CEP Report

By

NAME	RegistrationNumber
Hamza Jadoon	FA22-BCE-012
Abdur Rehman	FA22-BCE-027
Ibad-ur-Rehman	FA22-BCE-032

For the course

Object Oriented Programming

Semester Fall 2023

Supervised by:

CEP Course Teacher's Name

Dr. M. Bilal Qureshi

Department of Electrical & Computer Engineering

COMSATS University Islamabad – Abbottabad Campus

DECLARATION

We Hamza Jadoon FA22-BCE-012, Abdur Rehman FA22-BCE-027, Ibad-ur-Rehman FA22-BCE-032 hereby declare that we have produced the work presented in this report, during the scheduled period of study. We also declare that we have not taken any material from any source except referred to wherever due. If a violation of rules has occurred in this report, we shall be liable to punishable action.

Date: 29-12-2023

Hamza Jadoon

FA22-BCE-012

Abdur Rehman

FA22-BCE-027

Ibad-ur-Rehman

FA22-BCE-032

ABSTRACT

The existing hospital management system is outdated and inefficient, causing difficulties in managing various operations and hindering the overall efficiency of the hospital. The lack of a comprehensive and user-friendly system leads to challenges in patient registration, appointment scheduling, record management, and overall coordination among different departments. Additionally, the absence of real time data integration and reporting features hampers decision-making processes and impedes the overall quality of healthcare services. Therefore, there is a critical need to develop a basic hospital management system that addresses these challenges and provides a streamlined and efficient solution for managing hospital operations, improving patient care, and enhancing the overall performance of the healthcare facility

TABLE OF CONTENTS

LIST OF FIGURES

1 Introduction

The Basic Hospital Management System is a comprehensive software solution designed to streamline and optimize the operations of healthcare facilities. This project aims to develop a user-friendly system that facilitates efficient management of various hospital processes, including patient registration (along different characteristics of the patient), removal of existing patients, record management, and displaying the admitted patients along with their info. By implementing this system, hospitals can improve their overall efficiency, enhance patient care, and enable seamless communication among staff members. The Basic Hospital Management System will leverage modern technologies to provide real-time data integration and reporting capabilities, enabling effective decision-making and ensuring a high standard of healthcare services

Fig: 1.1 Title of the Picture

1.1 Objectives

- Design and implement a "**Basic Hospital Management System"** capable of adding and removing patients, as well as displaying a list of patients with their relevant details.
- Create a user-friendly interface for input and output.
- It provides a simple and user-friendly interface for hospital staff to perform essential tasks related to patient management.

1.2 Features and Cost Estimate of our Project

1. Functional:

Adding Patients/Doctors:

The system can add patients and doctors by asking for their ID, number, name etc.

Removing Patients:

The system can remove patients that no longer require to be in the hospital by pressing a specific key.

Displaying:

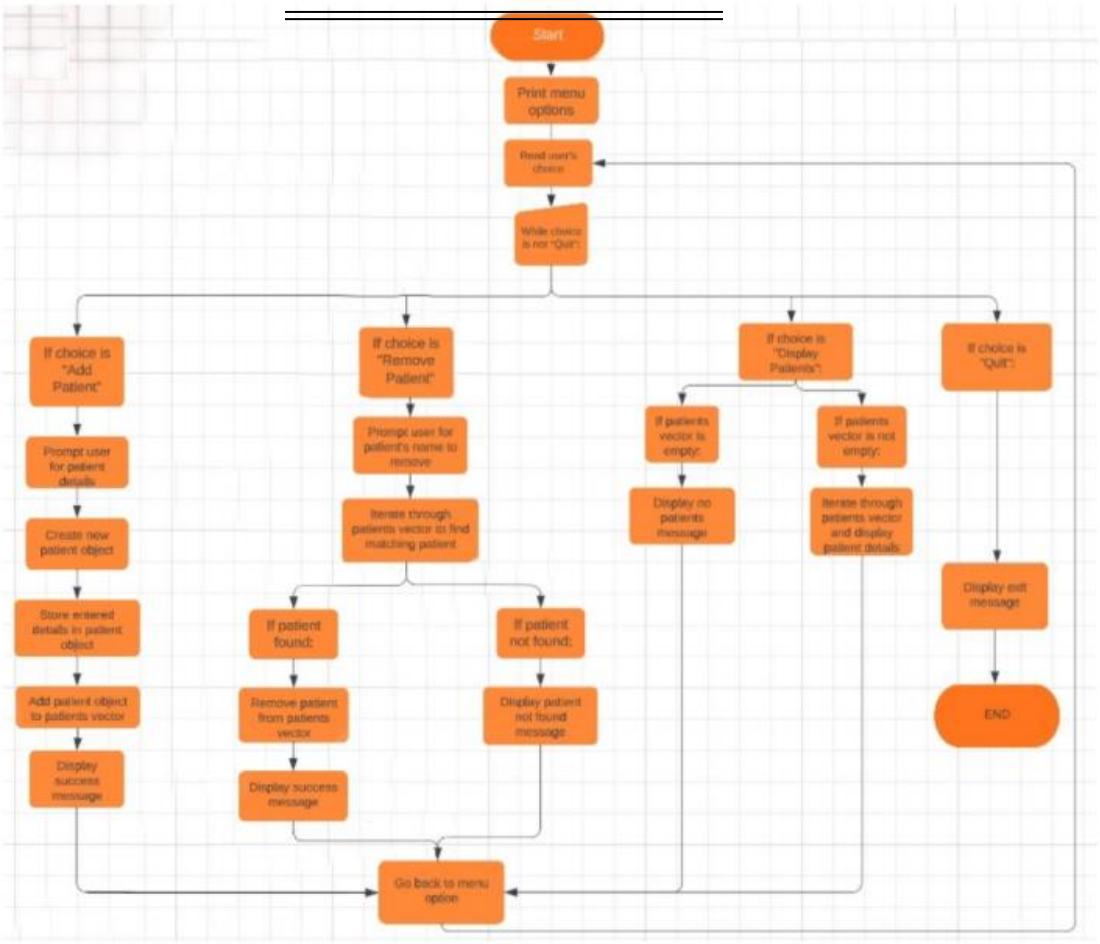
The system can display both doctors and patients alongside with their IDs.

2 Literature Survey

We didn't have to go for deep study. The project included simple structure knowledge and classes

The code is written in C++ language and uses the features of object oriented programming(class).

3 Flow chart



4

5 Simulation Results

→ CODE & IMPLEMENTATION:

```
#include <iostream>
#include <cstring>
using namespace std;
```

```
const int MAX_PATIENTS = 100;
const int MAX_DOCTORS = 10;
struct Doctor {
    int id;
    char name[50];
    char specialization[50];
};
struct Patient {
    int id;
    char name[50];
    int age;
    char gender;
    int doctorId;
};
class HospitalManagementSystem {
private:
    Doctor doctors[MAX_DOCTORS];
    Patient patients[MAX_PATIENTS];
    int doctorCount;
    int patientCount;

public:
    HospitalManagementSystem() : doctorCount(0), patientCount(0) {}

    void addDoctor() {
        if (doctorCount < MAX_DOCTORS) {
            Doctor  doctor;
            cout << "Enter Doctor ID: ";
            cin >> doctor.id;
            cout << "Enter Doctor Name: ";
            cin>>doctor.name;
            cout << "Enter Doctor Specialization: ";
            cin>>doctor.specialization;;
            doctors[doctorCount++] = doctor;
```

```
            cout << "Doctor added successfully.\n";
        }
        else {
            cout << "Maximum number of doctors reached.\n";
        }
    }

    void addPatient() {
        if (patientCount < MAX_PATIENTS) {
            Patient patient;
            cout << "Enter Patient ID: ";
            cin >> patient.id;
            cout << "Enter Patient Name: ";
            cin>>patient.name;
            cout << "Enter Patient Age: ";
            cin >> patient.age;
            cout << "Enter Patient Gender (M/F): ";
            cin >> patient.gender;
            cout << "Enter Doctor ID for the Patient: ";
            cin >> patient.doctorId;
            patients[patientCount++] = patient;
            cout << "Patient added successfully.\n";
        }
        else {
            cout << "Maximum number of patients reached.\n";
        }
    }
    void displayDoctors() {
        if (doctorCount > 0) {
           cout << "\nList of Doctors:\n";
            cout << "ID\tName\t\tSpecialization\n";
            for (int i = 0; i < doctorCount; ++i) {
                cout << doctors[i].id << "\t" << doctors[i].name << "\t\t" <<
doctors[i].specialization << "\n";
```

