HOSPITAL MANAGEMENT SYSTEM

**Code:-**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_PATIENTS 100

#define FILENAME "patients.txt"

// Patient structure

typedef struct {

int id;

char name[100];

int age;

char gender[10];

char disease[100];

} Patient;

// Appointment structure (Linked List)

typedef struct Appointment {

int patientId;

char date[20];

char time[10];

struct Appointment\* next;

} Appointment;

// Global Variables

Patient patients[MAX\_PATIENTS];

int patientCount = 0;

Appointment\* appointmentHead = NULL;

// Function declarations

void addPatient();

void viewPatients();

void editPatient();

void savePatients();

void loadPatients();

void sortPatients();

void addAppointment();

void viewAppointments();

void deleteAppointment();

void menu();

// Main function

int main() {

loadPatients();

menu();

return 0;

}

// Menu function

void menu() {

int choice;

while (1) {

printf("\n=== Hospital Management System ===\n");

printf("1. Add Patient\n");

printf("2. View Patients\n");

printf("3. Edit Patient\n");

printf("4. Add Appointment\n");

printf("5. View Appointments\n");

printf("6. Delete Appointment\n");

printf("7. Sort Patients by Name\n");

printf("8. Save and Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

getchar(); // Clear buffer

switch (choice) {

case 1:

addPatient();

break;

case 2:

viewPatients();

break;

case 3:

editPatient();

break;

case 4:

addAppointment();

break;

case 5:

viewAppointments();

break;

case 6:

deleteAppointment();

break;

case 7:

sortPatients();

break;

case 8:

savePatients();

printf("Data saved. Exiting...\n");

exit(0);

default:

printf("Invalid choice. Try again.\n");

}

}

}

// Add new patient

void addPatient() {

if (patientCount >= MAX\_PATIENTS) {

printf("Patient list full!\n");

return;

}

printf("\n--- Add Patient ---\n");

patients[patientCount].id = patientCount + 1;

printf("Enter Name: ");

fgets(patients[patientCount].name, sizeof(patients[patientCount].name), stdin);

patients[patientCount].name[strcspn(patients[patientCount].name, "\n")] = '\0';

printf("Enter Age: ");

scanf("%d", &patients[patientCount].age);

getchar();

printf("Enter Gender: ");

scanf("%s", patients[patientCount].gender);

getchar();

printf("Enter Disease: ");

fgets(patients[patientCount].disease, sizeof(patients[patientCount].disease), stdin);

patients[patientCount].disease[strcspn(patients[patientCount].disease, "\n")] = '\0';

printf("Patient added with ID: %d\n", patients[patientCount].id);

patientCount++;

}

// View all patients

void viewPatients() {

printf("\n--- Patient List ---\n");

for (int i = 0; i < patientCount; i++) {

printf("\nID: %d\nName: %s\nAge: %d\nGender: %s\nDisease: %s\n",

patients[i].id, patients[i].name, patients[i].age,

patients[i].gender, patients[i].disease);

}

}

// Edit a patient

void editPatient() {

int id, found = 0;

printf("\nEnter Patient ID to Edit: ");

scanf("%d", &id);

getchar();

for (int i = 0; i < patientCount; i++) {

if (patients[i].id == id) {

found = 1;

printf("Editing Patient %s:\n", patients[i].name);

printf("Enter New Name: ");

fgets(patients[i].name, sizeof(patients[i].name), stdin);

patients[i].name[strcspn(patients[i].name, "\n")] = '\0';

printf("Enter New Age: ");

scanf("%d", &patients[i].age);

getchar();

printf("Enter New Gender: ");

scanf("%s", patients[i].gender);

getchar();

printf("Enter New Disease: ");

fgets(patients[i].disease, sizeof(patients[i].disease), stdin);

patients[i].disease[strcspn(patients[i].disease, "\n")] = '\0';

printf("Patient updated!\n");

break;

}

}

if (!found) {

printf("Patient ID not found.\n");

}

}

// Save patients to file

void savePatients() {

FILE\* fp = fopen(FILENAME, "w");

if (fp == NULL) {

printf("Error saving file.\n");

return;

}

for (int i = 0; i < patientCount; i++) {

fprintf(fp, "%d;%s;%d;%s;%s\n", patients[i].id, patients[i].name,

patients[i].age, patients[i].gender, patients[i].disease);

}

fclose(fp);

}

// Load patients from file

void loadPatients() {

FILE\* fp = fopen(FILENAME, "r");

if (fp == NULL) {

// No file yet

return;

}

while (fscanf(fp, "%d;%[^;];%d;%[^;];%[^\n]\n",

&patients[patientCount].id,

patients[patientCount].name,

&patients[patientCount].age,

patients[patientCount].gender,

patients[patientCount].disease) == 5) {

patientCount++;

}

fclose(fp);

}

// Sort patients by name

void sortPatients() {

Patient temp;

for (int i = 0; i < patientCount - 1; i++) {

for (int j = i + 1; j < patientCount; j++) {

if (strcmp(patients[i].name, patients[j].name) > 0) {

temp = patients[i];

patients[i] = patients[j];

patients[j] = temp;

}

}

}

printf("Patients sorted by name!\n");

}

// Add new appointment

void addAppointment() {

int id, found = 0;

printf("\nEnter Patient ID for Appointment: ");

scanf("%d", &id);

getchar();

for (int i = 0; i < patientCount; i++) {

if (patients[i].id == id) {

found = 1;

break;

}

}

if (!found) {

printf("Patient not found!\n");

return;

}

Appointment\* newApp = (Appointment\*)malloc(sizeof(Appointment));

newApp->patientId = id;

printf("Enter Date (DD/MM/YYYY): ");

fgets(newApp->date, sizeof(newApp->date), stdin);

newApp->date[strcspn(newApp->date, "\n")] = '\0';

printf("Enter Time (HH:MM): ");

fgets(newApp->time, sizeof(newApp->time), stdin);

newApp->time[strcspn(newApp->time, "\n")] = '\0';

newApp->next = NULL;

// Insert at end

if (appointmentHead == NULL) {

appointmentHead = newApp;

} else {

Appointment\* temp = appointmentHead;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = newApp;

}

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

}

// View all appointments

void viewAppointments() {

Appointment\* temp = appointmentHead;

if (temp == NULL) {

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

return;

}

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

while (temp != NULL) {

printf("Patient ID: %d | Date: %s | Time: %s\n",

temp->patientId, temp->date, temp->time);

temp = temp->next;

}

}

// Delete an appointment

void deleteAppointment() {

int id;

printf("\nEnter Patient ID to Delete Appointment: ");

scanf("%d", &id);

Appointment \*temp = appointmentHead, \*prev = NULL;

while (temp != NULL) {

if (temp->patientId == id) {

if (prev == NULL) { // Deleting head

appointmentHead = temp->next;

} else {

prev->next = temp->next;

}

free(temp);

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

return;

}

prev = temp;

temp = temp->next;

}

printf("Appointment not found for Patient ID %d.\n", id);

}