DOCTOR APPOINTMENT SYSTEM {DAS}

Project Report

TABLE OF CONTENTS

1. DECLARATION
2. CERTIFICATE
3. ACKNOWLEDGEMENT
4. ABSTRACT
5. INTRODUCTION
6. PROBLEM STATEMENT
7. OBJECTIVE
8. LITERATURE REVIEW
9. METHODOLOGY

10. SAMPLE C CODES

11. SNAPSHOTS OF RUNNING APPLICATION

12. SYSTEM ANALYSIS

13. FUTURE ENHANCEMENTS

14. CONCLUSION

15. REFRENCES

|  |  |  |  |
| --- | --- | --- | --- |
|  | | |  |
|  | |  |  |
|  | | |  |
|  | |  |  |
|  | | |  |
|  |  | |  |
|  |  | |  |
|  |  | |  |
|  |  | |  |
|  |  | |  |
|  | |  |  |

ABSTRACT

*The purpose of DOCTOR APPOINTMENT SYSTEM is to automate the existing manual system by the help of computerized equipments and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. The required software and hardware are easily available and easy to work with.*

*DOCTOR APPOINTMENT SYSTEM, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus it will help organization in better utilization of resources. The organization can maintain computerized records without redundant entries. That means that one need not be distracted by information that is not relevant, and doctors can focus of treating the patients.*

The aim is to automate its existing manual system by the help of computerized equipments and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. Basically the project describes how to manage for good performance and better services for the patients.

INTRODUCTION

Doctor Appointment system is the system where the users can search the doctors that the needed for and also can take appointment. The system will book appointments with doctors. Admin can manage doctors, patients. Administration have full authority to add, delete and update doctors and patients. Helps patients to get the doctors appointment’s any time. It make the best medical care available to you in the simplest form. It helps you save time of interacting with the doctors in a manual fashion and makes the same possible with diverse set of options available in the application. It services available 24/7 and can help you get in touch with the doctors directly. This help patients to book appointments from anywhere and from any device.

PROBLEM STATEMENT

DURING THIS PANDEMIC OF CORONA VIRUS IN MANY AREAS THERE WAS VERY POOR MANAGEMENT OF MEDICAL FACILITIES SO THIS PROJECT IS A WAY TO HANDLE THESE TYPE OF SITUATIONS SO THAT EVERY PATIENT CAN GET GOOD TREATMENT AND THIS WILL HAPPEN BY APPOINTING SUITABLE DOCTORS BASED ON THEIR MEDICAL CONDITION.

OBJECTIVE

The main objective of the Project on DOCTOR APPOINTMENT SYSTEM is to manage the details of Doctors, Patient, Doctor availability. The project is totally built at administrative end and thus only the administrator is guaranteed the access. The purpose of the project is to build an application program to reduce the manual work for managing the Hospital, Doctors, Medicine, Patient. It tracks all the details about the Patient, Doctor reports.

Functionalities provided by DOCTOR APPOINTMENT SYSTEM are as follows :--

It tracks all the information of Doctors, Medicine, Employee etc

 Manage the information of Doctors

 Shows the information and description of the Patient

 To increase efficiency of managing the Hospital, Doctors

 It deals with monitoring the information .

 Manage the information of Hospital

 Editing, adding and updating of Records is improved which results in proper resource management of Hospital data.

 Manage the information of Employee

LITERATURE REVIEW

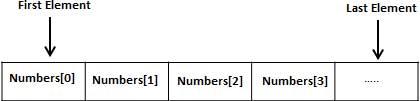
It may help collecting perfect management in details. In a very short time, the collection will be obvious, simple and sensible. It will help a person to know the management of passed year perfectly and vividly. It also helps in current all works the management & collection procedure will go on smoothly. Our project aims at Business process automation, i.e. we have tried to  In computer system, it is not necessary to create the manifest but we can directly print it, which saves our time.  To assist the staff in capturing the effort spent on their respective working areas.  To utilize resources in an efficient manner by increasing their productivity through automation.  The system generates types of information that can be used for various purposes.  It satisfy the user requirement  Be easy to understand by the user and operator  Be easy to operate  Have a good user interface  Be expandable  Delivered on schedule within the budget..

IN THIS PROJECT WE WILL USE MANY METHODS REGARDING C LANGUAGE LIKE ARRAYS , DATA STRUCTURES , FILES , FUNCTIONS, POINTERS..

**ARRAYS :**--- Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index.

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.



## Declaring Arrays

To declare an array in C, a programmer specifies the type of the elements and the number of elements required by an array as follows −

type arrayName [ arraySize ];

This is called a *single-dimensional* array. The arraySize must be an integer constant greater than zero and type can be any valid C data type. For example, to declare a 10-element array called balance of type double, use this statement −

double balance[10];

Here *balance* is a variable array which is sufficient to hold up to 10 double numbers.

## Initializing Arrays

You can initialize an array in C either one by one or using a single statement as follows −

double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};

**STRUCTURES** :- Structures are used to represent a record. Suppose you want to keep track of your books in a library. You might want to track the following attributes about each book −

* Title
* Author
* Subject
* Book ID

## Defining a Structure

To define a structure, you must use the struct statement. The struct statement defines a new data type, with more than one member. The format of the struct statement is as follows −

struct [structure tag] { member definition; member definition; ... member definition;} [one or more structure variables];

The structure tag is optional and each member definition is a normal variable definition, such as int i; or float f; or any other valid variable definition. At the end of the structure's definition, before the final semicolon, you can specify one or more structure variables but it is optional. Here is the way you would declare the Book structure −

struct Books { char title[50]; char author[50]; char subject[100]; int book\_id;} book; .

**FILES** :-- A file represents a sequence of bytes on the disk where a group of related data is stored. File is created for permanent storage of data. It is a ready made structure.

In C language, we use a structure pointer of file type to declare a file.

