KLE Technological University | Creating Value, Leveraging Knowledge
Dr. M. S. Sheshgiri Campus, Belagavi
Department of Electronics & Communication Engineering

Department of
Electronics and Communication Engineering

Course Project Report

on

Hospital Management System

By:

1. **Prajwal Halgi** USN:02FE21BEC059
2. **Sakshi Bagewadi** USN:02FE21BEC089
3. **Shivanand Ghevadi** USN:02FE22BEC420

Semester: VI, 2023-2024

Under the Guidance of

Prof. Suresh Murgod

KLE Technological University,
Dr. M. S. Sheshgiri College of Engineering and Technology
BELAGAVI-590 008
2023-2024

KLE Technological University | Creating Value, Leveraging Knowledge
Dr. M. S. Sheshgiri Campus, Belagavi
Department of Electronics & Communication Engineering

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that project entitled “ **Hospital Management System** ” is a bonafide work carried out by the student team of ” **Prajwal Halgi 02FE21BEC059, Sakshi Bagewadi 02FE21BEC089, Shivanand Ghevadi 02FE22BEC420**”. The project report has been approved as it satisfies the requirements with respect to the Course project work prescribed by the university curriculum for B.E. (VI Semester) in Department of Electronics and Communication Engineering of KLE Technological University Dr. M. S. Sheshgiri CET Belagavi campus for the academic year 2023-2024.

Prof. Suresh Murgod	**Dr. Dattaprasad A. Torse**	**Dr. S. F. Patil**
Guide	**Head of Department**	**Principal**

Hospital Managment System

Team members

June 12, 2024

SL.No	Name	USN	Roll.No
1	Prajwal Hlagi	02FE21BEC059	54
2	Sakshi Bagewadi	02FE21BEC089	09
3	Shivanand Ghevadi	02FE21BEC420	61

1 Guide: Proff Suresh.F.Murgod

2 Problem Statement

Design and implement a Hospital Management System (HMS) using Object-Oriented Programming (OOP) concepts in C++

3 Abstract

The Hospital Management System (HMS) is a comprehensive software solution designed to facilitate the efficient management of hospital operations using Object-Oriented Programming (OOP) concepts in C++. This system aims to streamline the administrative tasks associated with patient management, doctor scheduling, appointment booking, and medical record maintenance. By leveraging the principles of encapsulation, inheritance, and polymorphism, the HMS provides a robust and scalable framework for managing hospital data and workflows. The core components of the HMS include classes for Person, Patient, Doctor, Appointment, and MedicalRecord. These classes are designed to encapsulate relevant attributes and methods, enabling seamless interaction and data manipulation. The Person class serves as a base class, with Patient and Doctor classes inheriting common attributes while adding specific functionalities relevant to their roles. The Appointment and MedicalRecord classes facilitate the management of scheduling and medical documentation, respectively Key functionalities of the HMS include adding and updating patient and doctor information, scheduling and managing appointments, maintaining and retrieving medical records, and providing search capabilities for efficient data retrieval. The system incorporates data persistence mechanisms through file handling to ensure that data is saved and loaded across sessions, ensuring continuity and reliability.

4 Introduction

In today's rapidly evolving healthcare landscape, efficient management of hospital operations is crucial for providing high-quality patient care and ensuring smooth administrative workflows. The Hospital Management System (HMS) is a software application developed using Object-Oriented Programming (OOP) concepts in C++, designed to address the complexities and challenges associated with managing hospital data and operations. The HMS aims to streamline various aspects of hospital management, including patient registration, doctor scheduling, appointment booking, and medical record maintenance. By leveraging OOP principles, the system provides a modular, scalable, and maintainable framework that enhances the overall efficiency and effectiveness of hospital administration. The core components of the HMS are encapsulated within distinct classes: Person, Patient, Doctor, Appointment, and MedicalRecord. These classes are designed to model real-world entities and their interactions within the hospital environment. The Person class serves as a foundational base, encapsulating common attributes such as name, age, gender, address, and contact information. Derived from this base class, the Patient and Doctor classes extend functionalities specific to patient care and medical practice, respectively

