A Project report on

HOSPITAL MANAGEMENT SYSTEM

This project is part of DS II Lab evaluation

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING

Submitted by

V. YETHIN KUMAR REDDY – AP22110010935

SK. ERSHAD ALI – AP22110010941

G. ROHITH SAI – AP22110010954

K. PANEENDRA BHARGAV – AP22110010971

SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

23rd Nov 2023

TABLE OF CONTENTS:

Abstract:

- The Hospital Management System (HMS) is a comprehensive software solution designed to streamline and enhance the efficiency of managing patient and doctor records within a healthcare facility.
- In the dynamic landscape of healthcare, effective information management is critical for providing quality patient care, optimizing resource utilization, and ensuring seamless communication between healthcare professionals.
- The Hospital Management System integrates various modules to cover a wide range of functionalities, including patient medical records management, and doctor management.
- The system aims to automate and simplify the administrative and clinical processes, leading to

- improved patient outcomes and better operational performance for healthcare institutions.
- This project leverages modern technology to create a user-friendly interface for the hospital staff.
- By centralizing patient and doctor information, the Hospital Management System contributes to efficient decision-making, reduces paperwork, and enhances the overall quality of healthcare services.

Introduction:

- In the rapidly evolving landscape of healthcare, the need for efficient management of patient and doctor records has become paramount. Traditional manual methods of record-keeping are not only time-consuming but also prone to errors, hindering the seamless flow of information crucial for providing timely and accurate medical care. Recognizing this challenge, the Hospital Management System is introduced as a comprehensive solution to address the complexities of managing healthcare information.

- The primary objective of this system is to digitize and automate various aspects of hospital management, ranging from patient registration id to doctor appointments and medical record keeping. By adopting a centralized and integrated approach, the Hospital Management System seeks to streamline administrative processes, reduce paperwork, and enhance the overall quality of healthcare services.

- The system is designed to be user-friendly, ensuring that healthcare professionals, administrative staff, and patients can interact with it effortlessly. It incorporates features such as secure data storage, easy retrieval of patient information, appointment scheduling, and prescription management.

Hospital Management System Algorithm

1.Initialization:

- **Define two structures, Doctor and Patient, to store information about doctors and patients, respectively.**
- **Create global arrays to store doctor and patient data, and initialize counters for the number of doctors and patients.**

2.Hash Function:

- **Define a simple hash function, calculateHash(id), which calculates the hash index for a given ID using the modulo operator. This maps each ID to an index in the arrays for efficient data retrieval.**

3.Add Doctor Function:

- **Create an addDoctor() function:**
- **Prompt the user to enter the doctor's name and specialization.**
- **Assign a unique ID (incrementing the doctorCount), set the doctor as present, and calculate the hash index using the calculateHash function.**
- **Store the doctor's information in the array at the computed index.**
- **Increase the doctorCount to track the number of doctors.**

4.Add Patient Function:

- **Implement an addPatient() function:**
- **Prompt the user to enter the patient's name, age, gender, doctor ID, and diagnosis.**
- **Assign a unique ID (incrementing the patientCount), set the patient as present, and calculate the hash index using the calculateHash function.**
- **Store the patient's information in the array at the computed index.**
- **Increase the patientCount to track the number of patients.**

5.Display Doctors and Patients Functions:

- **Implement two functions, displayDoctors() and displayPatients():**
- **Iterate through the respective arrays and print the information for each doctor and patient that is marked as present.**

6.Check Patient Presence Function:

- **Create a checkPatientPresence(patientId) function:**
- **Accept a patient's unique ID as input.**
- **Calculate the hash index for the given ID.**
- **Check if a patient with the given ID exists and is marked as present.**
- **Display the patient's presence status accordingly.**

7.Check Doctor Presence Function:

- **Implement a checkDoctorPresence(doctorId) function:**
- **Accept a doctor's unique ID as input.**
- **Calculate the hash index for the given ID.**
- **Check if a doctor with the given ID exists and is marked as present.**
- **Display the doctor's presence status accordingly.**

8.Main Program Loop:

- **In the main() function, set up a menu-driven interface with options to:**
- **Add doctors.**
- **Add patients.**
- **Display the list of doctors.**
- **Display the list of patients.**
- **Check patient presence by ID.**
- **Check doctor presence by ID.**
- **Exit the program.**

