 **Geethanjali College of Engineering and Technology**

**(Autonomous)**

## Cheeryal (V), Keesara (M) – 501301 (T.S)

*Data Structures Project Report on* **“ANAMNESIS”**

Branch:-CSE-IOT

**Submitted to:** D. DEEPTHI SREE

**Submitted By:**

1. GVNS.TEJASWI(21R11A6926)
2. K.B. PRASANNA (21R11A6930)
3. S. LIKITHA (21R11A6959)
4. G. DINESH (21R11A6924)
5. B. CHAKRADHAR (22R15A6904)

# Abstract:-

The purpose of the project entitled as ANAMNESIS which is known for HOSPITAL MANAGEMENT SYSTEM. It is to computerized the Front Office Management of Hospital to develop software which is user friendly, simple, fast, and cost-effective. It deals with the collection of patients information like add patient, update patient, delete patient, search patient, view patient details etc. Traditionally, it was done manually. The main function of the system is register and store patient details and retrieve these details as and when required, and also to manipulate these details meaningfully. The Hospital Management System can be entered using a username and password. It is accessible by the Admin, Doctor, and Receptionist. The data are well protected for personal use and makes the data processing is very fast.

Keywords:-anamnesis,admin,receptionist

# **Introduction:-**

Hospital Management System is designed to improve the quality and management of hospital in the arears of clinical process analysis and activity -based costing. Hospital Management System enables you to develop your organization and improve its effectiveness and quality of work. Managing the key processes efficiently is critical to the success of the hospital helps you manage your processes. This includes registration of patients, storing their details into the system, and also computerized billing in the pharmacy, and labs. The software has the facility to give a unique id for every patient and stores the details of every patient and the staff automatically. It includes a search facility to know the current status of each room. The Hospital Management System can be entered using a username and password. It is accessible either by an administrator or receptionist. Only they can add data. The data can be retrieved easily. The interface is very user-friendly. The data are well protected for personal use and makes the data processing very fast. Hospital Management System is powerful, flexible, and easy to use and is designed and developed to deliver real conceivable benefits to hospitals.Hospital Management System is designed for multispeciality hospitals, to cover a wide range of hospital administration and management processes. It is an integrated end-to-end Hospital Management System that provides relevant information across the hospital to support effective decision making for patient care, hospital administration and critical financial accounting, in a seamless flow. Hospital Management System is a software product suite designed to improve the quality and management of hospital management in the areas of clinical process analysis and activity-based costing. Hospital Management System enables you to develop your organization and improve its effectiveness and quality of work. Managing the key processes efficiently is critical to the success of the hospital helps you manage your processes.

Purpose:- Main aim of developing ANAMNESIS is to provide an easy way not only to automate all functionalities involved managing leaves and Payroll for the employees of Company, but also to provide full functional reports to management of Hospital with the necessary details.

Proposed System:- Hospital Management System is designed for any hospital to replace their existing manual paper based system. The new system is to control the information of patients. Room availability, staff and operating room schedules and patient invoices. These services are to be provided in an efficient, cost effective manner, with the goal of reducing the time and resources currently required for such tasks.

**Objectives of the System:-** The project “Anamnesis” is aimed to develop the day-by- day state of admission of patients, reports generation and etc.It is designed to achieve the following objectives:

* To computerize all the details regarding patient details & hospital details.
* Scheduling the appointment of patient with doctors to make it convenient for both.
* It should be able to handle the test reports of patients conducted in the pathology lab of the hospital.
* The inventory should update automatically whenever a transaction is made.
* The information of the patients should be kept up to date and there record should be kept in the system for historical purposes.

**Project Modules:-** This application is based o[n](https://aticleworld.com/file-handling-in-c/) [file handling in C,](https://aticleworld.com/file-handling-in-c/) where I have used a filerelated function like [fopen,](https://aticleworld.com/fopen-in-c/) [fread,](https://aticleworld.com/fread-in-c/) [fwrite,](https://aticleworld.com/fwrite-in-c/)..etc. Good thing is that the “Hospital Management System Project” is password-protected, so only authorized persons are able to log in to this application. Also to increase the readability, I have broken the application into different functions. Each function of the project extensively uses [file handing function.](https://aticleworld.com/file-handling-in-c/)

The functions are

* isFileExists()
* init()
* printMessageCenter()
* headMessage()
* welcomeMessage()
* isNameValid()
* isValidDate()
* menu()
* addPatientInDataBase()
* searchPatient()
* viewPatient()
* deletePatient()
* updateCredential()

**isFileExists():**This function verifies that a file has been created or not. If the file exists, the function return 1 otherwise returns 0.

