username == usernameE[i] && password == passwordE[i])

            {

                contributer = username;

                choice = 2;

                break;

            }

            else

            {

                choice = 3;

            }

        }

    }

    return choice;

}

void adminMenu()

{

    int choice = 0;

    while (choice != 10)

    {

        system("cls");

        printHeader();

        cout << endl;

        cout << endl;

        string menu = "Login";

        string subMenu = "Admin Menu";

        menuName(menu, subMenu);

        printAMenu();

        choice = choiceCheck(choice);

        if (choice == 1)

        {

            system("cls");

            printHeader();

            addEmployee();

        }

        if (choice == 2)

        {

            system("cls");

            printHeader();

            deleteEmployee();

        }

        if (choice == 3)

        {

            system("cls");

            printHeader();

            updateEmployee();

        }

        if (choice == 4)

        {

            system("cls");

            printHeader();

            searchEmployee();

        }

        if (choice == 5)

        {

            system("cls");

            printHeader();

            viewEmployee();

        }

        if (choice == 6)

        {

            system("cls");

            printHeader();

            searchDonorbyAdmin();

        }

        if (choice == 7)

        {

            system("cls");

            printHeader();

            viewDonorbyAdmin();

        }

        if (choice == 8)

        {

            system("cls");

            printHeader();

            searchRecipientbyAdmin();

        }

        if (choice == 9)

        {

            system("cls");

            printHeader();

            viewRecipientbyAdmin();

        }

    }

}

void printAMenu()

{

    cout << endl;

    cout << "1.  Add New Employee..." << endl;

    cout << "2.  Delete Employee... " << endl;

    cout << "3.  Update Employee Details... " << endl;

    cout << "4.  Search For Employee... " << endl;

    cout << "5.  View all Employees... " << endl;

    cout << endl;

    cout << "6.  Search For Donor... " << endl;

    cout << "7.  View all Donors... " << endl;

    cout << endl;

    cout << "8.  Search For Recipient... " << endl;

    cout << "9.  View all Recipient... " << endl;

    cout << endl;

    cout << "10. Log out... " << endl;

    cout << endl;

    cout << "Enter Your Option: ";

}

void employeeMenu()

{

    int choice = 0;

    while (choice != 11)

    {

        system("cls");

        printHeader();

        cout << endl;

        cout << endl;

        string menu = "Login";

        string subMenu = "Employee Menu";

        menuName(menu, subMenu);

        printEMenu();

        choice = choiceCheck(choice);

        if (choice == 1)

        {

            system("cls");

            printHeader();

            addDonor();

        }

        if (choice == 2)

        {

            system("cls");

            printHeader();

            deleteDonor();

        }

        if (choice == 3)

        {

            system("cls");

            printHeader();

            updateDonor();

        }

        if (choice == 4)

        {

            system("cls");

            printHeader();

            searchDonor();

        }

        if (choice == 5)

        {

            system("cls");

            printHeader();

            viewDonor();

        }

        if (choice == 6)

        {

            system("cls");

            printHeader();

            addRecipient();

        }

        if (choice == 7)

        {

            system("cls");

            printHeader();

            deleteRecipient();

        }

        if (choice == 8)

        {

            system("cls");

            printHeader();

            updateRecipient();

        }

        if (choice == 9)

        {

            system("cls");

            printHeader();

            searchRecipient();

        }

        if (choice == 10)

        {

            system("cls");

            printHeader();

            viewRecipient();

        }

    }

}

void printEMenu()

{

    cout << endl;

    cout << "1.  Add New Donor..." << endl;

    cout << "2.  Delete Donor... " << endl;

    cout << "3.  Update Donor Details... " << endl;

    cout << "4.  Search For Donor... " << endl;

    cout << "5.  View Details of all Donors... " << endl;

    cout << endl;

    cout << "6.  Add New Recipient..." << endl;

    cout << "7.  Delete Recipient... " << endl;

    cout << "8.  Update Recipient Details... " << endl;

    cout << "9.  Search For Recipient... " << endl;

    cout << "10. View Details of all Recipients... " << endl;

    cout << endl;

    cout << "11. Log out... " << endl;

    cout << endl;

    cout << "Enter Your Option: ";

}

void addDonor()

{

    cout << endl;

    string menu = "Employee Menu";

    string subMenu = "Add Donor ";

    menuName(menu, subMenu);

    cout << endl;

    cout << endl;

    cout << "Enter Details of the New Donor:-" << endl;

    cout << endl;

    cout << "Enter Name: ";

    nameD[indexD] = isAlpha(nameD[indexD]);

    cout << "Enter Age(+18): ";

    ageD[indexD] = isNum(ageD[indexD]);

    cout << "Enter Bloodgroup: ";

    bloodgroupD[indexD] = isBG(bloodgroupD[indexD]);

    cout << "Enter City: ";

    cityD[indexD] = isAlpha(cityD[indexD]);

    cout << "Enter Contact No.(11 numbers): ";

    contactD[indexD] = contactCheck(contactD[indexD]);

    contributerD[indexD] = contributer;

    DonorToFile(nameD[indexD], ageD[indexD], bloodgroupD[indexD], cityD[indexD], contactD[indexD], contributerD[indexD]);

    indexD++;

    cout << endl;

    cout << "Donor Added Sucessfully...";

