# CHAPTER 1

**INTRODUCTION**

Among several diseases and sufferings, many people are suffering from diabetes. With this computer based diabetes detection software, user will able to do their self-checkup without taking help of a doctor. Diabetes health care management system is not only useful for the patients it also helps to maintain the records about the patients details, appointment can be booked, can get any of the frequently asked queries cleared, it even gives a diet plan.

# Aim

The main aim is to provide requirement needed for detecting diabetes such as maintaining data about the patient details and to book appointment.

# Objective

The main objective is to create a unique and useful “Diabetes Detector” with exceptional quality and services that differentiates it from other storage system.

# Scope

The scope of Diabetes Detector is as follows: It mainly comprises of four modules:

Insertion of data to the file Extraction of data from the file Report generation module Search patient/appointment system.

# Advantages and Disadvantages

* + 1. **Advantages**

This project is beneficial for a patient to use.

Decreases the time consumption Improves efficiency

Seeking time is reduced

Decreases the paper and labor work Manage the entire process

# Disadvantages

It is too tiring to give computerized timing Security Limitations

Only works on Intranet

# CHAPTER 2

**HASHING**

Hashing is a useful searching technique, which can be used for implementing indexes. The main motivation for hashing is improving searching time. The idea is to discover the location of a key by simply examining the key. For that we need to design a hash function.

A hash function is a function h(k) that transforms a key into an address. There is no obvious connection between the key and the location. Two different keys may be sent to the same address generating a “collision”.

To compute the hash function apply 3 steps:

Transform the key into a number

Fold and add and take the mod by a prime number Divide by the size of the address space

# CHAPTER 3

**SOFTWARE REQUIREMENTS AND SPECIFICATION**

To run the project on various platforms we need some software specifications and hardware requirements to support this project.

# Hardware Specification

**Processor:** Intel core

**Ram:** 8.00GB

# Software Specification

# 

# Dev C++

# System Constraints

* + 1. **User Interface Constraints**

Using this portal is simple. A user familiar with system application can understand the functionality provided by the portal.

# Hardware Constraints

The portal should work on home desktop, laptops and computers.

# Software Constraints

The portal is designed to run in Net Beans, Eclipse.

# CHAPTER 4

**IMPLEMENTATION**

The project entitled “Diabetes Detector” can be implemented within the college or various department having their own department libraries. The Diabetes Detector can maintain the records of the books that are issued or deposited by the patients on time hence by reducing the searching time.

Diabetes Detector can be used for

Adding the a new patient

Displaying the patient details Searching the patient details Modifying the patient details Deleting the patient details

# INSERT STATEMENT

String void diabetes::enterdata()

{

int i,num=0;

fout.open("DIABETESDATA.txt",ios::app);

system("cls");

cout<<"\t\t\t\t\tWelcome!\n";

cout<<"\n\t\t\tPLEASE FILL ALL THE REQUIRDATA\n";

cout<<"\n\n ";

cout<<" Enter The no. of paitents : ";

cin>>num;

cout<<"\n\n ";

for(i=0;i<num;i++)

{

system("cls");

cout<<"\t\t\t\t\t\t\t\t WELCOME TO THE DIABETES HEALTH CARE\t\t\n\t\t\t\t\t\t\t\t\tWe judge we understand!\n";

cout<<"\t\t\t\t\t\t\t\t ....SJS

Enterprise....\n\n";cout<<"\n\n ";

cout<<" PLEASE ASK THE IDno. AND ENTER IT : ";

cin>>id;

cout<<"\n\n";

cout<<" 1 : ENTER NAME : ";

scanf("%s",name);

fflush(stdin);

cout<<"\n\n 2 : AGE : ";

cin>>age;

cout<<"\n\n 3 : SEX(M/F) : ";

cin>>sex;

cout<<"\n\n 4 : Height : ";

cin>>height;

cout<<"\n\n 5 : Weight : ";

cin>>weight;

cout<<"\n\n 6 : Phone number : ";

cin>>phn;

fout.write((char\*)&mainobj,sizeof(mainobj));

}

fout.close();

}

int diabetes::display(int a,int b)

{

fin.open("DIABETESDATA.txt",ios::in);

system("cls");

cout<<"\t\t\t\tHere's the Data of all paitents\n\n\n";

cout<<"|\t ID\t | NAME | AGE | SEX | HEIGHT | WEIGHT |\n";

while(fin.read((char\*)&mainobj,sizeof(mainobj)))