//Check file exist or not

**int** isFileExists(const **char** \*path)

{

// Try to open file

FILE \*fp = fopen(path, "rb");

**int** status = 0;

// If file does not exists

**if** (fp != **NULL**)

{

status = 1;

// File exists hence close file fclose(fp);

}

**return** status;

}

**init():**This function creates the file if not exist and copies the default password (“aticleworld”) in file header structure.

**void** init()

{

FILE \*fp = **NULL**;

int status = 0;

const char defaultUsername[] ="aticleworld"; const char defaultPassword[] ="aticleworld"; sFileHeader fileHeaderInfo = {0};

status = isFileExists(FILE\_NAME);

**if**(!status)

{

//create the binary file

fp = fopen(FILE\_NAME,"wb");

**if**(fp != **NULL**)

{

//Copy default password strncpy(fileHeaderInfo.password,defaultPassword,sizeof(defaultPassword)); strncpy(fileHeaderInfo.username,defaultUsername,sizeof(defaultUsername)); fwrite(&fileHeaderInfo,FILE\_HEADER\_SIZE, 1, fp);

fclose(fp);

}

}

}

**printMessageCenter():**This function prints the message in the middle of the head massage. I have passed the message in this function as per the operation.

//Align the message

**void** printMessageCenter(const **char**\* message)

{

**int** len =0;

**int** pos = 0;

//calculate how many space need to print len = (78 - strlen(message))/2; printf("\t\t\t");

**for**(pos =0 ; pos < len ; pos++)

{

//print space printf(" ");

}

//print message printf("%s",message);

}

headMessage():It prints the message on the top of the console and prints the message as per operation.

//Head message

**void** headMessage(const **char** \*message)

{

system("cls");

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

printf("\n\t\t\t############ ############"); printf("\n\t\t\t############ Patient Record Management System Project in C ############"); printf("\n\t\t\t############ ############");

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

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

printMessageCenter(message);

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

}

.

**welcomeMessage():**This function displays the first welcomes screen of the “Hospital Management System Project in C” and asks the user to press any key to access the library application.

//Display message

**void** welcomeMessage()

{

headMessage("[www.aticleworld.com"](http://www.aticleworld.com/)); printf("\n\n\n\n\n");

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

printf("\n\t\t\t printf("\n\t\t\t printf("\n\t\t\t printf("\n\t\t\t printf("\n\t\t\t printf("\n\t\t\t

printf("\n\t\t\t

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=");

=

=

=

=

=

WELCOME TO

Patient Record

=");

=");

MANAGEMENT

SYSTEM

=");

=");

=");

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=");

printf("\n\t\t\t \*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*\n"); printf("\n\n\n\t\t\t Enter any key to continue ");

getchar();

}

**isNameValid():**It validates the user name, author name ..etc. I have permitted this function to take the space in names.

//Validate name

**int** isNameValid(const **char** \*name)

{

**int** validName = 1;

**int** len = 0;

**int** index = 0;

len = strlen(name);

**for**(index =0; index <len ; ++index)

{

**if**(!(isalpha(name[index])) && (name[index] != '\n') && (name[index] != ' '))

{

validName = 0;

**break**;

}

}

**return** validName;

}

**isValidDate():**This function reads the date in the format of dd/mm/yyyy also validates the entered date.

**int** IsLeapYear(**int** year)

{

**return** (((year % 4 == 0) && (year % 100 != 0)) ||

(year % 400 == 0));

}

// returns 1 if given date is valid.

**int** isValidDate(Date \*validDate)

{

//check range of year,month and day

**if** (validDate->yyyy > MAX\_YR || validDate->yyyy < MIN\_YR)

**return** 0;

**if** (validDate->mm < 1 || validDate->mm > 12)

**return** 0;

**if** (validDate->dd < 1 || validDate->dd > 31)

**return** 0;

//Handle feb days in leap year

**if** (validDate->mm == 2)

{

**if** (IsLeapYear(validDate->yyyy))

**return** (validDate->dd <= 29);

**else**

**return** (validDate->dd <= 28);

}

//handle months which has only 30 days

**if** (validDate->mm == 4 || validDate->mm == 6 || validDate->mm == 9 || validDate->mm == 11)

**return** (validDate->dd <= 30);

**return** 1;

}

**menu():**This function displays the library menu and asks the user to select the option. If the user selects 0, then the application will close