9.Program Execution:

- **Run the program in a loop until the user decides to exit.**
- **Based on the user's choice, call the respective functions to perform the desired actions.**

*** This algorithm provides a basic outline of the hospital management system, where data is efficiently managed using a hash-based approach for doctor and patient storage and retrieval. The program offers functionalities for adding, displaying, and checking the presence of doctors and patients in the hospital.**

HOSPITAL MANAGEMENT CODE

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Define structures for patient and doctor information

struct Doctor {

    int id;

    char name[50];

    char specialization[50];

    int isPresent; // New field to track doctor's presence

};

struct Patient {

    int id;

    char name[50];

    int age;

    char gender;

    char phoneNumber[10]; // New field to store patient phone number

    int doctorId;

    char diagnosis[100]; // New field to store patient diagnosis

    int isPresent; // New field to track patient's presence

};
```

```
// Define global arrays to store patient and doctor data
struct Doctor doctors[100];
struct Patient patients[100];
int doctorCount = 0;
int patientCount = 0;

// Function to calculate a hash value for a given ID
int calculateHash(int id) {
    return id % 100;
}
// Function to add a doctor
void addDoctor() {
    struct Doctor doctor;
    printf("Enter doctor name: ");
    scanf("%s", doctor.name);
    printf("Enter doctor's specialization: ");
    scanf("%s", doctor.specialization);
    doctor.id = doctorCount + 1;
 doctor.isPresent = 1;
```

```
// Assume the doctor is present when added
    int hash = calculateHash(doctor.id);
    doctors[hash] = doctor;
    doctorCount++;
    printf("Doctor added successfully.\n");
}
// Function to add a patient
void addPatient() {
    struct Patient patient;
    printf("Enter patient name: ");
    scanf("%s", patient.name);
    printf("Enter patient age: ");
    scanf("%d", &patient.age);
    printf("Enter patient gender (M/F): ");
    scanf(" %c", &patient.gender);
    printf("Enter patient phone number: ");
    scanf("%s", patient.phoneNumber);
    printf("Enter doctor's ID for the patient: ");
    scanf("%d", &patient.doctorId);
    printf("Enter patient diagnosis: ");
    scanf("%s", patient.diagnosis);
```

```
    patient.id = patientCount + 1;
    patient.isPresent = 1;

 // Assume the patient is present when added
    int hash = calculateHash(patient.id);
    patients[hash] = patient;
    patientCount++;
    printf("Patient added successfully.\n");
}

// Function to display the list of doctors
void displayDoctors() {
    printf("List of Doctors:\n");
    for (int i = 1; i <= doctorCount; i++) {
        if (doctors[i].isPresent) {
            printf("ID: %d, Name: %s, Specialization: %s, Present: %s\n",
     doctors[i].id, doctors[i].name,
      doctors[i].specialization, doctors[i].isPresent ? "Yes" : "No");
        }
       }
}
```

```
// Function to display the list of patients
void displayPatients() {
    printf("List of Patients:\n");
    for (int i = 1; i <= patientCount; i++) {
        if (patients[i].isPresent) {
            printf("ID: %d \nName: %s \nAge: %d \nGender: %c \nPhone Number: %s \nDoctor ID: %d \nDiagnosis: %s \nPresent: %s\n",
                patients[i].id, patients[i].name, patients[i].age, patients[i].gender, patients[i].phoneNumber,
                patients[i].doctorId, patients[i].diagnosis, patients[i].isPresent ? "Yes" : "No");

        }
    }
}

// Function to check if a patient is present in the hospital
void checkPatientPresence(int patientId) {
    int hash = calculateHash(patientId);
    if (patients[hash].id == patientId && patients[hash].isPresent) {
```

```
        printf("Patient ID: %d, Name: %s, Present: Yes\n", patients[hash].id,
    patients[hash].name);
    } else {
        printf("Patient with ID %d not found or not present.\n", patientId);
    }
}

