An Industrial Oriented Project On

HOSPITAL MANAGEMENT SYSTEM

Submitted In partial fulfIlment of the Requirements for the award of the Diploma of Technology

In

COMPUTER ENGINEERING

By

MANOJ BABU KOKKIRIGADDA (22030-CM-138)

SUJITH KUMAR MADASU (22030-CM-164)

SHANMUKH REDDY DALLI (22030-CM-075)

Under the Esteemed Guidance of

Mr. RAMA CHANDRA RAO POLAMARASETTI

IT Consultant & Chief Trainer

A.A.N.M.&V.V.R.S.R. POLYTECHNIC

Seshadri Rao Knowledge Village, Gudlavalleru, Krishna-521356, Andhra Pradesh

2022-2025

CERTIFICATE

This is to certify that is a Bonafde record of project work Entitled

HOSPITAL MANAGEMENT SYSTEM

Carried out by MANOJ BABU KOKKIRIGADDA (22030-CM-138), SUJITH KUMAR MADASU (22030-CM-164), SHANMUKH REDDY DALLI (22030-CM-075) during academic year 2024-2025, in partial fulfllment of the requirement of the award of the Diploma of Technology in Engineering ofered by A.A.N.M.&V.V.R.S.R. POLYTECHNIC

Gudlavalleru, Krishna-521356, Andhra Pradesh

Project Guide

MR. RAMA CHANDRA RAO POLAMARASETTI

HOD-CME

Mr. ANNE KRISHNA CHAITANYA

External Examiner

A.A.N.M.&V.V.R.S.R. POLYTECHNIC

Seshadri Rao Knowledge Village, Gudlavalleru, Krishna-521356, Andhra Pradesh

DEPARTMENT OF COMPUTER ENGINEERING

2022-2025

DECLARATION

We here by declare that the document entitled **HOSPITAL MANAGEMENT SYSTEM**

Submitted to the A.A.N.M.&V.V.R.S.R. Polytechnic in partial fulfllment of requirements for the award of the Diploma of Technology in Computer Engineering is a record of an original work done by us under the guidance of Mr. Rama Chandra Rao Polamarasetti and this document has not been submitted to any other university for the award of any other degree.

MANOJ BABU KOKKIRIGADDA

(22030-CM-138)

SUJITH KUMAR MADASU

(22030-CM-164)

SHANMUKH REDDY DALLI

(22030-CM-075)

ACKNOWLEDGEMENT

On the submission of our project entitled “HOSPITAL MANAGEMENT SYSTEM” Manage patients, doctors, and appointments,Concepts: OOPs

Using Java

we would like to extend our gratitude and Sincere Thank to our guide Mr. RAMA CHANDRA RAO POLAMARASETTI, IT Consultant & Chief Trainer in Melmaa Tech Pvt Ltd. for their valuable timely suggestions.

We would also like to extend our sincere thanks to Mr. A. KRISHNA CHAITANYA, Head of the Department of Computer Engineering for his valuable suggestions and motivating guidance during our project .

We would like to thank our Director Mr. SHAIK CHARUK for giving us the opportunity to carry out our project work.

We endow our sincere thanks to Principal Dr. N. RAJASHEKHAR for his consistent
cooperation and encouragement.

We are very thankful to our teachers for providing the required background during the project work. We would also like to extend our gratitude to our friends and those who are directly or indirectly helped us in completing our project work.

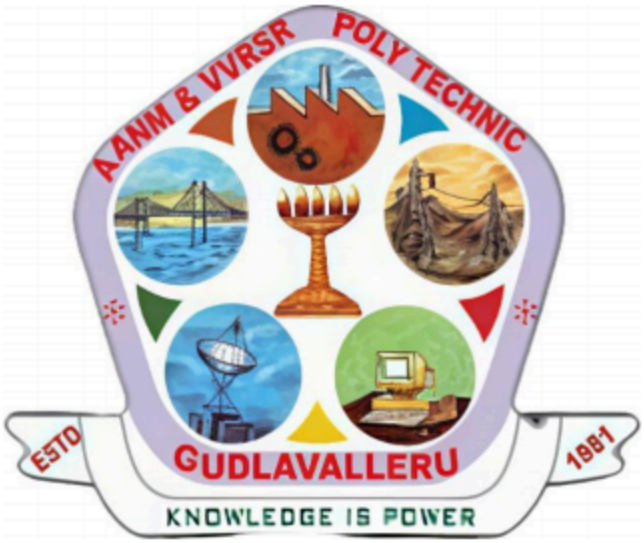
MANOJ BABU KOKKIRIGADDA (22030-CM-138)

SUJITH KUMAR MADASU (22030-CM-164)

SHANMUKH REDDY DALLI (22030-CM-075)

DEPARTMENT OF COMPUTER ENGINEERING

Vision and Mission of the Institute



Vision

To admit and groom students from rural background and be a truly technical institution, benefting society and nation as a whole institute.

Mission

The mission of the institute is to create, deliver and refne knowledge. Being a rural technical institute, we aim to:

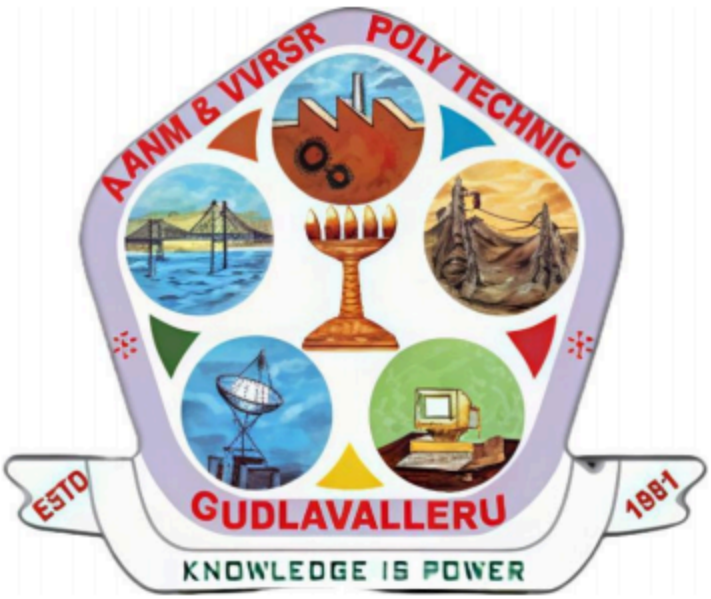
1. Enhance our position to one of the best technical institutions and to measure our performance against the highest defned standards.

2. Provide highest quality learning environment to our students for their greater wellbeing so as to equip them with highest technical and professional ethics.

3.Produce engineering graduates fully equipped to meet the ever-growing needs of industry and society

DEPARTMENT OF COMPUTER ENGINEERING

Department Vision and Mission



Vision

To be a center of eminence to mould young, fresh minds into challenging computer science professionals with ethical values.

Mission

1. Enrich the knowledge and wisdom with repository of books and modernized laboratory aided by dedicated resources.
2. Organize training and activities on upcoming techniques, and inter-personal skills.
3. Develop the ability to provide sustainable solutions to real world situations with collaboration.

Hospital Management System Documentation

Abstract

The Hospital Management System (HMS) is a Console User Interface (CUI)-based Java application developed to streamline hospital administration by efciently managing patients, doctors, and appointments. Utilizing Object-Oriented Programming (OOP) principles, the system ensures a structured, interactive, and user-friendly console-based experience.

This application simplifes hospital workfows by ofering efcient record management, reducing manual efort, and enhancing accessibility. With its CUI-based approach, the system provides a seamless and organized way to handle hospital operations, ensuring improved efciency healthcare management.