    Sleep(300);

    cout << endl;

    cout << "Press any key for back...";

    getch();

}

void deleteDonor()

{

    int index;

    string deleteName;

    cout << endl;

    string menu = "Employee Menu";

    string subMenu = "Delete Donor ";

    menuName(menu, subMenu);

    cout << endl;

    cout << endl;

    cout << "Enter Name of the Donor: ";

    cin.clear();

    cin.sync();

    getline(cin >> ws, deleteName);

    cout << endl;

    bool notFound = true;

    for (int idx = 0; idx < 100; idx++)

    {

        if ((deleteName == nameD[idx]) && (contributer == contributerD[idx]))

        {

            index = idx;

            cout << left << setw(20) << "Name" << left << setw(20) << "Age" << left << setw(20) << "Bloodgroup" << left << setw(20) << "City" << left << setw(20) << "Contact No." << endl;

            cout << endl;

            cout << left << setw(20) << nameD[index] << left << setw(20) << ageD[index] << left << setw(20) << bloodgroupD[index] << left << setw(20) << cityD[index] << left << setw(20) << contactD[index] << endl;

            for (int j = idx; j <= 100 - 1; j++)

            {

                nameD[j] = nameD[j + 1];

                ageD[j] = ageD[j + 1];

                bloodgroupD[j] = bloodgroupD[j + 1];

                cityD[j] = cityD[j + 1];

                contactD[j] = contactD[j + 1];

                contributerD[j] = contributerD[j + 1];

            }

            indexD--;

            updateDonorFile();

            cout << endl;

            cout << "Donor Removed..." << endl;

            notFound = true;

            break;

        }

        else

        {

            notFound = false;

        }

    }

    if (notFound == false)

    {

        setcolor(12);

        cout << endl;

        cout << "Donor Not Found" << endl;

        cout << endl;

        setcolor(15);

        cout << "Press any key for back...";

        getch();

    }

    else

    {

        cout << endl;

        cout << "Press any key for back...";

        getch();

    }

}

void updateDonor()

{

    int index;

    string updateName;

    cout << endl;

    string menu = "Employee Menu";

    string subMenu = "Update Donor ";

    menuName(menu, subMenu);

    cout << endl;

    cout << endl;

    cout << "Enter Name of the Donor: ";

    cin.clear();

    cin.sync();

    getline(cin >> ws, updateName);

    cout << endl;

    bool notFound = true;

    for (int idx = 0; idx < 100; idx++)

    {

        if (updateName == nameD[idx] && (contributer == contributerD[idx]))

        {

            index = idx;

            cout << left << setw(20) << "Name" << left << setw(20) << "Age" << left << setw(20) << "Bloodgroup" << left << setw(20) << "City" << left << setw(20) << "Contact No." << endl;

            cout << endl;

            cout << left << setw(20) << nameD[index] << left << setw(20) << ageD[index] << left << setw(20) << bloodgroupD[index] << left << setw(20) << cityD[index] << left << setw(20) << contactD[index] << endl;

            cout << endl;

            cout << "Enter Name: ";

            nameD[index] = isAlpha(nameD[index]);

            cout << "Enter Age(+18): ";

            ageD[index] = isNum(ageD[index]);

            cout << "Enter Bloodgroup: ";

            bloodgroupD[index] = isBG(bloodgroupD[index]);

            cout << "Enter City: ";

            cityD[index] = isAlpha(cityD[index]);

            cout << "Enter Contact No. (11 numbers): ";

            contactD[index] = contactCheck(contactD[index]);

            contributerD[index] = contributer;

            updateDonorFile();

            cout << endl;

            cout << "Donor Updated..." << endl;

            notFound = true;

            break;

        }

        else

        {

            notFound = false;

        }

    }

    if (notFound == false)

    {

        setcolor(12);

        cout << "Donor Not Found" << endl;

        cout << endl;

        setcolor(15);

        cout << "Press any key for back...";

        getch();

    }

    else

    {

        cout << endl;

        cout << "Press any key for back...";

        getch();

    }

}

void searchDonor()

{

    int index;

    string searchBG;

    string check;

    cout << endl;

    string menu = "Employee Menu";

    string subMenu = "Search Donor ";

    menuName(menu, subMenu);

    cout << endl;

    cout << endl;

    cout << "Enter Bloodgroup of the Donor: ";

    cin >> searchBG;

    bool notFound = true;

    bool one = false;

    cout << endl;

    cout << left << setw(20) << "Name" << left << setw(20) << "Age" << left << setw(20) << "Bloodgroup" << left << setw(20) << "City" << left << setw(20) << "Contact No." << endl;

    cout << endl;

    for (int idx = 0; idx < 100; idx++)

    {

        if (searchBG == bloodgroupD[idx] && (contributer == contributerD[idx]))

        {

            index = idx;

            cout << left << setw(20) << nameD[index] << left << setw(20) << ageD[index] << left << setw(20) << bloodgroupD[index] << left << setw(20) << cityD[index] << left << setw(20) << contactD[index] << endl;

            notFound = true;

            one = true;

        }

        else if (one == false)

        {

            notFound = false;

        }

    }

    if (notFound == false)

    {

        setcolor(12);

        cout << "Donor Not Found" << endl;

        cout << endl;

        setcolor(15);

        cout << "Press any key for back...";

        getch();