{

cout<<"|"<<setw(9)<<id<<setw(10)<<"|"<<setw(15)<<name<<setw(20)<<"|"<<setw(5)<<age<<setw(5)<<"|"<<setw(5)<<sex<<setw(5)<<"|"<<setw(7)<<height<<setw(7)<<"|"<<setw(7)<<weight<<setw(7)<<"\t|\n\n";

}

fin.close();

}

# DISPLAY STATEMENT

int diabetes::display(int a,int b)

{

fin.open("DIABETESDATA.txt",ios::in);

system("cls");

cout<<"\t\t\t\tHere's the Data of all paitents\n\n\n";

cout<<"|\t ID\t | NAME | AGE | SEX | HEIGHT | WEIGHT |\n";

while(fin.read((char\*)&mainobj,sizeof(mainobj)))

{

cout<<"|"<<setw(9)<<id<<setw(10)<<"|"<<setw(15)<<name<<setw(20)<<"|"<<setw(5)<<age<<setw(5)<<"|"<<setw(5)<<sex<<setw(5)<<"|"<<setw(7)<<height<<setw(7)<<"|"<<setw(7)<<weight<<setw(7)<<"\t|\n\n";

}

fin.close();

}

# SEARCH STATEMENT

void diabetes::search()

{

system("cls");

fin.open("DIABETESDATA.txt",ios::in|ios::out);

int tempid,k=0;

char check;

cout<<"\t\t\t\tDiabetes Health Care\t\n";

cout<<"\n Enter the id no. of patient : ";

cin>>tempid;

while(fin.read((char\*)&mainobj,sizeof(mainobj)))

{

if(tempid==mainobj.id)

{

k++;

break;

}

}

# DELETE STATEMENT

void diabetes::deldata()

{

int tempid,k=0;

system("cls");

fin.open("DIABETESDATA.txt",ios::in);

fout.open("Temp.txt",ios::out);

fturndoc.open("Doctor1.txt",ios::in);

ftemp.open("Tempdoc.txt",ios::out);

cout<<"\t\t\t\t\t\t\t\t WELCOME TO THE DIABETES HEALTH CARE\t\t\n\t\t\t\t\t\t\t\t\tWe judge we understand!\n";

cout<<"|\t ID\t | NAME | AGE | SEX | HEIGHT | WEIGHT |\n";

while(fin.read((char\*)&mainobj,sizeof(mainobj)))

{

cout<<"|"<<setw(9)<<id<<setw(10)<<"|"<<setw(15)<<name<<setw(20)<<"|"<<setw(5)<<age<<setw(5)<<"|"<<setw(5)<<sex<<setw(5)<<"|"<<setw(7)<<height<<setw(7)<<"|"<<setw(7)<<weight<<setw(7)<<"\t|\n\n";

}

fin.close();

fin.open("DIABETESDATA.txt",ios::in);

cout<<"\n Enter the id no. of patient : ";

cin>>tempid;

while(fin.read((char\*)&mainobj,sizeof(mainobj)))

{

if(tempid==mainobj.id)

{

k++;

}

else

{

fout.write((char\*)&mainobj,sizeof(mainobj));

}

}

fin.close();

fout.close();

fin.open("DIABETESDATA.txt",ios::out);

fout.open("Temp.txt",ios::in);

while(fout.read((char\*)&mainobj,sizeof(mainobj)))

{

fin.write((char\*)&mainobj,sizeof(mainobj));

}

cout<<"\n\n Deleted Record";

cout<<"\n\n";

fin.close();

fout.close();

cout<<"\n\n UPDATED RECORDS";

cout<<"\n\n";

fin.open("DIABETESDATA.txt",ios::in);

cout<<"|\t ID\t | NAME | AGE | SEX | HEIGHT | WEIGHT |\n";

while(fin.read((char\*)&mainobj,sizeof(mainobj)))

{

cout<<"|"<<setw(9)<<id<<setw(10)<<"|"<<setw(15)<<name<<setw(20)<<"|"<<setw(5)<<age<<setw(5)<<"|"<<setw(5)<<sex<<setw(5)<<"|"<<setw(7)<<height<<setw(7)<<"|"<<setw(7)<<weight<<setw(7)<<"\t|\n\n";

}

fin.close();

