**HOSPITAL MANAGEMENT SYSTEM**



Session: 2021 – 2024

**Submitted by:**

Muhammad Abubakar Siddique Farooqi 2021-CS-171

**Supervised by:**

Maida Shahid

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

**Description:**

This System automates the daily activities of hospital related to management. It keeps records of doctors, patients and different services provided by hospital. It provides the facility of retrieving particular doctor or patient from hospital storage system. It also provides the facility of recommending doctor to a patient. This system also provides validation on passwords and number type input. The whole system is password protected.

**Users of System:**

There are three users of system:

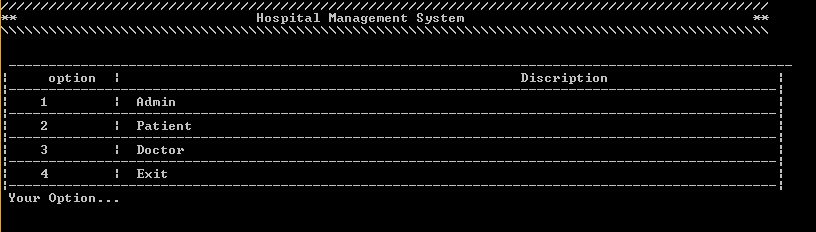
1. Admin
2. Patient
3. Doctor

**Functional Requirements:**

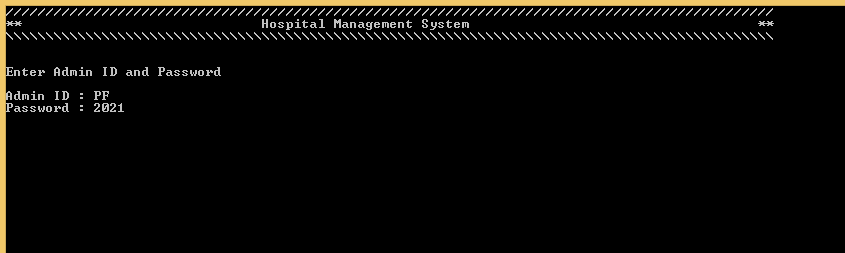
* Every user must log-in with an ID and Password to use the System.
* Admin can add Patients.
* Admin can add Doctor.
* Admin can add Services.
* Admin can see details of all Doctors.
* Admin can see details of all Patients.
* Admin can see Sorted list of Doctors.
* Admin can see particular Doctor.
* Admin can see particular Patient.
* Admin can see Doctor’s Status.
* Admin can see all Services
* Admin can change his password.
* Patient can see his details
* Patient can see Doctor’s status.
* Patient can see all Services.
* Patient can change his password.
* Doctor can see his details.
* Doctor can see particular Patient details.
* Doctor can see all Services.
* Doctor can change his Status.
* Doctor can change his password.

**Wireframes:**

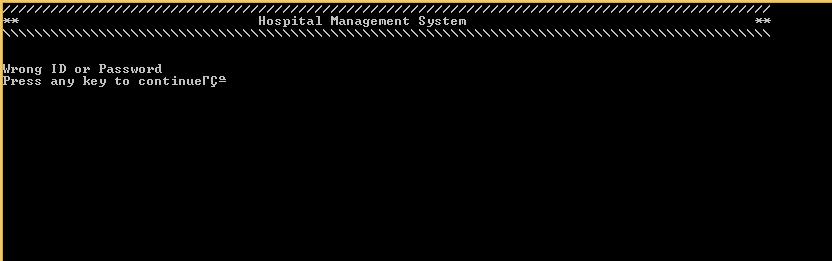
**User selection menu**:



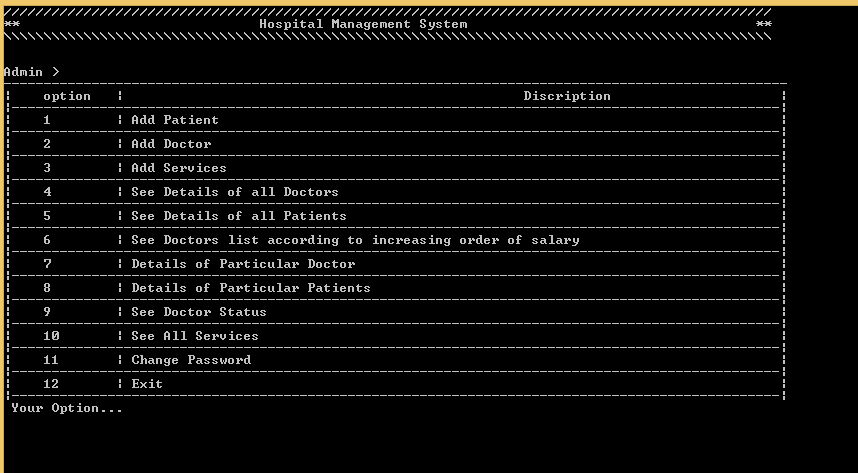
**Admin login page:**



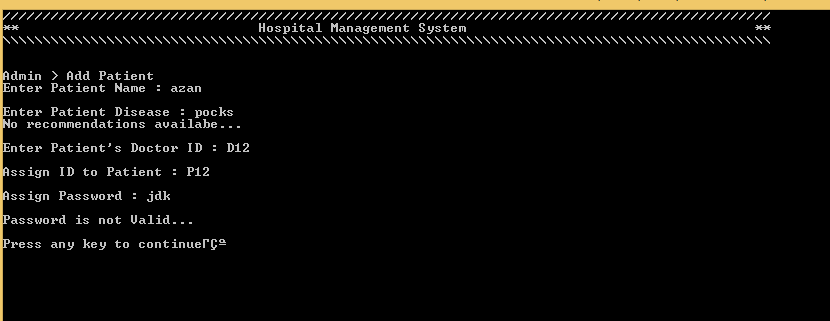
**If Admin enters wrong ID or Password:**



**Admin main menu:**



**Add patient (If password is not valid):**



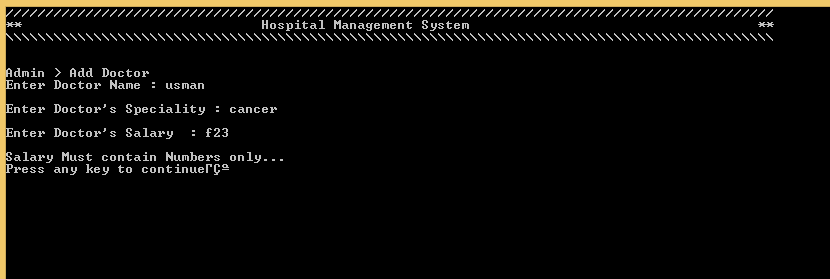
**Add patient (If ID of two patient becomes same):**



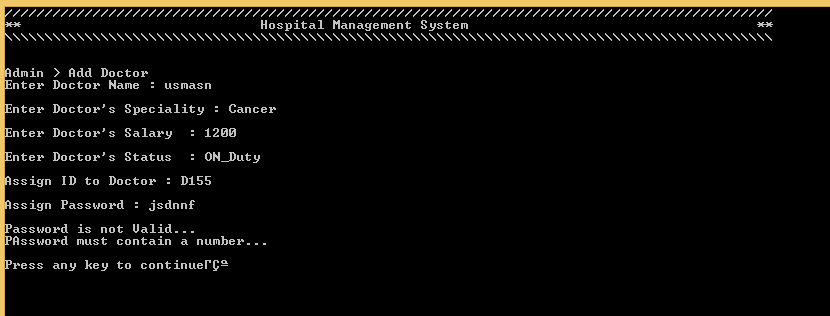
**Add patient (Successful):**



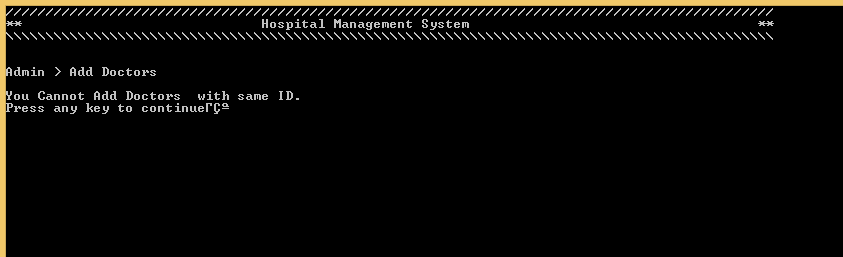
**Add Doctor (If salary is contain alphabet):**