// Function to check if a doctor is present in the hospital
void checkDoctorPresence(int doctorId) {
    int hash = calculateHash(doctorId);
    if (doctors[hash].id == doctorId && doctors[hash].isPresent) {
        printf("Doctor ID: %d, Name: %s, Present: Yes\n", doctors[hash].id,
    doctors[hash].name);
    } else {
        printf("Doctor with ID %d not found or not present.\n", doctorId);

    }
}
```

```
int main() {
    int choice;
    while (1) {
        printf("\nHospital Management System\n");
        printf("1. Add Doctor\n");
        printf("2. Add Patient\n");
        printf("3. Display Doctors\n");
        printf("4. Display Patients\n");
        printf("5. Check Patient Presence\n");
        printf("6. Check Doctor Presence\n");
        printf("7. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                addDoctor();
                break;
            case 2:
                addPatient();
                break;
            case 3:
```

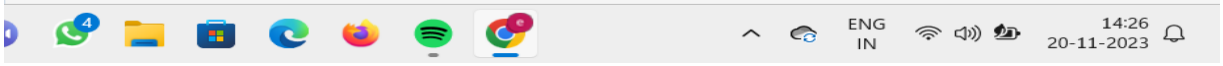
```
            displayDoctors();
            break;
        case 4:
            displayPatients();
            break;
        case 5:
            int patientId;
            printf("Enter patient ID to check presence: ");
            scanf("%d", &patientId);
            checkPatientPresence(patientId);
            break;
        case 6:
            int doctorId;
            printf("Enter doctor ID to check presence: ");
            scanf("%d", &doctorId);
            checkDoctorPresence(doctorId);
            break;
        case 7:
            printf("Exiting the program.\n");
            exit(0);
        default:
```

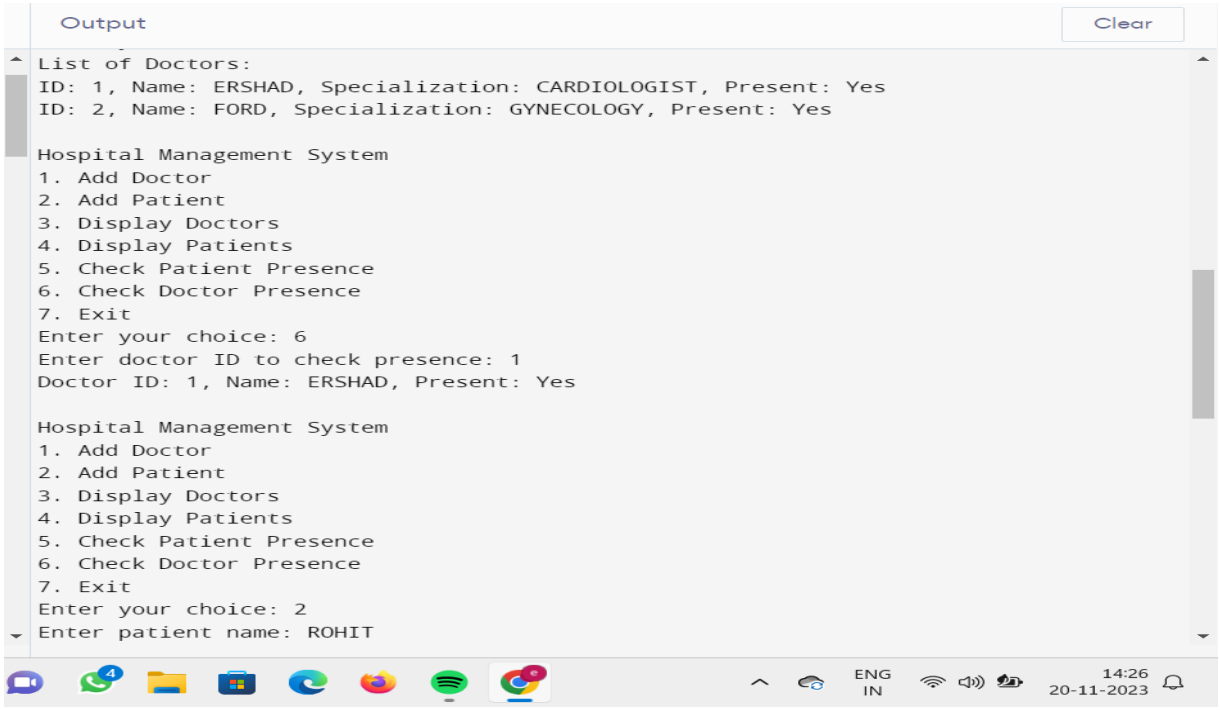
```
            printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```