***Key Features:

- Patient Management: Add, update, and view patient details.
- Doctor Management: Maintain doctor records, specialties, and availability.
- Appointment Scheduling: Book, modify, and cancel appointments.
- Search Functionality: Easily retrieve patient and doctor details.
- Authentication: Secure access for hospital staf.

Introduction

The **Hospital Management System (HMS)** is developed to streamline hospital operations by automating administrative tasks. Traditionally, hospitals rely on manual record-keeping, which is prone to errors, inefficiencies, and data loss. This system introduces a structured data management approach using **file handling techniques** in Java, ensuring secure and organized storage of **doctor, patient, and appointment records**.

By implementing **HMS**, hospitals can efficiently manage patient registrations, doctor specializations, appointment scheduling, and cancellations. The system reduces paperwork, minimizes errors, and provides **quick access to critical information**, improving overall **workflow efficiency and patient care**.

Literature Survey

Hospital Management Systems (HMS) are widely used in healthcare facilities to streamline administrative tasks, improve patient care, and enhance operational efciency. Most existing HMS solutions are database-driven, relying on SQL or NoSQL databases for storing and managing records. These systems require complex setup processes, dedicated servers, and database management knowledge, making them resource-intensive for small healthcare facilities.

***Comparison with Existing Systems:

Feature	Existing HMS (Database-Based)	Our HMS (File-Based)
Storage	Relational/NoSQL databases	File handling (text-based)
Complexity	Requires setup, maintenance	Simple and lightweight
Performance	High for large-scale usage	Efficient for small/mid-sized hospitals
Implementation	Needs database expertise	Core Java-based, easy to implement
Cost	Requires dedicated resources	Cost-effective

***Unique Features of Our HMS:

- File-Based Storage: Unlike traditional HMS that rely on databases, our system leverages fle handling in Java, reducing dependencies and making it easier to deploy.
- Lightweight & Portable: The system requires minimal resources, making it ideal for small and medium-sized hospitals.
- User-Friendly & Secure: The structured data management ensures easy retrieval, modifcation, and security of hospital records.

System Analysis

The system is analyzed based on the requirements of a hospital, focusing on key functionalities such as doctor and patient management, appointment scheduling, and data persistence using fle handling.

***Problem Statement: Manual hospital record management is time-consuming and error-prone.

***Objectives:

- Efciently manage doctor and patient records.
- Schedule and cancel appointments.
- Store records persistently in text fles.
- Provide a simple and interactive command-line interface.

***Scope:

- Manage doctors and patients.
- Facilitate appointment scheduling.
- Maintain hospital records.

System Design

***Architecture: A command-line interface with fle-based storage.

The system is structured with three primary components:1. Doctors - Manages doctor details such as name, specialization, and degree.2. Patients - Maintains patient information including age, gender, and disease.3. Appointments - Handles scheduling and cancellation of doctor-patient appointments.

***Modules:

- Doctor Management: Add and display doctors.
- Patient Management: Add and display patients.
- Appointment Management: Schedule and cancel appointments.
- File Handling: Save and retrieve data from text fles.

Hardware and Software Requirements

***__Hardware__: Processor: Intel Core i3 or higher- RAM: 4GB minimum- Storage: 500MB free disk space

***__Software__: Operating System: Windows/Linux/MacOS- Java Development Kit (JDK) 8 or higher- Text Editor or IDE (Eclipse, IntelliJ IDEA, VS Code)

Coding Templates

The project follows a modular coding approach, using separate classes for doctors, patients, and appointments. It utilizes fle handling to persist data and implements validation for input consistency.

- Doctor Class: Stores and manages doctor data.
- Patient Class: Handles patient records.
- Appointment Class: Manages appointment scheduling.
- Hospital_Management Class: Implements the core logic of the system.
- Main Class: Provides the user interface and menu system.

Testing

The system was tested using unit testing and manual validation techniques. Test cases included doctor and patient addition, appointment scheduling, and fle storage verifcation. The system performed successfully under multiple test scenarios.

- Unit Testing: Testing individual modules (Doctor, Patient, Appointment).
- Integration Testing: Ensuring seamless interaction between modules.
- Validation Testing: Checking for correct user inputs.
- File Handling Testing: Verifying data persistence in text fles.