FILE \*fp;

C provides a number of functions that helps to perform basic file operations. Following are the functions,

|  |  |
| --- | --- |
| Function | description |
| fopen() | create a new file or open a existing file |
| fclose() | closes a file |
| getc() | reads a character from a file |
| putc() | writes a character to a file |
| fscanf() | reads a set of data from a file |
| fprintf() | writes a set of data to a file |
| getw() | reads a integer from a file |
| putw() | writes a integer to a file |
| fseek() | set the position to desire point |
| ftell() | gives current position in the file |
| rewind() | set the position to the beginning point |

**FUNCTIONS** : --- A function is a group of statements that together perform a task. Every C program has at least one function, which is main(), and all the most trivial programs can define additional functions.

You can divide up your code into separate functions. How you divide up your code among different functions is up to you, but logically the division is such that each function performs a specific task.

A function declaration tells the compiler about a function's name, return type, and parameters. A function definition provides the actual body of the function.

The C standard library provides numerous built-in functions that your program can call. For example, strcat() to concatenate two strings, memcpy() to copy one memory location to another location, and many more functions.

A function can also be referred as a method or a sub-routine or a procedure, etc.

## Defining a Function

The general form of a function definition in C programming language is as follows −

return\_type function\_name( parameter list ) { body of the function}.

**POINTER** :--- The pointer in C language is a variable which stores the address of another variable. This variable can be of type int, char, array, function, or any other pointer. The size of the pointer depends on the architecture. However, in 32-bit architecture the size of a pointer is 2 byte.

Consider the following example to define a pointer which stores the address of an integer.

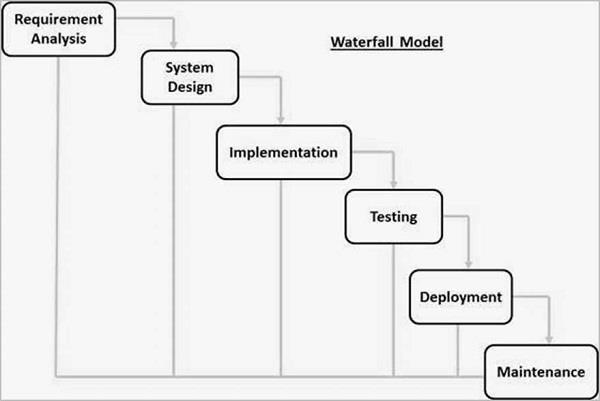
1. int n = 10;
2. int\* p = &n; // Variable p of type pointer is pointing to the address of the variable n of type integer.

## Declaring a pointer

The pointer in c language can be declared using \* (asterisk symbol). It is also known as indirection pointer used to dereference a pointer.

1. int \*a;//pointer to int
2. char \*c;//pointer to char

METHODOLOGY



Requirement Gathering and analysis − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

* System Design − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
* Implementation − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
* Integration and Testing − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
* Deployment of system − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
* Maintenance − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released.

C CODE

#include<stdio.h>

#include<windows.h>

#include<conio.h>

#include<ctype.h>

#include<string.h>

#include<stdlib.h>

char ans=0;

int ok;

int b, valid=0;

void WelcomeScreen(void);

void Title(void);

void MainMenu(void);

void LoginScreen(void);

void Add\_rec(void);

void func\_list();

void Search\_rec(void);

void Edit\_rec(void);

void Dlt\_rec(void);

void ex\_it(void);

//Defines gotoxy().

void gotoxy(short x, short y) {

COORD pos = {x, y};

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

}

struct patient

{

int age;

char Gender;

char First\_Name[20];

char Last\_Name[20];

char Contact\_no[15];

char Address[30];

char Email[30];

char Doctor[20];

char Problem[20];

};

struct patient p,temp\_c;

main(void)

{

WelcomeScreen();

Title();

LoginScreen();

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Welcome Screen \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

void WelcomeScreen(void)

{

printf("\n\n\n\n\n\n\n\t\t\t\t#########################################");

printf("\n\t\t\t\t#\t\t WELCOME TO\t\t#");

printf("\n\t\t\t\t# DOCTOR APPOINTMENT AND RECORDS \t#");

printf("\n\t\t\t\t#########################################");

printf("\n\n\n\n\n Press any key to continue......\n");

getch();

system("cls");

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Title Screen \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

void Title(void)

{

printf("\n\n\t\t----------------------------------------------------------------------------------");

printf("\n\t\t\t\t\t\t PORTAL ");

printf("\n\t\t----------------------------------------------------------------------------------");

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Main Menu \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

void MainMenu(void)

{

system("cls");

int choose;

Title();

printf("\n\n\n\n\n\t\t\t\t1. Add Patients Record\n");

printf("\n\t\t\t\t2. List Patients Record\n");

printf("\n\t\t\t\t3. Search Patients Record\n");

printf("\n\t\t\t\t4. Edit Patients Record\n");

printf("\n\t\t\t\t5. Delete Patients Record\n");

printf("\n\t\t\t\t6. Exit\n");

printf("\n\n\n \n\t\t\t\tChoose from 1 to 6:");

scanf("%i", &choose);

switch(choose)

{

case 1:

Add\_rec();

break;

case 2:

func\_list();

break;

case 3:

Search\_rec();

break;

case 4:

Edit\_rec();

break;

case 5:

Dlt\_rec();

break;

case 6:

ex\_it();

break;

default:

printf("\t\t\tInvalid entry. Please enter right option :)");

getch();

}

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Exit Screen \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

void ex\_it(void)

{

system("cls");

Title();

printf("\n\n\n\n\n\t\t\tTHANK YOU FOR VISITING :)");

getch();

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Login Screen \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

void LoginScreen(void)

{

int e=0 ;

char Username[15];

char Password[15];

char original\_Username[25]="UPES";

char original\_Password[15]="MINOR";

do

{

printf("\n\n\n\n\t\t\t\tEnter your Username and Password :)");

printf("\n\n\n\t\t\t\t\tUSERNAME:");

scanf("%s",&Username);

printf("\n\n\t\t\t\t\tPASSWORD:");

scanf("%s",&Password);

if (strcmp(Username,original\_Username)==0 && strcmp(Password,original\_Password)==0)