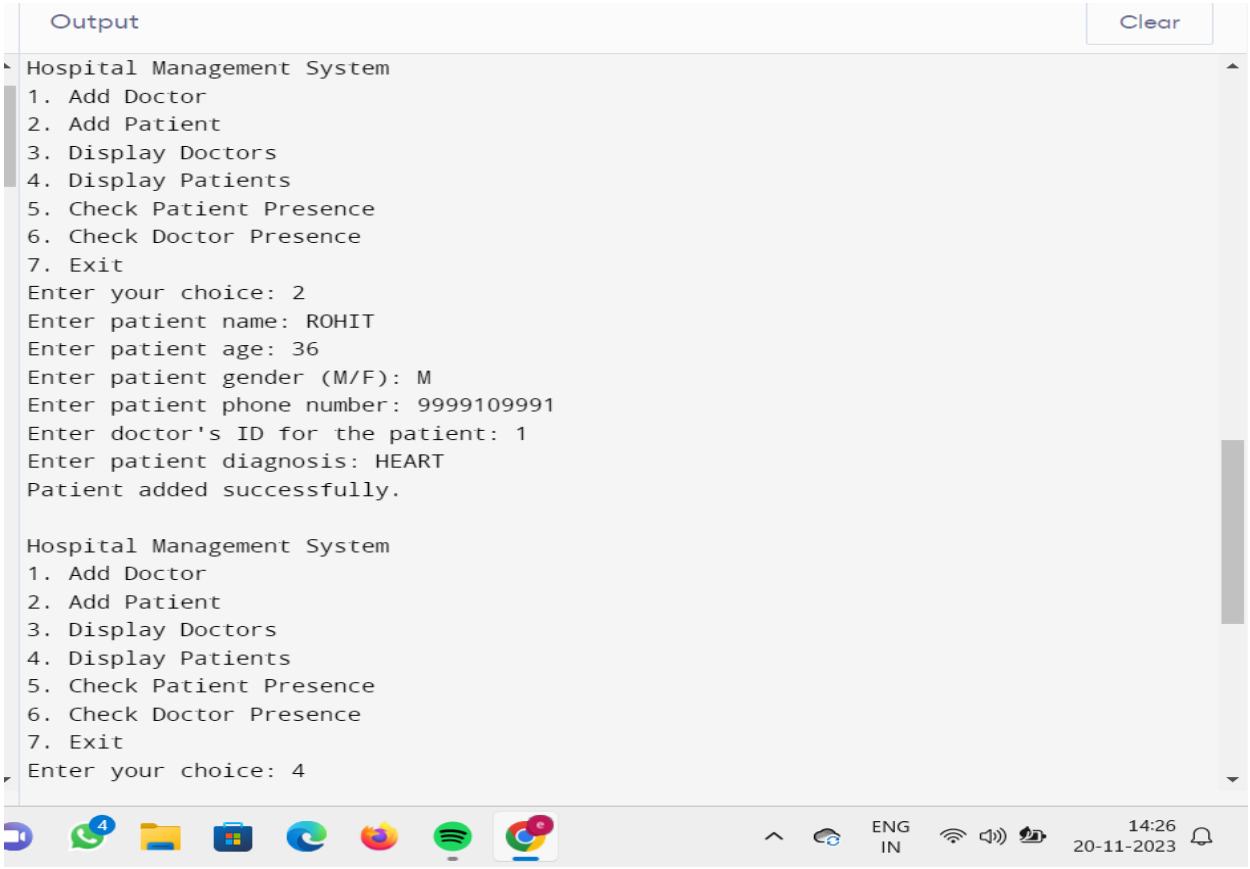
OUTPUT:

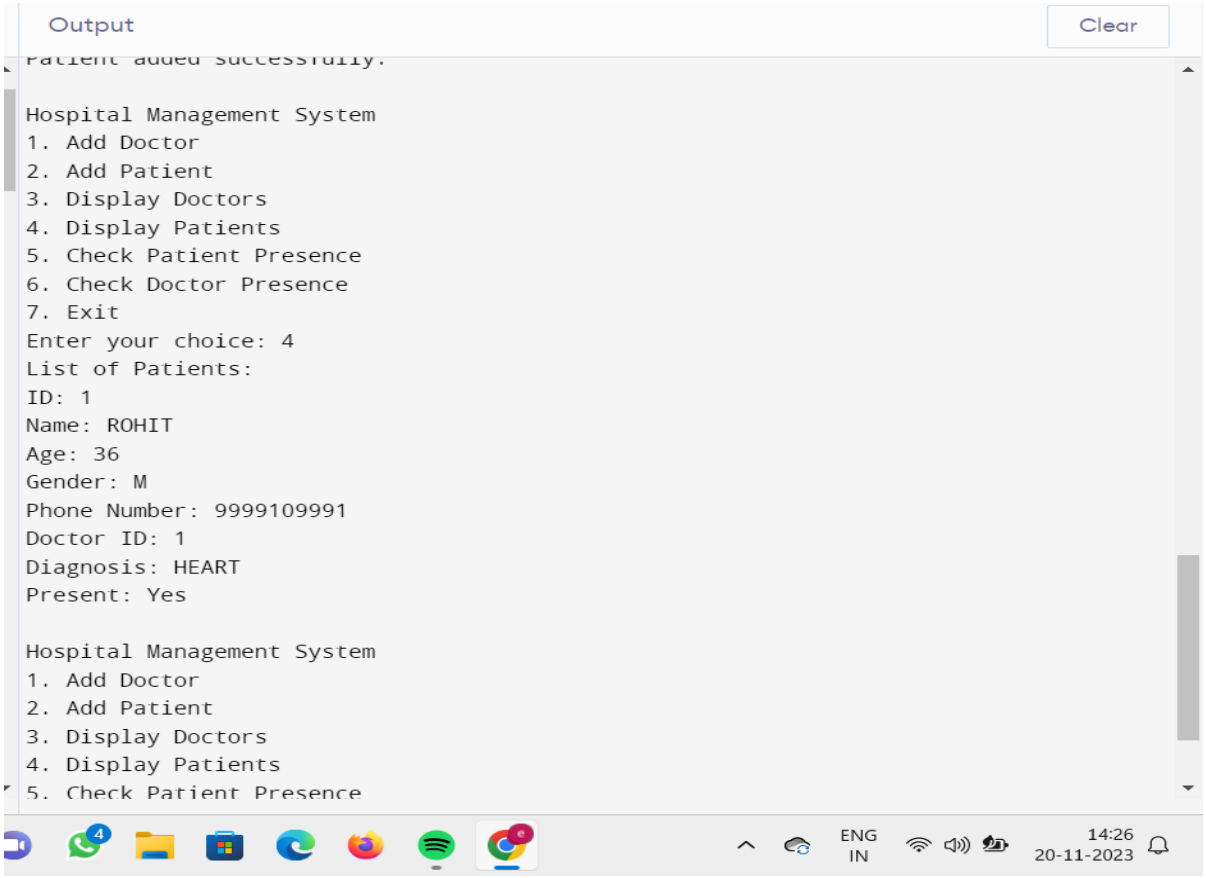
```
Output
/tmp/yvVB47u7QA.o
Hospital Management System
1. Add Doctor
2. Add Patient
3. Display Doctors
4. Display Patients
5. Check Patient Presence
6. Check Doctor Presence
7. Exit
Enter your choice: 1
Enter doctor name: ERSHAD
Enter doctor's specialization: CARDIOLOGIST
Doctor added successfully.

Hospital Management System
1. Add Doctor
2. Add Patient
3. Display Doctors
4. Display Patients
5. Check Patient Presence
6. Check Doctor Presence
7. Exit
Enter your choice: 1
Enter doctor name: FORD
Enter doctor's specialization: GYNECOLOGY
Doctor added successfully.
```