**void** menu()

{

**int** choice = 0;

**do**

{

headMessage("MAIN MENU"); printf("\n\n\n\t\t\t1.Add New Patient Record"); printf("\n\t\t\t2.Search Patient Record"); printf("\n\t\t\t3.View Patient Record"); printf("\n\t\t\t4.Delete Patient Record"); printf("\n\t\t\t5.Update Password"); printf("\n\t\t\t0.Exit"); printf("\n\n\n\t\t\tEnter choice => "); scanf("%d",&choice);

**switch**(choice)

{

**case** 1: addPatientInDataBase(); **break**;

**case** 2:

searchPatient();

**break**; **case** 3:

viewPatient();

**break**; **case** 4:

deletePatient();

**break**; **case** 5:

updateCredential();

**break**; **case** 0:

printf("\n\n\n\t\t\t\tThank you!!!\n\n\n\n\n"); exit(1);

**break**; **default**:

printf("\n\n\n\t\t\tINVALID INPUT!!! Try again...");

} //Switch Ended

}

**while**(choice!=0); //Loop Ended

}

**addPatientInDataBase():**This function opens the binary file in append mode and writes the patient information like name, admit date, address ..etc.

**void** addPatientInDataBase()

{

s\_PatientInfo addPatientInfoInDataBase = {0}; FILE \*fp = **NULL**;

**int** status = 0;

fp = fopen(FILE\_NAME,"ab+");

**if**(fp == **NULL**)

{

printf("File is not opened\n"); exit(1);

}

headMessage("ADD NEW PATIENT"); printf("\n\n\t\t\tENTER YOUR DETAILS BELOW:");

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

printf("\n\t\t\tPatient ID = "); fflush(stdin);

scanf("%u",&addPatientInfoInDataBase.patientRecordId);

**do**

{

printf("\n\t\t\tPatient Father Name = "); fflush(stdin);

fgetsRemovedNewLine(addPatientInfoInDataBase.patientFatherName,MAX\_FATHER

\_NAME,stdin);

status = isNameValid(addPatientInfoInDataBase.patientFatherName);

**if** (!status)

{

printf("\n\t\t\tName contain invalid character. Please enter again.");

}

}

**while**(!status); **do**

{

printf("\n\t\t\tPatient Name = "); fflush(stdin);

fgetsRemovedNewLine(addPatientInfoInDataBase.patientName,MAX\_PATIENT\_NAM E,stdin);

status = isNameValid(addPatientInfoInDataBase.patientName);

**if** (!status)

{

printf("\n\t\t\tName contain invalid character. Please enter again.");

}

}

**while**(!status); **do**

{

printf("\n\t\t\tPatient Address = "); fflush(stdin);

fgetsRemovedNewLine(addPatientInfoInDataBase.patientAddr,MAX\_FATHER\_NAME, stdin);

status = isNameValid(addPatientInfoInDataBase.patientAddr);

**if** (!status)

{

printf("\n\t\t\tName contain invalid character. Please enter again.");

}

}

**while**(!status); **do**

{

printf("\n\t\t\tPatient Disease = "); fflush(stdin);

fgetsRemovedNewLine(addPatientInfoInDataBase.patientDisease,MAX\_PATIENT\_DIS EASE,stdin);

status = isNameValid(addPatientInfoInDataBase.patientDisease);

**if** (!status)

{

printf("\n\t\t\tName contain invalid character. Please enter again.");

}

}

**while**(!status);

printf("\n\t\t\tPatient Total Charge = "); fflush(stdin);

scanf("%f",&addPatientInfoInDataBase.patientTotalFees); printf("\n\t\t\tPatient Initial Deposit = ");

fflush(stdin); scanf("%f",&addPatientInfoInDataBase.patientDepositMoney); **do**

{

printf("\n\t\t\tPatient Admit Date:- ");

//get date year,month and day from user printf("\n\t\t\tEnter date in format (dd/mm/yyyy): ");

scanf("%d/%d/%d",&addPatientInfoInDataBase.patientAdmitDate.dd,&addPatientI nfoInDataBase.patientAdmitDate.mm,&addPatientInfoInDataBase.patientAdmitDat e.yyyy);

//check date validity

status = isValidDate(&addPatientInfoInDataBase.patientAdmitDate);

**if** (!status)

{

printf("\n\t\t\tPlease enter a valid date.\n");

}

}

**while**(!status); fwrite(&addPatientInfoInDataBase,sizeof(addPatientInfoInDataBase), 1, fp); fclose(fp);

}

**searchPatient():**This function opens the binary file in reading mode and asks the user to enter the patient id number which wants to search. If the patient info is not available on the list, it shows the message patient not find in records.

**deletePatient():**This function is used to delete the record of a patient from the database on the basis of patient id.

