A Project Report

on

Hospital Management System

Submitted By:

AVANISH DESHPANDE (20BT04006)

DEVENDRA SHARMA (20BT04045)

VARADA DESHPANDE (20BT04051)

IN

COMPUTER SCIENCE AND ENGINEERING

Guided by:

Dr. Saurabh Shah

Professor and Dean, School of Technology

Academic Year: 2021-22

GSFC University
School of Technology
Computer Science and Engineering

CANDIDATES' DECLARATION

We, the students of Computer Science & Engineering hereby declare that the project report entitled "**Hospital Management System**" is our own work conducted under the supervision of the guide (Dr./Prof.) **Saurabh Shah** for the subject **Object Oriented Programming with Java (BTCS302)** in Semester III, Academic Year 2021-22 .

We further declare that to the best of our knowledge that the report contains the original work of the project carried out as the partial fulfillment of the assignment submission.

(1) Student's Name: AVANISH DESHPANDE **Enrollment No: 20BT04006**

(2) Student's Name: DEVENDRA SHARMA **Enrollment No: 20BT04045**

(3) Student's Name: VARADA DESHPANDE **Enrollment No: 20BT04051**

Date: November 24, 2021

CERTIFICATE

This is to certify that the project entitled **"*Hospital Management System*"** is a bonafied report of the work carried out by ***(1) Avanish Deshpande (20BT04006), (2) Devendra Sharma (20BT04045) and (3) Varada Deshpande (20T04051)*** for **Object Oriented Programming with Java (BTCS302)** subject in Semester III of Computer Science & Engineering under the guidance and supervision of **Dr. Saurabh Shah** for the partial fulfillment of assignment submission.

To the best of my knowledge and belief, this work embodies the work of candidates themselves, have duly completed, fulfills the requirement of assignment submission and is up to the standard in respect of content, presentation and language.

Date: November 23, 2021

Dr Saurabh Shah
Project Guide

GSFC University
School of Technology
Computer Science and Engineering

ACKNOWLEDGEMENT

In the successful accomplishment of this project, many people have bestowed upon us their blessings and heart-pledged support. We would therefore like to take this opportunity to thank all the people who have been concerned with the project.

Firstly, we would like to extend our sincere and heartfelt gratitude to our Object Oriented Programming with Java subject mentor, Dr. Saurabh Shah, who has helped us in this endeavor, and has always been very cooperative. Without his help, coordination, guidance and encouragement, this project couldn't have been what it evolved to be.

We would also like to thank our parents and friends who have helped us with their valuable suggestions and guidance throughout, which has been very helpful in various phases of the completion of this project.

GSFC University
School of Technology
Computer Science and Engineering

PROJECT ABSTRACT/ SYNOPSIS

The purpose of this project entitled "Hospital Management System" is to develop and provide a user-friendly software which is simple, fast, reliable, and cost-effective. The main function of the system is to store the details of doctors and patients, and retrieve them as and when required, and also to manipulate these details meaningfully.

This project on Hospital Management System using Java provides distinct functionalities to different types of users (doctors, patients and admin).

Doctor

A doctor has the options to:-

1. View the scheduled appointments for her/him and the details of her/his patient.

Patient

A patient has the options to:-

1. View the facilities provided by the hospital.
2. View the details of available doctors – their qualification, specialization and contact number.
3. Booking an appointment.
4. Cancelling an appointment.

Admin

The admin holds the rights to manage (add/remove/modify) the details of doctors and hospital facilities; and also to allot a room to a patient requiring hospitalization. This portal is, however, password-protected.

INDEX

INTRODUCTION

Project Summary ……..

This project on Hospital Management System using Java provides distinct functionalities to different types of users (doctors, patients and admin).

Doctor
A doctor has the options to:-

1. View the scheduled appointments for her/him and the details of her/his patient.

Patient
A patient has the options to:-

1. View the facilities provided by the hospital.
2. View the details of available doctors – their qualification, specialization and contact number.
3. Booking an appointment.
4. Cancelling an appointment.

Admin
The admin holds the rights to manage (add/remove/modify) the details of doctors and hospital facilities; and also to allot a room to a patient requiring hospitalization. This portal is, however, password-protected.

Purpose: Goals and Objectives ……

The purpose of this project entitled "Hospital Management System" is to develop and provide a user-friendly software which is simple, fast, reliable, and cost-effective. The main function of the system is to store the details of doctors and patients, and retrieve them as and when required, and also to manipulate these details meaningfully.