    }

    else

    {

        cout << endl;

        cout << "Press any key for back...";

        getch();

    }

}

void viewDonor()

{

    bool flag = false;

    cout << endl;

    string menu = "Employee Menu";

    string subMenu = "View All Donors ";

    menuName(menu, subMenu);

    cout << endl;

    cout << endl;

    cout << "Following Are the Donors: " << endl;

    cout << endl;

    cout << left << setw(20) << "Name" << left << setw(20) << "Age" << left << setw(20) << "Bloodgroup" << left << setw(20) << "City" << left << setw(20) << "Contact No." << endl;

    cout << endl;

    for (int index = 0; index < 100; index++)

    {

        if ((nameD[index] != "") && (contributer == contributerD[index]))

        {

            cout << left << setw(20) << nameD[index] << left << setw(20) << ageD[index] << left << setw(20) << bloodgroupD[index] << left << setw(20) << cityD[index] << left << setw(20) << contactD[index] << endl;

            flag = true;

        }

    }

    if (flag == false)

    {

        setcolor(12);

        cout << "Donors not Found" << endl;

        cout << "Add Donors to View Donors" << endl;

        cout << endl;

        setcolor(15);

    }

    cout << endl;

    cout << "Press any key for back...";

    getch();

}

void addRecipient()

{

    cout << endl;

    string menu = "Employee Menu";

    string subMenu = "Add Recipient ";

    menuName(menu, subMenu);

    cout << endl;

    cout << endl;

    cout << "Enter Details of the New Recipient:-" << endl;

    cout << endl;

    cout << "Enter Name: ";

    nameR[indexR] = isAlpha(nameR[indexR]);

    cout << "Enter Age(+18): ";

    ageR[indexR] = isNum(ageR[indexR]);

    cout << "Enter Bloodgroup: ";

    bloodgroupR[indexR] = isBG(bloodgroupR[indexR]);

    cout << "Enter City: ";

    cityR[indexR] = isAlpha(cityR[indexR]);

    cout << "Enter Contact No.(11 numbers): ";

    contactR[indexR] = contactCheck(contactR[indexR]);

    contributerR[indexR] = contributer;

    RecipientToFile(nameR[indexR], ageR[indexR], bloodgroupR[indexR], cityR[indexR], contactR[indexR], contributerR[indexR]);

    indexR++;

    cout << endl;

    cout << "Recipient Added Sucessfully...";

    Sleep(300);

    cout << endl;

    cout << "Press any key for back...";

    getch();

}

void deleteRecipient()

{

    int index;

    string deleteName;

    cout << endl;

    string menu = "Employee Menu";

    string subMenu = "Delete Recipient ";

    menuName(menu, subMenu);

    cout << endl;

    cout << endl;

    cout << "Enter Name of the Recipient: ";

    cin.clear();

    cin.sync();

    getline(cin >> ws, deleteName);

    cout << endl;

    bool notFound = true;

    for (int idx = 0; idx < 100; idx++)

    {

        if (deleteName == nameR[idx] && (contributer == contributerR[idx]))

        {

            index = idx;

            cout << left << setw(20) << "Name" << left << setw(20) << "Age" << left << setw(20) << "Bloodgroup" << left << setw(20) << "City" << left << setw(20) << "Contact No." << endl;

            cout << endl;

            cout << left << setw(20) << nameR[index] << left << setw(20) << ageR[index] << left << setw(20) << bloodgroupR[index] << left << setw(20) << cityR[index] << left << setw(20) << contactR[index] << endl;

            for (int j = idx; j <= 100 - 1; j++)

            {

                nameR[j] = nameR[j + 1];

                ageR[j] = ageR[j + 1];

                bloodgroupR[j] = bloodgroupR[j + 1];

                cityR[j] = cityR[j + 1];

                contactR[j] = contactR[j + 1];

                contributerR[j] = contributerR[j + 1];

            }

            indexR--;

            updateRecipientFile();

            cout << endl;

            cout << "Recipient Removed..." << endl;

            notFound = true;

            break;

        }

        else

        {

            notFound = false;

        }

    }

    if (notFound == false)

    {

        setcolor(12);

        cout << endl;

        cout << "Recipient Not Found" << endl;

        setcolor(15);

        cout << endl;

        cout << "Press any key for back...";

        getch();

    }

    else

    {

        cout << endl;

        cout << "Press any key for back...";

        getch();

    }

}

void updateRecipient()

{

    int index;

    string updateName;

    cout << endl;

    string menu = "Employee Menu";

    string subMenu = "Update Recipient ";

    menuName(menu, subMenu);

    cout << endl;

    cout << endl;

    cout << "Enter Name of the Recipient: ";

    cin.clear();

    cin.sync();

    getline(cin >> ws, updateName);

    cout << endl;

    bool notFound = true;

    for (int idx = 0; idx < 100; idx++)

    {

        if (updateName == nameR[idx] && (contributer == contributerR[idx]))

