Hospital Patient Management System

A Report Submitted
In Partial Fulfillment
for award of Bachelor of Technology

In
COMPUTER SCIENCE AND ENGINEERING
(CYBER SECURITY)

By

Mani Shankar Jha (Roll No. 0251dcys182)

Pakhi Agrawal (Roll No. 2401331720087)

Samridhi Shahi (Roll No. 2401331720110)

Under the Supervision of Mr. Aritra Mitra

Prof. Mr. Aritra Mitra
Professor, Department of Cyber Security

NIET Greater Noida
Autonomous Institute
Estd.2001
Get Future Ready

DECLARATION

I hereby declare that the project report entitled "Hospital Patient Management System" submitted to **Noida Institute of Engineering and Technology** is my original work. No part of this work has been submitted elsewhere for any other degree or diploma. All references and sources have been acknowledged.

Mani Shankar Jha

(Roll No. 0251dcys182)

Pakhi Agrawal

(Roll No. 2401331720087)

Samridhi Shahi

(Roll No. 2401331720110)

CERTIFICATE

Certified that Mani Shankar Jha (Roll No. 0251dcys182), Pakhi Agrawal (Roll No. 2401331720087), Samridhi Shahi (Roll No. 2401331720110) has carried out the Advanced Java project work presented in this Project Report at Hospital Patient Management System in partial fulfilment of the requirements for the award of Bachelor of Technology, in Cyber Security from Dr. APJ Abdul Kalam Technical University, Lucknow under our supervision.

Signature	Signature
Mr. Aritra Mitra Professor Cyber Security NIET Greater Noida	Dr. Tripti Sharma Head Of Department Cyber Security NIET Greater Noida

Date:

ACKNOWLEDGEMENT

I would like to express my gratitude towards Mr. Aritra Mitra for their guidance, support and constant supervision as well as for providing necessary information during my internship.

My thanks and appreciations to respected HOD Dr. Tripti Sharma, for their motivation and support throughout.

ABSTRACT

Hospital Patient Management System

"The Hospital Patient Management System is a Java Swing-based application intended to assist hospital staff by managing records of patients, doctors, and appointments. The system leverages generics and collections for efficient data handling, offering an interactive GUI for CRUD operations on patients, doctors, and appointment bookings. It streamlines administrative workflows such as appointment scheduling, record updating, and deletion, ensuring both ease of access and improved operational efficiency."

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Efficient hospital administration is crucial for patient care and resource management. Manual systems are prone to errors and inefficiencies. The Hospital Patient Management
System aims to digitize core administrative tasks using a robust Java-based GUI.

1.1.1 Background

Modern hospitals require timely access to patient and doctor data, hassle-free scheduling, and reliable record-keeping. Software solutions can greatly enhance these capabilities.

1.1.2 System Overview

- Java Swing provides GUI components for ease of use.
- Data managed using collection frameworks (Array List, HashMap, etc.).
- Supports interactive CRUD for patients, doctors, appointments.
- Enhances reliability and accessibility of records.

1.1.2.1 Patient Module

- Stores individual information (name, age, gender, contact, medical history).
- Add, update, delete, search functionalities.

1.1.2.2 Doctor Module

- Manages doctor profiles (specialization, qualification, work schedules).
- Assigns doctors to patients.
- Supports modifications and deletions.

1.1.2.3 Appointments Module

- Schedule, edit, and cancel appointments.
- Prevents clashes using validation logic.
- Links patient and doctor records.
- Print the Appointments

1.2 Objectives and Scope

Objectives:

- Provide error-free record keeping.
- Automate repetitive administrative tasks.
- Improve workflow and reduce manual labor.

Scope:

- Suitable for hospitals, clinics, and healthcare centers.
- Modular design enables extension (e.g., billing, pharmacy integration).

CHAPTER 2

LITERATURE REVIEW

2.1 Review of Existing Systems

- **Manual Recordkeeping:**
    - Most hospitals started with paper-based systems for storing patient records, appointments, and doctor information.
    - These systems had serious drawbacks:
        - Easy to misplace or damage records.
        - Tedious to update, search, or share information.
        - High risk of human error (incorrect entries, duplicates).
- **Basic Digital Solutions:**
    - Some facilities use spreadsheets or word documents to track information.
    - While faster than paper, these are not secure or integrated:
        - Difficult for multiple users to access or edit simultaneously.
        - No built-in validation to prevent errors.
        - Lacks proper backup and recovery features.
- **Enterprise Resource Planning (ERP) Packages:**
    - Large hospital chains deploy ERP platforms (like SAP Healthcare, Oracle Health Management).
    - ERPs are comprehensive, handling everything from patient records to pharmacy, billing, and insurance.
    - Challenges:
        - Substantial licensing and setup costs.
        - Need for trained staff and IT infrastructure.
        - Not scalable for smaller clinics or hospitals with limited resources.