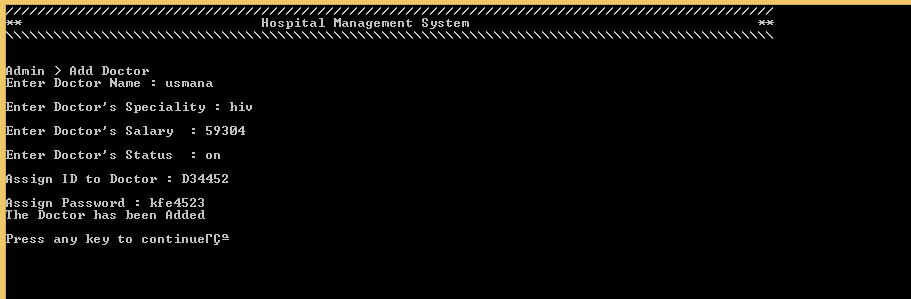
**Add Doctor (Password is not valid):**



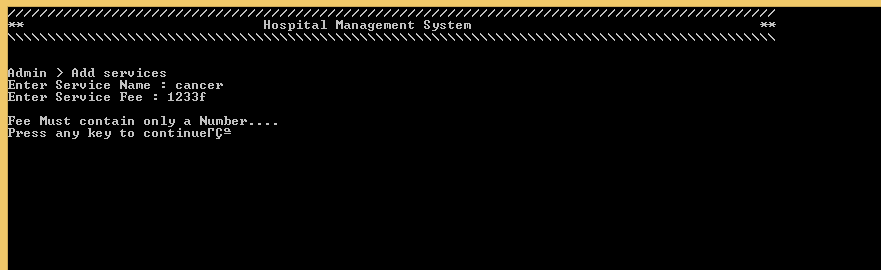
**Add Doctor (If ID of two Doctors become same):**



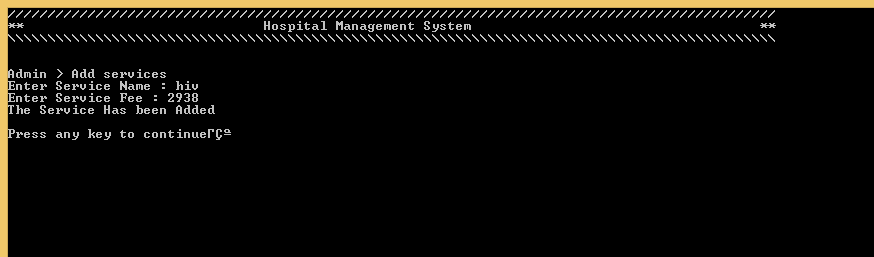
**Add Doctor (Successful):**



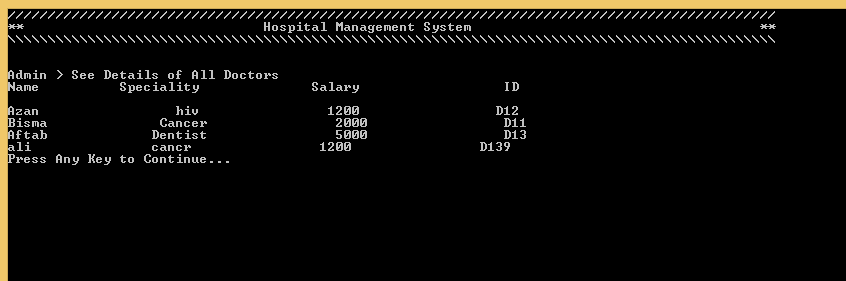
**Add Services (If Fee is not valid):**



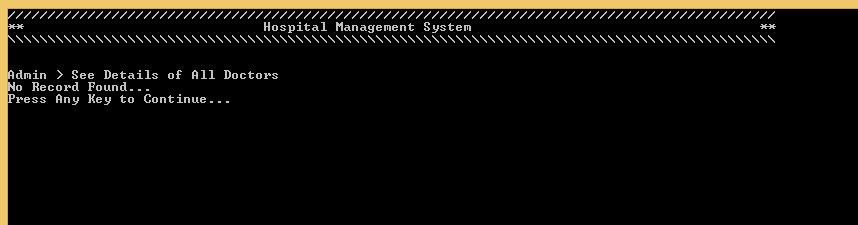
**Add Services (Successful):**



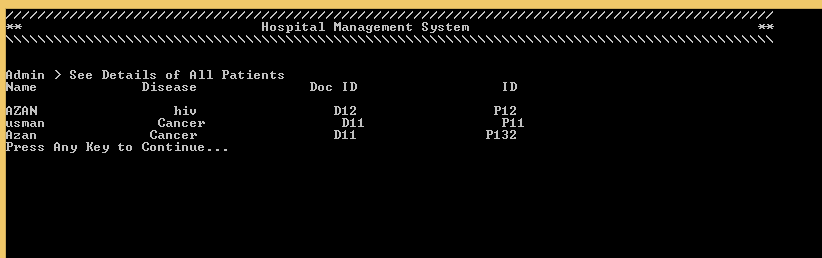
**Details of all Doctors:**



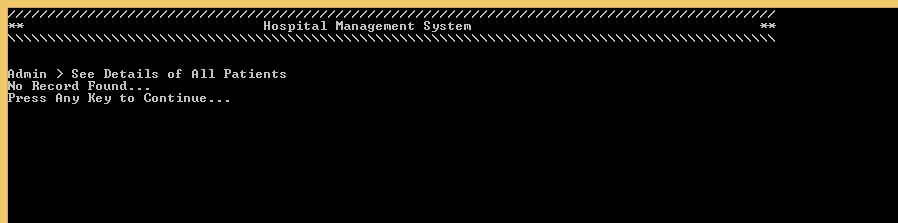
**Details of all Doctors (If no Record found):**



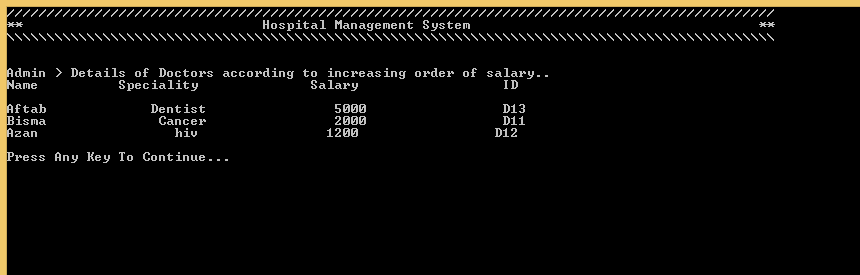
**Details of all Patients:**



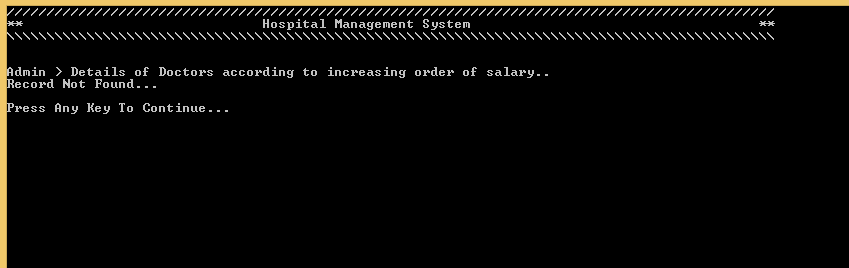
**Details of all Patients (If no Record found):**



**Sorted list of Doctors:**



**Sorted list of Doctors (If no Record found):**



**Details of particular Doctor:**



**Details of particular Doctor (If no Record found):**



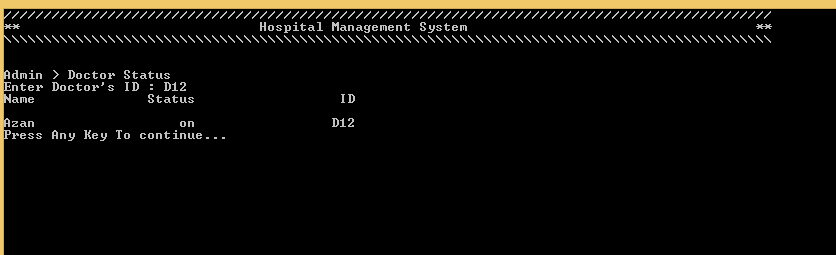
**Details of particular Patient:**



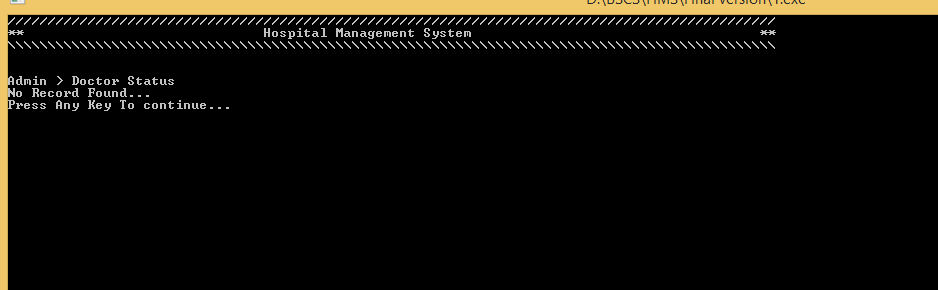
**Details of particular Patient (If no Record found):**