5 Literature Survey

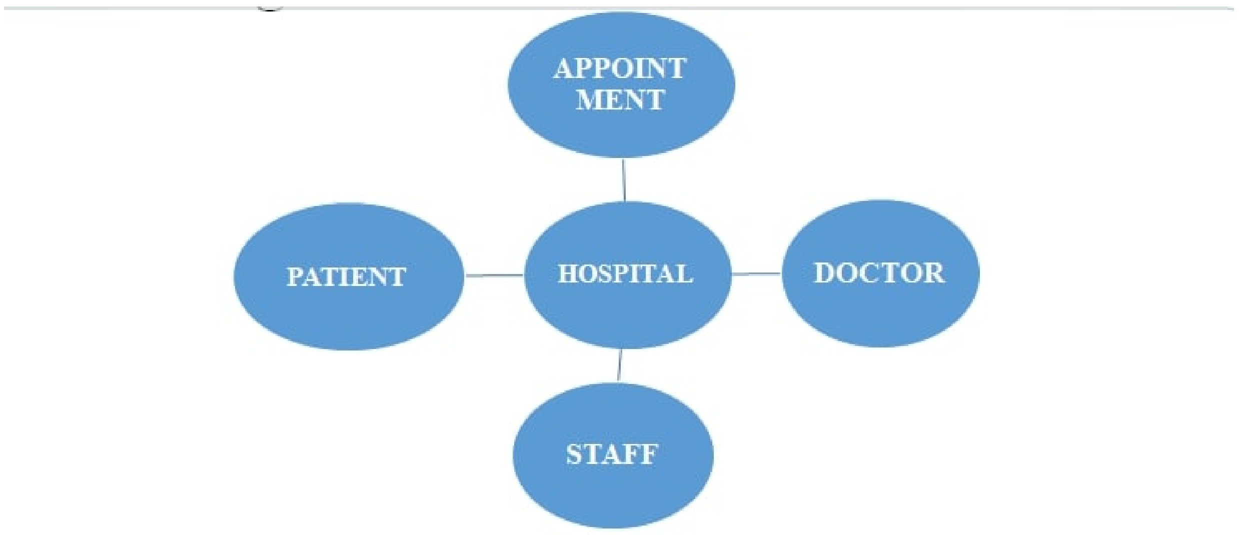
1. General Information: How many departments does your hospital have? What are the main services provided by your hospital/clinic?

2. Patient Management: What information do you need to store about patients? (e.g., name, age, contact details, medical history) How do you currently manage patient appointments and admissions? Do you require a system for managing patient billing and payments?

3. Doctor Management: How many doctors do you have in your hospital/clinic? What information do you need to store about doctors? (e.g., name, specialty, contact details) Do you need a system to manage doctor schedules and appointments?

4. Appointment Management: How are appointments currently scheduled in your hospital? Do you need the system to handle different types of appointments (e.g., routine check-ups, consultations, surgeries)? Are there any specific requirements for appointment reminders or notifications?

6 Block Diagram



7 Block Diagram Explaination

1. Patient (Derived Class from Person): Attributes: PatientID, MedicalHistory, CurrentDiagnosis, TreatmentPlan Methods: ScheduleAppointment(), AddMedicalRecord(), DisplayPatientDetails()

2. Doctor (Derived Class from Person): Attributes: DoctorID, Specialization, Availability, Appointments Methods: AddAvailability(), ScheduleAppointment(), DisplayDoctorDetails()

3. Appointment: Attributes: AppointmentID, Patient, Doctor, Date, Time, Status Methods: Schedule(), Cancel(), Reschedule(), DisplayAppointmentDetails()

4. Staff: Medical Staff: Doctors,Nurses,Technicians and Therapists. Administrative Staff: Administrative Officer,Receptionists and Billing and Coding Staff

8 Concepts Used

1. Inheritance: Inheritance is a mechanism where a new class, known as a child class, is derived from an existing class, known as a parent class. The child class inherits attributes and behaviors (methods) from the parent class, which allows for code reusability and the creation of a hierarchical relationship between classes. For example, if we have a general class Animal, a specific class Dog can inherit from Animal, gaining its properties and behaviors while also introducing its own specific features.