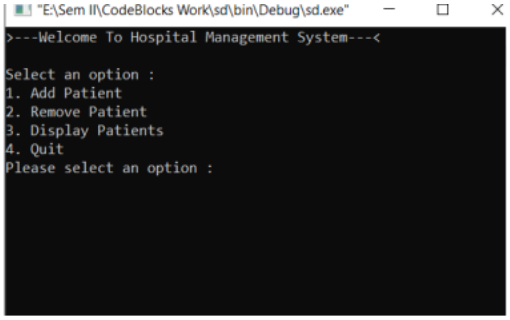
```
            }
        }
        else {
            cout << "No doctors available.\n";
        }
    }
    void displayPatients() {
        if (patientCount > 0) {
            cout << "\nList of Patients:\n";
            cout << "ID\tName\t\tAge\tGender\tDoctor ID\n";
            for (int i = 0; i < patientCount; ++i) {
                cout << patients[i].id << "\t" << patients[i].name << "\t\t" <<
patients[i].age << "\t" << patients[i].gender << "\t" << patients[i].doctorId << "\n";
            }
        }
        else {
            cout << "No patients available.\n";
        }
    }
};
int main() {
    HospitalManagementSystem hospital;
    int choice;
    do {
        cout << "\nHospital Management System Menu:\n";
        cout << "1. Add Doctor\n";
        cout << "2. Add Patient\n";
        cout << "3. Display Doctors\n";
        cout << "4. Display Patients\n";
        cout << "0. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice) {
        case 1:
```

```
            hospital.addDoctor();
            break;
        case 2:
            hospital.addPatient();
            break;
        case 3:
            hospital.displayDoctors();
            break;
        case 4:
            hospital.displayPatients();
            break;
        case 0:
            cout << "Exiting the program.\n";
            break;
        default:
            cout << "Invalid choice. Please try again.\n";
        }
    } while (choice != 0);
    return 0;
}
```

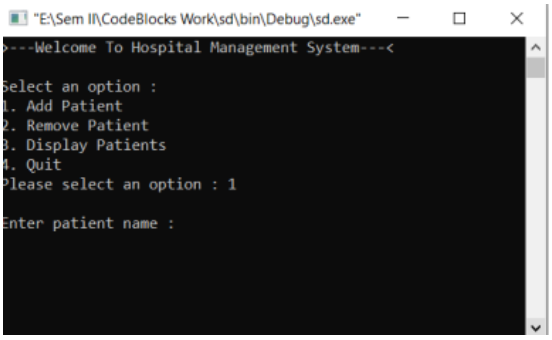
Fig: 4.1 simulation results1

→ RESULTS / OUTPUT :

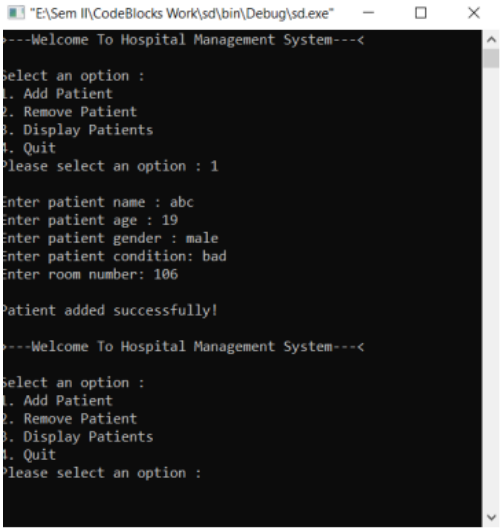
o Starting of code: Let us choose 1 of 4 options:



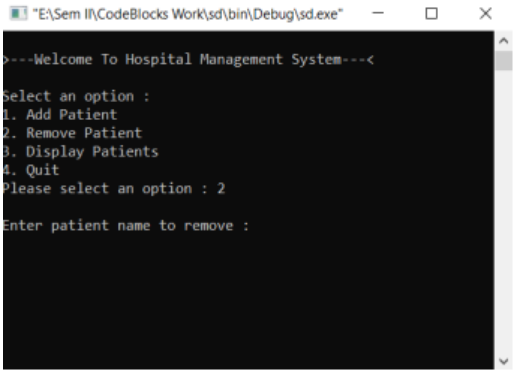
o If we select 1st option i.e., “Add Patient”: it will ask for diff detail including patient’s name, age, gender, condition and room numb.