}

# MODIFY STATEMENT

ftemp.open("send1.txt",ios::in);

fout.open("DIABETESDATA.txt",ios::in);

system("cls");

cout<<"\n\n "; int tempid;

fin.open("update.txt",ios::out);

cout<<"\n\n You may have requests to update record ";

cout<<"\n\n Press Y to see them ";

char tp;

cin>>tp;

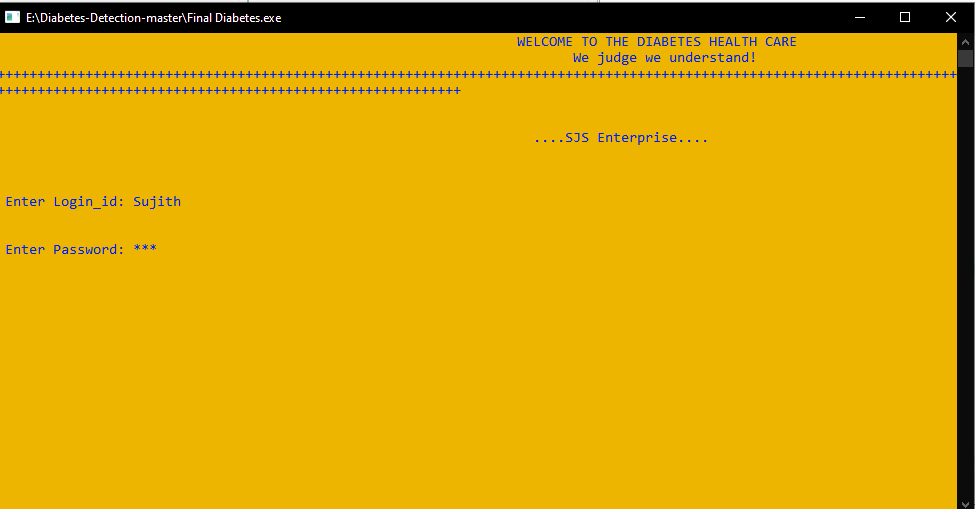
continue;

# CHAPTER 5

**RESULT ANALYSIS**

# LOGIN PAGE

The figure 1.1 says that the end user has to enter the user credentials such as name of the end user and his/her password. after all the credentials have been filled successfully press the login button, if all the details of the user is correct a message will be displayed referring that the user name and password has been entered successfully.



**Figure 1.1 Login page**

# 5.2 HOME PAGE

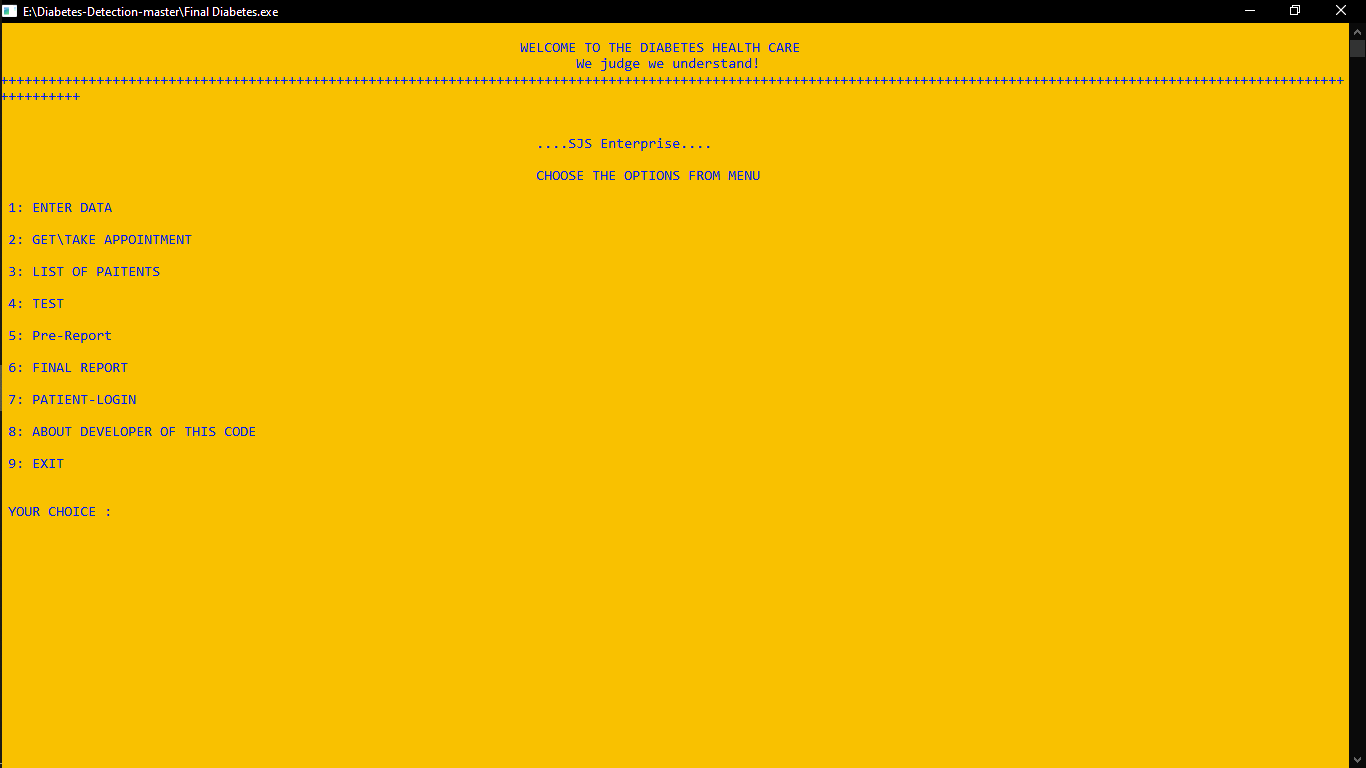
The figure 1.2 says that after successful logging in, the home page will be displayed which comprises of patient, quick checkup, administer menu, queries, queries reply, diet planner, log out and the exit button which refer to the previous page. The user can click on any of the button to proceed further.



