Mini-Project

Of

Data Structure &Algorithm

(CAN605)

Master Of Computer Applications

Academic Session 2025-2026

Submitted to:

Mr. Sanjay Kumar

Assistant Professor

School Of Computing

Submitted By:

Shashank bhardwaj

MCA – 2nd Sem

SAP ID - 1000024713

School Of Computing

DIT University, Dehradun

CERTIFICATE

Certified that project report entitled "**Hospital Management System**" submitted by "**Priyanshu Negi ,ID- 1000024514", "Shashank Bhardwaj, ID- 1000024713",**"**Varun Gupta ID- 1000024718"** during the period 2024-2025 in partial fulfilment of the requirements for the award of degree of MCA of DIT University, Dehradun, is a record of work carried out under my guidance and supervision. The project report embodies result of referred work and studies carried out by student themselves and the content of the report do not form the basis for the award of any other degree to the candidate or to anybody of the team.

Mr. Sanjay Kumar

Asst. Professor

DIT University, Dehradun

Hospital Management System: This project involves creating a hospital management system to perform menu-driven operations like manage patient records, manage patient appointments, manage doctor assignments and manage medical history.

- ➔ Use arrays or linked lists to store patient information such as patient ID, name, age, medical condition and appointments.

- ➔ Use trees to represent the hierarchical structure of medical staff, with nodes representing doctors and patients. Organize doctors and patients into departments and manage doctor-patient relationships.

Code:-

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

#define MAX_PATIENTS 100
#define MAX_DOCTORS 50
#define MAX_DEPARTMENTS 5

structPatient {
    intpatient_id;
    charname[100];
    intage;
    charmedical_condition[200];
    charappointment_date[20];
};

structDoctor {
    intdoctor_id;
    charname[100];
    chardepartment[50];
    structPatient*patients[MAX_PATIENTS];
    intpatient_count;
};

structDoctorNode {
    structDoctordoctor;
    structDoctorNode*left;
    structDoctorNode*right;
};

structPatientpatients[MAX_PATIENTS];
structDoctordoctors[MAX_DOCTORS];
structDoctorNode*department_roots[MAX_DEPARTMENTS];
intpatient_count=0;
intdoctor_count=0;

voiddisplayMenu() {
    printf("\n--- Hospital Management System ---\n");
    printf("1. Add Patient\n");
    printf("2. Add Doctor\n");
```

```
    printf("3. Assign Doctor to Patient\n");
    printf("4. View Patient Record\n");
    printf("5. View Doctor Assignment\n");
    printf("6. Exit\n");
    printf("Enter your choice: ");
}

voidaddPatient() {
    if (patient_count>=MAX_PATIENTS) {
        printf("Patient limit reached!\n");
        return;
    }
    printf("Enter patient ID: ");
    scanf("%d", &patients[patient_count].patient_id);
    printf("Enter patient name: ");
    getchar();
    fgets(patients[patient_count].name, 100, stdin);

patients[patient_count].name[strcspn(patients[patient_count].name,
"\n")] =0;
    printf("Enter patient age: ");
    scanf("%d", &patients[patient_count].age);
    printf("Enter medical condition: ");
    getchar();
    fgets(patients[patient_count].medical_condition, 200, stdin);

patients[patient_count].medical_condition[strcspn(patients[patient_c
ount].medical_condition, "\n")] =0;
    printf("Enter appointment date: ");
    fgets(patients[patient_count].appointment_date, 20, stdin);

patients[patient_count].appointment_date[strcspn(patients[patient_co
unt].appointment_date, "\n")] =0;
    patient_count++;
    printf("Patient added successfully!\n");
}

voidaddDoctor() {
    if (doctor_count>=MAX_DOCTORS) {
        printf("Doctor limit reached!\n");
        return;
    }
```

```
    printf("Enter doctor ID: ");
    scanf("%d", &doctors[doctor_count].doctor_id);
    printf("Enter doctor name: ");
    getchar();
    fgets(doctors[doctor_count].name, 100, stdin);
    doctors[doctor_count].name[strcspn(doctors[doctor_count].name,
"\n")] =0;
    printf("Enter department: ");
    fgets(doctors[doctor_count].department, 50, stdin);

doctors[doctor_count].department[strcspn(doctors[doctor_count].depar
tment, "\n")] =0;
    doctors[doctor_count].patient_count=0;
    doctor_count++;
    printf("Doctor added successfully!\n");
}

voidassignDoctorToPatient() {
    intpatient_id, doctor_id;
    printf("Enter patient ID: ");
    scanf("%d", &patient_id);
    printf("Enter doctor ID: ");
    scanf("%d", &doctor_id);

    structPatient*patient=NULL;
    structDoctor*doctor=NULL;

    for (inti=0; i<patient_count; i++) {
        if (patients[i].patient_id==patient_id) {
            patient=&patients[i];
            break;
        }
    }