- **Open-Source Healthcare Platforms:**
  - Solutions like OpenMRS, GNU Health offer customizable, community-driven record management.
  - Benefits:
    - Free to use, customizable.
    - Wide feature set for clinical workflows.
  - Drawbacks:
    - Difficult to install, configure, and maintain.
    - May require database and server management skills.
    - Excessive features can overwhelm small clinics with simple needs

2.2 Technological Landscape

- **Java in Healthcare Software:**
  - Java is a preferred choice for software development in healthcare due to:
    - Cross-platform compatibility (Windows, Mac, Linux).
    - Built-in security features to help protect sensitive patient data.
    - Mature GUI frameworks such as Swing for robust desktop applications.
- **Limitations of Current Solutions:**
  - Many existing systems use web technologies, or are built for large-scale enterprises.
  - Small clinics struggle with:
    - Complexity in setup and daily usage (database/server configurations).
    - Steep learning curve for staff with limited technical expertise.
- **Advantages of This Project's Approach:**
  - Designed as a simple, standalone Java desktop application requiring minimal technical setup.

- **Key features, as shown in the code:**
    - **Patient, Doctor, Appointment Classes:** Each entity represented by an object, storing relevant details, supporting easy CRUD operations.
    - **Use of Collections:** Array List and HashMap efficiently manage and search large amounts of data, ensuring scalability.
    - **Persistence with Serialization:** Data is saved and loaded from files, avoiding complex external database setups.
    - **Swing-based GUI:** User interface is organized into tabs for patients, doctors, and appointments, with easy-to-use buttons and tables.
    - **Printing/Exporting:** Appointment records can be printed directly to PDF, aiding paperless workflow and reporting.
    - **Validation and Error Handling:** Input fields and actions are checked to prevent incorrect or missing data.

- **Summary:**
    - This project bridges the gap by providing:
        - An affordable, effective record management system.
        - Ease of use for non-technical staff.
        - Extensibility (possible future modules for pharmacy, billing, etc.).
        - Greater reliability and data integrity compared to manual or basic digital methods.
    - It is especially suitable for small-to-medium hospitals and clinics, complementing their workflow without the overhead of full ERP software or web/server solutions.

CHAPTER 3

Requirements

3.1 Requirements

- **User Authentication**
  - Only authorized hospital staff should access the application and sensitive medical data.
  - In future versions, a login window (dialogs) or user roles (admin, staff) can be implemented.
  - Current code does not show authentication, but it's standard for production systems.
- **Patient Management**
  - Staff must be able to add, edit, search, and remove patient records efficiently.
  - The GUI provides an "Add Patient" button to collect patient details (ID, name, age, contact).
    - Data validation avoids duplicate or blank entries.
    - Updates and deletions allow the hospital to maintain accurate, up-to-date records.
    - Patient information is stored in an ArrayList<Patient>, serialized to a file (patients.ser), ensuring persistence between program runs.
    - Patient searches and updates operated by selecting rows in the table.
  - Each Patient object encapsulates identity and contact info, and can be displayed or printed.

- **Doctor Management**
    - Allows hospital staff to register, update, or remove doctor profiles, including specialization.
    - Doctors are managed within a HashMap<String, Doctor>, providing quick lookup by ID.
    - GUI tab "Doctors" includes add/update/delete operations, similar to patients.
    - Doctor info includes ID, name, and specialty, which helps match patients to appropriately qualified doctors.
    - Updates to doctor data appear system-wide, supporting staffing changes or schedule updates.
- **Appointment Scheduling**
    - Book, view, edit, and cancel appointments efficiently to minimize double-booking and scheduling conflicts.
    - The "Appointments" tab lets staff enter patient ID, doctor ID, date/time, and appointment notes.
        - **Validation:** Ensures IDs exist and date/time format is correct; can be expanded for conflict checking.
        - All appointments are stored in an ArrayList<Appointment>.
        - Appointments link patient and doctor objects for complete traceability.
        - Editing/canceling is table-based for easy correction of schedules.
        - Print functionality generates PDF summaries using iText, supporting administrative reports or patient handouts.
- **Search Functionality**
    - Tables for patients, doctors, and appointments make it easy to scan, sort, and select records.
    - Add-ons could allow text-based search (by name, ID, specialty, date, etc.), helping staff quickly locate or filter relevant data.
    - Underlying collection data structures (ArrayList, HashMap) ensure performance even as record counts grow.