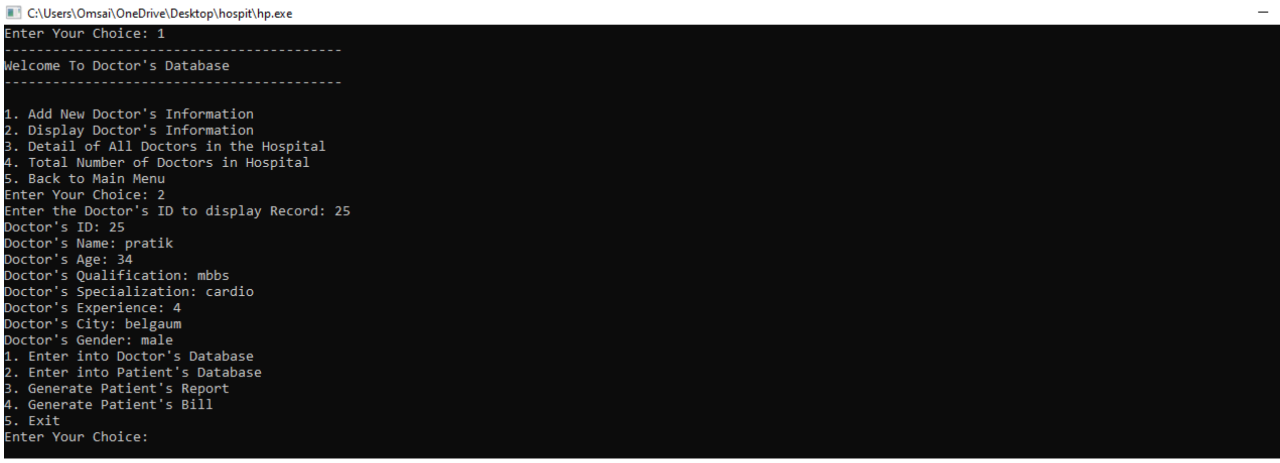
2. Polymorphism: Polymorphism allows objects of different classes to be treated as objects of a common superclass. It enables a single interface to represent different underlying forms (data types). The most common use of polymorphism is when a parent class reference is used to refer to a child class object. For instance, a function that takes a shape object can also accept objects of its subclasses like circle, square, or triangle, and each shape can define its own method for calculating the area.

3. File Handling: File handling is the process of reading from and writing to files. It involves opening a file, performing operations (like reading or writing), and then closing the file to ensure that resources are properly managed. This is essential for tasks like saving data, loading configurations, logging events, or even processing large amounts of data stored in files.

4. Abstraction: Abstraction is the concept of hiding the complex implementation details and showing only the necessary features of an object. It simplifies the user interaction with objects by providing a clear and simplified view. For example, when you use a smartphone, you interact with its user interface, which abstracts the complex inner workings of the device, such as the hardware and software operations that make the device function.

5. Encapsulation: Encapsulation is the bundling of data (attributes) and methods (functions) that operate on the data into a single unit, or class. It restricts direct access to some of an object's components, which is a means of preventing accidental interference and misuse of the data. For instance, a class may have private attributes that cannot be accessed directly from outside the class; instead, they can only be accessed through public methods, thus protecting the integrity of the data.