        {

            index = idx;

            cout << left << setw(20) << "Name" << left << setw(20) << "Age" << left << setw(20) << "Bloodgroup" << left << setw(20) << "City" << left << setw(20) << "Contact No." << endl;

            cout << endl;

            cout << left << setw(20) << nameR[index] << left << setw(20) << ageR[index] << left << setw(20) << bloodgroupR[index] << left << setw(20) << cityR[index] << left << setw(20) << contactR[index] << endl;

            cout << endl;

            cout << "Enter Name: ";

            nameR[index] = isAlpha(nameR[index]);

            cout << "Enter Age(+18): ";

            ageR[index] = isNum(ageR[index]);

            cout << "Enter Bloodgroup: ";

            bloodgroupR[index] = isBG(bloodgroupR[index]);

            cout << "Enter City: ";

            cityR[index] = isAlpha(cityR[index]);

            cout << "Enter Contact No.(11 numbers): ";

            contactR[index] = contactCheck(contactR[index]);

            contributerR[index] = contributer;

            updateRecipientFile();

            cout << endl;

            cout << "Recipient Updated..." << endl;

            notFound = true;

            break;

        }

        else

        {

            notFound = false;

        }

    }

    if (notFound == false)

    {

        setcolor(12);

        cout << "Recipient Not Found" << endl;

        cout << endl;

        setcolor(15);

        cout << "Press any key for back...";

        getch();

    }

    else

    {

        cout << endl;

        cout << "Press any key for back...";

        getch();

    }

}

void searchRecipient()

{

    int index;

    string searchBG;

    cout << endl;

    string menu = "Employee Menu";

    string subMenu = "Search Recipient ";

    menuName(menu, subMenu);

    cout << endl;

    cout << endl;

    cout << "Enter Bloodgroup of the Recipient: ";

    cin >> searchBG;

    bool notFound = true;

    bool one = false;

    cout << endl;

    cout << left << setw(20) << "Name" << left << setw(20) << "Age" << left << setw(20) << "Bloodgroup" << left << setw(20) << "City" << left << setw(20) << "Contact No." << endl;

    cout << endl;

    for (int idx = 0; idx < 100; idx++)

    {

        if (searchBG == bloodgroupR[idx] && (contributer == contributerR[idx]))

        {

            index = idx;

            cout << left << setw(20) << nameR[index] << left << setw(20) << ageR[index] << left << setw(20) << bloodgroupR[index] << left << setw(20) << cityR[index] << left << setw(20) << contactR[index] << endl;

            notFound = true;

            one = true;

        }

        else if (one == false)

        {

            notFound = false;

        }

    }

    if (notFound == false)

    {

        setcolor(12);

        cout << "Recipient Not Found" << endl;

        cout << endl;

        setcolor(15);

        cout << "Press any key for back...";

        getch();

    }

    else

    {

        cout << endl;

        cout << "Press any key for back...";

        getch();

    }

}

void viewRecipient()

{

    bool flag = false;

    cout << endl;

    string menu = "Employee Menu";

    string subMenu = "View All Recipients ";

    menuName(menu, subMenu);

    cout << endl;

    cout << endl;

    cout << "Following Are the Recipients: " << endl;

    cout << endl;

    cout << left << setw(20) << "Name" << left << setw(20) << "Age" << left << setw(20) << "Bloodgroup" << left << setw(20) << "City" << left << setw(20) << "Contact No." << endl;

    cout << endl;

    for (int index = 0; index < 100; index++)

    {

        if ((nameR[index] != "") && (contributer == contributerR[index]))

        {

            cout << left << setw(20) << nameR[index] << left << setw(20) << ageR[index] << left << setw(20) << bloodgroupR[index] << left << setw(20) << cityR[index] << left << setw(20) << contactR[index] << endl;

            flag = true;

        }

    }

    if (flag == false)

    {

        setcolor(12);

        cout << "Recipients not Found" << endl;

        cout << "Add Recipients to View Recpients" << endl;

        setcolor(15);

        cout << endl;

    }

    cout << endl;

    cout << "Press any key for back...";

    getch();

}

void addEmployee()

{

    cout << endl;

    string menu = "Admin Menu";

    string subMenu = "Add Employee";

    menuName(menu, subMenu);

    cout << endl;

    cout << endl;

    cout << "Enter Details of the New Employee:-" << endl;

    cout << endl;

    cout << "Enter Name: ";

    nameE[indexE] = isAlpha(nameE[indexE]);

    cout << "Enter Age(+18): ";

    ageE[indexE] = isNum(ageE[indexE]);

    cout << "Enter CNIC(13 numbers): ";

    cnicE[indexE] = cnicCheck(cnicE[indexE]);

    cout << "Enter Contact No(11 numbers): ";

    contactE[indexE] = contactCheck(contactE[indexE]);

    cout << "Enter Username: ";

    usernameE[indexE] = usercheck(usernameE[indexE]);

    cout << "Enter Password: ";

    cin >> passwordE[indexE];

    EmployeeToFile(nameE[indexE], ageE[indexE], cnicE[indexE], contactE[indexE], usernameE[indexE], passwordE[indexE]);

    indexE++;

    cout << endl;

    cout << "Employee Added Sucessfully...";

    Sleep(300);

    cout << endl;

    cout << "Press any key for back...";

    getch();

}

void deleteEmployee()

{

    int index;

    string deleteName;

    cout << endl;

    string menu = "Admin Menu";

    string subMenu = "Delete Employee";

    menuName(menu, subMenu);

    cout << endl;

    cout << endl;

    cout << "Enter Username of the Employee: ";

    cin >> deleteName;

    cout << endl;

    bool notFound = true;

    for (int idx = 0; idx < 100; idx++)