**Figure 1.2 Home page**

# 5.3 PATIENT

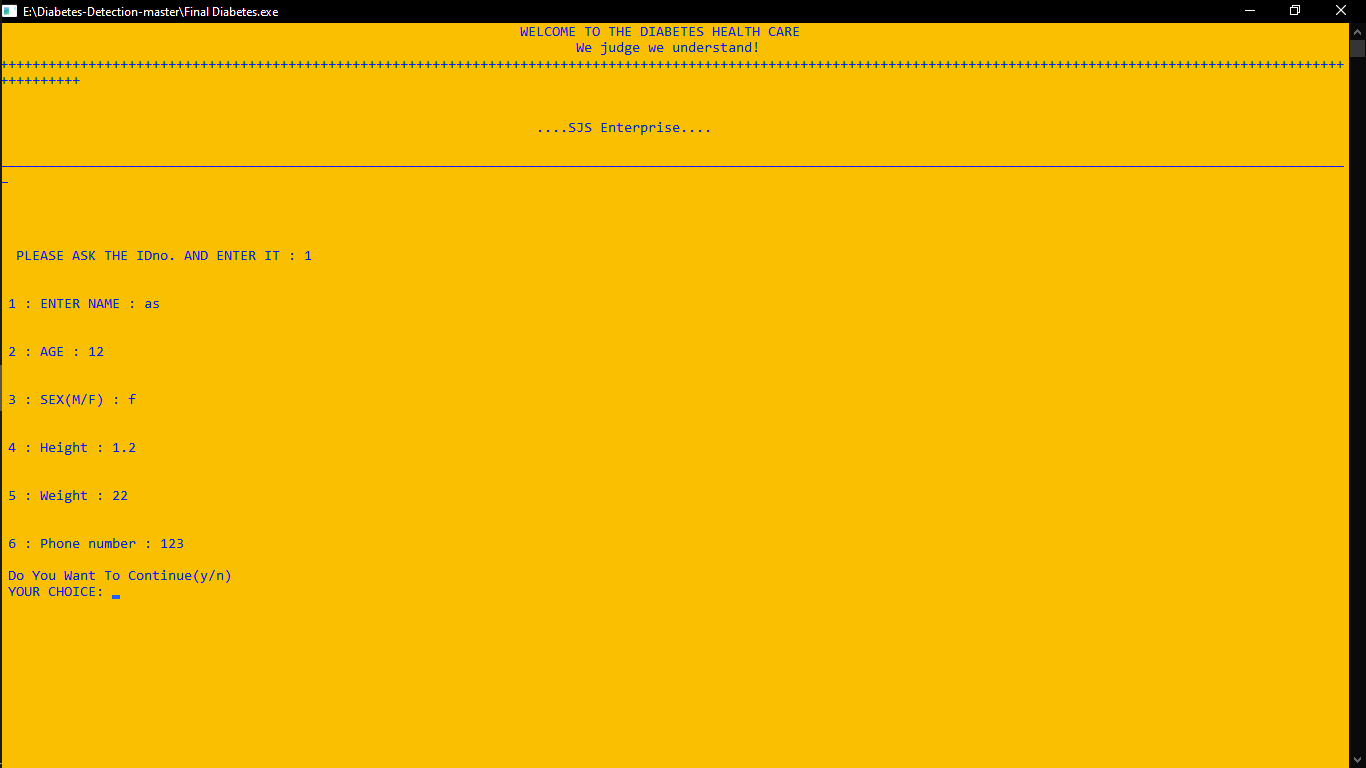
The figure 1.3 says that once the user clicks on the patient button, it comprises of enter data, take appointment, list of patients, test, pre-report, patient-login, final report, about developer of this code and exit .when clicked on list of patients button the patient name, id, sex, age, height and weight will be displayed successfully.