{

printf("\n\n\n\t\t\t\t\t...Login Successfull...");

getch();

MainMenu();

break;

}

else

{

printf("\n\t\t\tPassword is incorrect:( Try Again :)");

e++;

getch();

}

}

while(e<=2);

if(e>2)

{

printf("You have cross the limit. You cannot login. :( :(");

getch();

ex\_it();

}

system("cls");

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ADD RECORD\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Add\_rec(void)

{

system("cls");

Title();

char ans;

FILE\*ek;

ek=fopen("Record2.dat","a");

printf("\n\n\t\t\t!!!!!!!!!!!!!! Add Patients Record !!!!!!!!!!!!!\n");

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*First Name\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

A:

printf("\n\t\t\tFirst Name: ");

scanf("%s",p.First\_Name);

p.First\_Name[0]=toupper(p.First\_Name[0]);

if(strlen(p.First\_Name)>20||strlen(p.First\_Name)<2)

{

printf("\n\t Invalid :( \t The max range for first name is 20 and min range is 2 :)");

goto A;

}

else

{

for (b=0;b<strlen(p.First\_Name);b++)

{

if (isalpha(p.First\_Name[b]))

{

valid=1;

}

else

{

valid=0;

break;

}

}

if(!valid)

{

printf("\n\t\t First name contain Invalid character :( Enter again :)");

goto A;

}

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Last name \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

B:

printf("\n\t\t\tLast Name: ");

scanf("%s",p.Last\_Name);

p.Last\_Name[0]=toupper(p.Last\_Name[0]);

if(strlen(p.Last\_Name)>20||strlen(p.Last\_Name)<2)

{

printf("\n\t Invalid :( \t The max range for last name is 20 and min range is 2 :)");

goto B;

}

else

{

for (b=0;b<strlen(p.Last\_Name);b++)

{

if (isalpha(p.Last\_Name[b]))

{

valid=1;

}

else

{

valid=0;

break;

}

}

if(!valid)

{

printf("\n\t\t Last name contain Invalid character :( Enter again :)");

goto B;

}

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Gender \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

do

{

printf("\n\t\t\tGender[M/F]: ");

scanf(" %c",&p.Gender);

if(toupper(p.Gender)=='M'|| toupper(p.Gender)=='F')

{

ok =1;

}

else

{

ok =0;

}

if(!ok)

{

printf("\n\t\t Gender contain Invalid character :( Enter either F or M :)");

}

} while(!ok);

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Age \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

printf("\n\t\t\tAge:");

scanf(" %i",&p.age);

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Address \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

do

{

C:

printf("\n\t\t\tAddress: ");

scanf("%s",p.Address);

p.Address[0]=toupper(p.Address[0]);

if(strlen(p.Address)>20||strlen(p.Address)<4)

{

printf("\n\t Invalid :( \t The max range for address is 20 and min range is 4 :)");

goto C;

}

}while(!valid);

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Contact no. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

do

{

D:

printf("\n\t\t\tContact no: ");

scanf("%s",p.Contact\_no);

if(strlen(p.Contact\_no)>10||strlen(p.Contact\_no)!=10)

{

printf("\n\t Sorry :( Invalid. Contact no. must contain 10 numbers. Enter again :)");

goto D;

}

else

{

for (b=0;b<strlen(p.Contact\_no);b++)

{

if (!isalpha(p.Contact\_no[b]))

{

valid=1;

}

else

{

valid=0;

break;

}

}

if(!valid)

{

printf("\n\t\t Contact no. contain Invalid character :( Enter again :)");

goto D;

}

}

}while(!valid);

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Email \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

do

{

printf("\n\t\t\tEmail: ");

scanf("%s",p.Email);

if (strlen(p.Email)>30||strlen(p.Email)<8)

{

printf("\n\t Invalid :( \t The max range for email is 30 and min range is 8 :)");

}

}while(strlen(p.Email)>30||strlen(p.Email)<8);

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Problem \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

E:

printf("\n\t\t\tProblem: ");

scanf("%s",p.Problem);

p.Problem[0]=toupper(p.Problem[0]);

if(strlen(p.Problem)>15||strlen(p.Problem)<3)

{

printf("\n\t Invalid :( \t The max range for first name is 15 and min range is 3 :)");

goto E;

}

else

{

for (b=0;b<strlen(p.Problem);b++)

{

if (isalpha(p.Problem[b]))

{

valid=1;

}

else

{

valid=0;

break;

}

}

if(!valid)

{

printf("\n\t\t Problem contain Invalid character :( Enter again :)");

goto E;

}

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Prescribed Doctor \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

F:

printf("\n\t\t\tPrescribed Doctor:");

scanf("%s",p.Doctor);

p.Doctor[0]=toupper(p.Doctor[0]);

if(strlen(p.Doctor)>30||strlen(p.Doctor)<3)

{

printf("\n\t Invalid :( \t The max range for first name is 30 and min range is 3 :)");

goto F;

}

else

{

for (b=0;b<strlen(p.Doctor);b++)

{

if (isalpha(p.Doctor[b]))

{

valid=1;

}

else

{

valid=0;

break;

}

}

if(!valid)

{

printf("\n\t\t Doctor name contain Invalid character :( Enter again :)");

goto F;

}

}

fprintf(ek," %s %s %c %i %s %s %s %s %s\n", p.First\_Name, p.Last\_Name, p.Gender, p.age, p.Address, p.Contact\_no, p.Email, p.Problem, p.Doctor);

printf("\n\n\t\t\t.... Information Record Successful ...");

fclose(ek);

sd:

getch();

printf("\n\n\t\t\tDo you want to add more[Y/N]?? ");

scanf(" %c",&ans);

if (toupper(ans)=='Y')

{

Add\_rec();

}

else if(toupper(ans)=='N')

{

printf("\n\t\t Thank you :) :)");

getch();

MainMenu();

}

else

{

printf("\n\t\tInvalid Input\n");

goto sd;