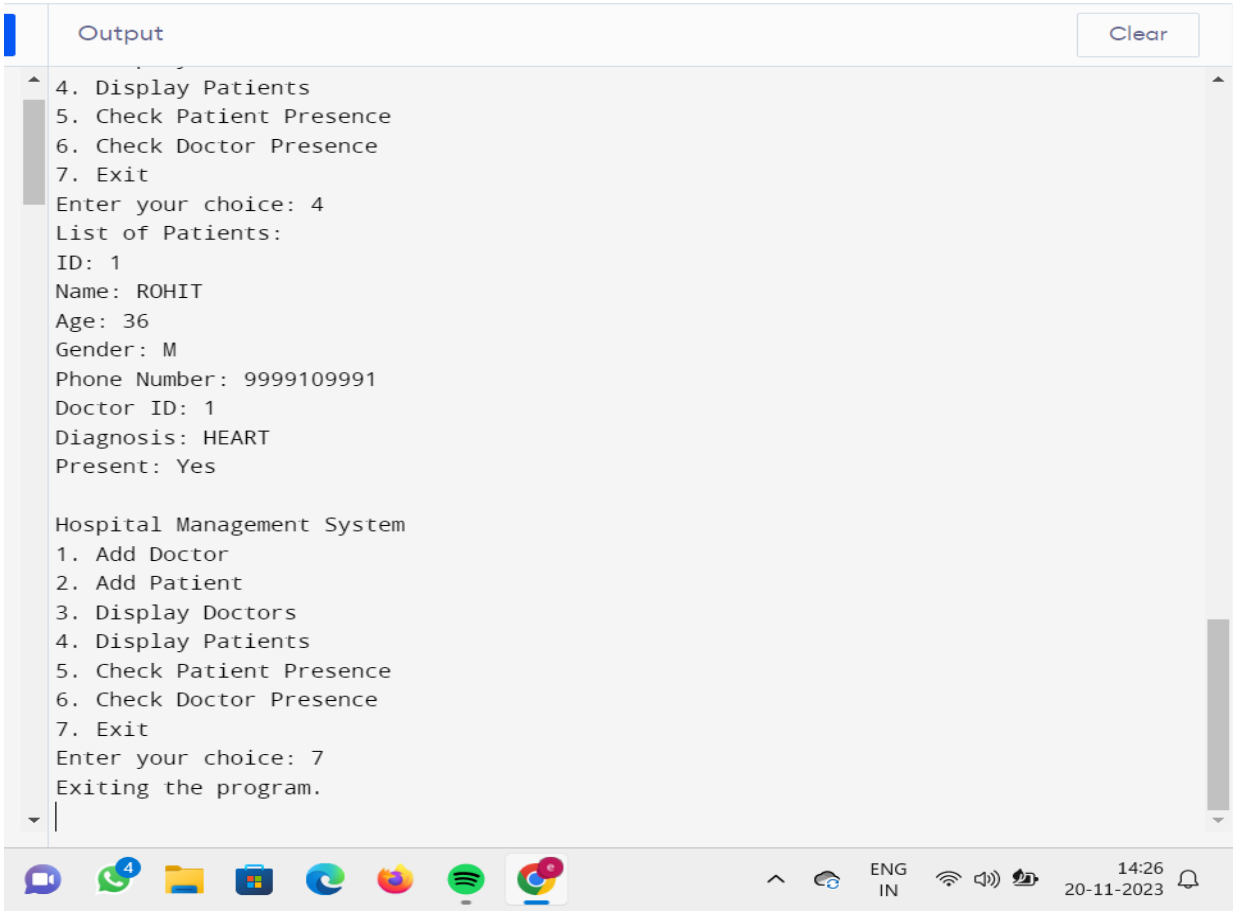
```
Output                                                    Clear

Hospital Management System
1. Add Doctor
2. Add Patient
3. Display Doctors
4. Display Patients
5. Check Patient Presence
6. Check Doctor Presence
7. Exit
Enter your choice: 1
Enter doctor name: FORD
Enter doctor's specialization: GYNECOLOGY
Doctor added successfully.

Hospital Management System
1. Add Doctor
2. Add Patient
3. Display Doctors
4. Display Patients
5. Check Patient Presence
6. Check Doctor Presence
7. Exit
Enter your choice: 3
List of Doctors:
ID: 1, Name: ERSHAD, Specialization: CARDIOLOGIST, Present: Yes
ID: 2, Name: FORD, Specialization: GYNECOLOGY, Present: Yes
```