Scope ……..

A computerized Hospital Management System has a range of need and benefits of implementation. These include:-

- Efficient management of patient data
- Simultaneous updating of changes made to any data/ item in the entire database.
- Faster as compared to manual systems

The project has a wide scope for implementation in small as well as large hospitals.

Technology and Literature ……..

Many private hospitals maintain their own management systems to efficiently manage and update patient details, staff details, hospital facilities and contact details etc. Some of these are web-based, while the others are "applications".

In case of web-based hospital management systems, although totally cloud-based, internet connectivity may pose problems. In critical situations, this may pose an issue. Hence, a dedicated management system, hosted on the hospital pc itself may be more helpful.

Java provides an easy way to design such a system through the various facilities it provides (such as Object Oriented Programming (OOP), File Handling, the ability to create Graphics User Interface (GUI) etc.). Due to its platform-independency, the Java code would have a better accessibility and usability, with similar user experiences across different platforms.

Hardware and Software Requirements ……..

Since an executable file has not been created for this project currently, the PC on which the code has been downloaded must have the JDK 11 installed on it. The file for the source code requires 18kB space for download. Another 18kB space is taken up by the "class" files that are created on compilation of the source code. This makes up for the minimum space/memory requirement of the project.

As and when data for patient or doctor is added/ removed from the memory, the memory requirements of the Hospital Management System might change. It may increase or decrease, based on the operation performed.

SYSTEM ANALYSIS AND DESIGN

Study of Current System ……..

Many private hospitals maintain their own management systems to efficiently manage and update patient details, staff details, hospital facilities and contact details etc. Some of these are web-based, while the others are “applications”.

Problems and Weaknesses of Current Systems …….

In case of web-based hospital management systems, although totally cloud-based, internet connectivity may pose problems. In critical situations, this may pose an issue. Hence, a dedicated management system, hosted on the hospital pc itself may be more helpful.

Requirements of New System ……..

Java provides an easy way to design such a system through the various facilities it provides (such as Object Oriented Programming (OOP), File Handling, the ability to create Graphics User Interface (GUI) etc.). Due to its platform-independency, the Java code would have a better accessibility and usability, with similar user experiences across different platforms.

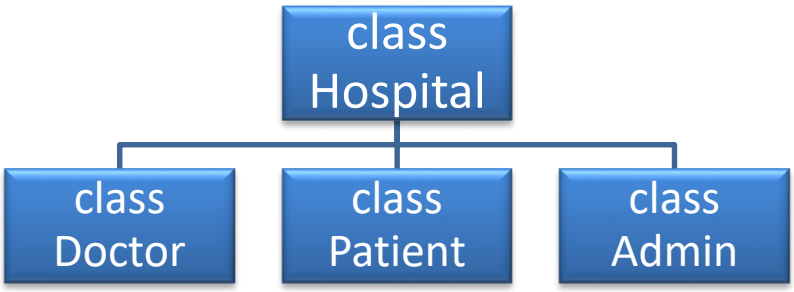
In addition to that, a computerized Hospital Management System has a range of need and benefits of implementation. These include:-

- Efficient management of patient data
- Simultaneous updating of changes made to any data/ item in the entire database.
- Faster as compared to manual systems.

The project has a wide scope for implementation in small as well as large hospitals.

System Design ……...

Hospital Management System has been equipped with 4 classes: Hospital, Patient, Doctor and Admin. These classes make use of concepts such as constructors, inheritance, static and non-static methods. The inheritance structure in the program can be visualized as follows:



The entire system is command-line based, where menus have been created to navigate between different portals smoothly. The scanner class, provided under the 'io' package of Java has been used to take user inputs. Object Oriented Programming Concepts such as classes and inheritance; static data members and methods; constructors; instance methods etc. have been used in the development of the project.

We are successfully able to add doctors to our database correctly, appending a new entry to the same file on each iteration. We are also able to store the details of patient in our database, again appending for each entry. For both, doctor as well as patient, the date and time of each entry is being recorded and stored in the respective databases, with the help of the Calender class provided by java under the 'util' package.

For maintaining the databases, we are making the use of File Handling, storing data in a as live objects using the 'Serializable' interface that has been provided by Java under the 'util' package. This interface must be implemented by the class whose object you want to persist.
The 'Serializable' interface is a marker interface, which means that it has no body, or it contains no methods. Therefore, a class implementing 'Serializable' does not have to implement any specific methods. Implementing this interface just tells the Java serialization classes that its object is intended for object Serialization. In other words, 'Serializable' is just used to "mark" Java classes which support a certain capability. In this code, 'ObjectInutStream' helps in object serialization (writing the data objects to file/database) and 'ObjectOutputStream' helps in object deserialization (reading the data objects from the file).