Source Code

```
import java.util.*;
import java.io.*;
class Doctors
{
String Dr_Name;
String Specialization;
String Degree;
int Dr_Id;
Doctors(String Dr_Name, String Specialization, String Degree, int Dr_Id)
{
this.Dr_Name = Dr_Name;
this.Specialization = Specialization;
this.Degree = Degree;
this.Dr_Id = Dr_Id;
}
void display()
{
System.out.println("Doctor ID: "+Dr_Id);
System.out.println("Name: Dr. "+Dr_Name);
System.out.println("Specialization:"+Specialization);
System.out.println("Degree: "+Degree);
System.out.println("------------------------------");
}
String toFileString(int sno)
{
return sno + ".\n" +
"Doctor ID       : " + Dr_Id + "\n" +
```

```
"Name          : Dr. " + Dr_Name + "\n" +
"Specialization : " + Specialization + "\n" +
"Degree        : " + Degree + "\n" +
"-------------------------";
}
}
class Patients
{
String Patient_Name;
int age;
char Gender;
String Disease;
int Patient_Id;
Patients(String Patient_Name, int age, char Gender, String Disease, int Patient_Id)
{
this.Patient_Name = Patient_Name;
this.age = age;
this.Gender = Gender;
this.Disease = Disease;
this.Patient_Id = Patient_Id;
}
void display()
{
System.out.println("Patient ID: "+Patient_Id);
System.out.println("Name: "+Patient_Name);
System.out.println("Age: " +age);
System.out.println("Gender: "+Gender);
System.out.println("Disease: "+ Disease);
```

```
System.out.println("------------------------------");
}
String toFileString(int sno)
{
return sno + ".\n" +
"Patient ID : " + Patient_Id + "\n" +
"Name       : " + Patient_Name + "\n" +
"Age        : " + age + "\n" +
"Gender     : " + Gender + "\n" +
"Disease    : " + Disease + "\n" +
"-------------------------";
}
}
class Appointment
{
int Appointment_id;
int Patient_id;
int Doctor_id;
String Date;
String Time;
Appointment(int Appointment_id, int Patient_id, int Doctor_id, String Date, String Time)
{
this.Appointment_id = Appointment_id;
this.Patient_id = Patient_id;
this.Doctor_id = Doctor_id;
this.Date = Date;
this.Time = Time;
}
```

```
void display()
{
System.out.println("Appointment ID: "+ Appointment_id);
System.out.println("Patient ID: "+Patient_id);
System.out.println("Doctor ID: " +Doctor_id);
System.out.println("Date: "+ Date);
System.out.println("Time: "+ Time);
System.out.println("------------------------------
"); }
String toFileString(int sno)
{
return sno + ".\n" +
"Appointment ID : " + Appointment_id + "\n" +
"Patient ID     : " + Patient_id + "\n" +
"Doctor ID      : " + Doctor_id + "\n" +
"Date           : " + Date + "\n" +
"Time           : " + Time + "\n" +
"-------------------------";
}
}
class Hospital_Management
{
ArrayList<Doctors> doctorList = new ArrayList<>();
ArrayList<Patients> patientList = new ArrayList<>();
ArrayList<Appointment> appointmentList = new ArrayList<>();
Hospital_Management()
{
createFile("doctors.txt");
```

```
createFile("patients.txt");
createFile("appointments.txt");
}
void createFile(String fleName)
{
try
{
File fle = new File(fleName);
if (!fle.exists())
{
fle.createNewFile();
//System.out.println(fleName + " created successfully.");
}
}
catch (IOException e)
{
System.out.println("Error creating fle: " + fleName);
}
}
Scanner sc = new Scanner(System.in);
void addDoctor()
{
int Dr_Id;
while (true)
{
System.out.print("Enter Doctor ID: ");
Dr_Id = sc.nextInt();
sc.nextLine();
```

```
boolean exists = false;
for (Doctors d : doctorList)
{
if (d.Dr_Id == Dr_Id)
{
System.out.println("Error: Doctor ID already exists. Please enter a unique ID.");
exists = true;
break;
}
}
if (!exists) break;
}
String Dr_Name;
while(true)
{
System.out.print("Enter Doctor Name: ");
Dr_Name = sc.nextLine();
if (Dr_Name.matches("[a-zA-Z ]+"))
{
break;
}
System.out.println("Not a valid name! Please enter Alphabets only");
}
System.out.print("Enter Specialization: ");
String Specialization = sc.nextLine();
System.out.print("Enter Degree: "); String
Degree = sc.nextLine();
doctorList.add(new Doctors(Dr_Name, Specialization, Degree, Dr_Id));
```

```
saveDoctors();
System.out.println("Doctor added successfully.");
System.out.println("------------------------------");
}
void addPatient()
{
int Patient_Id;
while (true)
{
System.out.print("Enter Patient ID: ");
Patient_Id = sc.nextInt();
sc.nextLine();
boolean exists = false;
for (Patients p : patientList)
{
if (p.Patient_Id == Patient_Id)
{
System.out.println("Error: Patient ID already exists. Please enter a unique ID.");
exists = true;
break;
}
}
if (!exists) break;
}
String Patient_Name;
while (true)
{
System.out.print("Enter Patient Name: ");
```

```
Patient_Name = sc.nextLine();
if (Patient_Name.matches("[a-zA-Z ]+"))
{
break;
}
System.out.println("Not a valid name! Please enter Alphabets only.");
}
int age;
while (true)
{
System.out.print("Enter Age: ");
if (sc.hasNextInt())
{
age = sc.nextInt();
sc.nextLine();
break;
}
else
{
System.out.println("Error: Please enter a valid numeric age.");
sc.next();
}
}
char Gender;
while (true)
{
System.out.print("Enter Gender (M/F): ");
String genderInput = sc.next().toUpperCase();
```

```
if (genderInput.equals("M") || genderInput.equals("F"))
{
Gender = genderInput.charAt(0);
break;
}
else
{
System.out.println("Error: Please enter 'M' for Male or 'F' for Female.");
}
}
sc.nextLine();
System.out.print("Enter Disease: ");
String Disease = sc.nextLine();
patientList.add(new Patients(Patient_Name, age, Gender, Disease,
Patient_Id)); savePatients();
System.out.println("Patient added
successfully."); System.out.println("----------------
----------"); }
void Schedule_Appointment()
{
int Appointment_id;
while (true)
{
System.out.print("Enter Appointment ID: ");
Appointment_id = sc.nextInt();
sc.nextLine();
boolean exists = false;
for (Appointment a : appointmentList)
```

```
{
if (a.Appointment_id == Appointment_id)
{
System.out.println("Error: Patient ID already exists. Please enter a unique ID.");
exists = true;
break;
}
}
if (!exists) break;
}
System.out.print("Enter Patient ID: ");
int Patient_id = sc.nextInt();
System.out.print("Enter Doctor ID: ");
int Doctor_id = sc.nextInt();
sc.nextLine();
System.out.print("Enter Date: ");
String Date = sc.nextLine();
System.out.print("Enter Time: ");
String Time = sc.nextLine();
appointmentList.add(new Appointment(Appointment_id, Patient_id, Doctor_id,
Date, Time));
saveAppointments();
System.out.println("Appointment scheduled
successfully."); System.out.println("----------------------------
-"); }
void Cancel_Appointment()
{
System.out.print("Enter Appointment ID to Cancel: ");
```

```
int id = sc.nextInt();
boolean removed = appointmentList.removeIf(a -> a.Appointment_id ==
id); if (removed)
System.out.println("Appointment cancelled
successfully."); else
System.out.println("Appointment ID not found.");
}
void Disp_allDoctors()
{
System.out.println("Doctors List:");
System.out.println("-------------------------------");
for (Doctors d : doctorList)
d.display();
}
void Disp_allPatients()
{
System.out.println("Patients List:");
System.out.println("-------------------------------");
for (Patients p : patientList)
p.display();
}
void Disp_allAppointments()
{
if (appointmentList.isEmpty())
{
System.out.println("No appointments found.");
return;
}
```

```
System.out.println("Appointments List:");
System.out.println("------------------------------
"); for (Appointment a : appointmentList)
a.display();
}
void saveDoctors()
{
try (BuferedWriter writer = new BuferedWriter(new FileWriter("doctors.txt")))
{
int sno = 1;
for (Doctors d : doctorList)
{
writer.write(d.toFileString(sno++) + "\n");
}
}
catch (IOException e)
{
System.out.println("Error saving doctors.");
}
}
void savePatients()
{
try (BuferedWriter writer = new BuferedWriter(new FileWriter("patients.txt")))
{
int sno = 1;
for (Patients p : patientList)
{
writer.write(p.toFileString(sno++) + "\n");
```

```
}
}
catch (IOException e)
{
System.out.println("Error saving patients.");
}
}
void saveAppointments()
{
try (BuferedWriter writer = new BuferedWriter(new FileWriter("appointments.txt")))
{
int sno = 1;
for (Appointment a : appointmentList)
{
writer.write(a.toFileString(sno++) + "\n");
}
}
catch (IOException e)
{
System.out.println("Error saving appointments.");
}
}
}
class Main
{
public static void main(String...args)
{
Hospital_Management ob = new Hospital_Management();
```

```
Scanner sc = new Scanner(System.in);
String D_Username = "Batch 04";
String D_Password = "Batch 04";
int attempts = 3;
while (attempts > 0)
{
System.out.print("Enter Username: ");
String username = sc.nextLine();
System.out.print("Enter Password: ");
String password = sc.nextLine();
if (username.equals(D_Username) && password.equals(D_Password))
{
System.out.println("Login Successful! Access
Granted"); System.out.println("------------------------------
"); break;
}
else
{
attempts--;
System.out.println("Invalid Credentials! " + attempts+ " Attempts left" );
if (attempts == 0)
{
System.out.println("Too many failed attempts! Exiting...");
return;
}
}
}
ob.doctorList.add(new Doctors("Sujith", "Cardiologist", "MBBS", 101));
```