}

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*VIEW RECORD\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void func\_list()

{

int row;

system("cls");

Title();

FILE \*ek;

ek=fopen("Record2.dat","r");

printf("\n\n\t\t\t!!!!!!!!!!!!!! List Patients Record !!!!!!!!!!!!!\n");

gotoxy(1,15);

printf("Full Name");

gotoxy(20,15);

printf("Gender");

gotoxy(32,15);

printf("Age");

gotoxy(37,15);

printf("Address");

gotoxy(49,15);

printf("Contact No.");

gotoxy(64,15);

printf("Email");

gotoxy(88,15);

printf("Problem");

gotoxy(98,15);

printf("Prescribed Doctor\n");

printf("=================================================================================================================");

row=17;

while(fscanf(ek,"%s %s %c %i %s %s %s %s %s\n", p.First\_Name, p.Last\_Name,

&p.Gender, &p.age, p.Address, p.Contact\_no, p.Email, p.Problem, p.Doctor)!=EOF)

{

gotoxy(1,row);

printf("%s %s",p.First\_Name, p.Last\_Name);

gotoxy(20,row);

printf("%c",p.Gender);

gotoxy(32,row);

printf("%i",p.age);

gotoxy(37,row);

printf("%s",p.Address);

gotoxy(49,row);

printf("%s",p.Contact\_no);

gotoxy(64,row);

printf("%s",p.Email);

gotoxy(88,row);

printf("%s",p.Problem);

gotoxy(98,row);

printf("%s",p.Doctor);

row++;

}

fclose(ek);

getch();

MainMenu();

}

void Search\_rec(void)

{

char name[20];

system("cls");

Title();// call Title function

FILE \*ek;

ek=fopen("Record2.dat","r");

printf("\n\n\t\t\t!!!!!!!!!!!!!! Search Patients Record !!!!!!!!!!!!!\n");

gotoxy(12,8);

printf("\n Enter Patient Name to be viewed:");

scanf("%s",name);

fflush(stdin);

name[0]=toupper(name[0]);

while(fscanf(ek,"%s %s %c %i %s %s %s %s %s\n", p.First\_Name, p.Last\_Name, &p.Gender, &p.age, p.Address, p.Contact\_no, p.Email, p.Problem, p.Doctor)!=EOF)

{

if(strcmp(p.First\_Name,name)==0)

{

gotoxy(1,15);

printf("Full Name");

gotoxy(25,15);

printf("Gender");

gotoxy(32,15);

printf("Age");

gotoxy(37,15);

printf("Address");

gotoxy(52,15);

printf("Contact No.");

gotoxy(64,15);

printf("Email");

gotoxy(80,15);

printf("Problem");

gotoxy(95,15);

printf("Prescribed Doctor\n");

printf("=================================================================================================================");

gotoxy(1,18);

printf("%s %s",p.First\_Name, p.Last\_Name);

gotoxy(25,18);

printf("%c",p.Gender);

gotoxy(32,18);

printf("%i",p.age);

gotoxy(37,18);

printf("%s",p.Address);

gotoxy(52,18);

printf("%s",p.Contact\_no);

gotoxy(64,18);

printf("%s",p.Email);

gotoxy(80,18);

printf("%s",p.Problem);

gotoxy(95,18);

printf("%s",p.Doctor);

printf("\n");

break;

}

}

if(strcmp(p.First\_Name,name)!=0)

{

gotoxy(5,10);

printf("Record not found!");

getch();

}

fclose(ek);

L:

getch();

printf("\n\n\t\t\tDo you want to view more[Y/N]??");

scanf("%c",&ans);

if (toupper(ans)=='Y')

{

Search\_rec();

}

else if(toupper(ans)=='N')

{

printf("\n\t\t Thank you :) :)");

getch();

MainMenu();

}

else

{

printf("\n\tInvalid Input.\n");

goto L;

}

}

void Edit\_rec(void)

{

FILE \*ek, \*ft;

int i,b, valid=0;

char ch;

char name[20];

system("cls");

Title();// call Title window

ft=fopen("temp2.dat","w+");

ek=fopen("Record2.dat","r");

if(ek==NULL)

{

printf("\n\t Can not open file!! ");

getch();

MainMenu();

}

printf("\n\n\t\t\t!!!!!!!!!!!!!! Edit Patients Record !!!!!!!!!!!!!\n");

gotoxy(12,13);

printf("Enter the First Name of the Patient : ");

scanf(" %s",name);

fflush(stdin);

name[0]=toupper(name[0]);

gotoxy(12,15);

if(ft==NULL)

{

printf("\n Can not open file");

getch();

MainMenu();

}

while(fscanf(ek,"%s %s %c %i %s %s %s %s %s\n", p.First\_Name, p.Last\_Name, &p.Gender, &p.age, p.Address, p.Contact\_no, p.Email, p.Problem, p.Doctor)!=EOF)

{

if(strcmp(p.First\_Name, name)==0)

{

valid=1;

gotoxy(25,17);

printf("\*\*\* Existing Record \*\*\*");

gotoxy(10,19);

printf("%s \t%s \t%c \t%i \t%s \t%s \t%s \t%s \t%s\n",p.First\_Name,p.Last\_Name,p.Gender, p.age,p.Address,p.Contact\_no,p.Email,p.Problem,p.Doctor);

gotoxy(12,22);

printf("Enter New First Name: ");

scanf("%s",p.First\_Name);

gotoxy(12,24);

printf("Enter Last Name: ");

scanf("%s",p.Last\_Name);

gotoxy(12,26);

printf("Enter Gender: ");

scanf(" %c",&p.Gender);

p.Gender=toupper(p.Gender);

gotoxy(12,28);

printf("Enter age: ");

scanf(" %i",&p.age);

gotoxy(12,30);

printf("Enter Address: ");

scanf("%s",p.Address);

p.Address[0]=toupper(p.Address[0]);

gotoxy(12,32);

printf("Enter Contact no: ");

scanf("%s",p.Contact\_no);

gotoxy(12,34);

printf("Enter New email: ");

scanf("%s",p.Email);

gotoxy(12,36);

printf("Enter Problem: ");

scanf("%s",p.Problem);

p.Problem[0]=toupper(p.Problem[0]);

gotoxy(12,38);

printf("Enter Doctor: ");

scanf("%s",p.Doctor);

p.Doctor[0]=toupper(p.Doctor[0]);

printf("\nPress U charecter for the Updating operation : ");

ch=getche();

if(ch=='u' || ch=='U')