The command-line is cleared at various instances throughout the program, with the help of the 'ProcessBuilder' class in Java, which is used to create operating system processes.
The readPassword() method helps take user input from the console, without displaying what is being typed.

IMPLEMENTATION AND SCREENSHOTS

Source Code ……...

```
import java.util.*;
import java.io.*;



class Hospital implements Serializable{

    static String MonthsOfYear[] = { "January", "February", "March", "April",
"May", "June", "July", "August", "September", "October", "November", "December"
};

    static String DaysOfWeek[] = { "Sunday", "Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday" };

    static String am_pm[] = { "AM", "PM" };

    static char[] adminPwd = "admin@xyzhospital".toCharArray();


    static void Display() {
        System.out.println("XYZ Hospital");
        System.out.println("Phone Number: 1234567890");
        System.out.println("Email: admin@xyzhospital.com");
        System.out.println("Website: https://www.xyzhospital.org");
    }


    public static void clear() {
        try {
            if (System.getProperty("os.name").contains("Windows"))
                new ProcessBuilder("cmd", "/c",
"cls").inheritIO().start().waitFor();
            else
                Runtime.getRuntime().exec("clear");
        } catch (IOException | InterruptedException ex) {}
    }
```

```java
    public static void WriteObjectToFile(Object serObj, File fileName) {
        try {

            FileOutputStream fileOut = new FileOutputStream(fileName, true);
            ObjectOutputStream objectOut = new ObjectOutputStream(fileOut);
            objectOut.writeObject(serObj);
            objectOut.close();
            System.out.println("The Object  was succesfully written to a file");
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }



    public static void printRecords(File RecordFile) {
        try {
            if (RecordFile.length()!=0) {
                FileInputStream F = new FileInputStream(RecordFile);
                ObjectInputStream input = new ObjectInputStream(F);
                Object obj = input.readObject();


                while(obj!=null) {
                    try {
                        if (obj != null) {

                            if (obj instanceof Doctor) {
                                System.out.println("\n Name" +  "\t\t\t Phone
Number" + "\t\t\t Qualification" + "\t\t\t Specialization");
                                Doctor d = (Doctor) obj;
                                System.out.println('\n' + d.Name +  "\t\t" +
d.PhoneNumber + "\t\t\t" + d.Qualification + "\t\t" + d.Specialization);
                            }

                            else if (obj instanceof Patient) {
                                System.out.println("\n Name" + "\t\t\t Age" +
"\t\t\t Address" + "\t\t\t Phone Number" + "\t\t\t Blood Group" + "\t\t\t Gender"
+ "\t\t\t Doctor Name" + "\t\t\t Appointment Date");
                                Patient p = (Patient) obj;
                                System.out.println('\n' + p.Name + "\t\t" + p.Age
+ "\t\t" + p.Address + "\t\t" + p.PhoneNumber + "\t\t" + p.BloodGroup + "\t\t" +
p.Gender + "\t\t" + p.DocName + "\t\t" + p.AppDate);
                            }
                            obj = input.readObject();
                        }
```

```
                    } catch (Exception e) {
                        break;
                    }
                }
                input.close();
                F.close();
            }
            else
                System.out.println("File Empty.");
        } catch (Exception ex) {
            System.out.println("An error occurred.");
        }
    }


    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int choice = 0;
        do {
            clear();
            Calendar calendar = Calendar.getInstance();
            System.out.print("Date: " +
MonthsOfYear[calendar.get(Calendar.MONTH)] + " " + calendar.get(Calendar.DATE) +
", " + calendar.get(Calendar.YEAR));
            System.out.print("\t\t\t\t\t\t\t\tDay: " +
DaysOfWeek[calendar.get(Calendar.DAY_OF_WEEK)]);
            System.out.print("\t\t\t\t\t\t\t\tTime: " +
calendar.get(Calendar.HOUR) + ":" + calendar.get(Calendar.MINUTE) + ":" +
calendar.get(Calendar.SECOND) + " " + am_pm[calendar.get(Calendar.AM_PM)]);
            System.out.print("\n\nWelcome to ");
            Display();
            System.out.println("\nWhich Portal would you like to access?" + "\n1.
Doctor" + "\n2. Patient" + "\n3. Admin" + "\n4. Exit");
            System.out.print("Enter option number: ");
            choice = sc.nextInt();
            switch(choice) {
                case 1: Doctor d = new Doctor();
                        break;
                case 2: Patient p = new Patient();
                        break;
                case 3: System.out.println("Enter Password: ");
                        Console con = System.console();
                        sc.nextLine();
                        char[] pwd = con.readPassword();
```

```
                        if (!Arrays.equals(pwd,adminPwd)) {
                            System.out.println("\nIncorrect Password. Entry to
admin portal denied!");
                            sc.nextLine();
                            continue;
                        }
                        Admin a = new Admin();
                        break;
                case 4: clear();
                        System.out.print("\nThank you for showing your trust in
");
                        Display();
                        break;