**Figure 1.3 Patient**

# 5.4 ENTER DATA

The figure 1.4 says that if the user wants to enter patient details, just by patient name, id, sex, age, height and weight.



**Figure 1.4 Enter Data**

# 5.5 ADMINISTER MENU

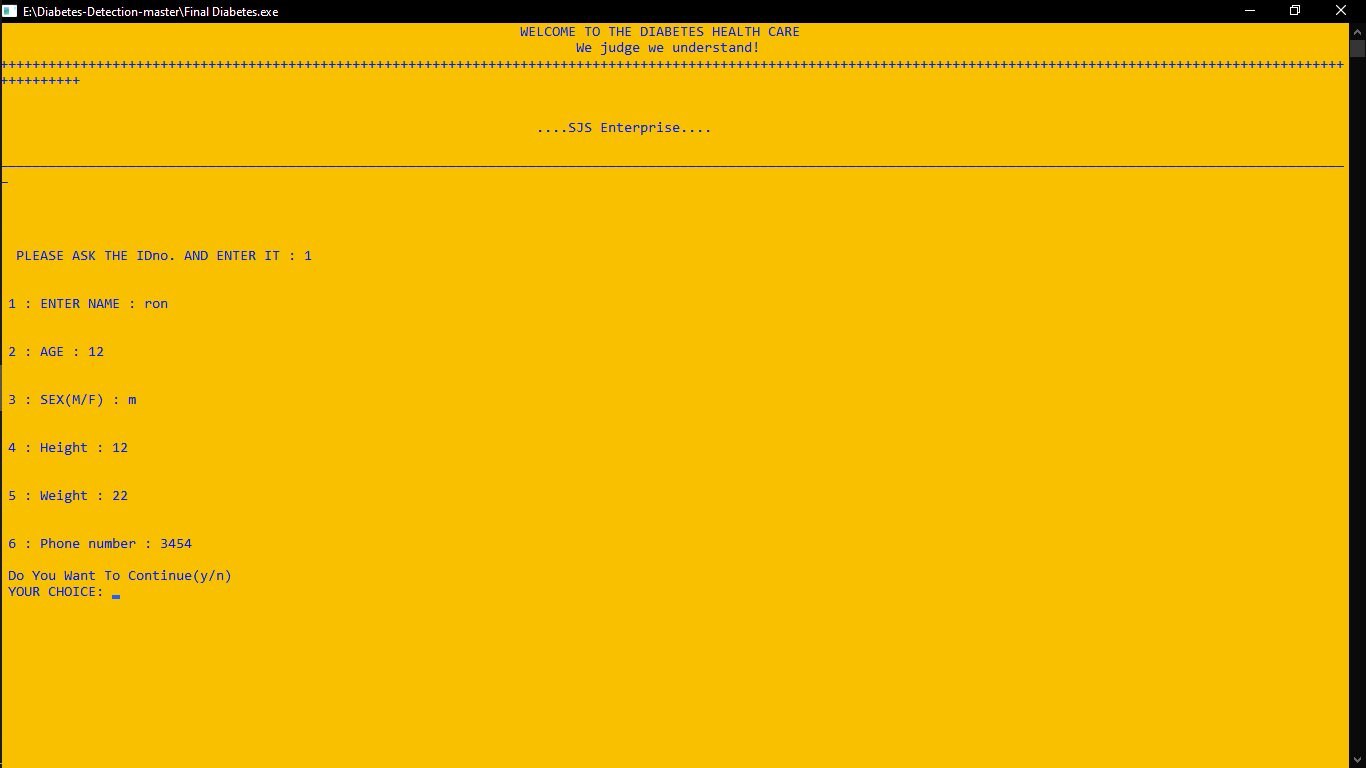
The figure 1.5 says that the administer menu includes list all data, search for patient, delete records, update records, count patients, admin records, admin signup, admin duties and logout . The same process is carried out by the books records.