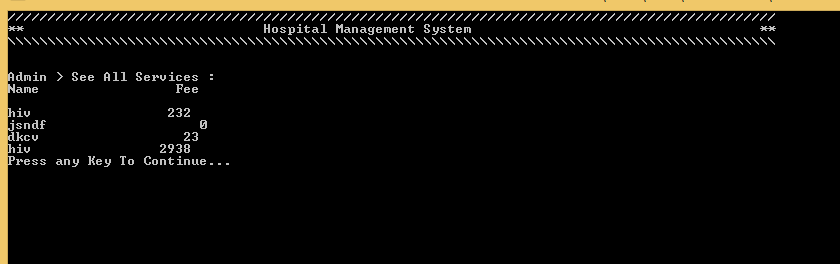
**See Doctor Status:**



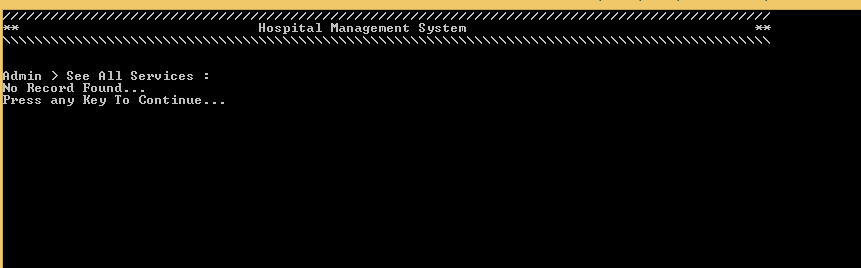
**See Doctor Status (If no Record found):**



**See All Services:**



**See All Services (If no Record found):**



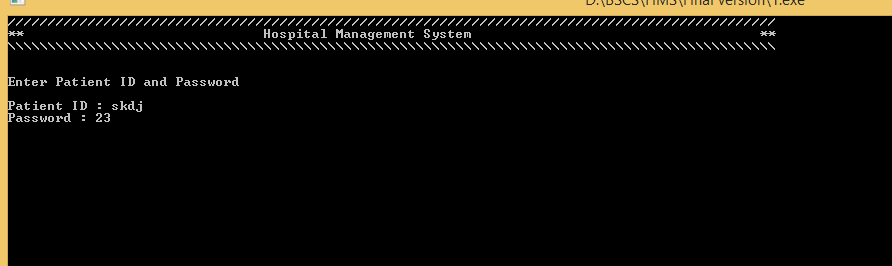
**Change Password (If not valid):**



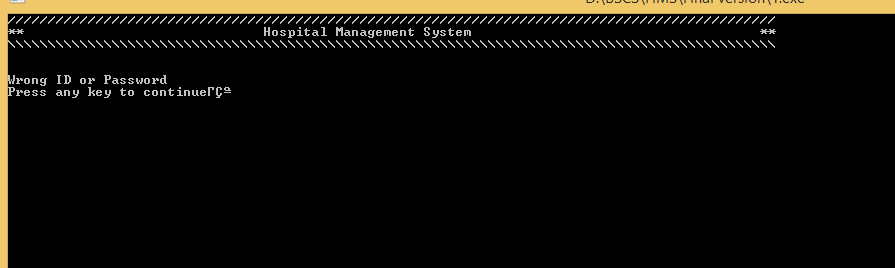
**Change Password:**



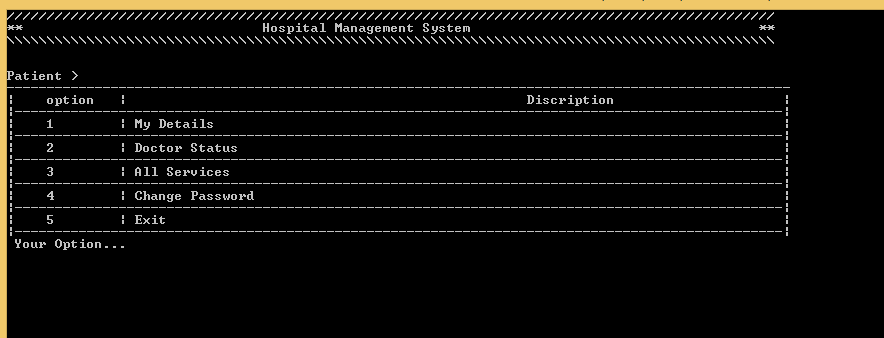
**Patient login Page:**



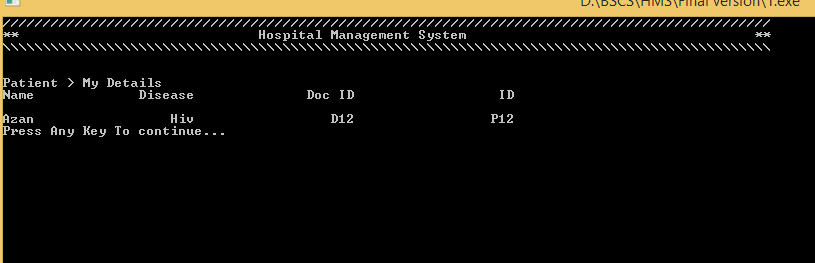
**Patient Login Page (If ID or Password is invalid):**



**Patient Main Menu:**



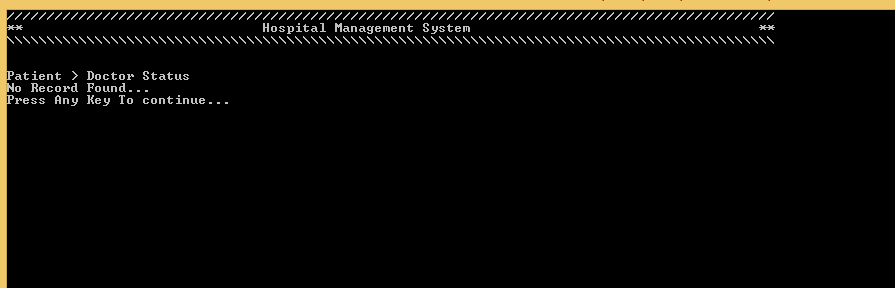
**My Details:**



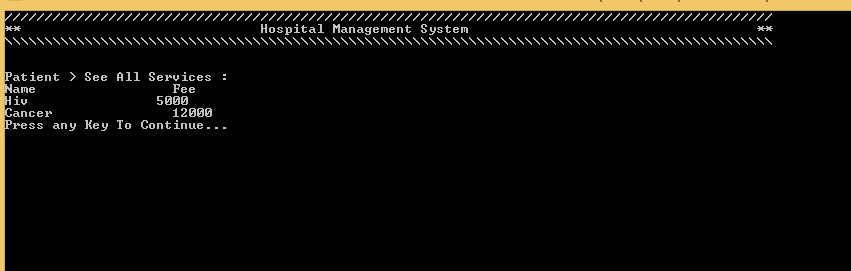
**Doctor Status:**



**Doctor Status (If no Record found):**



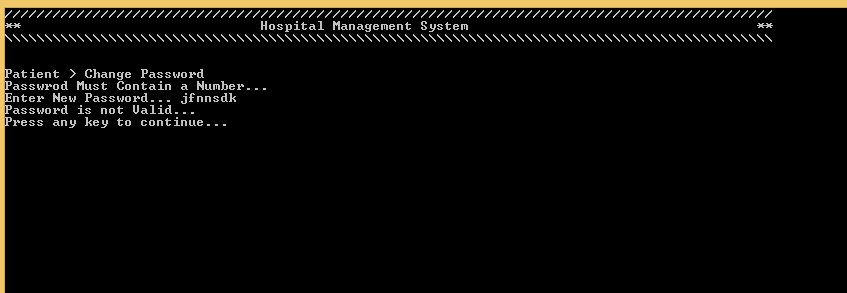
**See all Services:**



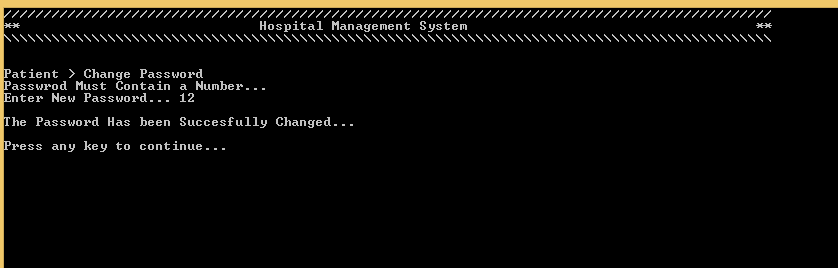
**See all Services (If no Record found):**