    {

        if (deleteName == usernameE[idx])

        {

            index = idx;

            cout << left << setw(20) << "Name" << left << setw(20) << "Age" << left << setw(20) << "CNIC" << left << setw(20) << "Contact No." << left << setw(20) << "Username" << left << setw(20) << "Password" << endl;

            cout << endl;

            cout << left << setw(20) << nameE[index] << left << setw(20) << ageE[index] << left << setw(20) << cnicE[index] << left << setw(20) << contactE[index] << left << setw(20) << usernameE[index] << left << setw(20) << passwordE[index] << endl;

            for (int j = idx; j <= 100 - 1; j++)

            {

                nameE[j] = nameE[j + 1];

                ageE[j] = ageE[j + 1];

                cnicE[j] = cnicE[j + 1];

                contactE[j] = contactE[j + 1];

                usernameE[j] = usernameE[j + 1];

                passwordE[j] = passwordE[j + 1];

            }

            indexR--;

            updateEmployeeFile();

            cout << endl;

            cout << "Employee Removed..." << endl;

            notFound = true;

            break;

        }

        else

        {

            notFound = false;

        }

    }

    if (notFound == false)

    {

        setcolor(12);

        cout << endl;

        cout << "Employee Not Found" << endl;

        cout << endl;

        setcolor(15);

        cout << "Press any key for back...";

        getch();

    }

    else

    {

        cout << endl;

        cout << "Press any key for back...";

        getch();

    }

}

void updateEmployee()

{

    int index;

    string updateName;

    cout << endl;

    string menu = "Admin Menu";

    string subMenu = "Update Employee";

    menuName(menu, subMenu);

    cout << endl;

    cout << endl;

    cout << "Enter CNIC of the Employee: ";

    cin >> updateName;

    cout << endl;

    bool notFound = true;

    for (int idx = 0; idx < 100; idx++)

    {

        if (updateName == cnicE[idx])

        {

            index = idx;

            cout << left << setw(20) << "Name" << left << setw(20) << "Age" << left << setw(20) << "CNIC" << left << setw(20) << "Contact No." << left << setw(20) << "Username" << left << setw(20) << "Password" << endl;

            cout << endl;

            cout << left << setw(20) << nameE[index] << left << setw(20) << ageE[index] << left << setw(20) << cnicE[index] << left << setw(20) << contactE[index] << left << setw(20) << usernameE[index] << left << setw(20) << passwordE[index] << endl;

            cout << endl;

            cout << "Enter Name: ";

            nameE[index] = isAlpha(nameE[index]);

            cout << "Enter Age(+18): ";

            ageE[index] = isNum(ageE[index]);

            cout << "Enter CNIC(13 numbers): ";

            cnicE[index] = cnicCheck(cnicE[index]);

            cout << "Enter Contact No(11 numbers): ";

            contactE[index] = contactCheck(contactE[index]);

            cout << "Enter Username: ";

            usernameE[index] = usercheck(usernameE[index]);

            cout << "Enter Password: ";

            cin >> passwordE[index];

            updateEmployeeFile();

            cout << endl;

            cout << "Employee Updated..." << endl;

            notFound = true;

            break;

        }

        else

        {

            notFound = false;

        }

    }

    if (notFound == false)

    {

        setcolor(12);

        cout << "Employee Not Found" << endl;

        cout << endl;

        setcolor(15);

        cout << "Press any key for back...";

        getch();

    }

    else

    {

        cout << endl;

        cout << "Press any key for back...";

        getch();

    }

}

void searchEmployee()

{

    int index;

    string searchName;

    cout << endl;

    string menu = "Admin Menu";

    string subMenu = "Search Employee";

    menuName(menu, subMenu);

    cout << endl;

    cout << endl;

    cout << "Enter CNIC of the Employee: ";

    cin >> searchName;

    bool notFound = true;

    bool one = false;

    cout << endl;

    cout << left << setw(20) << "Name" << left << setw(20) << "Age" << left << setw(20) << "CNIC" << left << setw(20) << "Contact No." << left << setw(20) << "Username" << left << setw(20) << "Password" << endl;

    cout << endl;

    for (int idx = 0; idx < 100; idx++)

    {

        if (searchName == cnicE[idx])

        {

            index = idx;

            cout << left << setw(20) << nameE[index] << left << setw(20) << ageE[index] << left << setw(20) << cnicE[index] << left << setw(20) << contactE[index] << left << setw(20) << usernameE[index] << left << setw(20) << passwordE[index] << endl;

            notFound = true;

            one = true;

        }

        else if (one == false)

        {

            notFound = false;

        }

    }

    if (notFound == false)

    {

        setcolor(12);

        cout << "Employee Not Found" << endl;

        cout << endl;

        setcolor(15);

        cout << "Press any key for back...";

        getch();

    }

    else

    {

        cout << endl;

        cout << "Press any key for back...";

        getch();

    }

}

void viewEmployee()