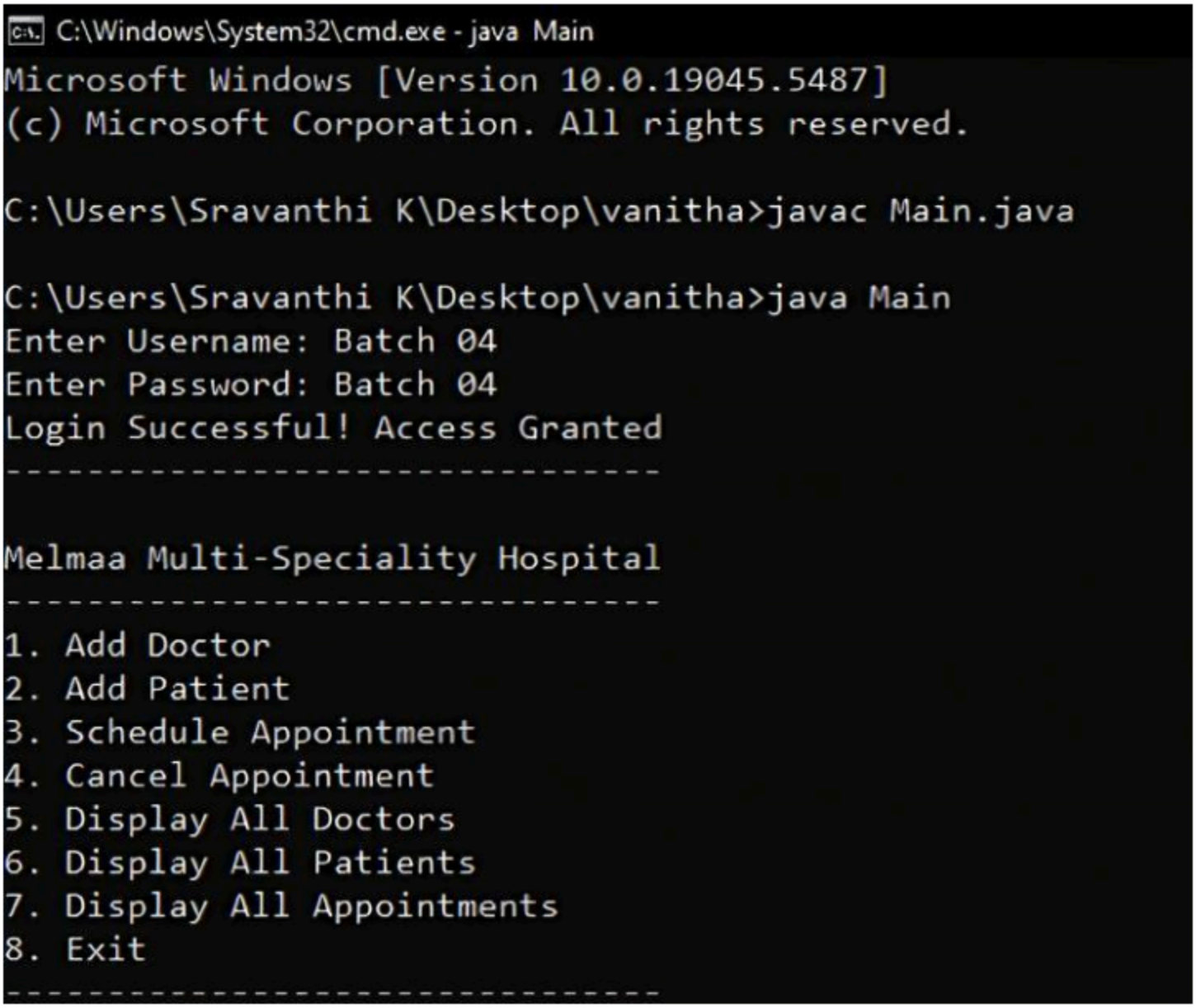
```
ob.doctorList.add(new Doctors("Shanmukh", "Neurologist", "MBBS",
102)); ob.patientList.add(new Patients("Uma", 16, 'F', "Fever", 201));
ob.patientList.add(new Patients("Manoj", 19, 'M', "Migraine", 202));
ob.appointmentList.add(new
Appointment(301, 201, 101, "2025-03-10", "1:00
pm")); ob.appointmentList.add(new
Appointment(302, 202, 102, "2025-03-12", "3:30 pm"));
ob.saveDoctors();
ob.savePatients();
ob.saveAppointments();
int choice;
do
{
System.out.println("\nMelmaa Multi-Speciality
Hospital"); System.out.println("------------------------------
"); System.out.println("1. Add Doctor");
System.out.println("2. Add Patient");
System.out.println("3. Schedule Appointment");
System.out.println("4. Cancel Appointment");
System.out.println("5. Display All Doctors");
System.out.println("6. Display All Patients");
System.out.println("7. Display All Appointments");
System.out.println("8. Exit");
System.out.println("------------------------------");
System.out.print("Enter your choice: ");
choice = sc.nextInt();
switch (choice)
{
```

```
case 1: ob.addDoctor();
break;
case 2: ob.addPatient();
break;
case 3: ob.Schedule_Appointment();
break;
case 4: ob.Cancel_Appointment();
break;
case 5: ob.Disp_allDoctors();
break;
case 6: ob.Disp_allPatients();
break;
case 7: ob.Disp_allAppointments();
break;
case 8: System.out.println("Exiting...");
System.out.println("\nThank You...");
break;
default: System.out.println("Invalid choice");
}
}while (choice != 8);
}
}
```

Output Screens

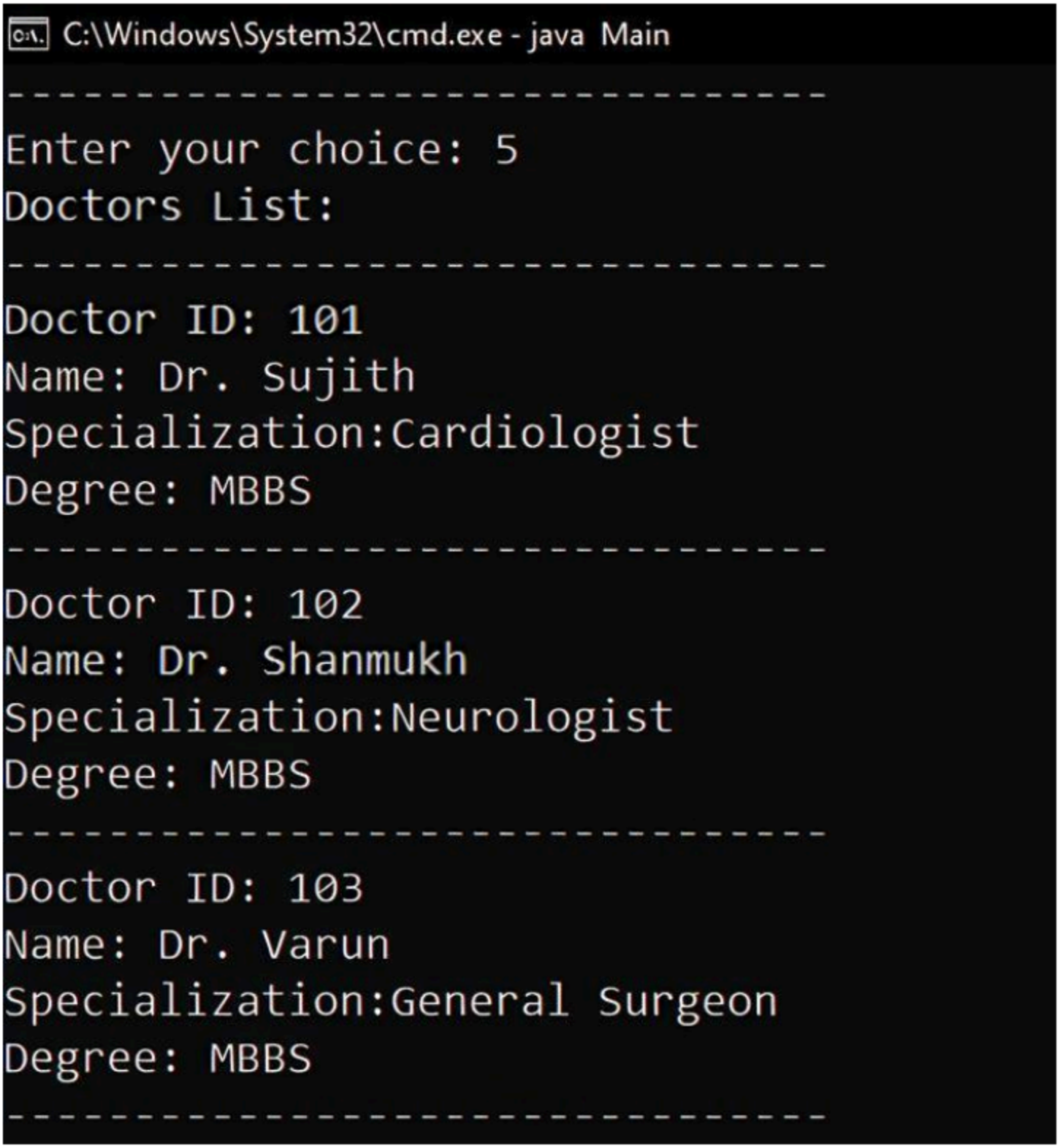
The HMS operates through a command-line interface with user-friendly prompts and formatted outputs. Screens include:- Doctor registration- Patient record management- Appointment scheduling and cancellation- Displaying records of doctors, patients, and appointments