{

fprintf(ft,"%s %s %c %i %s %s %s %s %s\n",p.First\_Name,p.Last\_Name,p.Gender, p.age,p.Address,p.Contact\_no,p.Email,p.Problem,p.Doctor);

printf("\n\n\t\t\tPatient record updated successfully...");

}

}

else

{

fprintf(ft,"%s %s %c %i %s %s %s %s %s\n",p.First\_Name,p.Last\_Name,p.Gender, p.age,p.Address,p.Contact\_no,p.Email,p.Problem,p.Doctor);

}

}

if(!valid) printf("\n\t\tNO RECORD FOUND...");

fclose(ft);

fclose(ek);

remove("Record2.dat");

rename("temp2.dat","Record2.dat");

getch();

MainMenu();

}

void Dlt\_rec()

{

char name[20];

int found=0;

system("cls");

Title();// call Title function

FILE \*ek, \*ft;

ft=fopen("temp\_file2.dat","w+");

ek=fopen("Record2.dat","r");

printf("\n\n\t\t\t!!!!!!!!!!!!!! Delete Patients Record !!!!!!!!!!!!!\n");

gotoxy(12,8);

printf("\n Enter Patient'S FIRST Name to delete: ");

fflush(stdin);

gets(name);

name[0]=toupper(name[0]);

while (fscanf(ek,"%s %s %c %i %s %s %s %s %s", p.First\_Name, p.Last\_Name, &p.Gender,

&p.age, p.Address, p.Contact\_no, p.Email, p.Problem, p.Doctor)!=EOF)

{

if (strcmp(p.First\_Name,name)!=0)

fprintf(ft,"%s %s %c %i %s %s %s %s %s\n", p.First\_Name, p.Last\_Name, p.Gender, p.age, p.Address, p.Contact\_no, p.Email, p.Problem, p.Doctor);

else

{

printf("%s %s %c %i %s %s %s %s %s\n", p.First\_Name, p.Last\_Name, p.Gender, p.age, p.Address, p.Contact\_no, p.Email, p.Problem, p.Doctor);

found=1;

}

}

if(found==0)

{

printf("\n\n\t\t\t Record not found....");

getch();

MainMenu();

}

else

{

fclose(ek);

fclose(ft);

remove("Record2.dat");

rename("temp\_file2.dat","Record2.dat");

printf("\n\n\t\t\t Record deleted successfully :) ");

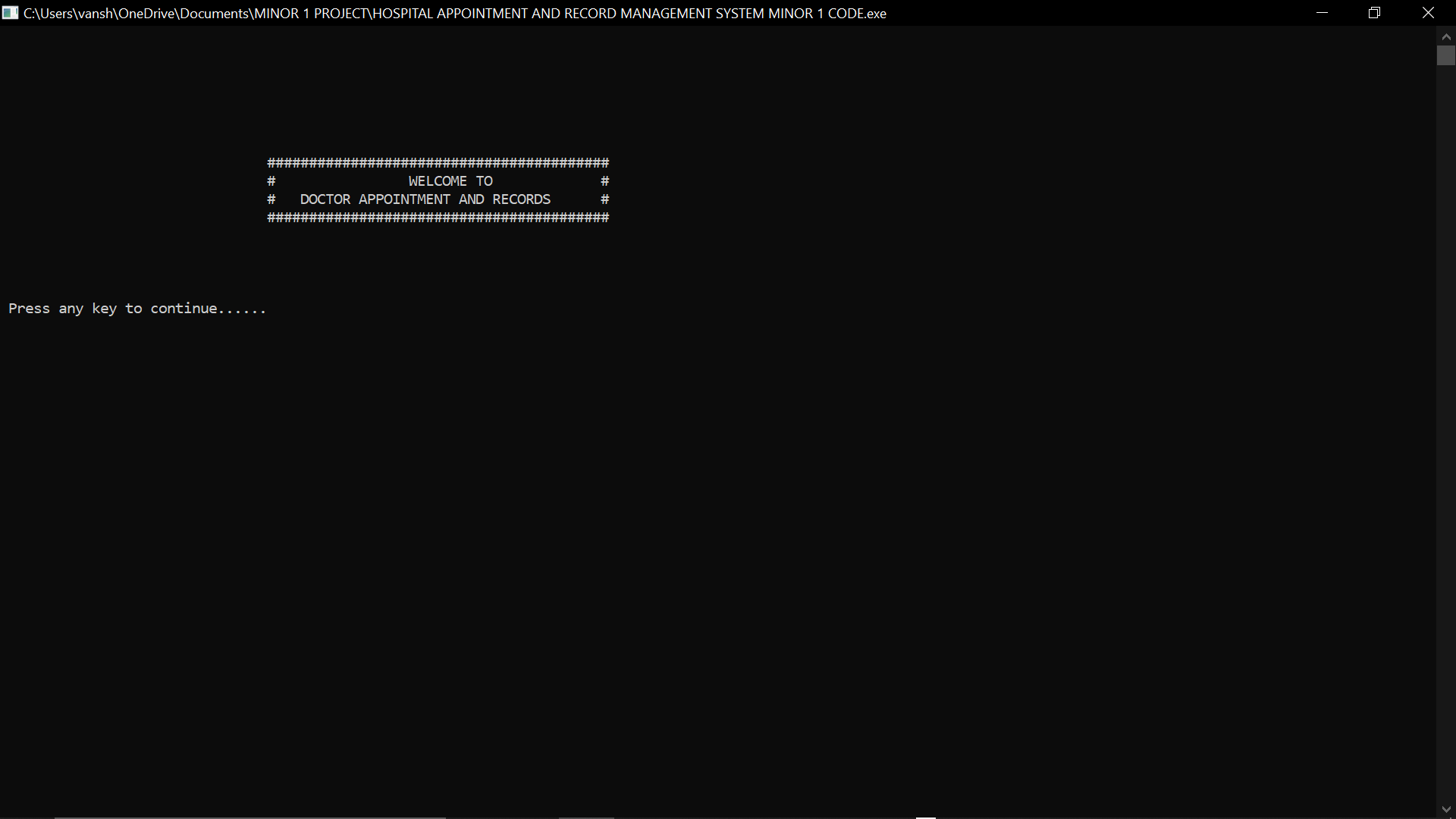
getch();