{

    bool flag = false;

    cout << endl;

    string menu = "Admin Menu";

    string subMenu = "View All Employees";

    menuName(menu, subMenu);

    cout << endl;

    cout << endl;

    cout << "Following Are the Employees: " << endl;

    cout << endl;

    cout << left << setw(20) << "Name" << left << setw(20) << "Age" << left << setw(20) << "CNIC" << left << setw(20) << "Contact No." << left << setw(20) << "Username" << left << setw(20) << "Password" << endl;

    cout << endl;

    for (int index = 0; index < 100; index++)

    {

        if (nameE[index] != "")

        {

            cout << left << setw(20) << nameE[index] << left << setw(20) << ageE[index] << left << setw(20) << cnicE[index] << left << setw(20) << contactE[index] << left << setw(20) << usernameE[index] << left << setw(20) << passwordE[index] << endl;

            flag = true;

        }

    }

    if (flag == false)

    {

        setcolor(12);

        cout << "Employees not Found" << endl;

        cout << "Add Employees to View Employees" << endl;

        cout << endl;

        setcolor(15);

    }

    cout << endl;

    cout << "Press any key for back...";

    getch();

}

void searchDonorbyAdmin()

{

    int index;

    string searchName;

    cout << endl;

    string menu = "Admin Menu";

    string subMenu = "Search Donor";

    menuName(menu, subMenu);

    cout << endl;

    cout << endl;

    cout << "Enter Bloodgroup of the Donor: ";

    cin >> searchName;

    bool notFound = true;

    bool one = false;

    cout << endl;

    cout << left << setw(20) << "Name" << left << setw(20) << "Age" << left << setw(20) << "Bloodgroup" << left << setw(20) << "City" << left << setw(20) << "Contact No." << left << setw(20) << "Contributer" << endl;

    cout << endl;

    for (int idx = 0; idx < 100; idx++)

    {

        if (searchName == bloodgroupD[idx])

        {

            index = idx;

            cout << left << setw(20) << nameD[index] << left << setw(20) << ageD[index] << left << setw(20) << bloodgroupD[index] << left << setw(20) << cityD[index] << left << setw(20) << contactD[index] << left << setw(20) << contributerD[index] << endl;

            notFound = true;

            one = true;

        }

        else if (one == false)

        {

            notFound = false;

        }

    }

    if (notFound == false)

    {

        setcolor(12);

        cout << "Donor Not Found" << endl;

        cout << endl;

        setcolor(15);

        cout << "Press any key for back...";

        getch();

    }

    else

    {

        cout << endl;

        cout << "Press any key for back...";

        getch();

    }

}

void viewDonorbyAdmin()

{

    bool flag = false;

    cout << endl;

    string menu = "Admin Menu";

    string subMenu = "View All Donors";

    menuName(menu, subMenu);

    cout << endl;

    cout << endl;

    cout << "Following Are the Donors: " << endl;

    cout << endl;

    cout << left << setw(20) << "Name" << left << setw(20) << "Age" << left << setw(20) << "Bloodgroup" << left << setw(20) << "City" << left << setw(20) << "Contact No." << left << setw(20) << "Contributer" << endl;

    cout << endl;

    for (int index = 0; index < 100; index++)

    {

        if (nameD[index] != "")

        {

            cout << left << setw(20) << nameD[index] << left << setw(20) << ageD[index] << left << setw(20) << bloodgroupD[index] << left << setw(20) << cityD[index] << left << setw(20) << contactD[index] << left << setw(20) << contributerD[index] << endl;

            flag = true;

        }

    }

    if (flag == false)

    {

        setcolor(12);

        cout << "Donors not Found" << endl;

        cout << "Add Donors to View Donors" << endl;

        setcolor(15);

        cout << endl;

    }

    cout << endl;

    cout << "Press any key for back...";

    getch();

}

void searchRecipientbyAdmin()

{

    int index;

    string searchName;

    cout << endl;

    string menu = "Admin Menu";

    string subMenu = "Search Recipient";

    menuName(menu, subMenu);

    cout << endl;

    cout << endl;

    cout << "Enter Bloodgroup of the Recipient: ";

    cin >> searchName;

    bool notFound = true;

    bool one = false;

    cout << endl;

    cout << left << setw(20) << "Name" << left << setw(20) << "Age" << left << setw(20) << "Bloodgroup" << left << setw(20) << "City" << left << setw(20) << "Contact No." << left << setw(20) << "Contributer" << endl;

    cout << endl;

    for (int idx = 0; idx < 100; idx++)

    {

        if (searchName == bloodgroupR[idx])

        {

            index = idx;

            cout << left << setw(20) << nameR[index] << left << setw(20) << ageR[index] << left << setw(20) << bloodgroupR[index] << left << setw(20) << cityR[index] << left << setw(20) << contactR[index] << left << setw(20) << contributerR[index] << endl;

            notFound = true;

            one = true;

        }

        else if (one == false)

        {

            notFound = false;

        }

    }

    if (notFound == false)

    {

        setcolor(12);

        cout << "Recipient Not Found" << endl;

        cout << endl;

        setcolor(15);

        cout << "Press any key for back...";

        getch();

    }

    else

    {

        cout << endl;

        cout << "Press any key for back...";

        getch();

    }

}

void viewRecipientbyAdmin()