**viewPatient():**This function is used to view the information of all patients whose information is saved in the database.

**updateCredential():**This function opens the file in rb+ mode (reading and writing). It asks the user for the new username and password. After taking the password and username it closes the application. Now users can use the application with a new password and username.

# Source code:-

## #include <stdio.h> #include <stdlib.h> #include <time.h> #include <string.h> #include <ctype.h> #define MAX\_YR 9999

**#define MIN\_YR 1900**

## #define MAX\_SIZE\_USER\_NAME 30

**#define MAX\_SIZE\_PASSWORD 20**

## #define FILE\_NAME "PatientRecordSystem.bin" #define MAX\_FATHER\_NAME 50

**#define MAX\_PATIENT\_NAME 50**

## #define MAX\_PATIENT\_ADDRESS 300

**#define MAX\_PATIENT\_DISEASE 300**

## #define MAX\_DEPARTMENT\_NAME 100

**#define FILE\_HEADER\_SIZE sizeof(sFileHeader) typedef struct**

{

**int** yyyy;

**int** mm;

**int** dd;

} Date;

## typedef struct

{

**char** username[MAX\_SIZE\_USER\_NAME];

**char** password[MAX\_SIZE\_PASSWORD];

} sFileHeader;

//Elements of structure

**typedef struct**// to call in program

{

unsigned **int** patientRecordId; // declare the integer data type

**float** patientTotalFees; **float** patientDepositMoney;

Date patientAdmitDate;// declare the integer data type

**char** patientFatherName[MAX\_FATHER\_NAME];// declare the patients fathers name **char** patientName[MAX\_PATIENT\_NAME];// declare the name in array **char** patientAddr[MAX\_PATIENT\_ADDRESS];// declare the address in array **char** patientDisease[MAX\_PATIENT\_DISEASE];// declare the array for disease } s\_PatientInfo;

**void** fgetsRemovedNewLine(**char** \* restrict buf, **int** n,FILE \* restrict stream)

{

**if** (fgets(buf, n, stream) == **NULL**)

{

printf("Fail to read the input stream");

}

## else

{

buf[strcspn(buf, "\n")] = '\0';

}

}

//Align the message

**void** printMessageCenter(const **char**\* message)

{

**int** len =0;

**int** pos = 0;

//calculate how many space need to print len = (78 - strlen(message))/2; printf("\t\t\t");

**for**(pos =0 ; pos < len ; pos++)

{

//print space printf(" ");

}

//print message printf("%s",message);

}

//Head message

**void** headMessage(const **char** \*message)

{

system("cls");

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

printf("\n\t\t\t############ ############"); printf("\n\t\t\t############ Patient Record Management System ############"); printf("\n\t\t\t############ ############");

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

printf("\n\t\t\t \n"); printMessageCenter(message);

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

//Display message

**void** welcomeMessage()

{

headMessage(["www.ati](http://www.aticleworld.com/)c[leworld.com"](http://www.aticleworld.com/)); printf("\n\n\n\n\n");

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

printf("\n\t\t\t = WELCOME ="); printf("\n\t\t\t = TO ="); printf("\n\t\t\t = Patient Record ="); printf("\n\t\t\t = MANAGEMENT =");

printf("\n\t\t\t = SYSTEM =");

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

printf("\n\t\t\t \*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*-\*\*\n"); printf("\n\n\n\t\t\t Enter any key to continue....."); getchar();

}

//Validate name **int**

isNameValid(const **char** \*name)

{

**int** validName = 1; **int** len = 0;

**int** index = 0; len

= strlen(name);

**for**(index =0; index <len ; ++index)

{

**if**(!(isalpha(name[index])) && (name[index] != '\n') && (name[index] != ' '))

{

validName = 0;

## break;

}

}

**return** validName;

}

// Function to check leap year. //Function returns 1 if leap year

**int** IsLeapYear(**int** year)

{

**return** (((year % 4 == 0) && (year % 100 != 0)) ||

(year % 400 == 0));

}

// returns 1 if given date is valid.