    for (inti=0; i<doctor_count; i++) {
        if (doctors[i].doctor_id==doctor_id) {
            doctor=&doctors[i];
            break;
        }
    }

    if (patient==NULL||doctor==NULL) {
```

```
        printf("Invalid patient or doctor ID!\n");
        return;
    }

    doctor->patients[doctor->patient_count] =patient;
    doctor->patient_count++;
    printf("Doctor assigned to patient successfully!\n");
}

voidviewPatientRecord() {
    intpatient_id;
    printf("Enter patient ID: ");
    scanf("%d", &patient_id);

    structPatient*patient=NULL;
    for (inti=0; i<patient_count; i++) {
        if (patients[i].patient_id==patient_id) {
            patient=&patients[i];
            break;
        }
    }

    if (patient==NULL) {
        printf("Patient not found!\n");
        return;
    }

    printf("\nPatient ID: %d\n", patient->patient_id);
    printf("Name: %s\n", patient->name);
    printf("Age: %d\n", patient->age);
    printf("Medical Condition: %s\n", patient->medical_condition);
    printf("Appointment Date: %s\n", patient->appointment_date);
}

voidviewDoctorAssignment() {
    intdoctor_id;
    printf("Enter doctor ID: ");
    scanf("%d", &doctor_id);

    structDoctor*doctor=NULL;
    for (inti=0; i<doctor_count; i++) {
        if (doctors[i].doctor_id==doctor_id) {
```

```
            doctor=&doctors[i];
            break;
        }
    }

    if (doctor==NULL) {
        printf("Doctor not found!\n");
        return;
    }

    printf("\nDoctor ID: %d\n", doctor->doctor_id);
    printf("Name: %s\n", doctor->name);
    printf("Department: %s\n", doctor->department);
    printf("Assigned Patients:\n");

    if (doctor->patient_count==0) {
        printf("No patients assigned yet.\n");
    } else {
        for (inti=0; i<doctor->patient_count; i++) {
            printf("Patient ID: %d, Name: %s\n", doctor-
>patients[i]->patient_id, doctor->patients[i]->name);
        }
    }
}

intmain() {
    intchoice;

    while (1) {
        displayMenu();
        scanf("%d", &choice);

        switch (choice) {
            case1:
                addPatient();
                break;
            case2:
                addDoctor();
                break;
            case3:
                assignDoctorToPatient();
                break;
```

```
            case4:
                viewPatientRecord();
                break;
            case5:
                viewDoctorAssignment();
                break;
            case6:
                printf("Exiting the system...\n");
                return0;
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }
    return0;
}
```

OUTPUT:-

- *Menu:-*

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS D:\Python\C Files\Projects> cd "d:\Python\C File

--- Hospital Management System ---
1. Add Patient
2. Add Doctor
3. Assign Doctor to Patient
4. View Patient Record
5. View Doctor Assignment
6. Exit
Enter your choice:
```

- Add Patient :-

```
Enter your choice: 1
Enter patient ID: 01
Enter patient name: Atul Verma
Enter patient age: 34
Enter medical condition: Corona
Enter appointment date: 25/04/2025
Patient added successfully!
```

- Add Doctor :-

```
Enter your choice: 2
Enter doctor ID: 1
Enter doctor name: Dr Rakesh Joshi
Enter department: Emergency Department
Doctor added successfully!
```

- Assign Doctor to Patient :-

```
Enter your choice: 3
Enter patient ID: 01
Enter doctor ID: 1
Doctor assigned to patient successfully!
```

- View Patient Record :-

```
Enter your choice: 4
Enter patient ID: 01

Patient ID: 1
Name: Atul Verma
Age: 34
Medical Condition: Corona
Appointment Date: 25/04/2025
```

- View Doctor Assignment :-

```
Enter your choice: 5
Enter doctor ID: 1

Doctor ID: 1
Name: Dr Rakesh Joshi
Department: Emergency Department
Assigned Patients:
Patient ID: 1, Name: Atul Verma
```

- Exit Program :-

```
6. Exit
Enter your choice: 6
Exiting the system...
PS D:\Python\C Files\Projects>
```

CONCLUSION

The **Hospital Management System** project developed here demonstrates the power of data structures in organizing and managing critical information in a healthcare environment. By using **binary trees** to structure medical staff and **arrays/linked lists** to manage patient data, we've created an efficient and scalable system that handles:

- **Patient Information**: Storing and managing detailed patient records, such as personal details, medical conditions, and appointments.
- **Doctor Assignment**: Organizing doctors in a hierarchical manner, where each doctor is easily linked to their specific department or specialty, and relationships with patients can be effectively maintained.
- **Medical History Management**: Handling patient medical histories, which allows for easy updates, retrieval, and review of past treatments.
- **Appointment Scheduling**: Managing patient appointments and doctor schedules, ensuring that appointments are booked efficiently and conflict-free.