{

    bool flag = false;

    cout << endl;

    string menu = "Admin Menu";

    string subMenu = "View All Recipients";

    menuName(menu, subMenu);

    cout << endl;

    cout << endl;

    cout << "Following Are the Recipients: " << endl;

    cout << endl;

    cout << left << setw(20) << "Name" << left << setw(20) << "Age" << left << setw(20) << "Bloodgroup" << left << setw(20) << "City" << left << setw(20) << "Contact No." << left << setw(20) << "Contributer" << endl;

    cout << endl;

    for (int index = 0; index < 100; index++)

    {

        if (nameR[index] != "")

        {

            cout << left << setw(20) << nameR[index] << left << setw(20) << ageR[index] << left << setw(20) << bloodgroupR[index] << left << setw(20) << cityR[index] << left << setw(20) << contactR[index] << left << setw(20) << contributerR[index] << endl;

            flag = true;

        }

    }

    if (flag == false)

    {

        setcolor(12);

        cout << "Recipients not Found" << endl;

        cout << "Add Recipients to View Recipients" << endl;

        setcolor(15);

        cout << endl;

    }

    cout << endl;

    cout << "Press any key for back...";

    getch();

}

void menuName(string menu, string subMenu)

{

    setcolor(02);

    cout << "  " << menu << " > " << subMenu << endl;

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

    setcolor(15);

}

string setcolor(unsigned short color)

{

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

    SetConsoleTextAttribute(hcon, color);

    return "";

}

string isAlpha(string input)

{

    cin.clear();

    cin.sync();

    getline(cin >> ws, input);

    int size;

    int check;

    bool flap;

    while (true)

    {

        size = input.length();

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

        {

            check = int(input[i]);

            if ((check >= 65 && check <= 90) || (check >= 97 && check <= 122) || input[i] == ' ')

            {

                flap = true;

            }

            else

            {

                flap = false;

                break;

            }

        }

        if (flap == true)

        {

            return input;

        }

        else

        {

            cin.clear();

            cin.sync();

            cout << "Wrong Charater..." << endl;

            cout << "Enter Again: ";

            getline(cin >> ws, input);

        }

    }

}

string isNum(string input)

{

    cin >> input;

    int x;

    int size;

    int check;

    bool flap;

    while (true)

    {

        size = input.length();

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

        {

            if (input[i] != ' ')

            {

                check = int(input[i]);

                if ((check >= 48 && check <= 57))

                {

                    flap = true;

                }

                else

                {

                    flap = false;

                    break;

                }

            }

        }

        if (flap == true)

        {

            x = stoi(input);

            if (x >= 18 && x <= 60)

            {

                return input;

            }

            else

            {

                cin.clear();

                cin.sync();

                cout << "Wrong Age..." << endl;

                cout << "Enter age: ";

                cin >> input;

            }

        }

        else

        {

            cin.clear();

            cin.sync();

            cout << "Wrong Age..." << endl;

            cout << "Enter age: ";

            cin >> input;

        }

    }

    return 0;

}

string isBG(string input)

{

    cin >> input;

    while (true)

    {

        if (input == "A+" || input == "A-" || input == "B+" || input == "B-" || input == "AB+" || input == "AB-" || input == "O+" || input == "O-")

        {

            break;

        }

        else

        {

            cin.clear();

            cin.sync();

            cout << "Wrong Bloodgroup..." << endl;

            cout << "Enter Blood: ";

            cin >> input;

        }

    }

    return input;

}

string contactCheck(string contact)

{

    cin >> contact;

    int size;

    int check;

    bool flap;

    while (true)

    {

        size = contact.length();

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

        {

            if (contact[i] != ' ')

            {

                check = int(contact[i]);

                if ((check >= 48 && check <= 57) && (size == 11))

                {

                    flap = true;

                }

                else

                {

                    flap = false;

                    break;

                }

            }

        }

        if (flap == true)

        {

            return contact;

        }

        else

        {

            cin.clear();

            cin.sync();

            cout << "Wrong Contact info..." << endl;

            cout << "Enter Contact No (11 numbers): ";

            cin >> contact;

        }

    }

}

string cnicCheck(string cnic)

{

    cin >> cnic;

    int size;

    int check;

    bool flap;

    while (true)

    {

        size = cnic.length();

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

        {

            if (cnic[i] != ' ')

            {

                check = int(cnic[i]);

                if ((check >= 48 && check <= 57) && (size == 13))

                {

                    flap = true;

                }

                else

                {

                    flap = false;

                    break;

                }

            }

        }

        if (flap == true)

        {

            return cnic;

        }

        else

        {

            cin.clear();

            cin.sync();

            cout << "Wrong CNIC..." << endl;

            cout << "Enter CNIC (13 numbers): ";

            cin >> cnic;

        }

    }

}

int choiceCheck(int choice)

{

    cin >> choice;

    while (true)

    {

        if (cin.fail())

        {

            cin.clear();

            cin.sync();

            cout << "Wrong Option..." << endl;

            cout << "Enter Option: ";

            cin >> choice;

        }

        if (!cin.fail())

        {

            break;

        }

    }

    return choice;

}

string usercheck(string username)

{

    cin >> username;

    for (int i = 0; i < 100; i++)

    {

        if (username == usernameE[i])

        {

            cin.clear();

            cin.sync();

            cout << "Username Already Present..." << endl;

            cout << "Enter Username: ";

            cin >> username;

