Fidha Fathima

Programming in C 7/17/24

CLINIC APPOINTMENT

SYSTEM

OVERVIEW

Page2

In the era of technology, where everything needs to be done efficiently and effectively the existence of Clinic Management System becomes necessary. The use of CMS can enhance the service and also the work flow of most of the activities that happens in hospitals where it helps in reducing the workload of medical staff, the number of man power needed and it also make hospitals management become more manageable and easier to control.

The appointment system is a digital platform designed to streamline and enhance the process of scheduling and managing appointments between patients and staffs. This system offers an efficient and user-friendly interface for patients to view available time slots, select preferred doctors and book appointment at their convenience. Hospitals or clinics cam manage the doctor’s schedules, track patient appointments and efficiently allocate their time. The system aims to reduce waiting times, enhance patient-doctor communication and improve overall healthcare service delivery.

## PROBLEM STATEMENT

In the current healthcare landscape, the traditional methods of scheduling and managing doctor appointments are often marred by inefficiencies, resulting in suboptimal patient experiences and operational challenges for healthcare providers. The prevalent manual appointments system contributes to long waiting times, scheduling errors and a lack of real-time communication between patients and healthcare professionals.

# OBJECTIVES

The system gives remedies for the problem that are currently being faced by the people Some of the objectives of the system are as follows:

* Enhance patient satisfaction by proving a user-friendly booking platform.
* Optimise healthcare provider workflow for efficient time management.
* Ensure accessibility and convenience through online scheduling options.
* Reduce scheduling errors and conflicts to enhance overall system accuracy.
* Increase transparency in appointment availability for patients.

SYSTEM REQUIREMENTS

Page3

### Hardware Requirements

#### Minimum Requirements

1. Processor: Any modern CPU (Inter i3 or equivalent)
2. RAM: 2 GB
3. Storage: 500 MB
4. Operating System: Any modern OS (Windows, Linux, macOS).
5. Display: Standard resolution monitor (1024x768 or higher).
6. Compiler: GCC, Clang, or any other C compiler

#### Additional Requirements

1. Text Editor/IDE: An IDE like visual studio code, code block, or any text editor of choice.
2. Debugging Tools: GDB (GNU Debugger) for debugging your C programs
   * Software Requirements

#### Operating System:

* + Any modern OS (Windows 10 or later, macOS 10.15 or later, or a recent Linux distribution).

#### C Compiler:

* + GCC: Commonly used on Linux and available for Windows via MinGW.
  + Clang: Another popular compiler, especially on macOS.
  + Microsoft Visual C++ Compiler: Available as a part of Visual studio on Windows

### Integrated Development Environment (IDE) or text editor:

* + IDE Options:
    - Visual studio Code
    - Code::Blocks
    - Eclipse
    - Visual studio
  + Text Editor Options:
    - Sublime Text
    - Notepad++
    - Vim/Emacs

## Debugger

* + GDB (GNU Debugger):

Page4

A portable debugger that works for many programming languages including C

Design and Development

Page5

## Program Logic

Program Structure:

1. Structures:
   * Doctor: Contains information about doctors such as name, specializations and available slots.
   * Patients: Contains information about patients such as name and age.
   * Appointment: Links a patient with a doctor and includes the appointment time.
2. Function:
   * printDoctor: prints details of doctor
   * printPatient: prints the details of the patient.
   * printAppointment: Prints the details of an appointments.
   * sheduleAppointment: Schedules an appointment by associating a patient with a doctor at specific time.
   * print Appointments: Prints all scheduled appointments.
   * sendReminders: Sends reminders for upcoming appointments.
   * main: the main function which provides a menu for fast users to interact with the system.

### Execution Flow

* Starting the program.

The user executes the program, and the menu appears.

* Viewing Doctor Data.

The user chooses the option to view doctor data.

The program prints the information for all doctors.

* Schedule an appointment.

The user chooses the option to schedule an appointment.

The program allows the user to enter patient information, doctor's name, and

appointment time.

The computer checks the doctor's availability and, if possible, arranges an

appointment.

* Viewing all appointments.

The user gets the option to view all appointments.

The program prints information about all organized appointments.

Page6

* Sending reminders.

The user chooses the option to send reminders.

The application produces reminder messages for each scheduled appointment.

* Exiting the Program

The user chooses the option to exit the software.

The program terminates.