                default: System.out.println("Invalid Choice. Please Re-enter.");
            }
        }while(choice!=4);
    }
}





class Doctor extends Hospital {
    String Name, PhoneNumber, Qualification, Specialization, AddDate, AddTime;
    Doctor() {
        Scanner sc = new Scanner(System.in);
        int choice = 0;
        do {
            clear();
            Calendar calendar = Calendar.getInstance();
            System.out.print("Date: " +
MonthsOfYear[calendar.get(Calendar.MONTH)] + " " + calendar.get(Calendar.DATE) +
", " + calendar.get(Calendar.YEAR));
            System.out.print("\t\t\t\t\t\t\t\tDay: " +
DaysOfWeek[calendar.get(Calendar.DAY_OF_WEEK)]);
            System.out.print("\t\t\t\t\t\t\t\tTime: " +
calendar.get(Calendar.HOUR) + ":" + calendar.get(Calendar.MINUTE) + ":" +
calendar.get(Calendar.SECOND) + " " + am_pm[calendar.get(Calendar.AM_PM)]);
            System.out.print("\n\nWelcome to the Doctor Portal of ");
            Display();
            System.out.println("\nWhat would you like to do?" + "\n1. View
Patient Appointments" + "\n2. Back to Main Menu");
            System.out.print("Enter option number: ");
            choice = sc.nextInt();
```

```
            switch(choice) {
                case 1: clear();
                        ViewAppointments();
                        sc.nextLine();
                        sc.nextLine();
                        break;
                case 2: break;
                default: System.out.println("Invalid Choice. Please Re-enter.");
            }
        }while(choice!=2);
    }



    Doctor (String Name, String PhoneNumber, String Qualification, String
Specialization, String AddDate, String AddTime) {
        this.Name = Name;
        this.PhoneNumber = PhoneNumber;
        this.Qualification = Qualification;
        this.Specialization = Specialization;
        this.AddDate = AddDate;
        this.AddTime = AddTime;
        File F = new File("DoctorRecords.txt");
        WriteObjectToFile(this, F);
    }



    static void ViewAppointments() {
        Display();
        Scanner sc = new Scanner(System.in);
        String nm;
        System.out.print("\n\nEnter your name to view the appointments booked
with you: ");
        nm = sc.nextLine();
        boolean deleted = false;
        try {
            if ("PatientRecords.txt".length()!=0) {
                FileInputStream F = new FileInputStream("PatientRecords.txt");
                ObjectInputStream input = new ObjectInputStream(F);
                Object obj = input.readObject();
                File temp = new File("temp.txt");
```

```
                while(obj!=null) {
                    try {
                        if (obj != null) {
                            if (obj instanceof Patient) {
                                Patient p= (Patient) obj;
                                if (nm.equalsIgnoreCase(p.DocName)) {
                                    p.getDetails();
                                }
                            }
                            obj = input.readObject();
                        }
                    } catch (Exception e) {
                        break;
                    }
                }
                input.close();
                F.close();
            }
            else
                System.out.println("File Empty.");
        } catch (Exception ex) {
            System.out.println("An error occurred.");
        }
    }
}




class Patient extends Hospital {
    String Name, Address, BloodGroup, Gender, PhoneNumber, DocName, AppDate,
BookDate, BookTime;
    int Age;
    Patient() {
        Scanner sc = new Scanner(System.in);
        int choice = 0;
        do {
            clear();
            Calendar calendar = Calendar.getInstance();
            System.out.print("Date: " +
MonthsOfYear[calendar.get(Calendar.MONTH)] + " " + calendar.get(Calendar.DATE) +
", " + calendar.get(Calendar.YEAR));
            System.out.print("\t\t\t\t\t\t\t\tDay: " +
DaysOfWeek[calendar.get(Calendar.DAY_OF_WEEK)]);
```

```
            System.out.print("\t\t\t\t\t\t\t\t\tTime: " +
calendar.get(Calendar.HOUR) + ":" + calendar.get(Calendar.MINUTE) + ":" +
calendar.get(Calendar.SECOND) + " " + am_pm[calendar.get(Calendar.AM_PM)]);
            System.out.print("\nWelcome to the Patient Portal of ");
            Display();

            System.out.println("\nWhat would you like to do?" + "\n1. View
Hospital Facilities" + "\n2. View Doctor List" + "\n3. Book an Appointment" +
"\n4. Back to Main Menu");

            System.out.print("Enter option number: ");
            choice = sc.nextInt();


            switch(choice) {
                case 1: clear();
                        viewFacilities();
                        sc.nextLine();
                        sc.nextLine();
                        break;

                case 2: clear();
                        File f = new File("DoctorRecords.txt");
                        printRecords(f);
                        sc.nextLine();
                        sc.nextLine();
                        break;

                case 3: getDetails();
                        File F = new File("PatientRecords.txt");
                        WriteObjectToFile(this, F);
                        System.out.println("\n\nBooking Successful. Here are the
details we got from you:-");
                        printDetails();
                        sc.nextLine();
                        sc.nextLine();
                        break;

                case 4: break;
                default: System.out.println("Invalid Choice. Please Re-enter.");
            }
        }while(choice!=4);
    }
```

```