***Login Screen

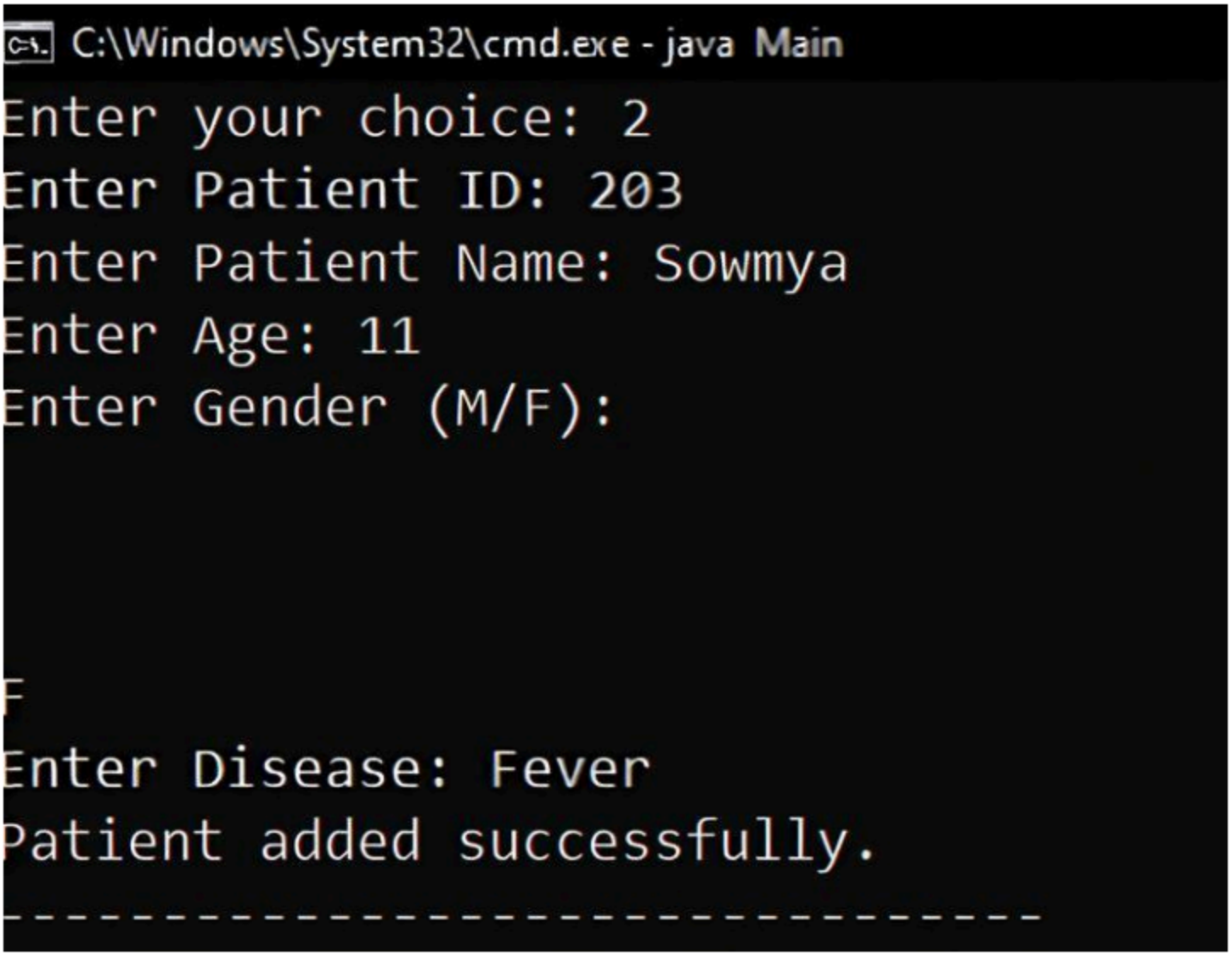


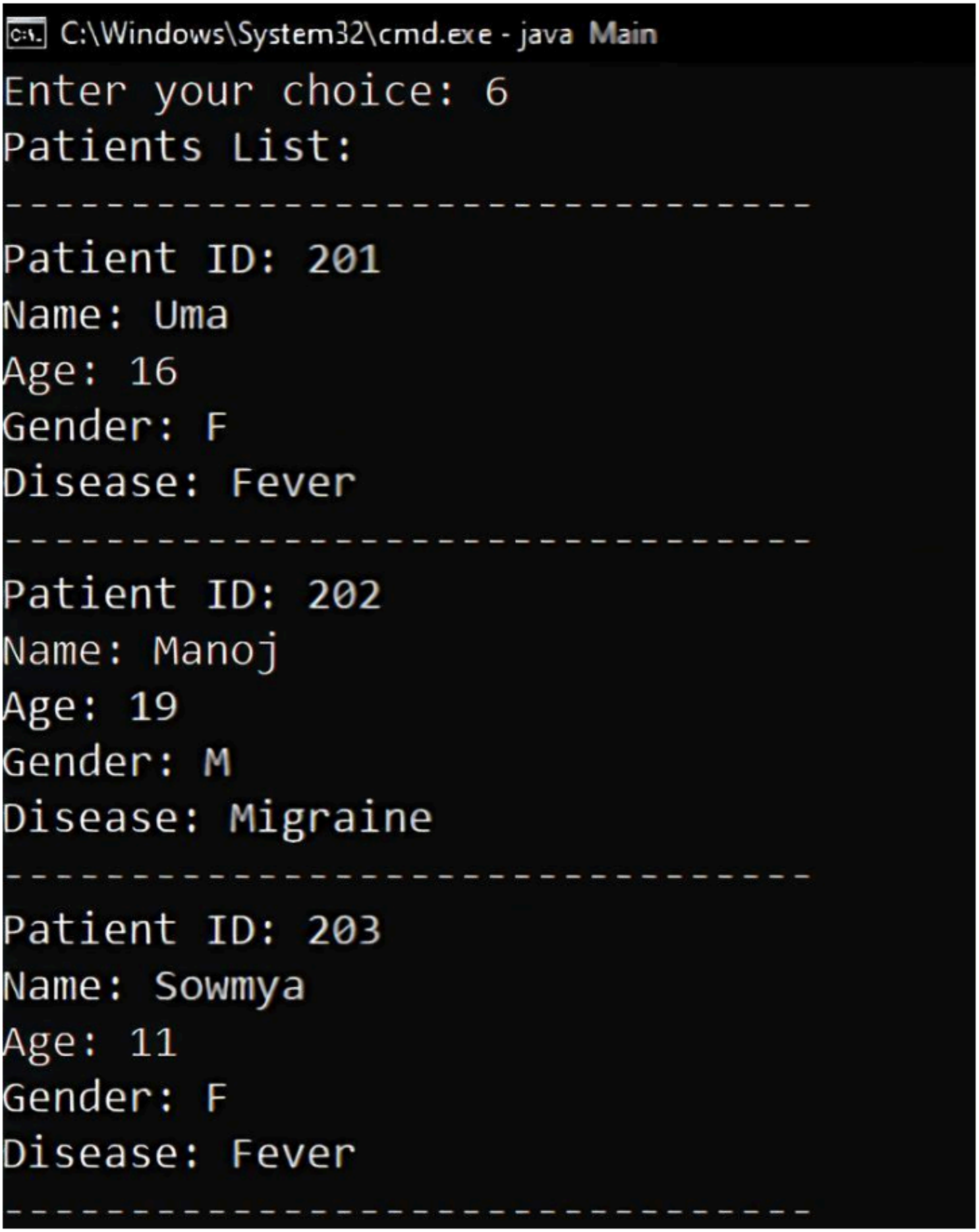
***Doctor Addition and Display

```
---------------------------------

Melmaa Multi-Speciality Hospital
---------------------------------
1. Add Doctor
2. Add Patient
3. Schedule Appointment
4. Cancel Appointment
5. Display All Doctors
6. Display All Patients
7. Display All Appointments
8. Exit
---------------------------------
Enter your choice: 1
Enter Doctor ID: 103
Enter Doctor Name: Varun
Enter Specialization: General Surgeon
Enter Degree: MBBS
Doctor added successfully.
---------------------------------
```