**int** isValidDate(Date \*validDate)

{

//check range of year,month and day **if** (validDate->yyyy > MAX\_YR || validDate->yyyy < MIN\_YR)

**return** 0;

**if** (validDate->mm < 1 || validDate->mm > 12)

**return** 0;

**if** (validDate->dd < 1 || validDate->dd > 31)

**return** 0;

//Handle feb days in leap year

**if** (validDate->mm == 2)

{

**if** (IsLeapYear(validDate->yyyy))

## else

}

**return** (validDate->dd <= 29);

**return** (validDate->dd <= 28);

//handle months which has only 30 days **if** (validDate->mm == 4 || validDate->mm == 6 || validDate->mm == 9 || validDate->mm == 11)

**return** (validDate->dd <= 30);

**return** 1;

}

// Add patient in list

**void** addPatientInDataBase()

{

s\_PatientInfo addPatientInfoInDataBase = {0}; FILE \*fp = **NULL**;

**int** status = 0;

fp = fopen(FILE\_NAME,"ab+");

**if**(fp == **NULL**)

{

printf("File is not opened\n"); exit(1);

}

headMessage("ADD NEW PATIENT"); printf("\n\n\t\t\tENTER YOUR DETAILS BELOW:");

printf("\n\t\t\t

\n"); printf("\n\t\t\tPatient ID = "); fflush(stdin); scanf("%u",&addPatientInfoInDataBase.patientRecordId);

## do

printf("\n\t\t\tPatient Father Name = "); fflush(stdin);

fgetsRemovedNewLine(addPatientInfoInDataBase.patientFatherName,MAX\_FATHER\_ NAME,stdin);

status = isNameValid(addPatientInfoInDataBase.patientFatherName);

**if** (!status)

{

printf("\n\t\t\tName contain invalid character. Please enter again.");

}

}

**while**(!status); **do**

{

printf("\n\t\t\tPatient Name = "); fflush(stdin);

fgetsRemovedNewLine(addPatientInfoInDataBase.patientName,MAX\_PATIENT\_NAME, stdin);

status = isNameValid(addPatientInfoInDataBase.patientName);

**if** (!status)

{

printf("\n\t\t\tName contain invalid character. Please enter again.");

}

}

**while**(!status); **do**

{

printf("\n\t\t\tPatient Address = "); fflush(stdin);

fgetsRemovedNewLine(addPatientInfoInDataBase.patientAddr,MAX\_FATHER\_NAME,st din);

status = isNameValid(addPatientInfoInDataBase.patientAddr);

**if** (!status)

{

printf("\n\t\t\tName contain invalid character. Please enter again.");

}

}

**while**(!status); **do**

{

printf("\n\t\t\tPatient Disease = ");

**if** (!status)

{

printf("\n\t\t\tName contain invalid character. Please enter again.");

}

}

**while**(!status);

printf("\n\t\t\tPatient Total Charge = "); fflush(stdin);

scanf("%f",&addPatientInfoInDataBase.patientTotalFees); printf("\n\t\t\tPatient Initial Deposit = ");

fflush(stdin); scanf("%f",&addPatientInfoInDataBase.patientDepositMoney);

## do

{

printf("\n\t\t\tPatient Admit Date:- ");

//get date year,month and day from user printf("\n\t\t\tEnter date in format (dd/mm/yyyy): ");

scanf("%d/%d/%d",&addPatientInfoInDataBase.patientAdmitDate.dd,&addPatie ntInfoInDataBase.patientAdmitDate.mm,&addPatientInfoInDataBase.patientAd mitDate.yyyy);

//check date validity

status = isValidDate(&addPatientInfoInDataBase.patientAdmitDate);

**if** (!status)

{

printf("\n\t\t\tPlease enter a valid date.\n");

}

}

**while**(!status);

|  |  |
| --- | --- |
| sizeof | (addPatientInfoInDataBase), 1, fp); |
| fwrite(&addPatientInfoInDataBase, |  |
| fclose(fp); |  |

}

// search patient Record

**void** searchPatient()

{

**int** found = 0;

**int** patientId =0;

s\_PatientInfo addPatientInfoInDataBase =

{0}; FILE \*fp = **NULL**; fp =

fopen(FILE\_NAME,"rb");

**if**(fp == **NULL**)

{

printf("\n\t\t\tFile is not opened\n"); exit(1);

}

printf("\n\n\t\t\tEnter patient ID NO to search:");

fflush(stdin); scanf("%u",&patientId);

|  |  |  |
| --- | --- | --- |
| **while** (fread (&addPatientInfoInDataBase, | sizeof | (addPatientInfoInDataBase), 1, fp)) |
| { |  |  |

**if**(addPatientInfoInDataBase.patientRecordId == patientId)

{

found = 1;