    void getDetails() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Patient Name: ");
        Name = sc.nextLine();
        System.out.print("Patient Age: ");
        Age = sc.nextInt();
        System.out.print("Patient Gender: ");
        sc.nextLine();
        Gender = sc.nextLine();
        System.out.print("Patient Address: ");
        Address = sc.nextLine();
        System.out.print("Patient Contact Number: ");
        PhoneNumber = sc.nextLine();
        System.out.println("\n\n");

        if ("PatientRecords.txt".length()==0) {
            do {
                File f = new File("DoctorRecords.txt");
                printRecords(f);
                System.out.print("\n\nDoctor Name: ");
                DocName = sc.nextLine();
                Calendar calendar = Calendar.getInstance();
                BookDate = MonthsOfYear[calendar.get(Calendar.MONTH)] + " " +
calendar.get(Calendar.DATE) + " " + calendar.get(Calendar.YEAR);
                System.out.print("Appointment date eg: " + BookDate + ": ");
                AppDate = sc.nextLine();
                BookTime = calendar.get(Calendar.HOUR) + ":" +
calendar.get(Calendar.MINUTE) + ":" + calendar.get(Calendar.SECOND);
            } while (!checkFreeApp(DocName, AppDate));
        }

        else {
                File f = new File("DoctorRecords.txt");
                printRecords(f);
                System.out.print("\n\nDoctor Name: ");
                DocName = sc.nextLine();
                Calendar calendar = Calendar.getInstance();
                BookDate = MonthsOfYear[calendar.get(Calendar.MONTH)] + " " +
calendar.get(Calendar.DATE) + " " + calendar.get(Calendar.YEAR);
                System.out.print("Appointment date eg: " + BookDate + ": ");
                AppDate = sc.nextLine();
                BookTime = calendar.get(Calendar.HOUR) + ":" +
calendar.get(Calendar.MINUTE) + ":" + calendar.get(Calendar.SECOND);
        }
    }
```

```java
    void printDetails() {
        System.out.println("Patient Name: " + Name);
        System.out.println("Patient Age: " + Age);
        System.out.println("Patient Gender: " + Gender);
        System.out.println("Patient Address: " + Address);
        System.out.println("Patient Contact Number: " + PhoneNumber);
        System.out.println("Book Date: " + BookDate);
        System.out.println("Book Time: " + BookTime);
    }




    public static boolean checkFreeApp(String DocName, String AppDate) {
        int count=0;
        try {
            System.out.println("Doctor Name\t\t Contact Number \t
Qualification\t\tSpecialization");
            Scanner pw = new Scanner(new BufferedReader(new
FileReader("DoctorRecords.txt")));
            int i=0;
            while(pw.hasNext()) {
                String line = pw.nextLine();
                String[] lineparts = line.split("\t", -1);
                if (lineparts[7].equalsIgnoreCase(DocName) &&
lineparts[8].equalsIgnoreCase(AppDate))
                    count = count + 1;
            }
        }   catch (IOException e) {
                    System.out.println("An error occurred.");
                    e.printStackTrace();
                }
        if (count>10) {
            System.out.println("Appointments for " + DocName + " for " + AppDate
+ " is full. Please try again for another doctor/date.");
            return false;
        }
        else {
            System.out.println("Appointment Successful.");
            return true;
        }
    }
```

```
    void viewFacilities() {
        System.out.println("\n Radiology (X-Ray) \n Sonography \n CT Scan \n
Physiotherapy \n ECG \n Ambulance Services \n Laboratory Services");
    }
}




class Admin extends Hospital {
    Admin() {
        Scanner sc = new Scanner(System.in);
        int choice = 0;
        do {
            clear();
            Calendar calendar = Calendar.getInstance();
            System.out.print("Date: " +
MonthsOfYear[calendar.get(Calendar.MONTH)] + " " + calendar.get(Calendar.DATE) +
", " + calendar.get(Calendar.YEAR));
            System.out.print("\t\t\t\t\t\t\t\t\tDay: " +
DaysOfWeek[calendar.get(Calendar.DAY_OF_WEEK)]);
            System.out.print("\t\t\t\t\t\t\t\t\tTime: " +
calendar.get(Calendar.HOUR) + ":" + calendar.get(Calendar.MINUTE) + ":" +
calendar.get(Calendar.SECOND) + " " + am_pm[calendar.get(Calendar.AM_PM)]);
            System.out.print("\n\nWelcome to the Admin Portal of ");
            Display();
            System.out.println("\nWhat would you like to do?" + "\n1. Add to
Doctor List" + "\n2. Delete from Doctor List" + "\n3. Allot Room" + "\n4. View
Room Allotment Details" + "\n5. Back to Main Menu");
            System.out.print("Enter option number: ");
            choice = sc.nextInt();

            switch(choice) {
                case 1: AddDoc();
                        sc.nextLine();
                        sc.nextLine();
                        break;

                case 2: if (DeleteDoc())
                            System.out.println("\nRecord Deleted Successfully");
                        else
                            System.out.println("\nRecord not Found");
                        sc.nextLine();
                        sc.nextLine();
                        break;
```

```
                case 3: AllotRoom();
                        sc.nextLine();
                        sc.nextLine();
                        break;

                case 4: ViewRoomAllotment();
                        sc.nextLine();
                        sc.nextLine();
                        break;

                case 5: break;

                default: System.out.println("Invalid Choice. Please Re-enter.");