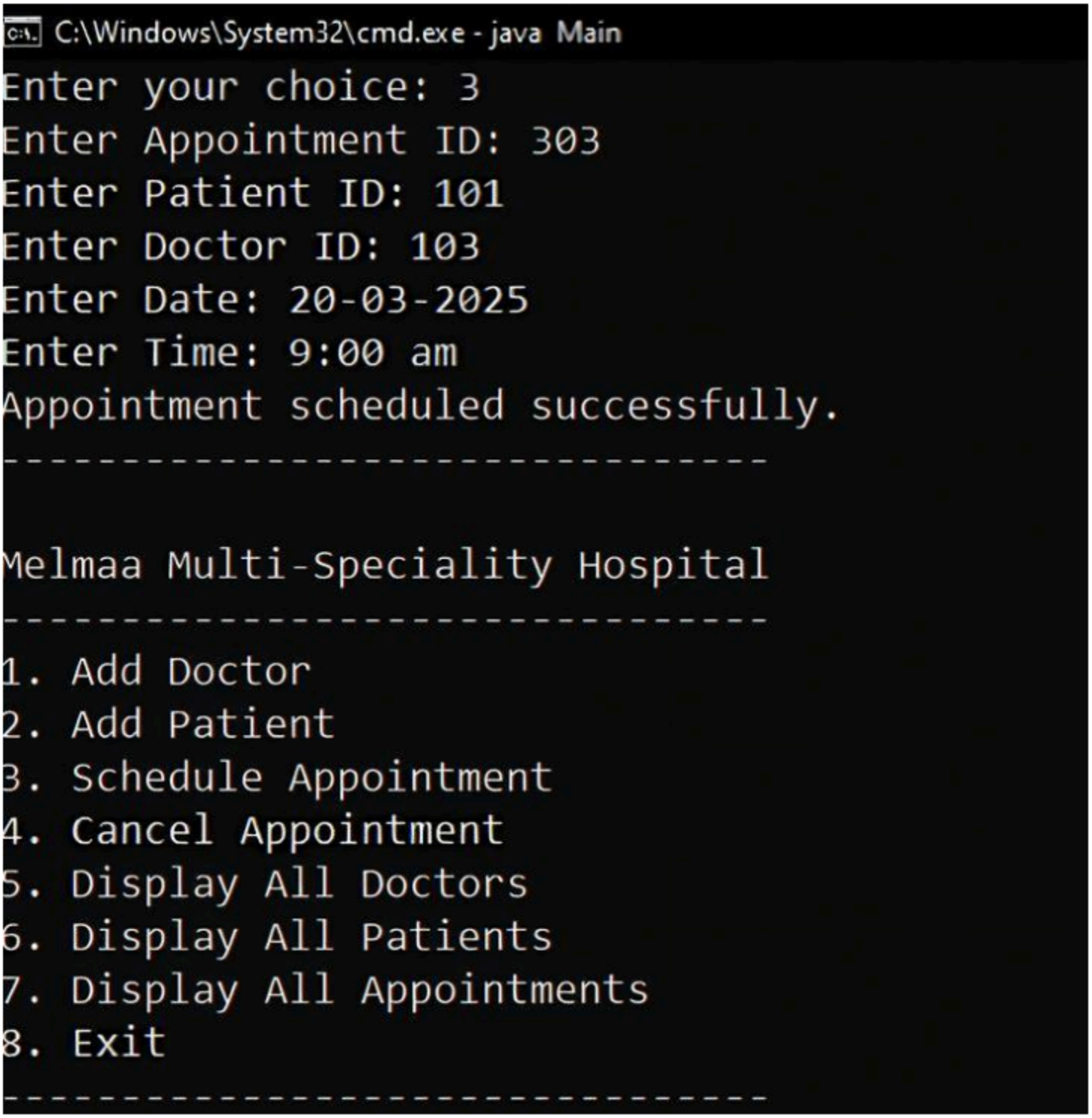


***Patient Addition and Display

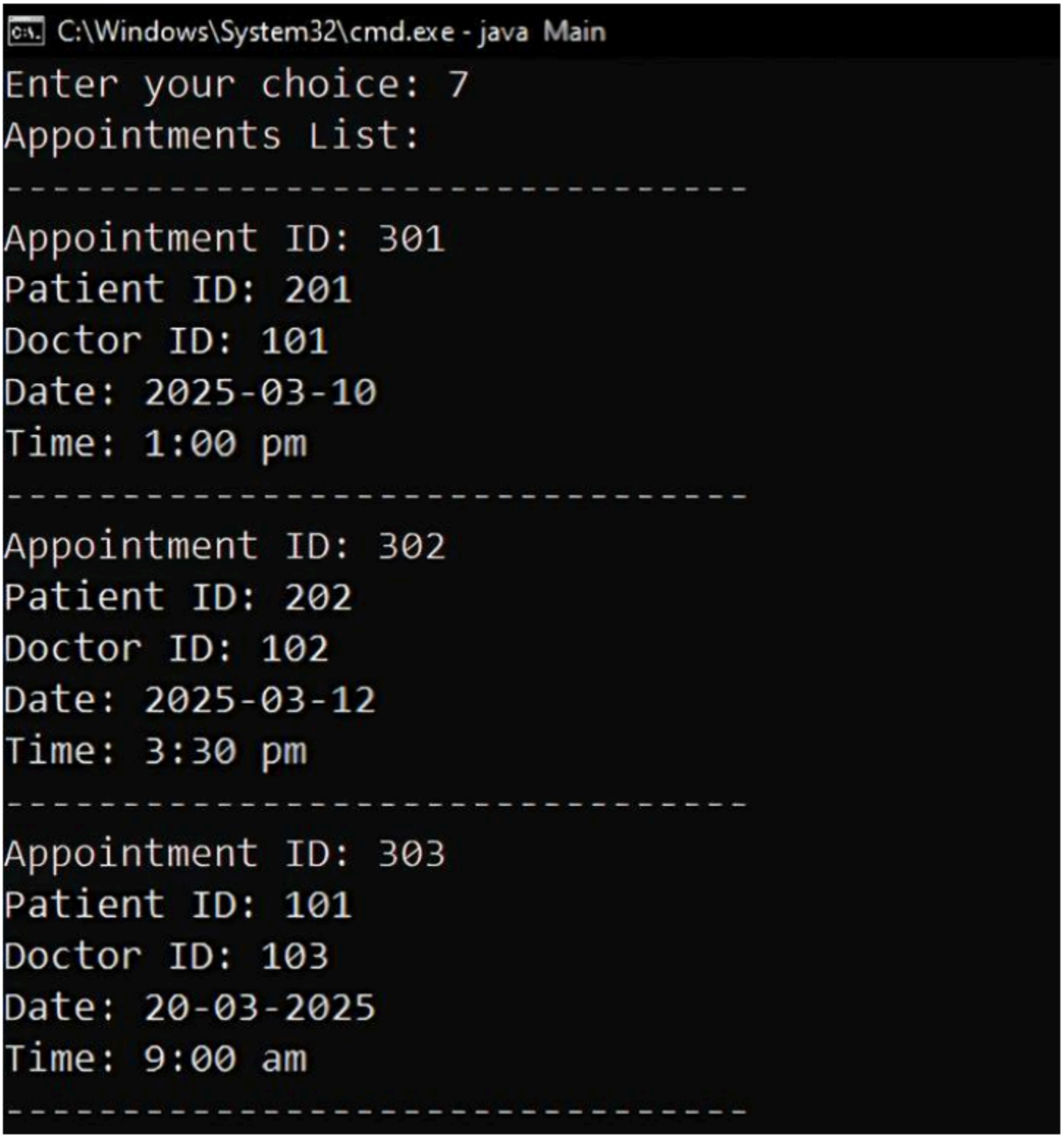

C:\Windows\System32\cmd.exe - java Main
Enter your choice: 2
Enter Patient ID: 203
Enter Patient Name: Sowmya
Enter Age: 11
Enter Gender (M/F):
F
Enter Disease: Fever
Patient added successfully.



***Appointment Scheduling and Display


C:\Windows\System32\cmd.exe - java Main
Enter your choice: 3
Enter Appointment ID: 303
Enter Patient ID: 101
Enter Doctor ID: 103
Enter Date: 20-03-2025
Enter Time: 9:00 am
Appointment scheduled successfully.

Melmaa Multi-Speciality Hospital
---
1. Add Doctor
2. Add Patient
3. Schedule Appointment
4. Cancel Appointment
5. Display All Doctors
6. Display All Patients
7. Display All Appointments
8. Exit