**Figure 1.5 Administer menu**

# 5.6 ADDING PATIENT DETAILS

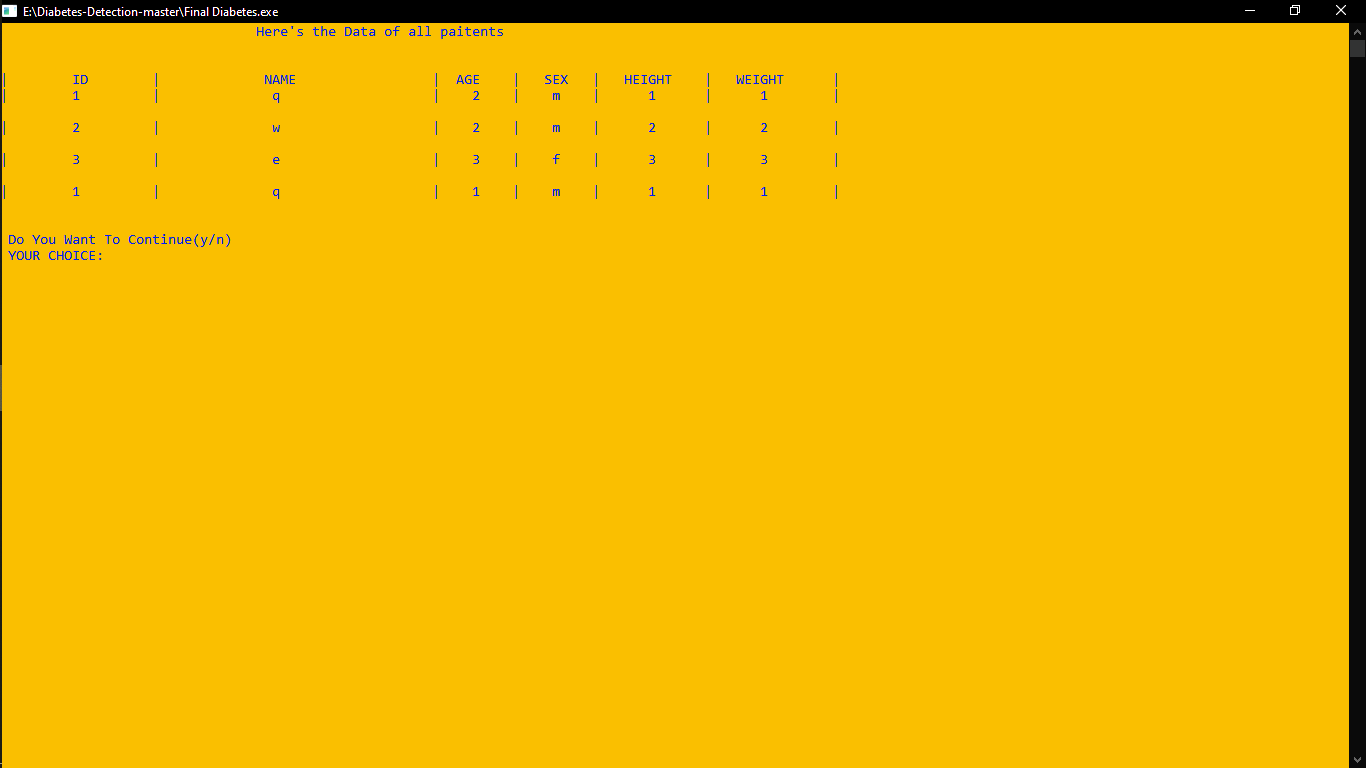
The figure 1.6 says that in order to add a patient details admission number and name of the patient has to be entered. After successfully entering the patient details it can be displayed on the table which includes admission number, hash number, name of the patient.