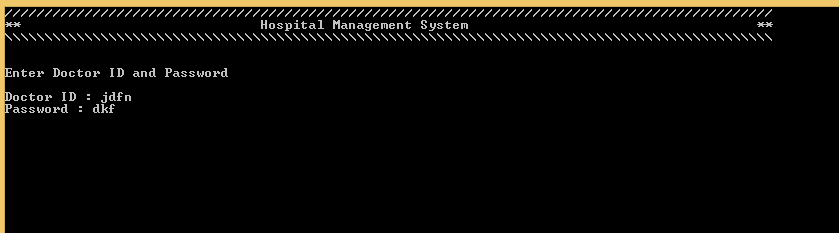
**Change Password (If not valid):**



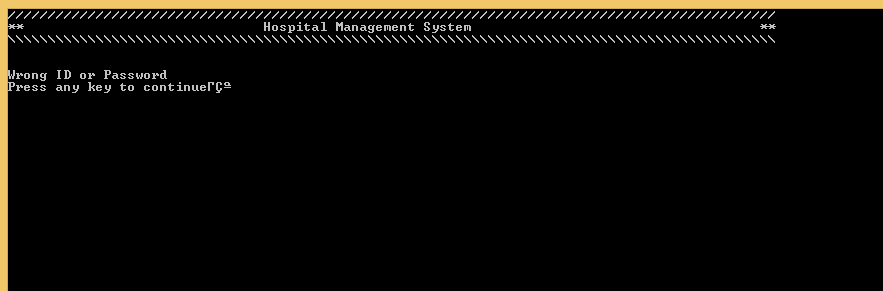
**Change Password (Successful):**



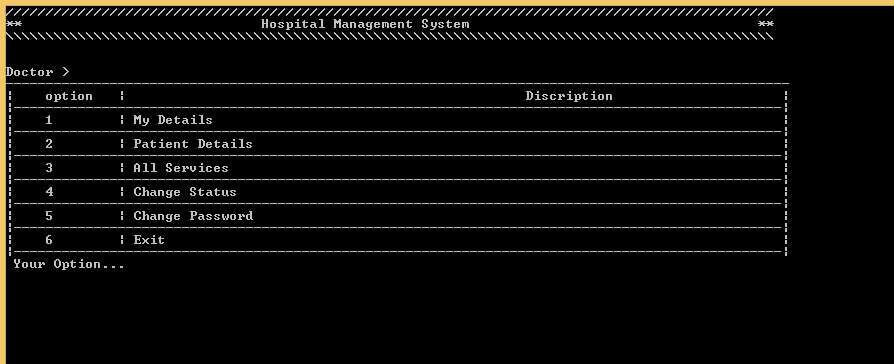
**Doctor login Page:**



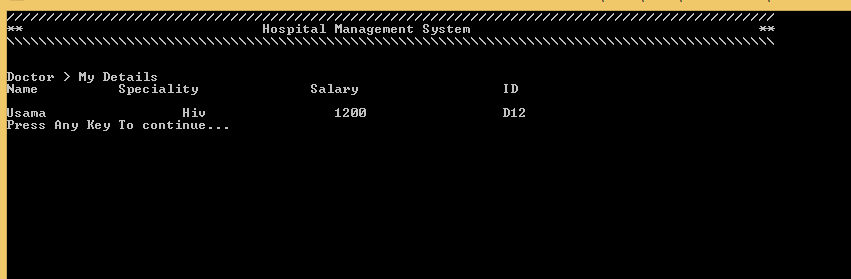
**Doctor Login Page (If ID or Password is invalid):**



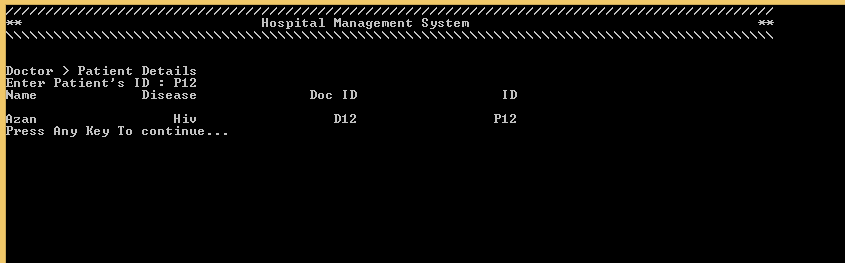
**Doctor Main Menu:**



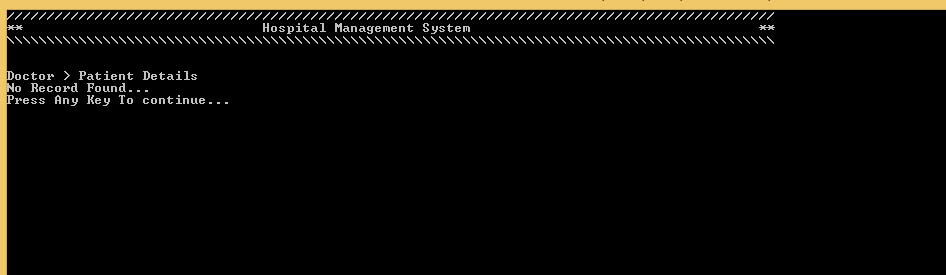
**My Details:**



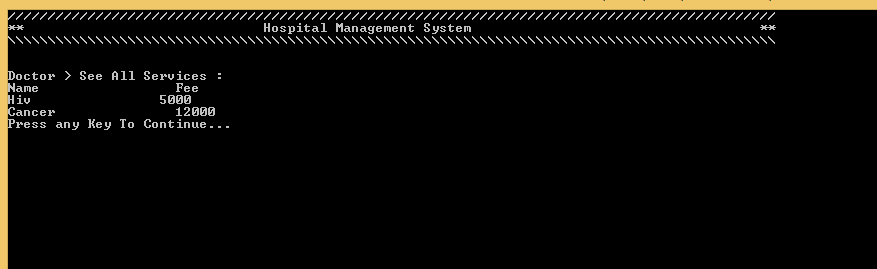
**Patient Details:**



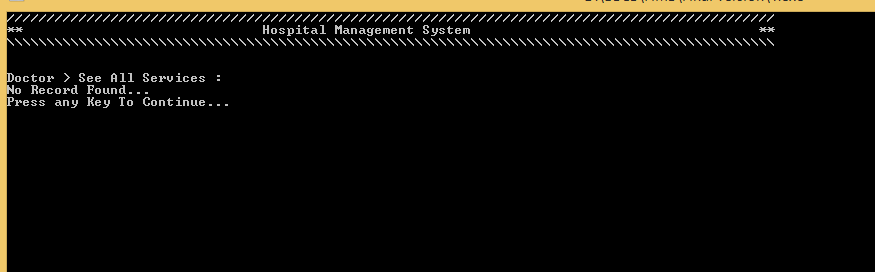
**Patient Details (If no Record found):**



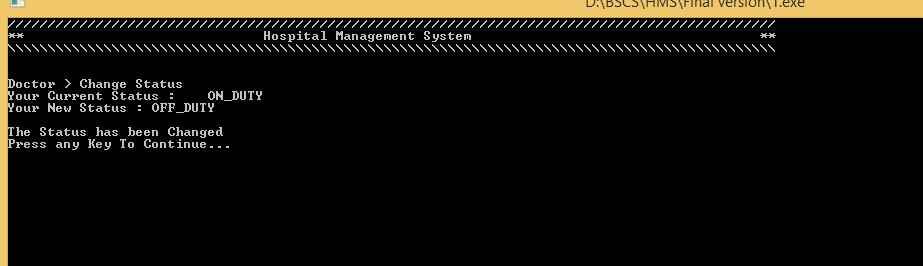
**See all Services:**



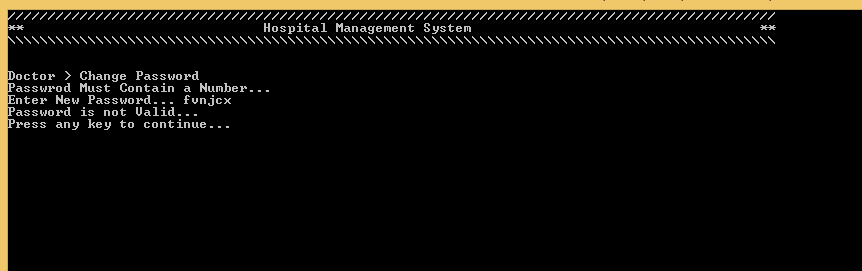
**See all Services (If no Record found):**



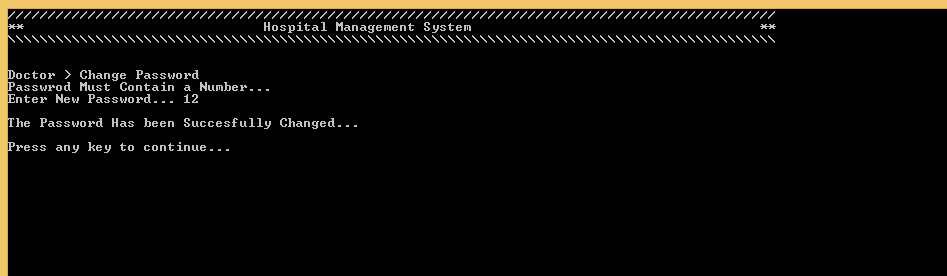
**Change Status:**



**Change Password (If not valid):**



**Change Password (Successful):**



**Data Structures:**

**Patient:**

string Patient\_Name[SIZE]

string Patient\_Disease[SIZE]

string Patient\_Doc\_ID[SIZE]

string Patient\_ID[SIZE]

string Patient\_Password[SIZE]

**Doctor:**

string Doctor\_Name[SIZE]

string Doctor\_Speciality[SIZE]

int Doctor\_Salary[SIZE]

string Doctor\_ID[SIZE]

string Doctor\_Password[SIZE]

string Doctor\_Status[SIZE]

**Services:**

string Service\_Name[SIZE]

int Service\_Fee[SIZE]

**Functions Prototypes:**

void header()

int User\_Selection\_Menu()

int Admin\_login()

int Admin\_Main\_Menu()

void Display\_Doctor\_info(int index)

bool Validation\_of\_Password(string pass)

void Change\_Password(int, int)

void Display\_Patient\_info(int index)

void Doc\_Status(int index)

void Display\_Services(int index)

void Recommend\_Doctor(string Disease)

void Add\_Patient()

void Add\_Services()

void Sorting\_Doctors()

void Add\_Doctor()

int Doctor\_Login()

int Doctor\_Main\_Menu()

int Patient\_Login()

int Patient\_Main\_Menu()

string Extraction\_of\_Specific\_Field(string Source, int field)

void Populating\_Patient\_Structures()

void Populating\_Doctor\_File()

void Populating\_Doctor\_Structures()

void Populating\_Patient\_File()

void Extraction\_of\_Admin\_password()

void Store\_Admin\_password()

void Populating\_Service\_File()

void Populating\_Service\_Structures()

bool ValidationOnNumberInput(string)