            }
        } while(choice!=5);
    }




    static void AddDoc() {
        Scanner sc = new Scanner(System.in);
        String Name, PhoneNumber, Qualification, Specialization, AddDate,
AddTime;
        System.out.print("Doctor Name: ");
        Name = sc.nextLine();
        System.out.print("Doctor Contact Number: ");
        PhoneNumber = sc.nextLine();
        System.out.print("Doctor Qualification: ");
        Qualification = sc.nextLine();
        System.out.print("Doctor Specialization: ");
        Specialization = sc.nextLine();
        Calendar calendar = Calendar.getInstance();
        AddDate = MonthsOfYear[calendar.get(Calendar.MONTH)] + " " +
calendar.get(Calendar.DATE) + " " + calendar.get(Calendar.YEAR);
        AddTime = calendar.get(Calendar.HOUR) + ":" +
calendar.get(Calendar.MINUTE) + ":" + calendar.get(Calendar.SECOND);
        Doctor d = new Doctor(Name, PhoneNumber, Qualification, Specialization,
AddDate, AddTime);
    }
```

```
    static boolean DeleteDoc() {
        Scanner sc = new Scanner(System.in);
        String nm;
        System.out.print("Enter doctor name to delete her/his details from the
list: ");
        nm = sc.nextLine();
        boolean deleted = false;

        try {
            if ("DoctorRecords.txt".length()!=0) {
                FileInputStream F = new FileInputStream("DoctorRecords.txt");
                ObjectInputStream input = new ObjectInputStream(F);
                Object obj = input.readObject();
                File temp = new File("temp.txt");
                while(obj!=null) {
                    try {
                        if (obj != null) {
                            if (obj instanceof Doctor) {
                                Doctor d = (Doctor) obj;
                                if (nm.equalsIgnoreCase(d.Name)) {
                                    deleted = true;
                                }
                                else {
                                    WriteObjectToFile(d, temp);
                                }
                            }
                            obj = input.readObject();
                        }
                    } catch (Exception e) {
                        break;
                    }
                }
                input.close();
                F.close();
                File oldFile = new File("DoctorRecords.txt");
                File newFile = new File("temp.txt");
                oldFile.delete();
                newFile.renameTo(oldFile);
            }
            else
                System.out.println("File Empty.");
        } catch (Exception ex) {
            System.out.println("An error occurred.");
        }
        return deleted;
    }
```

```java
    static void AllotRoom() {
        Scanner sc = new Scanner(System.in);
        String  PatientName, PatientId, RoomNo, TypeOfRoom="";
        int ServiceCharges = 0;
        System.out.print("\nPatient Name: ");
        PatientName = sc.nextLine();
        System.out.print("Patient id: ");
        PatientId = sc.nextLine();
        System.out.print("Room no: ");
        RoomNo = sc.nextLine();
        int RoomType = 0;


        do {
            System.out.print("\nSelect your Room Type:- \n1. General Ward \n2.
Semi-ward \n3. Deluxe Room \nEnter option number: ");
            RoomType = sc.nextInt();

            switch(RoomType){
                case 1:
                    System.out.println("You have alloted General Ward.");
                    ServiceCharges = 500;
                    TypeOfRoom = "General Ward";
                    break;

                case 2:
                    System.out.println("You have alloted Semi-ward.");
                    ServiceCharges = 1000;
                    TypeOfRoom = "Semi-ward";
                    break;

                case 3:
                    System.out.println("You have alloted Deluxe room.");
                    ServiceCharges = 1500;
                    TypeOfRoom = "Deluxe Room";
                    break;

                default:
                    System.out.println("Invalid option. Please re-enter!");
            }
        } while(RoomType<1 || RoomType>3);
```

```
        FileWriter fw = null;
        BufferedWriter bw = null;
        PrintWriter pw = null;
        try {
            fw = new FileWriter("AllotedRooms.txt", true);
            bw = new BufferedWriter(fw);
            pw = new PrintWriter(bw);
            pw.write("\n" + PatientName + "\t" + PatientId + "\t" + RoomNo + "\t"
+ TypeOfRoom + "\t" + ServiceCharges);
            pw.close();
        }   catch (IOException e) {
                    System.out.println("An error occurred.");
            }
    }




    static void ViewRoomAllotment() {
        if("AllotedRooms.txt".length()>0) {
            System.out.println("\n\nPatient Name \t\t Patient Id \t Room Number
\t\t TypeOfRoom \t\t Service Charges");
            try {
                Scanner pw = new Scanner(new BufferedReader(new
FileReader("AllotedRooms.txt")));
                int i;
                while(pw.hasNext()) {
                    String line = pw.nextLine();
                    String[] lineparts = line.split("\t", -1);
                    i=0;
                    for(String data : lineparts)
                        if (i<5) {
                            System.out.print(data + "\t\t");
                            i++;
                        }
                    System.out.println();
                }
            }   catch (IOException e) {
                        System.out.println("File Empty.");
                    }
        }
    }
}
```