CHAPTER 4

Implementation & Testing

4.1 System Implementation

- **Technology Stack:**
  - **Programming Language:** Java (version as set in Maven, e.g., 24)
  - **GUI Framework:** Java Swing (JFrame, JTabbedPane, JTable)
  - **Persistence:** Java Serialization (Serializable interface, .ser files)
  - **PDF Export:** iText PDF library, configured via Maven
- **Project Structure:**
  - **Patient, Doctor, Appointment Classes:**
    - Each class encapsulates relevant fields and implements Serializable to support data persistence.
    - Patient: ID, Name, Age, Contact
    - Doctor: ID, Name, Specialty
    - Appointment: ID, references to Patient & Doctor, LocalDateTime, Notes
  - **Data Management:**
    - Uses ArrayList and HashMap for runtime storage of objects.
    - Data is automatically saved to and loaded from local files (patients.ser, doctors.ser, appointments.ser) using object streams.

- **Graphical User Interface:**
    - Uses a tabbed interface:
        - **Patients Tab:** Display and manage patients
        - **Doctors Tab:** Display and manage doctors
        - **Appointments Tab:** View, book, cancel, and print PDF summaries for appointments
    - Each tab contains a table for listing records and buttons for CRUD operations.
- **Functional Features:**
    - **Add, Update, Delete:** Dialog prompts allow entry, validation, editing, and deletion of patients, doctors, and appointments.
    - **Appointment Scheduling:** Validates patient and doctor IDs, parses date/time input, avoids incomplete entries.
    - **PDF Generation:** Uses iText to print appointment details as PDFs for documentation or sharing.
- **Build & Dependencies:**
    - Utilizes Maven for dependency management and build automation.
    - Includes itextpdf dependency for PDF operations.
    - Configured to run the main class directly (exec-maven-plugin).

4.2 Testing

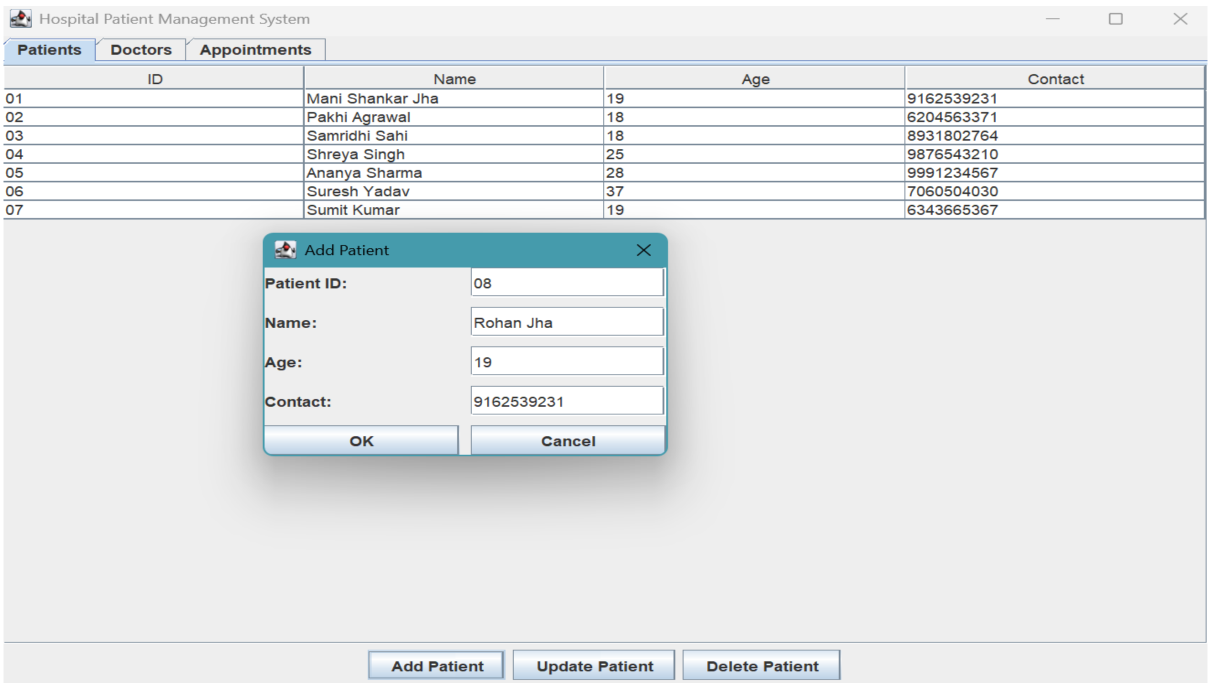
- **Unit Testing:**
    - Manual validation of object creation and manipulation for each main class (Patient, Doctor, Appointment).
    - Checked setters, getters, toString(), and object serialization/deserialization.