Pseudocode

1. START
2. Initialize doctors array
3. Initialize appointments array
4. Initialize numAppointment to 0
5. Initialize numDoctors to the number of doctors
6. Loop until the user chooses to exit
   * Display menu options
   * Get user choice
   * Switch user choice
     + Case 1: View Doctor Data
       - Print “Doctor Data”
       - For each doctor in doctors array

Call printDoctor function with doctor data

* + - * End for
      * Break
      * Prompt user for doctor name and appointment time
      * For each doctor name
        + If doctor name matches user input

If doctor has available slots

Create newAppointment with patient and doctor data

Add newAppointment to appointments array

Increment numAppointments

Decrement doctor’s available slots

Print “appointment scheduled successfully”

Return to menu.

Else

Print “Sorry! No available slots for this doctor!”

Return to me

* + - * + End if
      * End for
    - Case 3: view all Appointments
      * If no appointments available
        + Print “No appointments available”
      * Else
        + For each appointment in appointments array

Call printAppointment function with appointment data

* + - * + End For
      * End if
      * Break
    - Case 4: Send appointment reminders:
      * If no appointments scheduled
        + Print “No appointments scheduled”
      * Else
* Print “Sending reminders for upcoming appointments:”
* For each appointment in appointment array

Print “Reminder for patient with doctor at the time”

* + End for

Page7

* + - * End if
      * Break
    - Case 5: Exit
      * Print “Exiting….”
      * Break Loop
    - Default
      * Print “Invalid choice. Please enter a valid option.”
* End switch

1. End Loop
2. End

# TESTING AND RESULTS

Page8

CODE:

#include<ctype.h> #include<stdio.h> #include<string.h> struct Doctor

//Doctor

{

char name[ 50];

char specialization[ 50];

int avalableslots;

};

//Patient struct Patient

{

char name[ 50];

int age;

};

//appoinment

struct Appointment

{

struct Patient patient; struct Doctor doctor; char time[ 50];

};

//doctor data

void printDoctor( struct Doctor doc){

printf( "Doctor data\n " );

printf( "Doctor Name:%s\n " , doc. name);

printf( "Specialization: %s\n " , doc. specialization); printf( "Available slots:%d\n " , doc. avalableslots); printf( " \n " );

}

//patient data

void printPatient( struct Patient patient){ printf( "Patient Name:%s\n " , patient. name); printf( "Age:%d\n " , patient. age);

printf( " \n " );

}

//appoinment data

void printAppointment( struct Appointment app){ printf( "Appointmeng Details:\n " ); printPatient( app. patient);

printf( "Doctor;%s\n " , app. doctor. name);

Page9

printf( "specialization:%s\n " , app. doctor. specialization);

printf( "Appointment Time:%s\n " , app. time);

printf( " \n " );

}

###### //shedule

void scheduleAppointment( struct Doctor doctors[] , int numDoctors, struct

Appointment appointments[] , int \* numAppointments){

char doctorName[ 50];

char patientName[ 50];

int patientAge;

char time[ 20];

printf( "Enter patient name: "); scanf( " %s" , patientName); printf( "Enter patient age: "); scanf( " %d" , &patientAge);

printf( "Available Doctors:\n " );

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

{

printDoctor( doctors[ i ]);

}

printf( "Enter doctor name: ");

scanf( " %s" , doctorName);

printf( "Enter the appointment time(HH:MM): ");

scanf( " %s" , time);

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

{

if ( strcasecmp( doctors[ i ]. name, doctorName) ==0)

{

if ( doctors[ i ]. avalableslots>0)

{

struct Appointment newAppointment;

strcpy( newAppointment. patient. name, patientName); newAppointment. patient. age=patientAge; newAppointment. doctor=doctors[ i ];

strcpy( newAppointment. time, time);

appointments[ \* numAppointments] =newAppointment;

(\* numAppointments) ++;

doctors[ i ]. avalableslots-- ;

printf( "Appointment scheduled successfully!\n " );

##### return;

}

###### else

{

}

printf( "Sorry! No available slots for this doctor.\n " );

##### return;

Page10

}

}

printf( "Doctor not found.\n " );

}

void printAppointments( struct Appointment appointments[] , int

numAppointments){

if ( numAppointments==0)

{

printf( "No appointments available.\n " );

##### return;

}

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

{

printAppointment( appointments[ i ]);

}

}

void sendReminders( struct Appointment appointments[] , int numAppointments){

if ( numAppointments==0)

{

printf( "No appointments sheduled.\n " );

##### return;

}

printf( "Sending reminders for upcoming appointments:\n " );

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

{

printf( "Reminder:Appointment for %s with Dr.%s at

%s\n " , appointments[ i ]. patient. name, appointments[ i ]. doctor. name, appointments[ i ]

. time);

}

}

//sample

int main(){

struct Doctor doctors[ 3] ={

{"Dr.Ebenzer", "Cardiology", 3},

{"Dr.Kim", "Dermatology", 2},

{"Dr Milan", "Neuorology", 4}

};

struct Appointment appointments[ 50];