**Figure 1.6 Adding patient details**

# 5.7 DISPLAYING PATIENT DETAILS

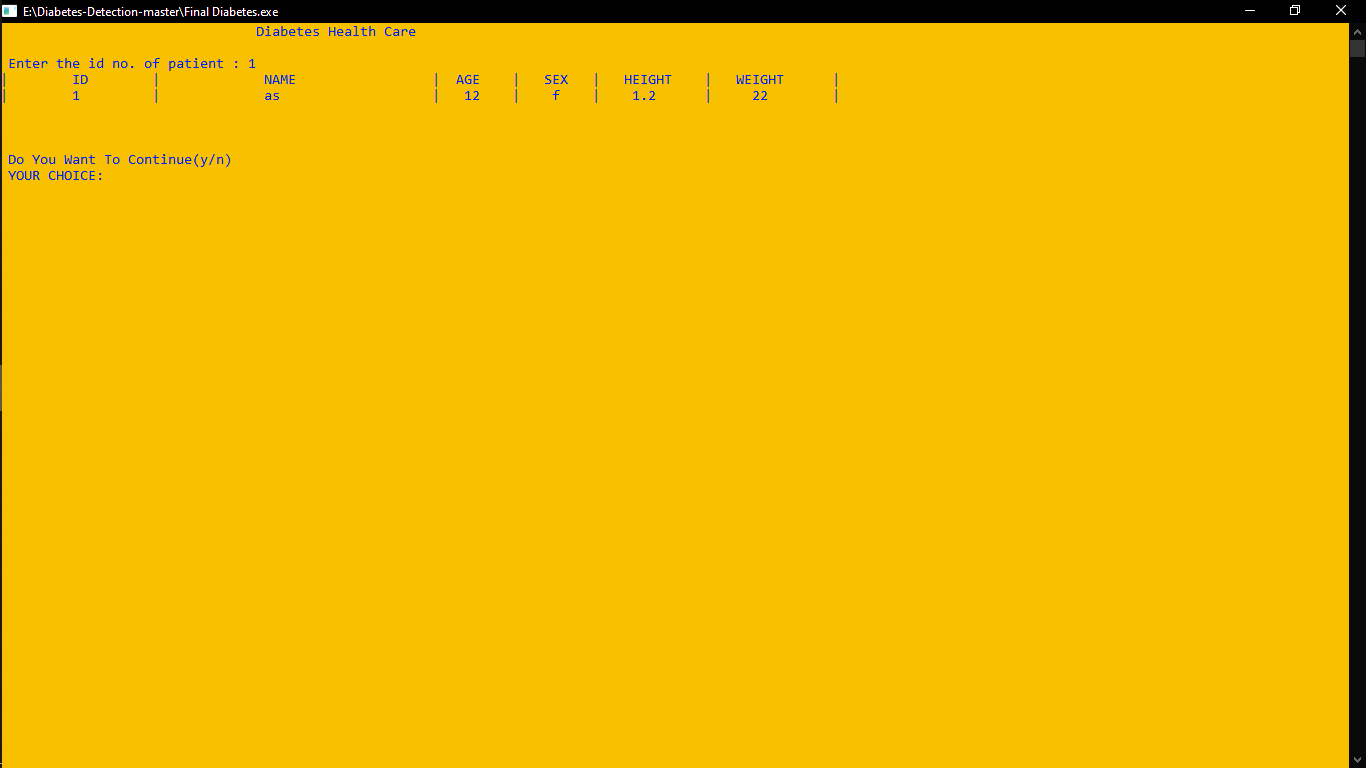
The figure 1.7 says that after successfully adding the patient details ,the newly entered record can be displayed successfully along with the existing records.



**Figure 1.7 Display patient details**

# 5.8 SEARCHING PATIENT DETAILS

The figure 1.8 says that the details of the patient can be searched by entering the patient id, once the search button has been selected the patient name will be shown automatically.