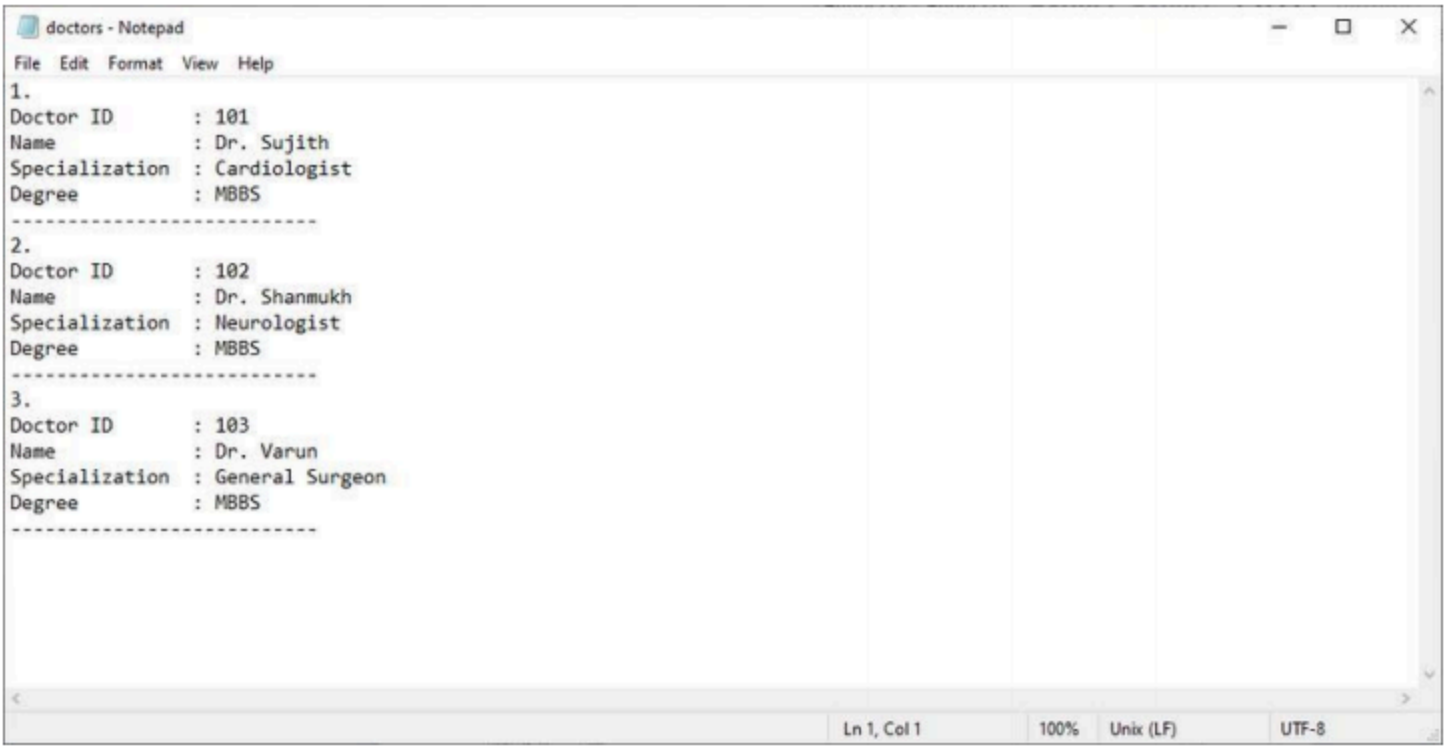


***Appointment Cancellation and Display

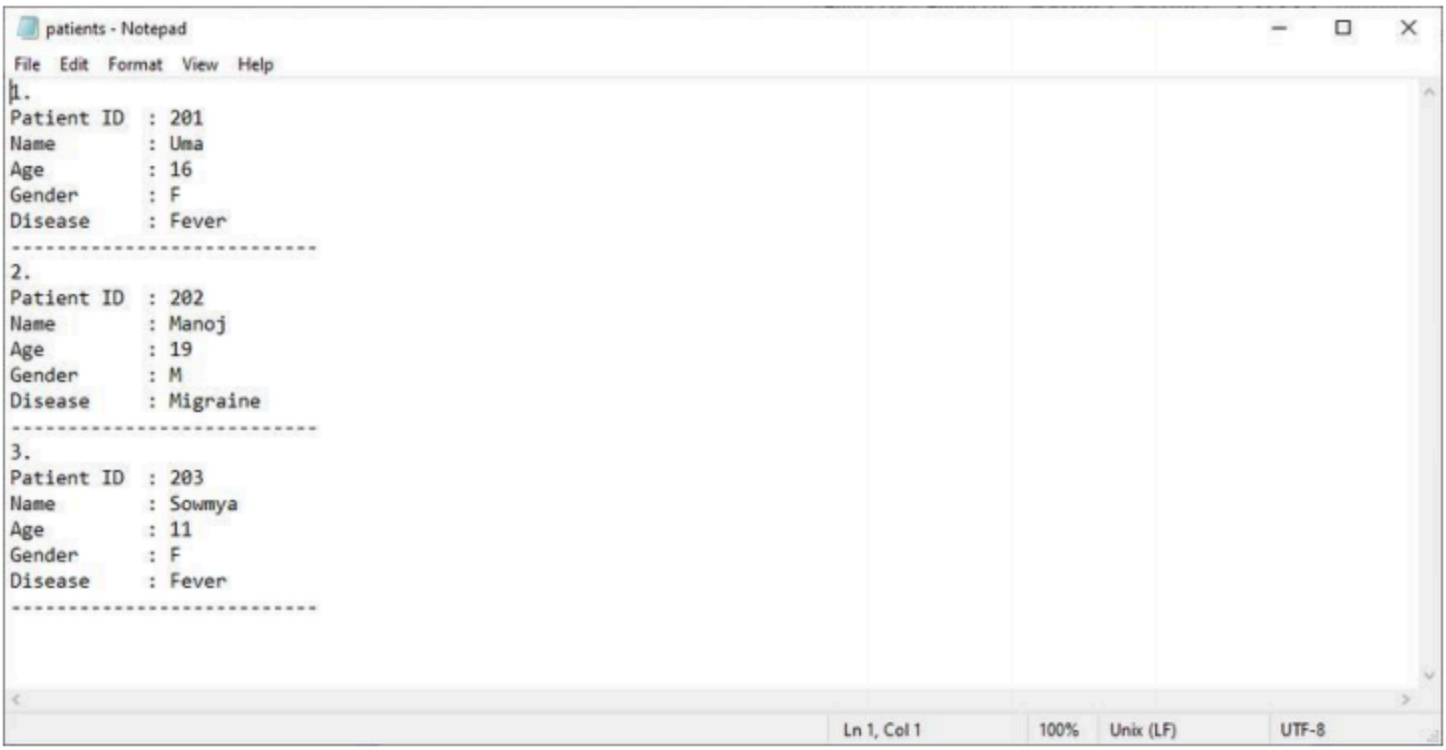
```
C:\Windows\System32\cmd.exe - java Main
Enter your choice: 4
Enter Appointment ID to Cancel: 301
Appointment cancelled successfully.

Melmaa Multi-Speciality Hospital
---------------------------------
1. Add Doctor
2. Add Patient
3. Schedule Appointment
4. Cancel Appointment
5. Display All Doctors
6. Display All Patients
7. Display All Appointments
8. Exit
---------------------------------
Enter your choice: 7
Appointments List:
---------------------------------
Appointment ID: 302
Patient ID: 202
Doctor ID: 102
Date: 2025-03-12
Time: 3:30 pm
---------------------------------
Appointment ID: 303
Patient ID: 101
Doctor ID: 103
Date: 20-03-2025
Time: 9:00 am
---------------------------------
```

***File Storage Verifcation

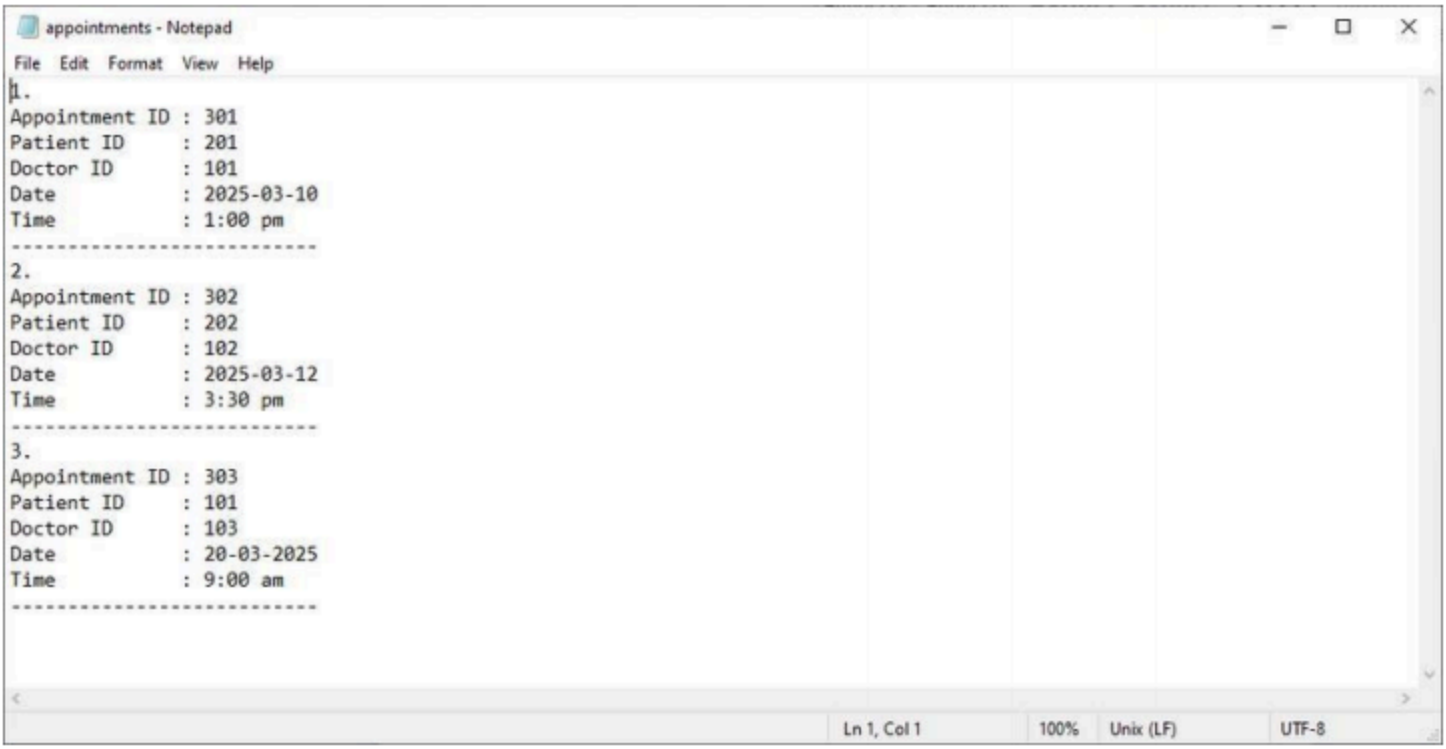
1. docotors.txt



2. patients.txt



3. appointments.txt



Conclusion

The **Hospital Management System (HMS)** is a **user-friendly and efcient** application designed to assist **small to mid-sized healthcare facilities** in managing their daily operations. By utilizing **Core Java and fle handling techniques**, the system ensures **structured and reliable data storage** for doctors, patients, and appointments.

This project enhances **administrative efciency** by automating **record-keeping, appointment scheduling, and patient management**, eliminating the need for manual processes. Its **simple yet efective** design makes it accessible for hospitals with limited technical resources while ensuring **data integrity and quick access to essential information**.

Overall, the HMS signifcantly improves **workfow, reduces paperwork, minimizes errors**, and enhances **patient care services**, making it a **practical solution for healthcare administration**.