Page11

int numAppointments=0;

int numDoctors=3;

int choice;

do

{

printf( " \n\n\n\*\*\*\*\*\*\*\*\*Clinic Management System

Menu:\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n" );

printf( "1.View DoctOR data\n " ); printf( "2.Schedule an appointment\n " ); printf( "3.View All appointments\n " );

printf( "4.Send appointment Reminders\n " );

printf( "5.Exit\n\n" );

printf( "Enter your choice: ");

scanf( " %d" , &choice);

switch ( choice)

{

case 1:

printf( " \n Doctor Data:\n\n" );

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

{

printDoctor( doctors[ i ]);

printf( " \n " );

}

break;

case 2:

scheduleAppointment( doctors, numDoctors, appointments, &numAppointments);

break;

case 3:

printAppointments( appointments, numAppointments);

break;

case 4:

sendReminders( appointments, numAppointments);

break;

case 5:

printf( "Exiting. \n " );

break;

default:

printf( "Invalid Choice.Please enter valid option.\n " );

break;

}

} while ( choice!= 5);

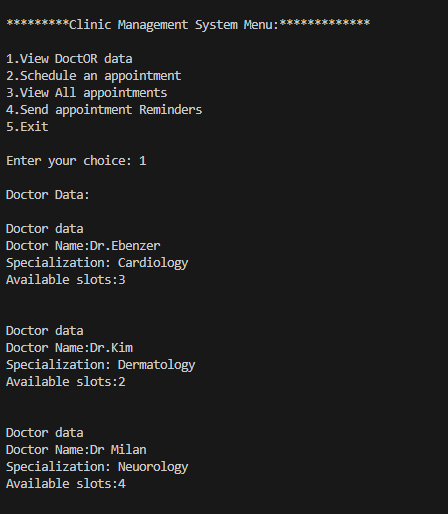
return 0;

}

Outputs:

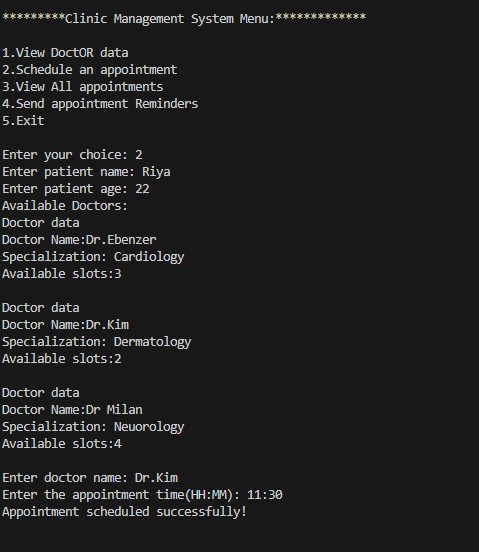
* For viewing Doctor data:

Page12

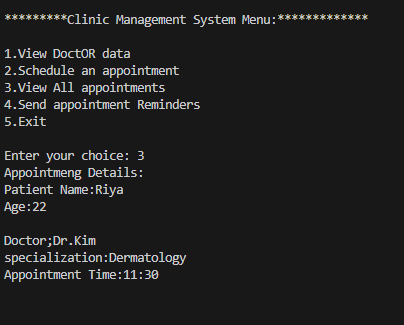


* For Scheduling an appointment

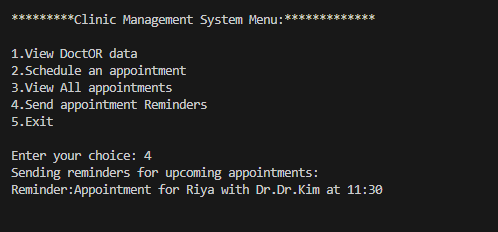
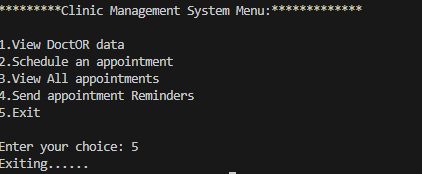
Page13



* To view all appointments:



Page14

* Sending Reminders:
* Exit:

## Conclusion

Page15

In conclusion, the Clinic appointment system stand as a transformative solution to the challenges inherent in traditional healthcare appointment management. Above provided is just the basic program to develop this system. After a systematic upgrading while acknowledging certain limitations, such as technological barriers and privacy concerns this system overall impact on improving patient experiences and operational efficiency underscores its significance in ever evolving landscape of healthcare administration. The clinical Appointment system represents a crucial step towards a more patient-centric, accessible and efficient healthcare ecosystem.