MainMenu();

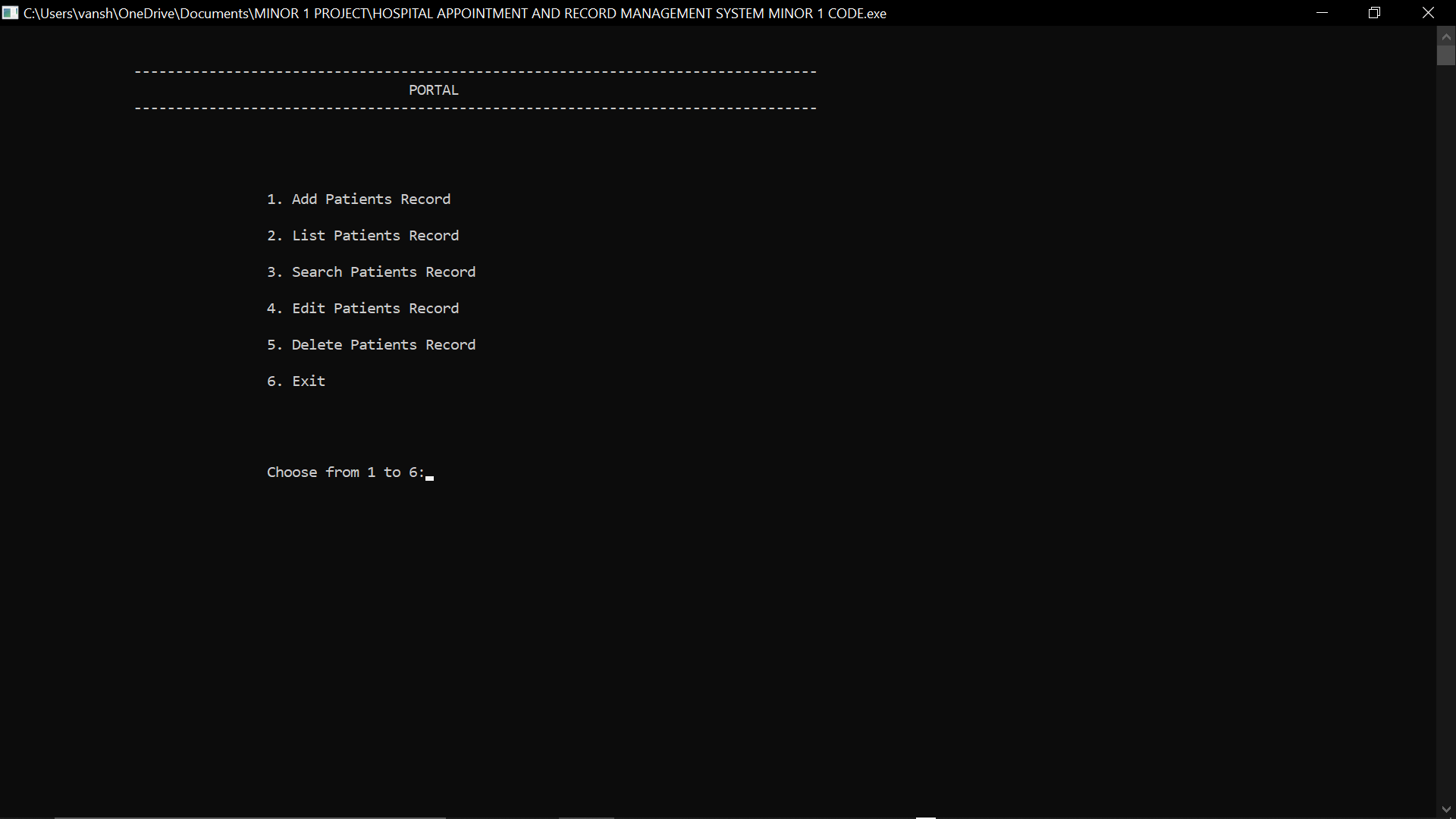
}

}

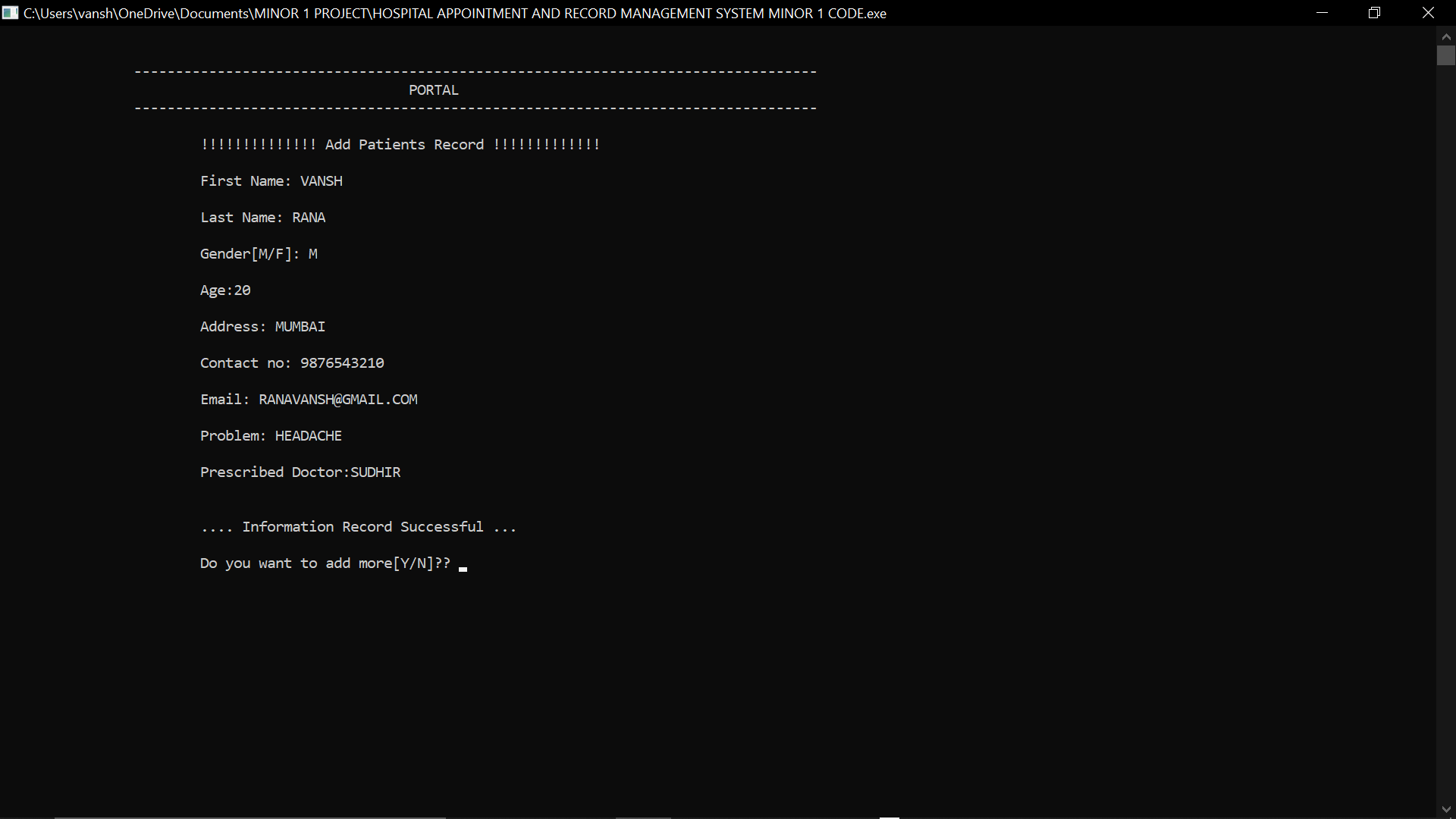
SNAPSHOTS OF RUNNING APPLICATION



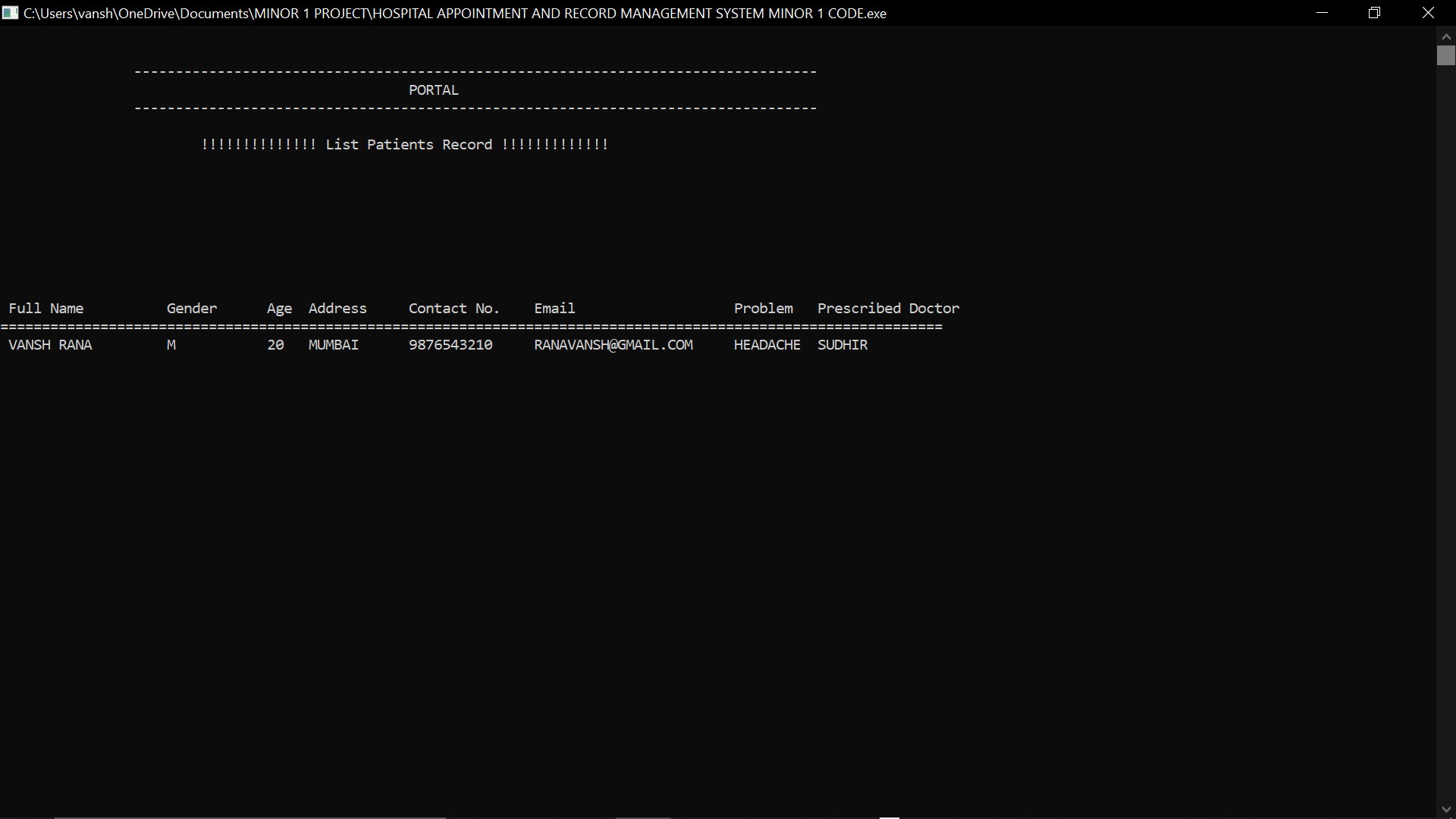
**I) WELCOME WINDOW**



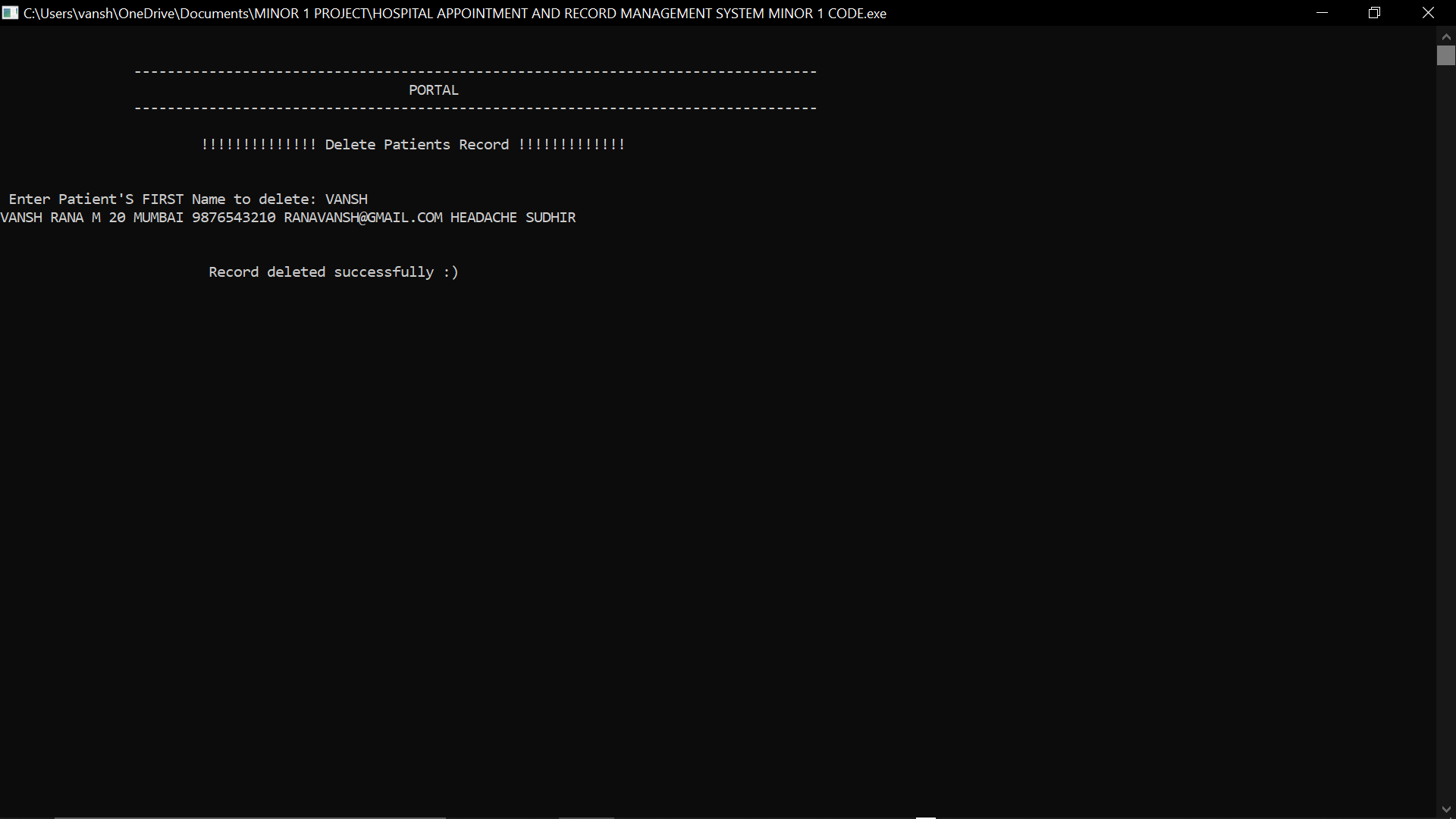
**II) MAIN MENU**



**III) ADDITION OF PATIENT RECORD AND APPOINTING DOCTOR**



**IV) DETAILS OF PATIENTS IN FILE**



**V) DELETION OF RECORDS**

SYSTEM ANALYSIS

System Analysis It is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components. System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose. The objective of the system analysis activity is to develop structured system specification for the proposed system. The structured system specification should describe what the proposed system would do; independent of the technology, which will be used to implement these requirements. The structured system specification will be used to implement these requirements. The essential model may itself consist of multiple models, modeling different aspect of the system. The data flow diagrams may model the data and there relationships and the state transition diagram may model time dependent behavior of the system.

FUTURE ENHANCEMENTS

1. MANY MEDICAL INSTITUTES CAN APPLY THIS METHOD IN THEIR MANAGEMENT.
2. WE CAN ADD DATABASES CONTAINING INFORMATION ABOUT EQUIPMENTS NEW FOUND DISEASES SO THAT NECESSARY THINGS CAN BE APPOINT TO PEOPLES IN NEED.
3. WE CAN INCREASE THE SCOPE OF THE PROJECT FROM A SINGLE INSTITUTE TO A GROUP OF INSTITUTES.

CONCLUSION

* We are able to successfully appoint doctors to specified patients according to their conditions.
* Information of Doctors and patients are stored in the FILES successfully.
* We created a successful application tool for maintaining the records of patients and their respective doctors assigned to them.