- **Integration Testing:**
    - Full workflow tested:
        - Add new patient/doctor
        - Book and update appointment between valid records
        - Delete patient or doctor and verify record removal
        - Print appointment PDF and verify file validity
- **GUI/User Testing:**
    - Ensured that:
        - All buttons and table actions work as intended
        - Errors prompt informative messages (e.g., invalid input, selection error)
        - Data updates are reflected live after each operation
- **Persistence Testing:**
    - Restarted the application and confirmed all data was loaded back correctly from .ser files.
- **Performance Testing:**
    - Populated patients, doctors, and appointments with 30+ entries each to confirm smooth operation and fast search/display.
- **Exception Handling:**
    - Verified that invalid input (empty, wrong format, nonexistent ID) was properly caught and reported to the user.

Test Case Example:

- **Add a new patient, add a new Doctor, book appointment, reschedule appointment, delete doctor.**

ADD A NEW PATIENT:



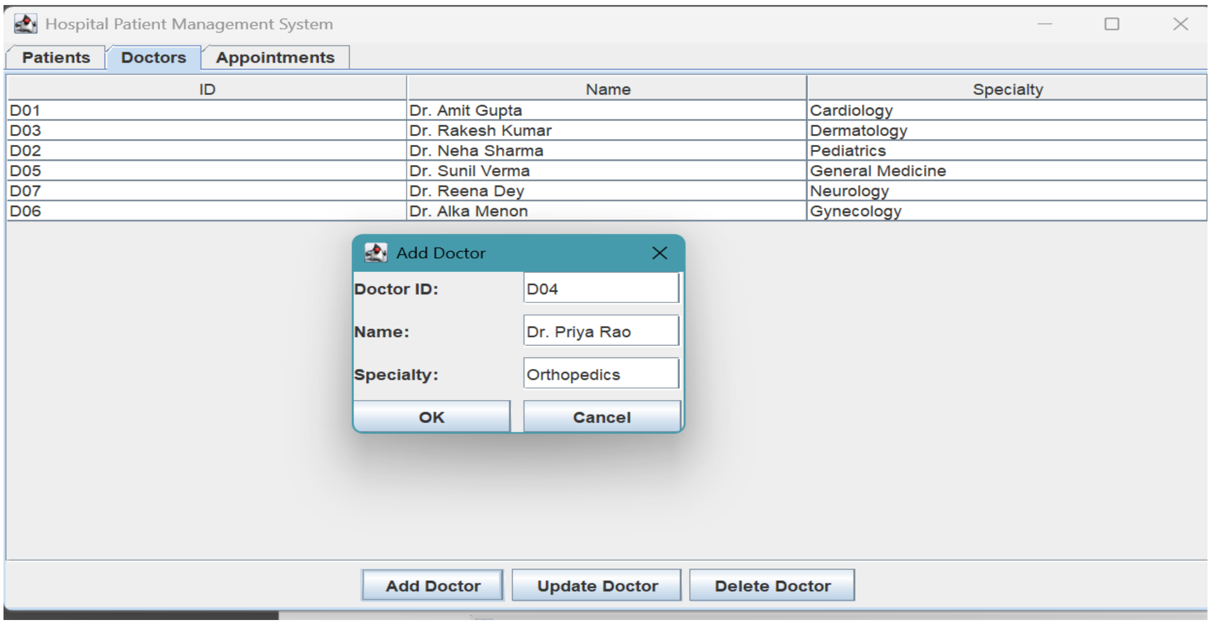
Hospital Patient Management System

ID	Name	Age	Contact
01	Mani Shankar Jha	19	9162539231
02	Pakhi Agrawal	18	6204563371
03	Samridhi Sahi	18	8931802764
04	Shreya Singh	25	9876543210
05	Ananya Sharma	28	9991234567
06	Suresh Yadav	37	7060504030
07	Sumit Kumar	19	6343665367

Add Patient

Patient ID:	08
Name:	Rohan Jha
Age:	19
Contact:	9162539231

ADD A NEW DOCTOR:



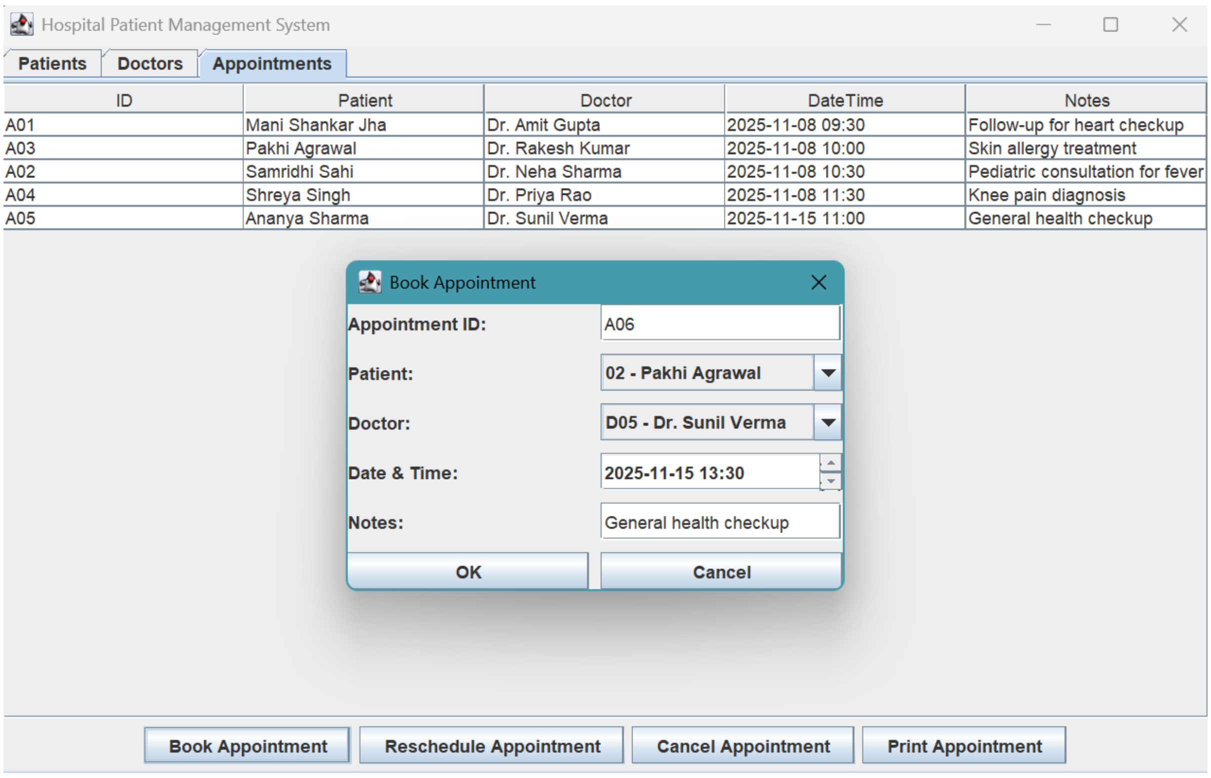
Hospital Patient Management System

ID	Name	Specialty
D01	Dr. Amit Gupta	Cardiology
D03	Dr. Rakesh Kumar	Dermatology
D02	Dr. Neha Sharma	Pediatrics
D05	Dr. Sunil Verma	General Medicine
D07	Dr. Reena Dey	Neurology
D06	Dr. Alka Menon	Gynecology

Add Doctor

Doctor ID:	D04
Name:	Dr. Priya Rao
Specialty:	Orthopedics

BOOK APPOINTMENT:



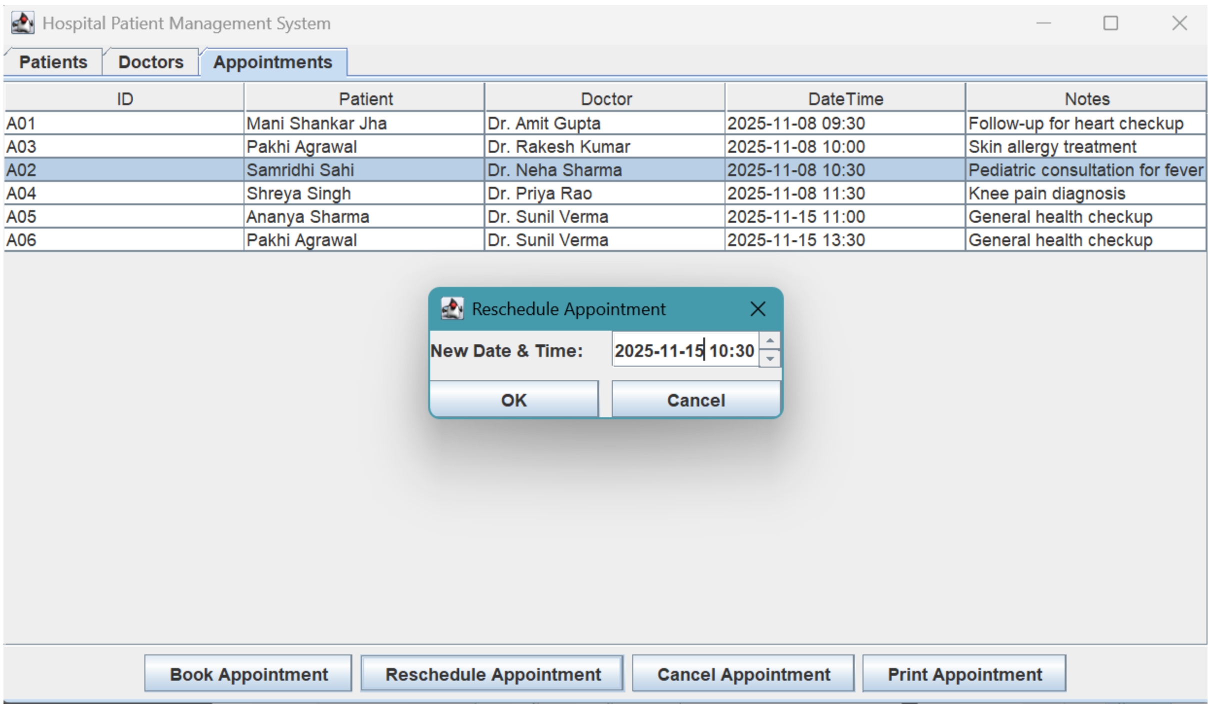
Hospital Patient Management System

ID	Patient	Doctor	DateTime	Notes
A01	Mani Shankar Jha	Dr. Amit Gupta	2025-11-08 09:30	Follow-up for heart checkup
A03	Pakhi Agrawal	Dr. Rakesh Kumar	2025-11-08 10:00	Skin allergy treatment
A02	Samridhi Sahi	Dr. Neha Sharma	2025-11-08 10:30	Pediatric consultation for fever
A04	Shreya Singh	Dr. Priya Rao	2025-11-08 11:30	Knee pain diagnosis
A05	Ananya Sharma	Dr. Sunil Verma	2025-11-15 11:00	General health checkup

Book Appointment

Appointment ID:	A06
Patient:	02 - Pakhi Agrawal
Doctor:	D05 - Dr. Sunil Verma
Date & Time:	2025-11-15 13:30
Notes:	General health checkup

RESCHEDULE APPOINTMENT:



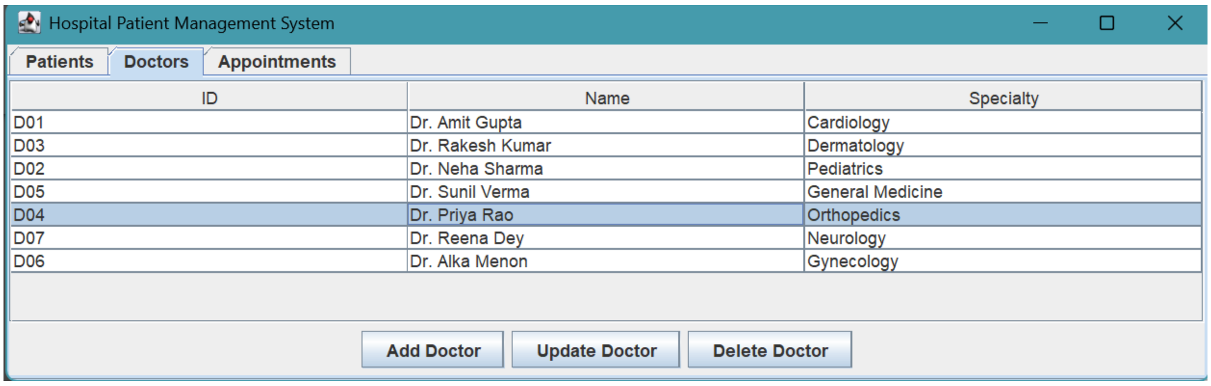
Hospital Patient Management System

ID	Patient	Doctor	DateTime	Notes
A01	Mani Shankar Jha	Dr. Amit Gupta	2025-11-08 09:30	Follow-up for heart checkup
A03	Pakhi Agrawal	Dr. Rakesh Kumar	2025-11-08 10:00	Skin allergy treatment
A02	Samridhi Sahi	Dr. Neha Sharma	2025-11-08 10:30	Pediatric consultation for fever
A04	Shreya Singh	Dr. Priya Rao	2025-11-08 11:30	Knee pain diagnosis
A05	Ananya Sharma	Dr. Sunil Verma	2025-11-15 11:00	General health checkup
A06	Pakhi Agrawal	Dr. Sunil Verma	2025-11-15 13:30	General health checkup