## break;

}

}

**if**(found)

{

**float** remainingAmount = (addPatientInfoInDataBase.patientTotalFees >= addPatientInfoInDataBase.patientDepositMoney)?

(addPatientInfoInDataBase.patientTotalFees - addPatientInfoInDataBase.patientDepositMoney):(**float**)(0.0);

printf("\n\t\t\tPatient id = %d\n",addPatientInfoInDataBase.patientRecordId); printf("\n\t\t\tPatient name = %s\n",addPatientInfoInDataBase.patientName); printf("\n\t\t\tPatient Disease = %s\n",addPatientInfoInDataBase.patientDisease); printf("\t\t\tPatient Total Charge =

%f\n",addPatientInfoInDataBase.patientTotalFees); printf("\t\t\tPatient Deposit Amount =

%f\n",addPatientInfoInDataBase.patientDepositMoney); printf("\t\t\tPatient Remaining Amount = %f\n",remainingAmount);

printf("\t\t\tPatient Father Name =

%s\n",addPatientInfoInDataBase.patientFatherName);

printf("\t\t\tPatient Address = %s\n",addPatientInfoInDataBase.patientAddr); printf("\t\t\tPatient Admited Date(day/month/year) = (%d/%d/%d)\n",addPatientInfoInDataBase.patientAdmitDate.dd, addPatientInfoInDataBase.patientAdmitDate.mm, addPatientInfoInDataBase.patientAdmitDate.yyyy);

}

## else

{

printf("\n\t\t\tNo Record");

}

fclose(fp);

printf("\n\n\n\t\t\tPress any key to go to main menu. ");

fflush(stdin); getchar();

}

// view Patient function

**void** viewPatient()

{

**int** found = 0;

unsigned **int** countPatient = 1;

headMessage("VIEW PATIENT DETAILS");

fp = fopen(FILE\_NAME,"rb");

**if**(fp == **NULL**)

{

printf("File is not opened\n"); exit(1);

}

**if** (fseek(fp,FILE\_HEADER\_SIZE,SEEK\_SET) != 0)

{

fclose(fp);

printf("Facing issue while reading file\n"); exit(1);

}

//Print patient count

printf("\n\t\t\tPatient Count = %d\n\n",countPatient);

|  |  |  |
| --- | --- | --- |
| **while** (fread (&addPatientInfoInDataBase, | sizeof | (addPatientInfoInDataBase), 1, fp)) |
| { |  |  |

**float** remainingAmount = (addPatientInfoInDataBase.patientTotalFees >= addPatientInfoInDataBase.patientDepositMoney)?

(addPatientInfoInDataBase.patientTotalFees - addPatientInfoInDataBase.patientDepositMoney):(**float**)(0.0); printf("\t\t\tPatient id = %u\n",addPatientInfoInDataBase.patientRecordId); printf("\t\t\tPatient Name = %s\n",addPatientInfoInDataBase.patientName);

printf("\n\t\t\tPatient Disease = %s\n",addPatientInfoInDataBase.patientDisease); printf("\t\t\tPatient Total Charge = %f\n",addPatientInfoInDataBase.patientTotalFees); printf("\t\t\tPatient Deposit Amount -%f\n",addPatientInfoInDataBase.patientDepositMoney); printf("\t\t\tPatient Remaining Amount = %f\n",remainingAmount);

printf("\t\t\tPatient Father Name = %s\n",addPatientInfoInDataBase.patientFatherName); printf("\t\t\tPatient Address = %s\n",addPatientInfoInDataBase.patientAddr);

printf("\t\t\tPatient Admited Date(day/month/year) = (%d/%d/%d)\n\n",addPatientInfoInDataBase.patientAdmitDate.dd, addPatientInfoInDataBase.patientAdmitDate.mm,

addPatientInfoInDataBase.patientAdmitDate.yyyy); found = 1;

++countPatient;

}

fclose(fp);

printf ("\n\t\t\tNo Record");

}

printf("\n\n\t\t\tPress any key to go to main menu. ");

fflush(stdin); getchar();

}

// Delete patient Record entry

**void** deletePatient()

{

**int** found = 0; **int** patientDelete = 0; sFileHeader fileHeaderInfo = {0}; s\_PatientInfo addPatientInfoInDataBase = {0};

FILE \*fp = **NULL**;

FILE \*tmpFp =

## NULL;

headMessage("Delete patient Record Details"); fp = fopen(FILE\_NAME,"rb");

**if**(fp == **NULL**)

{

printf("File is not opened\n"); exit(1);

}

tmpFp = fopen("tmp.bin","wb");

**if**(tmpFp == **NULL**)

{

fclose(fp);

printf("File is not opened\n"); exit(1);