**Complete Code:**

#include <iostream>

#include <conio.h>

#include <fstream>

#include <iomanip>

using namespace std;

const int SIZE = 100;

//.............................................................Variables for admin.............................

string Admin\_ID = "PF";

string Admin\_Password = "2021";

//............................................................Data Structures For Patient..........................

string Patient\_Name[SIZE];

string Patient\_Disease[SIZE];

string Patient\_Doc\_ID[SIZE];

string Patient\_ID[SIZE];

string Patient\_Password[SIZE];

int Patient\_Counter = 0;

//............................................................. Data Structures For Doctor.........................

string Doctor\_Name[SIZE];

string Doctor\_Speciality[SIZE];

int Doctor\_Salary[SIZE];

string Doctor\_ID[SIZE];

string Doctor\_Password[SIZE];

string Doctor\_Status[SIZE];

int Doctor\_Counter = 0;

//............................................................. Data Structures For Services..........................

string Service\_Name[SIZE];

int Service\_Fee[SIZE];

int Service\_Counter = 0;

//..................................................................PROTOTYPES........................................

void header();

int User\_Selection\_Menu();

int Admin\_login();

int Admin\_Main\_Menu();

void Display\_Doctor\_info(int index);

bool Validation\_of\_Password(string pass);

void Change\_Password(int, int);

void Display\_Patient\_info(int index);

void Doc\_Status(int index);

void Display\_Services(int index);

void Recommend\_Doctor(string Disease);

void Add\_Patient();

void Add\_Services();

void Sorting\_Doctors();

void Add\_Doctor();

int Doctor\_Login();

int Doctor\_Main\_Menu();

int Patient\_Login();

int Patient\_Main\_Menu();

string Extraction\_of\_Specific\_Field(string Source, int field);

void Populating\_Patient\_Structures();

void Populating\_Doctor\_File();

void Populating\_Doctor\_Structures();

void Populating\_Patient\_File();

void Extraction\_of\_Admin\_password();

void Store\_Admin\_password();

void Populating\_Service\_File();

void Populating\_Service\_Structures();

bool ValidationOnNumberInput(string);

//........................................................................ Header Function.................................................

void header()

{

system("cls");

cout << "////////////////////////////////////////////////////////////////////////////////////////////////" << endl;

cout << "\*\* Hospital Management System \*\* " << endl;

cout << "\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\" << endl

<< endl

<< endl;

} //...End of Header

// ..............................................................Menu That Select User of The Application.................................................

int User\_Selection\_Menu()

{

system("cls");

header();

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

cout << "| option | Discription |" << endl;

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

cout << "| 1 | Admin |" << endl;

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

cout << "| 2 | Patient |" << endl;

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

cout << "| 3 | Doctor |" << endl;

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

cout << "| 4 | Exit |" << endl;

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

cout << " Your Option... ";

int option;

cin >> option;

return option;

} //...End of User Selection Menu

//........................................................................ Admin Login screen.................................................

int Admin\_login()

{

header();

string ID, Password;

cout << "Enter Admin ID and Password" << endl

<< endl;

cin.ignore();

cout << "Admin ID : ";

getline(cin, ID);

cout << "Password : ";

getline(cin, Password);

if (ID == Admin\_ID && Password == Admin\_Password)

{

return 1;

}

else

{

header();

cout << "Wrong ID or Password" << endl;

cout << "Press any key to continue…";

getch();

return 0;

}

} // end of admin login

//........................................................................ Admin Main Menu.................................................

int Admin\_Main\_Menu()

{

int option;

header();

cout << "Admin > " << endl;

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

cout << "| option | Discription |" << endl;

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

cout << "| 1 | Add Patient |" << endl;

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

cout << "| 2 | Add Doctor |" << endl;

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

cout << "| 3 | Add Services |" << endl;

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

cout << "| 4 | See Details of all Doctors |" << endl;

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

cout << "| 5 | See Details of all Patients |" << endl;

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

cout << "| 6 | See Doctors list according to increasing order of salary |" << endl;

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

cout << "| 7 | Details of Particular Doctor |" << endl;

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

cout << "| 8 | Details of Particular Patients |" << endl;

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

cout << "| 9 | See Doctor Status |" << endl;

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

cout << "| 10 | See All Services |" << endl;

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

cout << "| 11 | Change Password |" << endl;

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

cout << "| 12 | Exit |" << endl;

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

cout << " Your Option...";

cin >> option;

return option;

} // end of admin main menu

// ........................................................................ Details of Doctor.................................................

void Display\_Doctor\_info(int index)

{

cout.setf(ios::left);

cout.width(30);

cout << Doctor\_Name[index];

cout.width(30);

cout << Doctor\_Speciality[index];

cout.width(30);

cout << Doctor\_Salary[index];

cout.width(30);

cout << Doctor\_ID[index] << endl;

} //...End of Display\_Doctor\_info

//.................................................................Validation on number type input...............................................

bool ValidationOnNumberInput(string source)

{

int i = 0;

while (source[i] != '\0')

{

if (!(source[i] >= 48 && source[i] <= 57))

return 0;

i++;

}

return 1;

}

//...................................................................... Validation\_of\_Password.................................................

bool Validation\_of\_Password(string pass)

{

int i = 0;

while (pass[i] != '\0')

{

// It will CHeck that password must contain a number

if (pass[i] >= 48 && pass[i] <= 57)

{

return 1;

break;

}

i++;

}

return 0;

} // Validation\_of\_Password

// ....................................................................... Change Password.................................................

void Change\_Password(int user, int index = 0)

{

header();

string pass;

if (user == 1) // for admin

{

cout << "Admin > Change Password" << endl;

cout << "Passwrod Must Contain a Number..." << endl;

cout << "Enter New Password... ";

cin.ignore();

getline(cin, pass);

if (Validation\_of\_Password(pass))

{

Admin\_Password = pass;

cout << endl

<< "The Password Has been Succesfully Changed..." << endl

<< endl;

}

else

cout << "Password is not Valid..." << endl;

}

if (user == 2) // for Patient

{

cout << "Patient > Change Password" << endl;

cout << "Passwrod Must Contain a Number..." << endl;

cout << "Enter New Password... ";

cin.ignore();

getline(cin, pass);

if (Validation\_of\_Password(pass))

{

Patient\_Password[index] = pass;

cout << endl

<< "The Password Has been Succesfully Changed..." << endl

<< endl;

}

else

cout << "Password is not Valid..." << endl;

}

if (user == 3) // for Doctor

{

cout << "Doctor > Change Password" << endl;

cout << "Passwrod Must Contain a Number..." << endl;

cout << "Enter New Password... ";

cin.ignore();

getline(cin, pass);

if (Validation\_of\_Password(pass))

{

Doctor\_Password[index] = pass;

cout << endl

<< "The Password Has been Succesfully Changed..." << endl

<< endl;

}

else

cout << "Password is not Valid..." << endl;

}

cout << "Press any key to continue..." << endl;

getch();

} // end of Change\_Password

// ........................................................................ Details of Patient.................................................

void Display\_Patient\_info(int index)

{

cout.setf(ios::left);

cout.width(30);

cout << Patient\_Name[index];

cout.width(30);

cout << Patient\_Disease[index];

cout.width(30);

cout << Patient\_Doc\_ID[index];

cout.width(30);

cout << Patient\_ID[index] << endl;

} // End of PAtient info

// ........................................................................ See Doctor Status.................................................

void Doc\_Status(int index)

{

cout.setf(ios::left);

cout.width(30);

cout << Doctor\_Name[index];

cout.width(30);

cout << Doctor\_Status[index];

cout.width(30);

cout << Doctor\_ID[index] << endl;

} // End of doc status

// ........................................................................ Display Services.................................................

void Display\_Services(int index)

{

cout.setf(ios ::left);

cout.width(30);

cout << Service\_Name[index];

cout.width(30);

cout << Service\_Fee[index] << endl;

} // end of dislay services

// ........................................................................ Recommend DOctor.................................................

void Recommend\_Doctor(string Disease)

{

bool flag = 1; // if it is true then no doctor is availbe for recomendation

for (int i = 0; i < Doctor\_Counter; i++)

{

if (Doctor\_Speciality[i] == Disease)

{

cout << "Doctor " << Doctor\_Name[i] << " with ID : " << Doctor\_ID[i] << " is recommended for this patient"

<< endl;

flag = 0;

break;

}

}

if (flag)

{

cout << "No recommendations availabe..." << endl;

}

} // End of Recommend Doctor

// ........................................................................ Add Patient.................................................

void Add\_Patient()

{

header();

cout << "Admin > Add Patient" << endl;

cout << "Enter Patient Name : ";

cin.ignore();

getline(cin, Patient\_Name[Patient\_Counter]);

cout << endl

<< "Enter Patient Disease : ";

getline(cin, Patient\_Disease[Patient\_Counter]);