Screenshots ……

```
C:\Windows\System32\cmd.exe - java  Hospital
Date: November 24, 2021                    Day: Thursday                    Time: 11:45:46 PM
Welcome to the Patient Portal of XYZ Hospital
Phone Number: 1234567890
Email: admin@xyzhospital.com
Website: https://www.xyzhospital.org

What would you like to do?
1. View Hospital Facilities
2. View Doctor List
3. Book an Appointment
4. Back to Main Menu
Enter option number:
```

```
C:\Windows\System32\cmd.exe - java  Hospital
Date: November 24, 2021                    Day: Thursday                    Time: 11:46:50 PM

Welcome to XYZ Hospital
Phone Number: 1234567890
Email: admin@xyzhospital.com
Website: https://www.xyzhospital.org

Which Portal would you like to access?
1. Doctor
2. Patient
3. Admin
4. Exit
Enter option number: 3
Enter Password:


Incorrect Password. Entry to admin portal denied!
```

```
C:\Windows\System32\cmd.exe - java  Hospital
Date: November 24, 2021                                    Day: Thursday                                    Time: 11:47:24 PM

Welcome to the Admin Portal of XYZ Hospital
Phone Number: 1234567890
Email: admin@xyzhospital.com
Website: https://www.xyzhospital.org

What would you like to do?
1. Add to Doctor List
2. Delete from Doctor List
3. Allot Room
4. View Room Allotment Details
5. Back to Main Menu
Enter option number:
```

```
C:\Windows\System32\cmd.exe - java  Hospital
Date: November 24, 2021                                    Day: Thursday                                    Time: 11:47:24 PM

Welcome to the Admin Portal of XYZ Hospital
Phone Number: 1234567890
Email: admin@xyzhospital.com
Website: https://www.xyzhospital.org

What would you like to do?
1. Add to Doctor List
2. Delete from Doctor List
3. Allot Room
4. View Room Allotment Details
5. Back to Main Menu
Enter option number: 1
Doctor Name: Dr. Umesh Panchal
Doctor Contact Number: 9875934673
Doctor Qualification: M.D., D.C.H.
Doctor Specialization: Critical Care Specialist
The Object  was succesfully written to a file
```

```
C:\Windows\System32\cmd.exe - java  Hospital
Date: November 24, 2021                                    Day: Thursday                                    Time: 11:48:47 PM

Welcome to the Admin Portal of XYZ Hospital
Phone Number: 1234567890
Email: admin@xyzhospital.com
Website: https://www.xyzhospital.org

What would you like to do?
1. Add to Doctor List
2. Delete from Doctor List
3. Allot Room
4. View Room Allotment Details
5. Back to Main Menu
Enter option number: 4


Patient Name            Patient Id      Room Number            TypeOfRoom             Service Charges

Varada Deshpande               123            12             General Ward           500
Avanish Deshpande              345            23             Semi-ward              1000
Devendra Sharma         456            54             Deluxe Room            1500
Ajay Sharma             109            45             Deluxe Room            1500
Saurabh shah            5              2              Deluxe Room            1500
Tarun sinha             33             5              Semi-ward              1000
Bhuvam bam              34             5              Semi-ward              1000
Shubham Patel           8328           21             Semi-ward              1000
```

```
C:\Windows\System32\cmd.exe - java  Hospital
Date: November 24, 2021                                    Day: Thursday                                    Time: 11:49:20 PM
Welcome to the Patient Portal of XYZ Hospital
Phone Number: 1234567890
Email: admin@xyzhospital.com
Website: https://www.xyzhospital.org

What would you like to do?
1. View Hospital Facilities
2. View Doctor List
3. Book an Appointment
4. Back to Main Menu
Enter option number: 3
Patient Name: Umeshkumar Yadav
Patient Age: 43
Patient Gender: Male
Patient Address: B-23, Indiranagar Society, Lucknow, Uttar Pradesh
Patient Contact Number: 7382974673



 Name                   Phone Number                   Qualification                  Specialization

Dr. Alka Soni           0265 1200 8000                 M.D., D.C.H.           Pediatrician, Critical Care Specialist


Doctor Name: Dr. Alka Soni
Appointment date eg: November 24 2021: November 26 2021
The Object  was succesfully written to a file


Booking Successful. Here are the details we got from you:-
Patient Name: Umeshkumar Yadav
Patient Age: 43
Patient Gender: Male
Patient Address: B-23, Indiranagar Society, Lucknow, Uttar Pradesh
Patient Contact Number: 7382974673
Book Date: November 24 2021
Book Time: 11:50:29
```