}

fread (&fileHeaderInfo,FILE\_HEADER\_SIZE, 1, fp); fwrite(&fileHeaderInfo,FILE\_HEADER\_SIZE, 1, tmpFp);

printf("\n\t\t\tEnter patient ID NO. for delete:"); scanf("%d",&patientDelete);

|  |  |  |
| --- | --- | --- |
| **while** (fread (&addPatientInfoInDataBase, | sizeof | (addPatientInfoInDataBase), 1, fp)) |
| { |  |  |

**if**(addPatientInfoInDataBase.patientRecordId != patientDelete)

{

|  |  |
| --- | --- |
| sizeof | (addPatientInfoInDataBase), 1, tmpFp); |
| fwrite(&addPatientInfoInDataBase, |  |
| } |  |

## else

{

found = 1;

}

}

(found)? printf("\n\t\t\tRecord deleted successfully. "):printf("\n\t\t\tRecord not found");

fclose(fp); fclose(tmpFp); remove(FILE\_NAME);

rename("tmp.bin",FILE\_NAME);

}

**void** updateCredential(**void**)

{

sFileHeader fileHeaderInfo = {0}; FILE \*fp = **NULL**;

**char** userName[MAX\_SIZE\_USER\_NAME]

= {0}; **char**

password[MAX\_SIZE\_PASSWORD] = {0}; headMessage("Update Credential"); fp = fopen(FILE\_NAME,"rb+");

**if**(fp == **NULL**)

{

printf("File is not opened\n"); exit(1);

}

fread (&fileHeaderInfo,FILE\_HEADER\_SIZE, 1, fp); **if** (fseek(fp,0,SEEK\_SET) != 0)

{

fclose(fp);

printf("\n\t\t\tFacing issue while updating password\n"); exit(1);

}

printf("\n\n\t\t\tNew Username:"); fflush(stdin);

fgetsRemovedNewLine(userName,MAX\_SIZE\_USER\_NAME,stdin); printf("\n\n\t\t\tNew Password:");

fflush(stdin);

fgetsRemovedNewLine(password,MAX\_SIZE\_PASSWORD,stdi n); strncpy(fileHeaderInfo.username,userName,sizeof(userName)); strncpy(fileHeaderInfo.password,password,sizeof(password)); fwrite(&fileHeaderInfo,FILE\_HEADER\_SIZE, 1, fp);

fclose(fp);

printf("\n\t\t\tYour Password has been changed successfully"); printf("\n\t\t\tRe-Run Application and Login with new Credential:"); fflush(stdin); getchar(); exit(1);

}

//Display menu

**void** menu()

{

**int** choice = 0;

## do

{

headMessage("MAIN MENU"); printf("\n\n\n\t\t\t1.Add New Patient Record"); printf("\n\t\t\t2.Search Patient Record"); printf("\n\t\t\t3.View Patient Record"); printf("\n\t\t\t4.Delete Patient Record"); printf("\n\t\t\t5.Update Password");

printf("\n\t\t\t0.Exit"); printf("\n\n\n\t\t\tEnter choice => "); scanf("%d",&choice);

**switch**(choice)

{

**case** 1:

addPatientInDataBase(); **break**; **case** 2: searchPatient(); **break**; **case** 3: viewPatient();

**break**; **case** 4: deletePatient(); **break**; **case** 5:

updateCredential();

## break;

**case** 0:

printf("\n\n\n\t\t\t\tThank you!!!\n\n\n\n\n");

exit(1); **break**; **default**: printf("\n\n\n\t\t\tINVALID INPUT!!! Try again...");

} //Switch Ended

}

**while**(choice!=0); //Loop Ended

}

//login password

**void** login()

{

**char** userName[MAX\_SIZE\_USER\_NAME] = {0};

**char** password[MAX\_SIZE\_PASSWORD]

= {0}; **int** L=0; sFileHeader fileHeaderInfo

= {0};

sFileHeader fileHeaderInfo = {0};

FILE \*fp = **NULL**;

headMessage("Login"); fp = fopen(FILE\_NAME,"rb");

**if**(fp == **NULL**)

{

printf("Data base is not opened\n"); exit(1);

}

fread (&fileHeaderInfo,FILE\_HEADER\_SIZE, 1, fp); fclose(fp);

## do

{

printf("\n\n\n\t\t\t\tUsername:"); fgetsRemovedNewLine(userName,MAX\_SIZE\_USER\_NAME,stdin);

printf("\n\t\t\t\tPassword:"); fgetsRemovedNewLine(password,MAX\_SIZE\_PASSWORD,stdin);

**if**((!strcmp(userName,fileHeaderInfo.username)) && (!strcmp(password,fileHeaderInfo.password)))

{

menu();