Recommend\_Doctor(Patient\_Disease[Patient\_Counter]);

cout << endl

<< "Enter Patient's Doctor ID : ";

getline(cin, Patient\_Doc\_ID[Patient\_Counter]);

cout << endl

<< "Assign ID to Patient : ";

getline(cin, Patient\_ID[Patient\_Counter]);

cout << endl

<< "Assign Password : ";

string pass;

getline(cin, pass);

//.............validaiton on password............

if (Validation\_of\_Password(pass))

{

Patient\_Password[Patient\_Counter] = pass;

//.........CHECKING FOR UNIQUENESS IN ID............

if (Patient\_Counter > 0)

{

bool flag = 1;

for (int i = Patient\_Counter - 1; i >= 0; i--)

{

if (Patient\_Counter == 0)

continue;

if (Patient\_ID[Patient\_Counter] == Patient\_ID[i])

{

Patient\_Counter--;

system(" cls ");

header();

cout << "Admin > Add Patients" << endl;

cout << "You Cannot Add Patients with same ID. " << endl;

cout << endl

<< "Press any key to continue… ";

flag = 0;

getch();

}

}

if (flag)

{

cout << "The patient has been Added" << endl;

cout << endl

<< "Press any key to continue… ";

getch();

}

}

Patient\_Counter++;

}

else

{

cout << endl

<< "Password is not Valid..." << endl;

cout << "PAssword must contain a number..." << endl;

cout << endl

<< "Press any key to continue… ";

getch();

}

} //...End of Add patient

//........................................................................... Add Services.................................................

void Add\_Services()

{

header();

cout << "Admin > Add services" << endl;

cout << "Enter Service Name : ";

cin.ignore();

getline(cin, Service\_Name[Service\_Counter]);

cout << "Enter Service Fee : ";

string fee;

cin >> fee;

if (ValidationOnNumberInput(fee))

{

Service\_Fee[Service\_Counter] = stoi(fee);

cout << "The Service Has been Added" << endl;

Service\_Counter++;

}

else

{

cout << endl

<< "Fee Must contain only a Number....";

}

cout << endl

<< "Press any key to continue… ";

getch();

} // End of Add Services

// .........................................................................Sorting of Doctors.................................................

void Sorting\_Doctors()

{

/\* temporary arrays for storing values of DOctor\_salary and their respective indices \*/

// I use this logic it prevent my original array from rearranging

int temp[Doctor\_Counter];

int index[Doctor\_Counter];

//....................................................................................

// filling of index

for (int i = 0; i < Doctor\_Counter; i++)

{

index[i] = i;

}

// filling of temp

for (int i = 0; i < Doctor\_Counter; i++)

{

temp[i] = Doctor\_Salary[i];

}

// sorting indices in index array

for (int i = 0; i < Doctor\_Counter; i++)

{

for (int j = 0; j < Doctor\_Counter - 1; j++)

{

int tempValue = temp[j];

int tempIndex = index[j];

if (temp[j] < temp[j + 1])

{

temp[j] = temp[j + 1];

temp[j + 1] = tempValue;

index[j] = index[j + 1];

index[j + 1] = tempIndex;

}

}

}

// Printing Sorted List of doctors

cout.setf(ios::left);

cout.width(30);

cout << "Name";

cout.width(30);

cout << "Speciality";

cout.width(30);

cout << "Salary";

cout.width(30);

cout << "ID" << endl

<< endl;

for (int i = 0; i < Doctor\_Counter; i++)

{

Display\_Doctor\_info(index[i]);

}

} // End of Sorting Doctors

// ...........................................................................Add Doctor.................................................

void Add\_Doctor()

{

header();

cout << "Admin > Add Doctor" << endl;

cout << "Enter Doctor Name : ";

cin.ignore();

getline(cin, Doctor\_Name[Doctor\_Counter]);

cout << endl

<< "Enter Doctor's Speciality : ";

getline(cin, Doctor\_Speciality[Doctor\_Counter]);

cout << endl

<< "Enter Doctor's Salary : ";

string salary;

cin >> salary;

//..............validaiton on salary...............

if (ValidationOnNumberInput(salary))

{

Doctor\_Salary[Doctor\_Counter] = stoi(salary);

cout << endl

<< "Enter Doctor's Status : ";

cin.ignore();

getline(cin, Doctor\_Status[Doctor\_Counter]);

cout << endl

<< "Assign ID to Doctor : ";

getline(cin, Doctor\_ID[Doctor\_Counter]);

string pass;

cout << endl

<< "Assign Password : ";

getline(cin, pass);

//.............validaiton on password............

if (Validation\_of\_Password(pass))

{

Doctor\_Password[Doctor\_Counter] = pass;

//.........CHECKING FOR UNIQUENESS IN ID............

if (Doctor\_Counter > 0)

{

bool flag = 1;

for (int i = Doctor\_Counter - 1; i >= 0; i--)

{

if (Doctor\_Counter == 0)

continue;

if (Doctor\_ID[Doctor\_Counter] == Doctor\_ID[i])

{

Doctor\_Counter--;

system(" cls ");

header();

cout << "Admin > Add Doctors" << endl;

cout << endl

<< "You Cannot Add Doctors with same ID. " << endl;

cout << "Press any key to continue… ";

flag = 0;

getch();

}

}

if (flag)

{

cout << "The Doctor has been Added" << endl;

cout << endl

<< "Press any key to continue… ";

getch();

}

}

Doctor\_Counter++;

}

else

{

cout << endl

<< "Password is not Valid..." << endl;

cout << "PAssword must contain a number..." << endl;

cout << endl

<< "Press any key to continue… ";

getch();

}

}

else

{

cout << endl

<< "Salary Must contain Numbers only...";

cout << endl

<< "Press any key to continue… ";

getch();

}

} //...End of Add\_Doctor

//........................................................................... Doctor\_Login.................................................

int Doctor\_Login()

{

header();

string ID, Password;

cout << "Enter Doctor ID and Password" << endl

<< endl;

cout << "Doctor ID : ";

cin.ignore();

getline(cin, ID);

cout << "Password : ";

getline(cin, Password);

//...........Checking whether doctor exist or not...........

for (int i = 0; i < Doctor\_Counter; i++)

{

if (ID == Doctor\_ID[i] && Password == Doctor\_Password[i])

return i;

}

return -1;

} // End of Doctor login

// ...........................................................................Doctor Main\_menu.................................................

int Doctor\_Main\_Menu()

{

header();

cout << "Doctor > " << endl;

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

cout << "| option | Discription |" << endl;

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

cout << "| 1 | My Details |" << endl;

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

cout << "| 2 | Patient Details |" << endl;

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

cout << "| 3 | All Services |" << endl;

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

cout << "| 4 | Change Status |" << endl;

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

cout << "| 5 | Change Password |" << endl;

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

cout << "| 6 | Exit |" << endl;

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

cout << " Your Option...";

int option;

cin >> option;

return option;

} // ENd of Doctor MAin menu

// ...........................................................................Patient Login.................................................

int Patient\_Login()

{

header();

string ID, Password;

cout << "Enter Patient ID and Password" << endl

<< endl;

cout << "Patient ID : ";

cin.ignore();

getline(cin, ID);

cout << "Password : ";

getline(cin, Password);

//...........Checking whether doctor exist or not...........

for (int i = 0; i < Patient\_Counter; i++)

{

if (ID == Patient\_ID[i] && Password == Patient\_Password[i])

return i;

}

return -1;

} // End of PAtient login

//........................................................................... Paient Main Menu.................................................

int Patient\_Main\_Menu()

{

header();

cout << "Patient > " << endl;

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

cout << "| option | Discription |" << endl;

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

cout << "| 1 | My Details |" << endl;

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

cout << "| 2 | Doctor Status |" << endl;

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

cout << "| 3 | All Services |" << endl;

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

cout << "| 4 | Change Password |" << endl;

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

cout << "| 5 | Exit |" << endl;

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

cout << " Your Option...";

int option;

cin >> option;

return option;

} // ENd of Patient main menu

// ..................................................................Extraction of specific Field from File.................................................

string Extraction\_of\_Specific\_Field(string Source, int field)

{

int count\_comma = 0;

int i = 0;

//............CHeking for Comma According to field

while (count\_comma < field - 1)

{

if (Source[i] == ',')

count\_comma++;

i++;

}

//.............Extracting the field

string temp = "";

while (1)

{

temp = temp + Source[i];

i++;

if (Source[i] == ',' || Source[i] == '\0')

break;

}

return temp;

} // End of Extraction of SPecific Field

//...................................................................Populatiing Patient's Data Structures from File.................................................

void Populating\_Patient\_Structures()

{

fstream patientFile("Patients.txt", ios ::in);

string temp = " ";

//............................cheking whether file is empty or not...........

char chk;

patientFile >> chk;

if (!patientFile.eof()) //.........................control goes inside If file is not empty

{

patientFile.seekg(0, ios::beg); //..........To go back to starting of file

while (!patientFile.eof())

{

getline(patientFile, temp);

//...The below condition is used to prevent reading a blank line because file contains a blank line at end

if (patientFile.eof())

{

break;

}

//.........................................................................................................