Reschedule Appointment

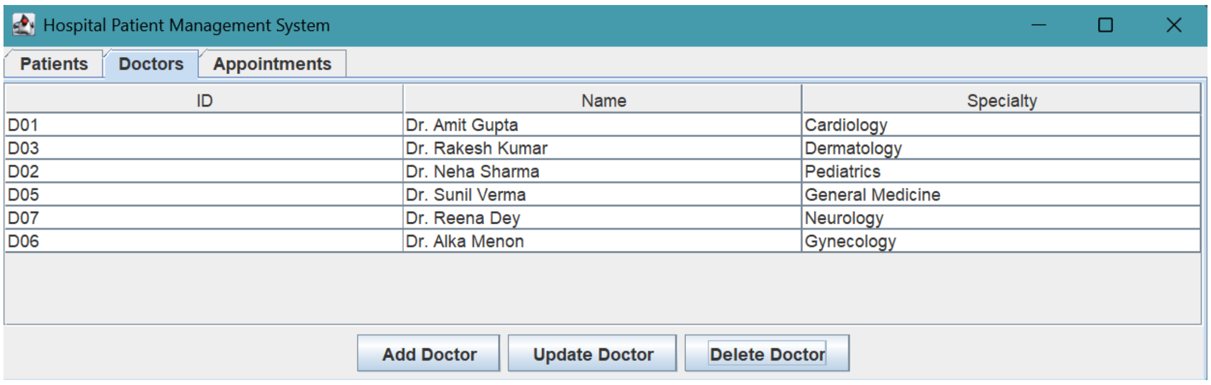
New Date & Time: 2025-11-15 10:30

DELETING A DOCTOR:



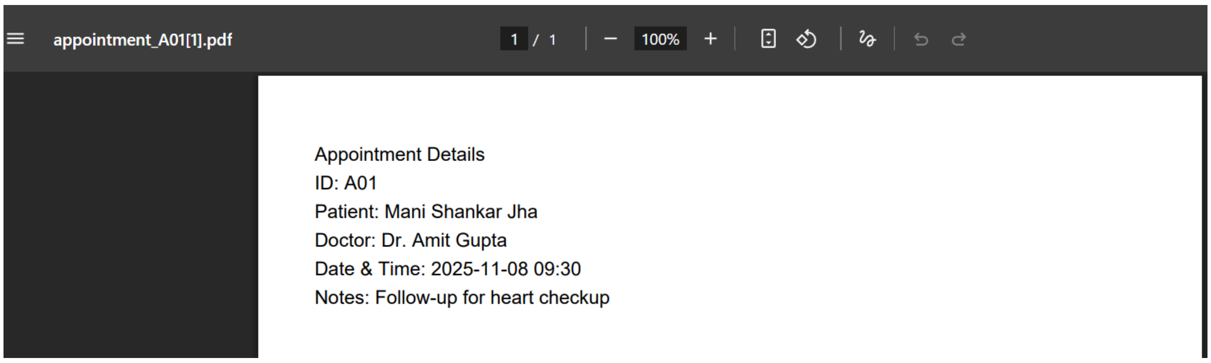
ID	Name	Specialty
D01	Dr. Amit Gupta	Cardiology
D03	Dr. Rakesh Kumar	Dermatology
D02	Dr. Neha Sharma	Pediatrics
D05	Dr. Sunil Verma	General Medicine
D04	Dr. Priya Rao	Orthopedics
D07	Dr. Reena Dey	Neurology
D06	Dr. Alka Menon	Gynecology

AFTER DELETING A DOCTOR:



ID	Name	Specialty
D01	Dr. Amit Gupta	Cardiology
D03	Dr. Rakesh Kumar	Dermatology
D02	Dr. Neha Sharma	Pediatrics
D05	Dr. Sunil Verma	General Medicine
D07	Dr. Reena Dey	Neurology
D06	Dr. Alka Menon	Gynecology

THIS IS THE PRINT OF APPOINTMENT:



Appointment Details
ID: A01
Patient: Mani Shankar Jha
Doctor: Dr. Amit Gupta
Date & Time: 2025-11-08 09:30
Notes: Follow-up for heart checkup

CHAPTER 5

CONCLUSION AND FUTURE WORK:

Conclusion

- The Hospital Patient Management System developed in Java Swing offers hospitals and clinics a **reliable, efficient, and user-friendly** solution for daily data management.
- It successfully replaces manual and error-prone recordkeeping by **digitizing patient, doctor, and appointment data** through tabbed panels and intuitive forms.
- Features like **CRUD operations, GUI tables, PDF export for appointments, and persistent local storage via serialization** provide comprehensive support for hospital staff.
- By keeping all operations quick, simple, and accessible, the system **reduces administrative overhead** and **minimizes scheduling clashes and data loss**, allowing workers to focus more on patient care.
- Backend collections (ArrayList, HashMap) ensure **high performance** and the ability to handle numerous patients, doctors, and appointments over time.
- Serialization of records allows the hospital to retain data between uses, even without a network or complex database, greatly **increasing accessibility** and **reliability** for small and medium facilities.

Future Work

- **Integration with JDBC and Database Storage**
  - Upgrade from file-based serialization to a relational database (e.g., MySQL, PostgreSQL) using JDBC, supporting:
    - Greater record volume
    - Enhanced security and backup controls
    - Multi-user simultaneous access and web deployment
- **Additional Modules**
  - Implement future modules for:
    - **Billing System:** Track and bill hospital charges per patient.
    - **Inventory Management:** Maintain stock levels of medicines, supplies, equipment.