**Figure 1.8 Search pateint details**

# 5.9 MODIFYING PATIENT DETAILS

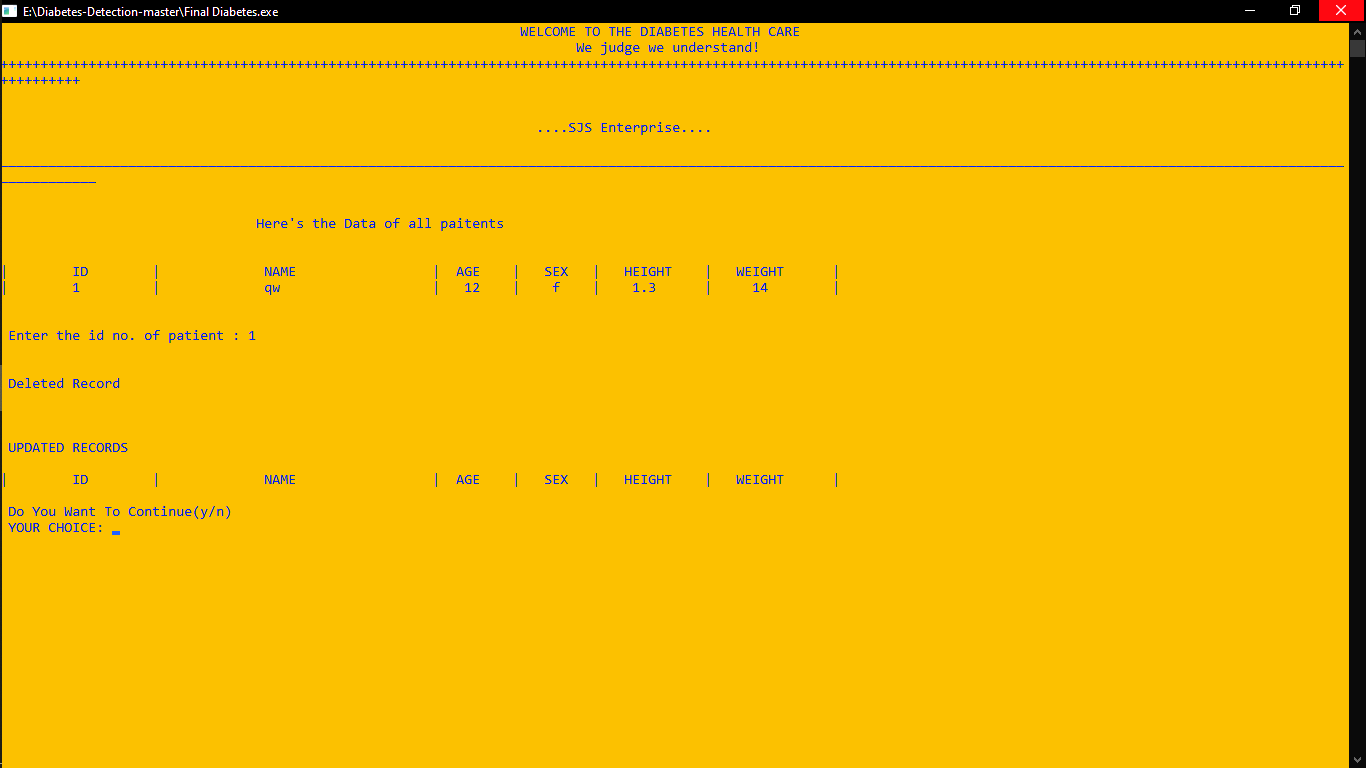
The figure 1.9 says that if any changes to be made in the patient details, it can be modified by altering the patient name and clicking on the modify button where the patient name will be modified successfully.



**Figure 1.9 Modify details**

# 5.8 DELETE PATIENT DETAILS

The figure 1.10 says that the patient details can be deleted and gets a conformation message saying that the patient details have been deleted successfully.



**Figure 1.10 Delete patient details**

# CONCLUSION AND FUTURE ENHANCEMENT

The project “DIABETES DETECTOR” is designed in order to reduce the burden of maintaining bulk records of all patient details. The main goal of building this file structure project is to retrieve the details faster and easier when compared to the manual Diabetes Detector. Maintaining the project and understanding the project is very easy. Maintaining the details in the files is manageable.

As the technology emerges, it is possible to upgrade the system and can be adaptable to the desired environment. Based on future security issues, security can be improved using emerging technologies. Sub admin module can be added. This can also be used at your home or your work place. This application can be easily implemented under various situations. We can add new features as and when required.

**BOOK**

# REFERENCES

FILE STRUCTURES- FOLK ZOELLICK RICCARDI EDITION 3

**WEBSITES**

# BIBLIOGRAPHY

Geeksforgeeks - [www.geeksforgeeks.com](http://www.geeksforgeeks.com/)

Stack overflow - [www.stackoverflow.com](http://www.stackoverflow.com/)