Patient\_Name[Patient\_Counter] = Extraction\_of\_Specific\_Field(temp, 1);

Patient\_Disease[Patient\_Counter] = Extraction\_of\_Specific\_Field(temp, 2);

Patient\_Doc\_ID[Patient\_Counter] = Extraction\_of\_Specific\_Field(temp, 3);

Patient\_ID[Patient\_Counter] = Extraction\_of\_Specific\_Field(temp, 4);

Patient\_Password[Patient\_Counter] = Extraction\_of\_Specific\_Field(temp, 5);

Patient\_Counter++;

}

}

patientFile.close();

} // End of Populating PAtient Structures

// ...................................................................Populatiing Doctor's Data Structures from File.................................................

void Populating\_Doctor\_Structures()

{

fstream DoctorFile("Doctors.txt", ios ::in);

string temp = " ";

//............................cheking whether file is empty or not...........

char chk;

DoctorFile >> chk;

if (!DoctorFile.eof()) //.........................control goes inside If file is not empty

{

DoctorFile.seekg(0, ios::beg); //..........To go back to starting of file

while (!DoctorFile.eof())

{

getline(DoctorFile, temp);

//...The below condition is used to prevent reading a blank line because file contains a blank line at end

if (DoctorFile.eof())

{

break;

}

//.........................................................................................................

Doctor\_Name[Doctor\_Counter] = Extraction\_of\_Specific\_Field(temp, 1);

Doctor\_Speciality[Doctor\_Counter] = Extraction\_of\_Specific\_Field(temp, 2);

Doctor\_Salary[Doctor\_Counter] = stoi(Extraction\_of\_Specific\_Field(temp, 3));

Doctor\_Status[Doctor\_Counter] = Extraction\_of\_Specific\_Field(temp, 4);

Doctor\_ID[Doctor\_Counter] = Extraction\_of\_Specific\_Field(temp, 5);

Doctor\_Password[Doctor\_Counter] = Extraction\_of\_Specific\_Field(temp, 6);

Doctor\_Counter++;

}

}

DoctorFile.close();

} // ENd of populating Doctor Structures

// ........................................................................populating File with Doctors info.................................................

void Populating\_Doctor\_File()

{

fstream DoctorFile("Doctors.txt", ios ::out);

for (int i = 0; i < Doctor\_Counter; i++)

{

DoctorFile << Doctor\_Name[i] << "," << Doctor\_Speciality[i] << "," << Doctor\_Salary[i] << "," << Doctor\_Status[i] << "," << Doctor\_ID[i] << "," << Doctor\_Password[i];

DoctorFile << endl;

}

DoctorFile.close();

} // ENd of populating doctor file

// .......................................................................populating File with PAtients info.................................................

void Populating\_Patient\_File()

{

fstream patientFile("Patients.txt", ios ::out);

for (int i = 0; i < Patient\_Counter; i++)

{

patientFile << Patient\_Name[i] << "," << Patient\_Disease[i] << "," << Patient\_Doc\_ID[i] << "," << Patient\_ID[i] << "," << Patient\_Password[i];

patientFile << endl;

}

patientFile.close();

} // end of populating patient file

// .......................................................................Extraction of Admin password from file.................................................

void Extraction\_of\_Admin\_password()

{

fstream adminFile("Admin.txt", ios ::in);

//............................cheking whether file is empty or not...........

char chk;

adminFile >> chk;

if (!adminFile.eof()) //.........................control goes inside If file is not empty

{

adminFile.seekg(0, ios::beg); //..........To go back to starting of file

getline(adminFile, Admin\_Password);

}

adminFile.close();

} // end of extraction of admin password

//........................................................................Storing admin password in file.................................................

void Store\_Admin\_password()

{

fstream adminFile("Admin.txt", ios ::out);

adminFile << Admin\_Password;

adminFile.close();

} // ENd of store Admin PAssword

//...................................................................... Populating File with Services.................................................

void Populating\_Service\_File()

{

fstream serviceFile("Services.txt", ios ::out);

for (int i = 0; i < Service\_Counter; i++)

{

serviceFile << Service\_Name[i] << "," << Service\_Fee[i];

serviceFile << endl;

}

serviceFile.close();

} // end of populating Service File

//......................................................................Populating Service Structures from file.................................................

void Populating\_Service\_Structures()

{

fstream serviceFile("Services.txt", ios ::in);

string temp = " ";

//............................cheking whether file is empty or not...........

char chk;

serviceFile >> chk;

if (!serviceFile.eof()) //.........................control goes inside If file is not empty

{

serviceFile.seekg(0, ios::beg); //..........To go back to starting of file

while (!serviceFile.eof())

{

getline(serviceFile, temp);

//...The below condition is used to prevent reading a blank line because file contains a blank line at end

if (serviceFile.eof())

{

break;

}

//.........................................................................................................

Service\_Name[Service\_Counter] = Extraction\_of\_Specific\_Field(temp, 1);

Service\_Fee[Service\_Counter] = stoi(Extraction\_of\_Specific\_Field(temp, 2));

Service\_Counter++;

}

}

serviceFile.close();

} // end Populating\_Service\_Structures()

//........................................................................... Start of main.................................................

int main()

{

//...................Filling oF all Data Structures From FIle....................

Populating\_Service\_Structures();

Extraction\_of\_Admin\_password();

Populating\_Doctor\_Structures();

Populating\_Patient\_Structures();

//................................................................................

int option; // It will store USer type

while (1)

{

header();

option = User\_Selection\_Menu();

//.....................If user enter wrong input in selecting user.............

if (option > 4 || option < 1)

continue;

//.............................................................................

// Selection According Value of option Variable

if (option == 1) // For Admin

{

int admin\_option = 1; // For storing admin selected option

header();

if (Admin\_login())

{

while (admin\_option != 12)

{

header();

admin\_option = Admin\_Main\_Menu();

if (admin\_option == 1)

{

Add\_Patient();

} //.....ENd of Option 1

else if (admin\_option == 2)

{

Add\_Doctor();

} //.....ENd of Option 2

else if (admin\_option == 3)

{

Add\_Services();

} //.....ENd of Option 3

else if (admin\_option == 4)

{

header();

if (Doctor\_Counter == 0) // If no Doctor is Added

{

cout << "Admin > See Details of All Doctors" << endl;

cout << "No Record Found..." << endl;

}

else

{

cout << "Admin > See Details of All Doctors" << endl;

cout.setf(ios::left);

cout.width(30);

cout << "Name";

cout.width(30);

cout << "Speciality";

cout.width(30);

cout << "Salary";

cout.width(30);

cout << "ID" << endl

<< endl;

for (int i = 0; i < Doctor\_Counter; i++)

{

Display\_Doctor\_info(i);

}

}

cout << "Press Any Key to Continue..." << endl;

getch();

} //.....ENd of Option 4

else if (admin\_option == 5)

{

header();

if (Patient\_Counter == 0) // If no patient is added

{

cout << "Admin > See Details of All Patients" << endl;

cout << "No Record Found..." << endl;

}

else

{

cout << "Admin > See Details of All Patients" << endl;

cout.setf(ios::left);

cout.width(30);

cout << "Name";

cout.width(30);

cout << "Disease";

cout.width(30);

cout << "Doc ID";

cout.width(30);

cout << "ID" << endl

<< endl;

for (int i = 0; i < Patient\_Counter; i++)

Display\_Patient\_info(i);

}

cout << "Press Any Key to Continue..." << endl;

getch();

} //.....ENd of Option 5

else if (admin\_option == 6)

{

header();

if (Doctor\_Counter == 0) // if no Doctor is added

{

cout << "Admin > Details of Doctors according to increasing order of salary.." << endl;

cout << "Record Not Found..." << endl;

}

else

{

cout << "Admin > Details of Doctors according to increasing order of salary.." << endl;

Sorting\_Doctors();

}

cout << endl

<< "Press Any Key To Continue..." << endl;

getch();

} //.....ENd of Option 6

else if (admin\_option == 7)

{

header();

bool flag = 0; // If it is false then ID will be wrong

cout << "Admin > Details of Particular Doctor" << endl;

string ID;

cout << "Enter Doctor's ID : ";

cin.ignore();

getline(cin, ID);

cout.setf(ios::left);

cout.width(30);

cout << "Name";

cout.width(30);

cout << "Speciality";

cout.width(30);

cout << "Salary";

cout.width(30);

cout << "ID" << endl

<< endl;

for (int i = 0; i < Doctor\_Counter; i++)

{

if (ID == Doctor\_ID[i]) // If user enter right ID

{

Display\_Doctor\_info(i);

flag = 1;

}

}

if (flag == 0)

{

header();

cout << "Admin > Details of Particular Doctor" << endl;

cout << "Record Not Found..." << endl;

}

cout << "Press Any Key To continue...";

getch();

} //.....ENd of Option 7

else if (admin\_option == 8)

{

header();

bool flag = 0; // If it is false then ID will be wrong

cout << "Admin > Details of Particular Patient" << endl;

string ID;

cout << "Enter Patient's ID : ";

cin.ignore();

getline(cin, ID);

cout.setf(ios::left);

cout.width(30);

cout << "Name";

cout.width(30);

cout << "Disease";

cout.width(30);

cout << "Doc ID";

cout.width(30);

cout << "ID" << endl

<< endl;

for (int i = 0; i < Patient\_Counter; i++)