9 Results

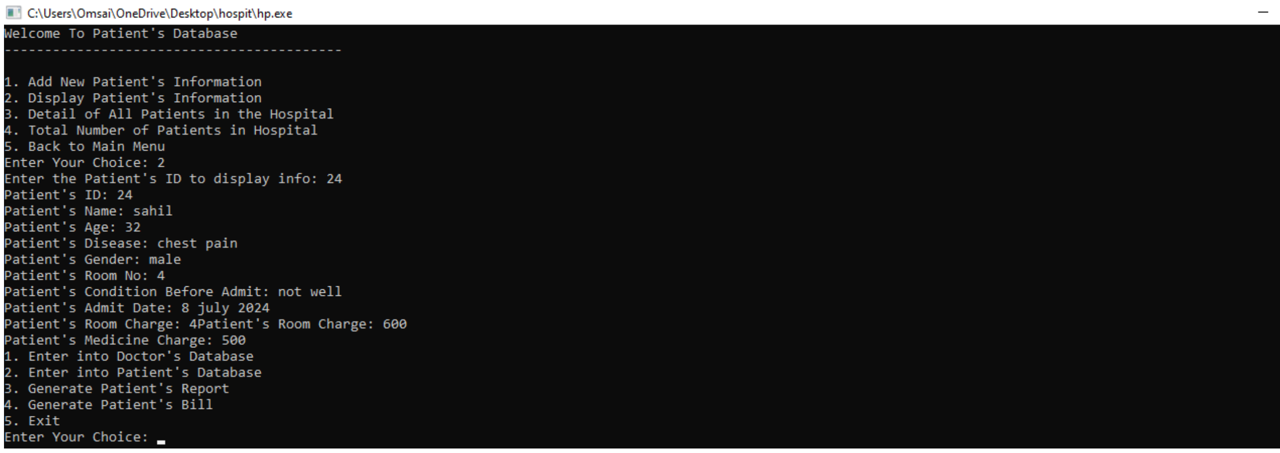


C:\Users\Omsai\OneDrive\Desktop\hospit\hp.exe

```
Enter Your Choice: 1
------------------------------------------
Welcome To Doctor's Database
------------------------------------------

1. Add New Doctor's Information
2. Display Doctor's Information
3. Detail of All Doctors in the Hospital
4. Total Number of Doctors in Hospital
5. Back to Main Menu
Enter Your Choice: 2
Enter the Doctor's ID to display Record: 25
Doctor's ID: 25
Doctor's Name: pratik
Doctor's Age: 34
Doctor's Qualification: mbbs
Doctor's Specialization: cardio
Doctor's Experience: 4
Doctor's City: belgaum
Doctor's Gender: male
1. Enter into Doctor's Database
2. Enter into Patient's Database
3. Generate Patient's Report
4. Generate Patient's Bill
5. Exit
Enter Your Choice:
```