LIMITATIONS AND FUTURE ENHANCEMENTS

The Hospital Management System that has been created is unable to validate data at its current stage. In addition to that, the interface that has been provided seems to be monotonous and somewhat boring for the user. This is where the scope for future enhancements for this project lies.

The future updates for this project can contain data validation for user inputs. This would ensure that we are getting a valid input from the user. The system can also make use of GUI (Graphics User Interface) in order to make the system more attractive, accessible and easy to use.

As an additional functionality, we can also extend the project by syncing/attaching it to web portals as well. This way, a patient would not have to search elsewhere or call up or go to the hospital to book an appointment. She/he can do it easily within the comfort of her/his home over the internet as well.

CONCLUSION

Hospital Management System is a broad concept which can easily be implemented through Java. Although the project has its own limitations at this stage of development, it can definitely be enhanced to achieve the motto it has been developed for: to provide an easy, user-friendly way to manage hospital data.

REFERENCES

https://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html

https://stackoverflow.com/questions/27409718/java-reading-multiple-objects-from-a-file-as-they-were-in-an-array

https://stackoverflow.com/questions/17293991/how-to-write-and-read-java-serialized-objects-into-a-file

https://stackoverflow.com/questions/2979383/how-to-clear-the-console

https://stackoverflow.com/questions/1625234/how-to-append-text-to-an-existing-file-in-java