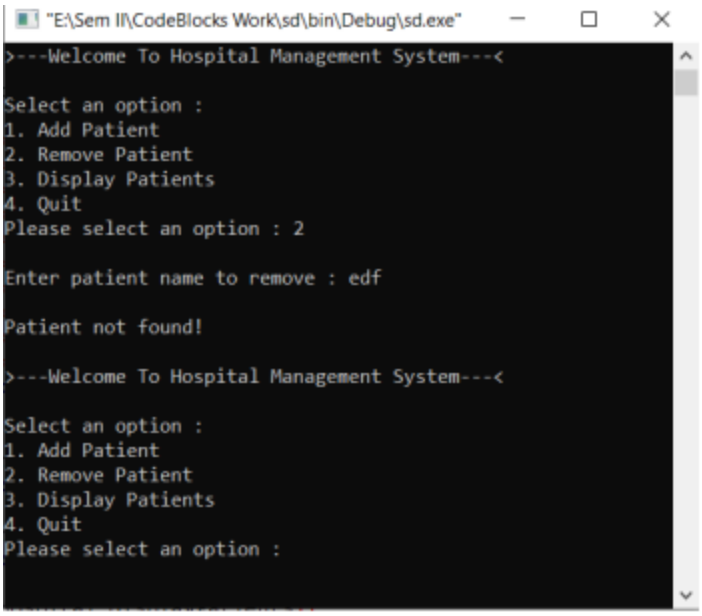
o After entering all the details, it will store patient’s info and display a success message. Then it will bring user back to menu option for further choices



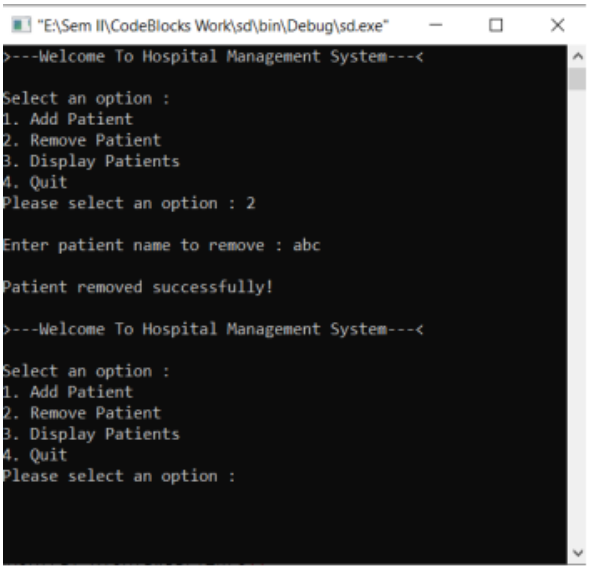
o Now if the we select 2nd option i.e., Remove Patient: It will ask for patient’s name in order to be removed:



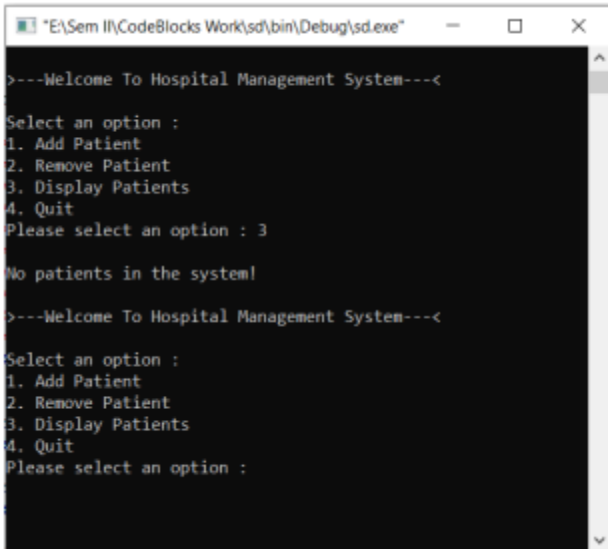
o If the entered name patient is not registered: It will display a message that “Patient not found!”, and will bring user back to menu option:



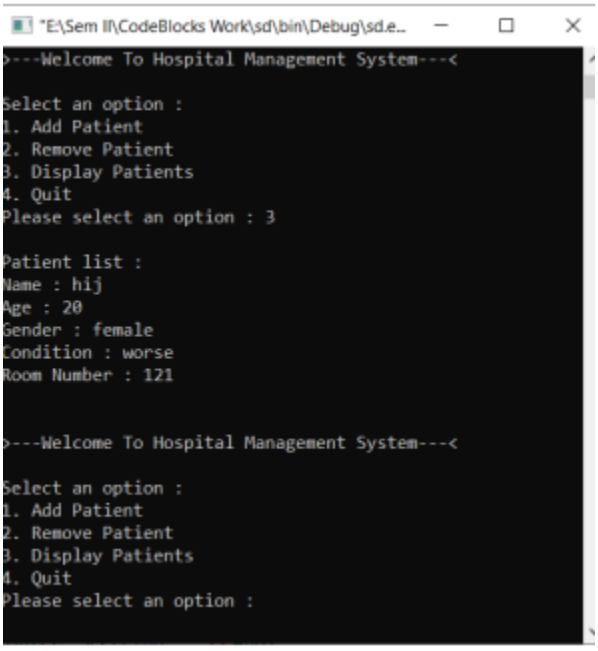
o After again selecting option “2”: If the entered named patient is registered, it will remove that patient’s info. with a success message:



o Now if we select 3rd option i.e., “Display Patients”: If there aren’t any patients registered, it will display the message:



o Now if there are some patient's registered and we choose to display them: It will display all the patients registered along with their respective info.



o Now if we select 4th option i.e., "Quit": The program will be ended with a message:

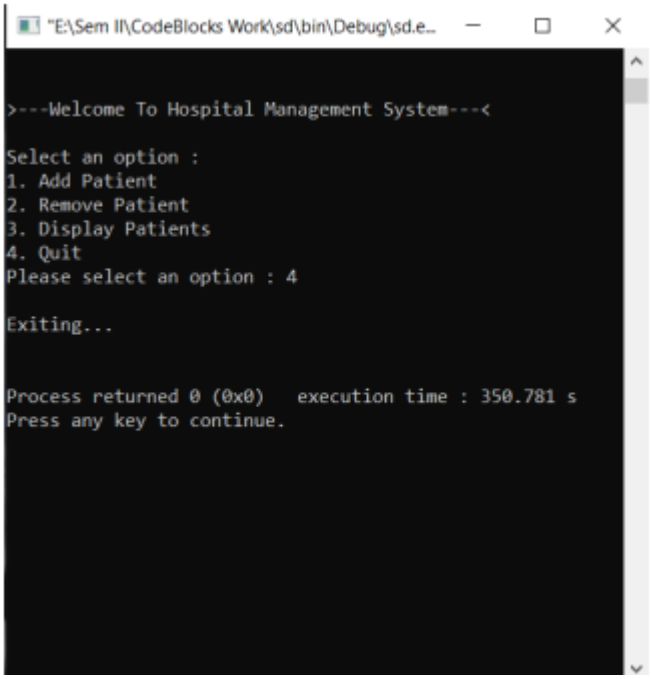


Fig: 4.1 simulation results2

5. Conclusion:

The code implements a basic hospital management system that allows users to add and remove patients, as well as display a list of patients with their relevant details. The code demonstrates the use of classes, structures and input/output operations to facilitate the management of patient records. It provides a simple and user-friendly interface for hospital staff to perform essential tasks related to patient management. Although the code is a minimal implementation, it can serve as a foundation for further enhancements and the addition of more advanced features to meet specific requirements in a hospital setting

6. References

Should be in standard IEEE format…

[1]. Author Name (year of publication). Title Journal/Conference/website [online] Available at: http:// Accessed on (dd-mm--yyyy)

[2]. Author Name (20xx). Title [online] Available at: https://Accessed on (dd-mm--yyyy)

Teachers should assess CLO2, CLO3 and CLO4 based on the given rubrics (overall weightage 20%)

Recommended Percentage Breakdown

CLO	**Percentage**
CLO2 (Investigation)	10%
CLO3 (Referencing/Citations) *(Turnitin report should be generated.)*	5%
CLO4 (Communication)	5%