Output
Clear
List of Doctors:
ID: 1, Name: ERSHAD, Specialization: CARDIOLOGIST, Present: Yes
ID: 2, Name: FORD, Specialization: GYNECOLOGY, Present: Yes
Hospital Management System
1. Add Doctor
2. Add Patient
3. Display Doctors
4. Display Patients
5. Check Patient Presence
6. Check Doctor Presence
7. Exit
Enter your choice: 6
Enter doctor ID to check presence: 1
Doctor ID: 1, Name: ERSHAD, Present: Yes
Hospital Management System
1. Add Doctor
2. Add Patient
3. Display Doctors
4. Display Patients
5. Check Patient Presence
6. Check Doctor Presence
7. Exit
Enter your choice: 2
Enter patient name: ROHIT
ENG
IN
14:26
20-11-2023


Output
Clear
Hospital Management System
1. Add Doctor
2. Add Patient
3. Display Doctors
4. Display Patients
5. Check Patient Presence
6. Check Doctor Presence
7. Exit
Enter your choice: 2
Enter patient name: ROHIT
Enter patient age: 36
Enter patient gender (M/F): M
Enter patient phone number: 9999109991
Enter doctor's ID for the patient: 1
Enter patient diagnosis: HEART
Patient added successfully.
Hospital Management System
1. Add Doctor
2. Add Patient
3. Display Doctors
4. Display Patients
5. Check Patient Presence
6. Check Doctor Presence
7. Exit
Enter your choice: 4
ENG
IN
14:26
20-11-2023


Output
Clear
Patient added successfully.
Hospital Management System
1. Add Doctor
2. Add Patient
3. Display Doctors
4. Display Patients
5. Check Patient Presence
6. Check Doctor Presence
7. Exit
Enter your choice: 4
List of Patients:
ID: 1
Name: ROHIT
Age: 36
Gender: M
Phone Number: 9999109991
Doctor ID: 1
Diagnosis: HEART
Present: Yes
Hospital Management System
1. Add Doctor
2. Add Patient
3. Display Doctors
4. Display Patients
5. Check Patient Presence
ENG
IN
14:26
20-11-2023


Output
Clear
4. Display Patients
5. Check Patient Presence
6. Check Doctor Presence
7. Exit
Enter your choice: 4
List of Patients:
ID: 1
Name: ROHIT
Age: 36
Gender: M
Phone Number: 9999109991
Doctor ID: 1
Diagnosis: HEART
Present: Yes
Hospital Management System
1. Add Doctor
2. Add Patient
3. Display Doctors
4. Display Patients
5. Check Patient Presence
6. Check Doctor Presence
7. Exit
Enter your choice: 7
Exiting the program.
ENG
IN
14:26
20-11-2023

CONCLUSION:

In conclusion, the provided C program constitutes a simple Hospital Management System that allows users to manage and track information about doctors and patients. The program utilizes structures to represent doctor and patient data, and global arrays to store this information. It incorporates functionalities such as adding doctors and patients, displaying lists of doctors and patients, and checking the presence of a specific patient or doctor in the hospital.

Key features of the program include the use of hash values for efficient data retrieval, the introduction of new fields for enhanced data representation (such as phone numbers, diagnosis, and presence status), and a user-friendly interface that operates through a menu system.

Overall, this Hospital Management System serves as a starting point for a more comprehensive solution, and its functionality can be expanded and refined based on specific project requirements and objectives.

CONTRIBUTION

In the course of this project, Weplayed a pivotal role in various aspects, contributing significantly to its success. Our team contributions can be categorized into the following key areas:

1). Project Planning

2). Designing of algorithm

3). Writing code

4). Project Report

5).PPT

- **Our team member YETHIN played an important role in initial parts of the project. He had done the whole project planning and ideology of the whole project.**
- **ERSHAD played a vital role in building of Project Report. He made the whole Project Report.**
- **ROHITH played an important role in the making of PPT. He made the whole PPT.**
- **BHARGAV played an important role in the whole project by involving in the designing of the ALGORITHM.**
- **ALL team members equally participated in Execution of the CODE.**

******THE END******