{

if (ID == Patient\_ID[i]) // If user enter right ID

{

Display\_Patient\_info(i);

flag = 1;

}

}

if (flag == 0)

{

header();

cout << "Admin > Details of Particular Patient" << endl;

cout << "Record Not Found..." << endl;

}

cout << "Press Any Key To continue...";

getch();

} //.....ENd of Option 8

else if (admin\_option == 9)

{

header();

bool flag = 0; // If it is false then ID will be wrong

cout << "Admin > Doctor Status" << endl;

string ID;

cout << "Enter Doctor's ID : ";

cin.ignore();

getline(cin, ID);

cout.setf(ios::left);

cout.width(30);

cout << "Name";

cout.width(30);

cout << "Status";

cout.width(30);

cout << "ID" << endl

<< endl;

for (int i = 0; i < Doctor\_Counter; i++)

{

if (ID == Doctor\_ID[i]) // If user enter right ID

{

Doc\_Status(i);

flag = 1;

}

}

if (flag == 0)

{

header();

cout << "Admin > Doctor Status" << endl;

cout << "No Record Found..." << endl;

}

cout << "Press Any Key To continue...";

getch();

} //.....ENd of Option 9

else if (admin\_option == 10)

{

header();

cout << "Admin > See All Services :" << endl;

if (Service\_Counter == 0) // If no Service is added yets

{

cout << "No Record Found..." << endl;

}

else

{

cout.setf(ios::left);

cout.width(30);

cout << "Name";

cout.width(30);

cout << "Fee" << endl

<< endl;

for (int i = 0; i < Service\_Counter; i++)

Display\_Services(i);

}

cout << "Press any Key To Continue..." << endl;

getch();

} //.....ENd of Option 10

else if (admin\_option == 11)

{

Change\_Password(1);

} //.....ENd of Option 11

} //..End of Admin While Loop

} //..End of IF ( admin\_login () )

else

continue;

} //......end of Admin If

else if (option == 2) // FOr PAtient

{

int Patient\_Option = 1; // For storing PAtient selected option

header();

int Patient\_number = Patient\_Login(); // it return -1 if password is incorrect

if (Patient\_number != -1)

{

while (Patient\_Option != 5)

{

header();

Patient\_Option = Patient\_Main\_Menu();

if (Patient\_Option == 1)

{

header();

cout << "Patient > My Details" << endl;

cout.setf(ios::left);

cout.width(30);

cout << "Name";

cout.width(30);

cout << "Disease";

cout.width(30);

cout << "Doc ID";

cout.width(30);

cout << "ID" << endl

<< endl;

Display\_Patient\_info(Patient\_number);

cout << "Press Any Key To continue...";

getch();

} //...End of option 1

else if (Patient\_Option == 2)

{

header();

cout << "Patient > Doctor Status" << endl;

string ID;

cout << "Enter Doctor's ID : ";

cin.ignore();

getline(cin, ID);

cout.setf(ios::left);

cout.width(30);

cout << "Name";

cout.width(30);

cout << "Status";

cout.width(30);

cout << "ID" << endl

<< endl;

bool flag = 0; // if its value is false then entered ID will be wrong

for (int i = 0; i < Doctor\_Counter; i++)

{

if (ID == Doctor\_ID[i])

{

Doc\_Status(i);

flag = 1;

}

}

if (flag == 0)

{

header();

cout << "Patient > Doctor Status" << endl;

cout << "No Record Found..." << endl;

}

cout << "Press Any Key To continue...";

getch();

} //...End of option 2

else if (Patient\_Option == 3)

{

header();

cout << "Patient > See All Services :" << endl;

if (Service\_Counter == 0) // If no Service is added yet

{

cout << "No Record Found..." << endl;

}

else

{

cout.setf(ios::left);

cout.width(30);

cout << "Name";

cout.width(30);

cout << "Fee" << endl;

for (int i = 0; i < Service\_Counter; i++)

Display\_Services(i);

}

cout << "Press any Key To Continue..." << endl;

getch();

} //...End of option 3

else if (Patient\_Option == 4)

{

Change\_Password(2, Patient\_number);

} //...End of option 4

else if (option == 5)

{

break;

} //...End of option 5

} //...End of PAtient While

} // End of Patient\_number != -1

else // If ID or PAssword is incorrect

{

header();

cout << "Wrong ID or Password" << endl;

cout << "Press any key to continue…";

getch();

}

} //.......... End of PAtient IF

else if (option == 3) // FOr Doctor

{

int Doctor\_option = 1; // SFor storing PAtient selected option

header();

int Doctor\_Number = Doctor\_Login();

if (Doctor\_Number != -1) // it return -1 if password is incorrect

{

while (Doctor\_option != 6)

{

header();

Doctor\_option = Doctor\_Main\_Menu();

if (Doctor\_option == 1)

{

header();

cout << "Doctor > My Details" << endl;

cout.setf(ios::left);

cout.width(30);

cout << "Name";

cout.width(30);

cout << "Speciality";

cout.width(30);

cout << "Salary";

cout.width(30);

cout << "ID" << endl

<< endl;

Display\_Doctor\_info(Doctor\_Number);

cout << "Press Any Key To continue...";

getch();

} //...ENd of option 1

else if (Doctor\_option == 2)

{

header();

cout << "Doctor > Patient Details" << endl;

string ID;

cout << "Enter Patient's ID : ";

cin.ignore();

getline(cin, ID);

cout.setf(ios::left);

cout.width(30);

cout << "Name";

cout.width(30);

cout << "Disease";

cout.width(30);

cout << "Doc ID";

cout.width(30);

cout << "ID" << endl

<< endl;

bool flag = 0; // if its value is false then entered ID will be wrong

for (int i = 0; i < Patient\_Counter; i++)

{

if (ID == Patient\_ID[i])

{

Display\_Patient\_info(i);

flag = 1;

}

}

if (flag == 0)

{

header();

cout << "Doctor > Patient Details" << endl;

cout << "No Record Found..." << endl;

}

cout << "Press Any Key To continue...";

getch();

} //...ENd of option 2

else if (Doctor\_option == 3)

{

header();

cout << "Doctor > See All Services :" << endl;

if (Service\_Counter == 0) // If no Service is added yet

{

cout << "No Record Found..." << endl;

}

else

{

cout.setf(ios::left);

cout.width(30);

cout << "Name";

cout.width(30);

cout << "Fee" << endl;

for (int i = 0; i < Service\_Counter; i++)

Display\_Services(i);

}

cout << "Press any Key To Continue..." << endl;

getch();

} //...ENd of option 3

else if (Doctor\_option == 4)

{

header();

cout << "Doctor > Change Status" << endl;

cout << "Your Current Status :\t " << Doctor\_Status[Doctor\_Number] << endl;

cout << "Your New Status : ";

cin.ignore();

getline(cin, Doctor\_Status[Doctor\_Number]);

cout << endl

<< "The Status has been Changed" << endl;

cout << "Press any Key To Continue..." << endl;

getch();

} //...ENd of option 4

else if (Doctor\_option == 5)

{

Change\_Password(3, Doctor\_Number);

} //...ENd of option 5

} //..End of Doctor While Loop

} //.. End of Doctor\_Number != -1

else // If ID or PAssword is incorrect

{

header();

cout << "Wrong ID or Password" << endl;

cout << "Press any key to continue…";

getch();

}

} //..End of DOctor IF

else if (option == 4) // FOr exit

{

break;

} //..End of Exit

} //............................End of LEading While

//............Populating FIles With Data Structures.....................

Populating\_Patient\_File();

Populating\_Doctor\_File();

Populating\_Service\_File();

Store\_Admin\_password();

//......................................................................

//.........Exit MEssage............

header();

cout << "\t\t\t\t\t Thanks For Using Our Application" << endl;

cout << "Press Any Key To Continue...";

//.................................

getch();

} //..End of main

­

**Functions Working Flow:**

**Test Cases:**

**Already Mentioned in Wireframes**

**Student Reg. No. :**  2021-CS-171  **Student Name.**  M.Abubakar Siddique Farooqi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **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 - **Algorithms** and Flow Charts of all functions- **Test Cases** are defined -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. |
| Sorting Features  **Grade:** | Sort working 100% and generating useful report | Sorting Feature is working but sorted data is not useful for project. | Sorting feature is partial implemented | Project do not contain sorting |
| 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)- Demo Data Functionality Added-At least Two Unit Tests are defined. | | | | |
| 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 |
| Recommendation Feature | Proper meaning full recommendation is present into system | Partial Recommendation is implemented | Implemented but not meaning full. | 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:** |  |

**Student Reg. No. :**  2021-CS-171  **Student Name.**  M.Abubakar Siddique Farooqi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **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 - **Algorithms** and Flow Charts of all functions- **Test Cases** are defined -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. |
| Sorting Features  **Grade:** | Sort working 100% and generating useful report | Sorting Feature is working but sorted data is not useful for project. | Sorting feature is partial implemented | Project do not contain sorting |
| 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)- Demo Data Functionality Added-At least Two Unit Tests are defined. | | | | |
| 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 |
| Recommendation Feature | Proper meaning full recommendation is present into system | Partial Recommendation is implemented | Implemented but not meaning full. | 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:** |  |