Doctors Database

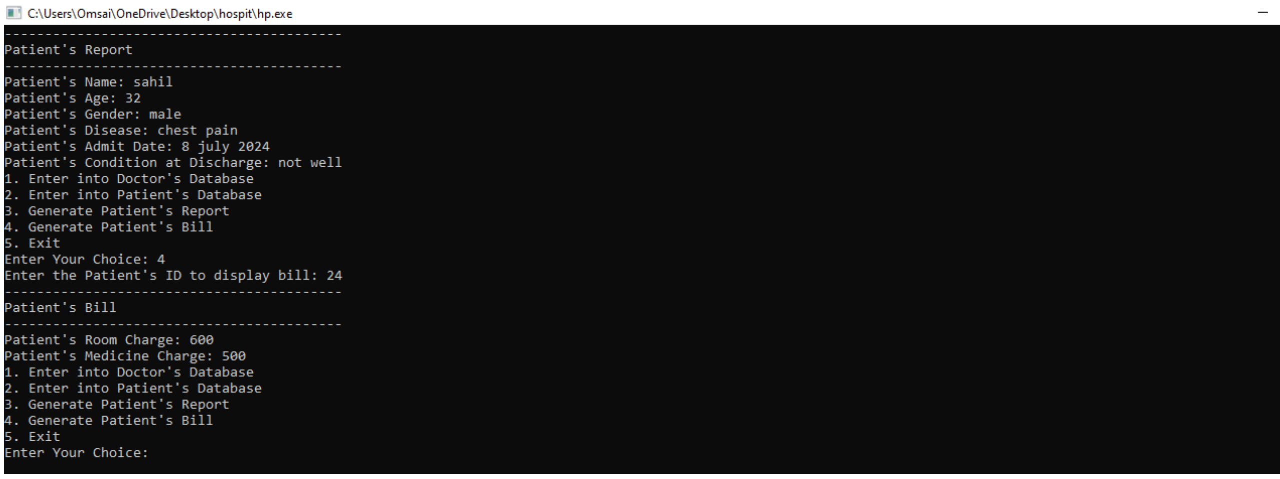


C:\Users\Omsai\OneDrive\Desktop\hospit\hp.exe

```
Welcome To Patient's Database
------------------------------------------

1. Add New Patient's Information
2. Display Patient's Information
3. Detail of All Patients in the Hospital
4. Total Number of Patients in Hospital
5. Back to Main Menu
Enter Your Choice: 2
Enter the Patient's ID to display info: 24
Patient's ID: 24
Patient's Name: sahil
Patient's Age: 32
Patient's Disease: chest pain
Patient's Gender: male
Patient's Room No: 4
Patient's Condition Before Admit: not well
Patient's Admit Date: 8 july 2024
Patient's Room Charge: 4Patient's Room Charge: 600
Patient's Medicine Charge: 500
1. Enter into Doctor's Database
2. Enter into Patient's Database
3. Generate Patient's Report
4. Generate Patient's Bill
5. Exit
Enter Your Choice: _
```

Patients Database



```
C:\Users\Omsai\OneDrive\Desktop\hospit\hp.exe
----------------------------------------
Patient's Report
----------------------------------------
Patient's Name: sahil
Patient's Age: 32
Patient's Gender: male
Patient's Disease: chest pain
Patient's Admit Date: 8 july 2024
Patient's Condition at Discharge: not well
1. Enter into Doctor's Database
2. Enter into Patient's Database
3. Generate Patient's Report
4. Generate Patient's Bill
5. Exit
Enter Your Choice: 4
Enter the Patient's ID to display bill: 24
----------------------------------------
Patient's Bill
----------------------------------------
Patient's Room Charge: 600
Patient's Medicine Charge: 500
1. Enter into Doctor's Database
2. Enter into Patient's Database
3. Generate Patient's Report
4. Generate Patient's Bill
5. Exit
Enter Your Choice:
```

Patients Report and bill

10 Conclusion

The implementation of a Hospital Management System using Object-Oriented Programming in C++ has demonstrated significant benefits in managing hospital operations. The OOP approach provides a robust framework for modeling real-world entities and their interactions, leading to an organized and maintainable codebase.

11 References

1. Reed M. Gardner and M. Michael Shabot, “Patient-Monitoring Systems,” in Biomedical Informatics, New York: Springer, new York, NY, 1995, pp. 585–625

2. Hripcsak G, Clayton PD, Pryor TA, Haug P, Wigertz OB, Van der lei J. The Arden Syntax for Medical Logic Modules. Proc 14th SCAMC, IEEE Comp Soc Press, 1990; 200-204.

3. Hripcsak G, Cimino, JJ, Johnson, SB, Clayton PD. The Columbia-Presbyterian Medical Center Decision-Support System as a Model for Implementing the Arden Syntax. Proc 15th SCAMC, McGraw-Hill, 1991; 248-252.

4. McDonald CJ. Action-Oriented Decisions in Ambulatory Medicine. Chicago: Year Book Medical Publishers, 1981.

5. O’Kane KC, McColligan EE. A C++ class library foundation for developing an electronic medical record. Comput Biol Med 1995; 25: 415-23.