        }

        else

        {

            continue;

        }

    }

    return username;

}

void gotoxy(int x, int y)

{

    COORD coordinates;

    coordinates.X = x;

    coordinates.Y = y;

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

}

void DonorToFile(string name, string age, string bloodgroup, string city, string contact, string contributer)

{

    fstream donorData;

    donorData.open("DonorData.txt", ios::app);

    donorData << name << "," << age << "," << bloodgroup << "," << city << "," << contact << "," << contributer << endl;

    donorData.close();

}

void RecipientToFile(string name, string age, string bloodgroup, string city, string contact, string contributer)

{

    fstream recipientData;

    recipientData.open("RecipientData.txt", ios::app);

    recipientData << name << "," << age << "," << bloodgroup << "," << city << "," << contact << "," << contributer << endl;

    recipientData.close();

}

void EmployeeToFile(string name, string age, string cnic, string contact, string username, string password)

{

    fstream employeeData;

    employeeData.open("EmployeeData.txt", ios::app);

    employeeData << name << "," << age << "," << cnic << "," << contact << "," << username << "," << password << endl;

    employeeData.close();

}

void LoadDonor()

{

    fstream donorData;

    string line = "";

    donorData.open("DonorData.txt", ios::in);

    while (!donorData.eof())

    {

        getline(donorData, line);

        nameD[indexD] = Dataparse(line, 1);

        ageD[indexD] = Dataparse(line, 2);

        bloodgroupD[indexD] = Dataparse(line, 3);

        cityD[indexD] = Dataparse(line, 4);

        contactD[indexD] = Dataparse(line, 5);

        contributerD[indexD] = Dataparse(line, 6);

        indexD++;

    }

    donorData.close();

}

void LoadRecipient()

{

    fstream recipientData;

    string line = "";

    recipientData.open("RecipientData.txt", ios::in);

    while (!recipientData.eof())

    {

        getline(recipientData, line);

        nameR[indexR] = Dataparse(line, 1);

        ageR[indexR] = Dataparse(line, 2);

        bloodgroupR[indexR] = Dataparse(line, 3);

        cityR[indexR] = Dataparse(line, 4);

        contactR[indexR] = Dataparse(line, 5);

        contributerR[indexR] = Dataparse(line, 6);

        indexR++;

    }

    recipientData.close();

}

void LoadEmployee()

{

    fstream employeeData;

    string line = "";

    employeeData.open("EmployeeData.txt", ios::in);

    while (!employeeData.eof())

    {

        getline(employeeData, line);

        nameE[indexE] = Dataparse(line, 1);

        ageE[indexE] = Dataparse(line, 2);

        cnicE[indexE] = Dataparse(line, 3);

        contactE[indexE] = Dataparse(line, 4);

        usernameE[indexE] = Dataparse(line, 5);

        passwordE[indexE] = Dataparse(line, 6);

        indexE++;

    }

    employeeData.close();

}

void updateDonorFile()

{

    fstream donorData;

    donorData.open("DonorData.txt", ios::out);

    for (int i = 0; i < indexD; i++)

    {

        if (nameD[i] != "")

        {

            donorData << nameD[i] << "," << ageD[i] << "," << bloodgroupD[i] << "," << cityD[i] << "," << contactD[i] << "," << contributerD[i] << endl;

        }

    }

    donorData.close();

}

void updateRecipientFile()

{

    fstream recipientData;

    recipientData.open("RecipientData.txt", ios::out);

    for (int i = 0; i < indexR; i++)

    {

        if (nameR[i] != "")

        {

            recipientData << nameR[i] << "," << ageR[i] << "," << bloodgroupR[i] << "," << cityR[i] << "," << contactR[i] << "," << contributerR[i] << endl;

        }

    }

    recipientData.close();

}

void updateEmployeeFile()

{

    fstream employeeData;

    employeeData.open("EmployeeData.txt", ios::out);

    for (int i = 0; i < indexE; i++)

    {

        if (nameE[i] != "")

        {

            employeeData << nameE[i] << "," << ageE[i] << "," << cnicE[i] << "," << contactE[i] << "," << usernameE[i] << "," << passwordE[i] << endl;

        }

    }

    employeeData.close();

}

string Dataparse(string line, int field)

{

    int comma = 1;

    string item = "";

    int length = line.length();

    for (int i = 0; i < length; i++)

    {

        if (line[i] == ',')

        {

            comma++;

        }

        else if (field == comma)

        {

            item = item + line[i];

        }

    }

    return item;

}

# **Weakness in the Application:**

* The option of stock of different blood group separately which is donated is not available in the application.
* Employee information is not changed by his own.
* The application is less user friendly because user need to type option number to use application.

# **Future Directions:**

* If recipient does not get his desire blood group, then an option of requests is enabled and employee get notify which blood group is needed to recipient.
* If employee wanted to change his password and username, he can change it by his own without the permission of admin.
* I also add the option of blood bank where the stock of donated blood is show. This option is only enabled for admin.
* Make it more user friendly by adding more graphics and use mouse pointer for selecting options instead of adding option number.

**Student Reg. No.:**  2022-CS-65  **Student Name:**  Muhammad Wali Ahmad

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **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-65  **Student Name:**  Muhammad Wali Ahmad

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **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:** |  |