}

## else

{

printf("\t\t\t\tLogin Failed Enter Again Username & Password\n\n"); L++;

}

}

**while**(L<=3); **if**(L>3)

{

headMessage("Login Failed"); printf("\t\t\t\tSorry,Unknown User.");

getchar(); system("cls");

}

}

//Check file exist or not

**int** isFileExists(const **char** \*path)

{

// Try to open file FILE

\*fp = fopen(path, "rb"); **int**

status = 0;

// If file does not exists

// If file does not exists

**if** (fp != **NULL**)

{

status = 1;

// File exists hence close file fclose(fp);

}

**return** status;

}

**void** init()

{

FILE \*fp = **NULL**;

**int** status = 0;

const **char** defaultUsername[]

="aticleworld"; const **char** defaultPassword[]

="aticleworld"; sFileHeader fileHeaderInfo =

{0}; status = isFileExists(FILE\_NAME);

**if**(!status)

{

//create the binary file

fp = fopen(FILE\_NAME,"wb");

**if**(fp != **NULL**)

{

//Copy default password strncpy(fileHeaderInfo.password,defaultPassword,sizeof(defaultPassword));

strncpy(fileHeaderInfo.username,defaultUsername,sizeof(defaultUsername)); fwrite(&fileHeaderInfo,FILE\_HEADER\_SIZE, 1, fp); fclose(fp);

}

}

}

**int** main()

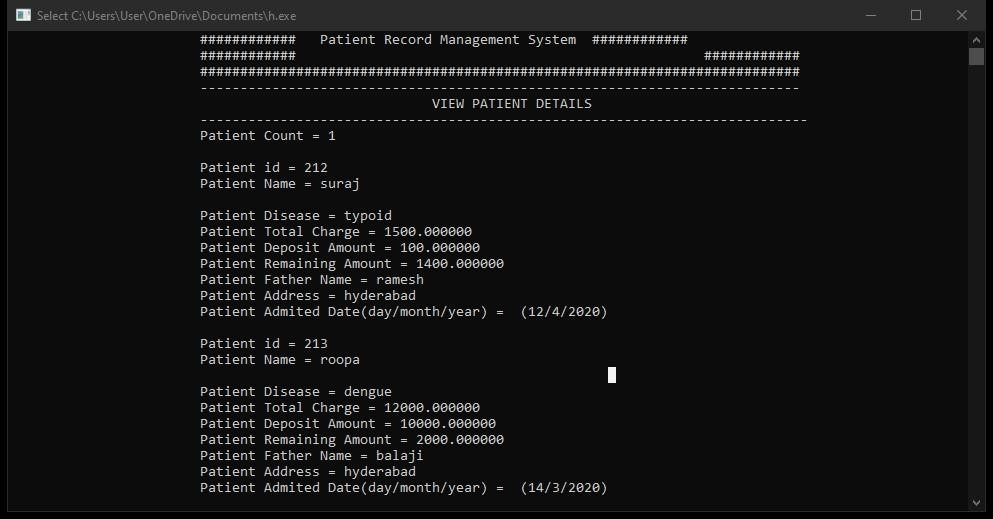
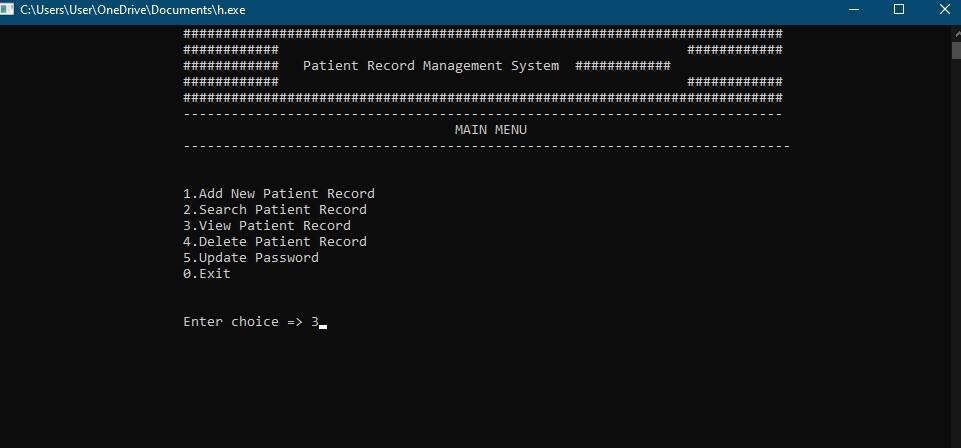
{

init();

welcomeMessage(); login();

**return** 0;

}

**Sample output:-**