        - **Pharmacy Module:** Link prescribed medicines with pharmacy, allowing automated stock deduction and billing.

- **Cloud Backup and Multi-User Accounts**
    - Support for **cloud synchronization** of patient and schedule records so data is not lost even in case of hardware failures.
    - Add **user account management**—roles for admin, doctors, nurses—enabling secure, controlled access to sensitive data.
- **Web-Based Upgrades**
    - Rebuild as a **web application** using technologies such as Java EE, Spring Boot, or REST APIs:
        - Enable remote access from any location within hospital/campus.
        - Enable appointments to be booked or checked from outside by staff/patients.
- **Enhanced Data Security**
    - Incorporate **encryption for data at rest and in transit** to follow healthcare data privacy laws (HIPAA, GDPR).
    - Implement security features such as login authentication, audit logging, and role-based access control.
- **Other Potential Features**
    - Advanced reporting, analytics dashboards for administration.
    - SMS/email notifications for appointments or changes.
    - Integration with third-party health devices, electronic medical records (EMR/EHR).

REFERENCES

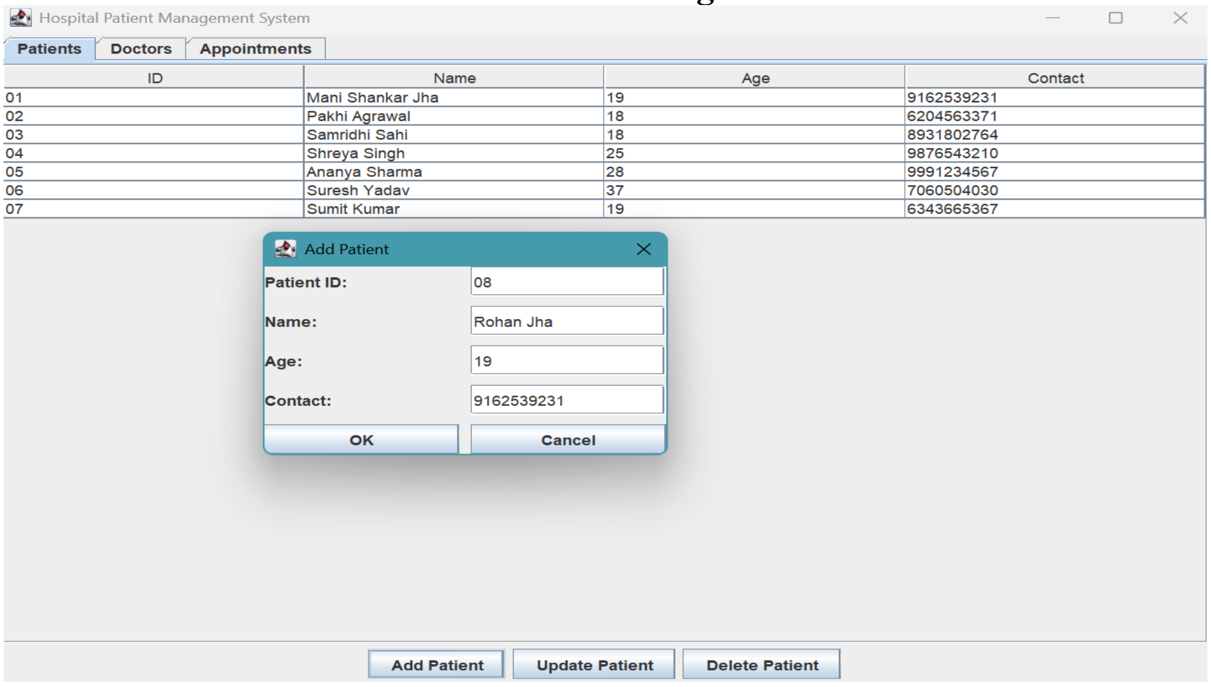
References

- Oracle, **Java Swing Documentation**
  Used for designing the application's graphical user interface (GUI)
- Bloch, Joshua. **Effective Java**
  Provides best practices for Java generics, serialization, and object-oriented design
- Documentation for **javax.swing, JTable, JPanel, DefaultTableModel, JTabbedPane**
  Essential for managing interactive tables and tabbed GUI panels
- **iText PDF Library Documentation**
  Referenced for generating PDF reports of appointments directly from Java code
- **Java Serialization & Collections Tutorials** (ArrayList, HashMap)
  Used for persistent data storage and fast record lookup in the system
- Healthcare software research articles on digitization and hospital management systems
- Official Java tutorials on event handling and dialog creation (JOptionPane)
- [Add any textbooks or important online tutorials you used for reference]

APPENDICES

A. GUI Screenshots:

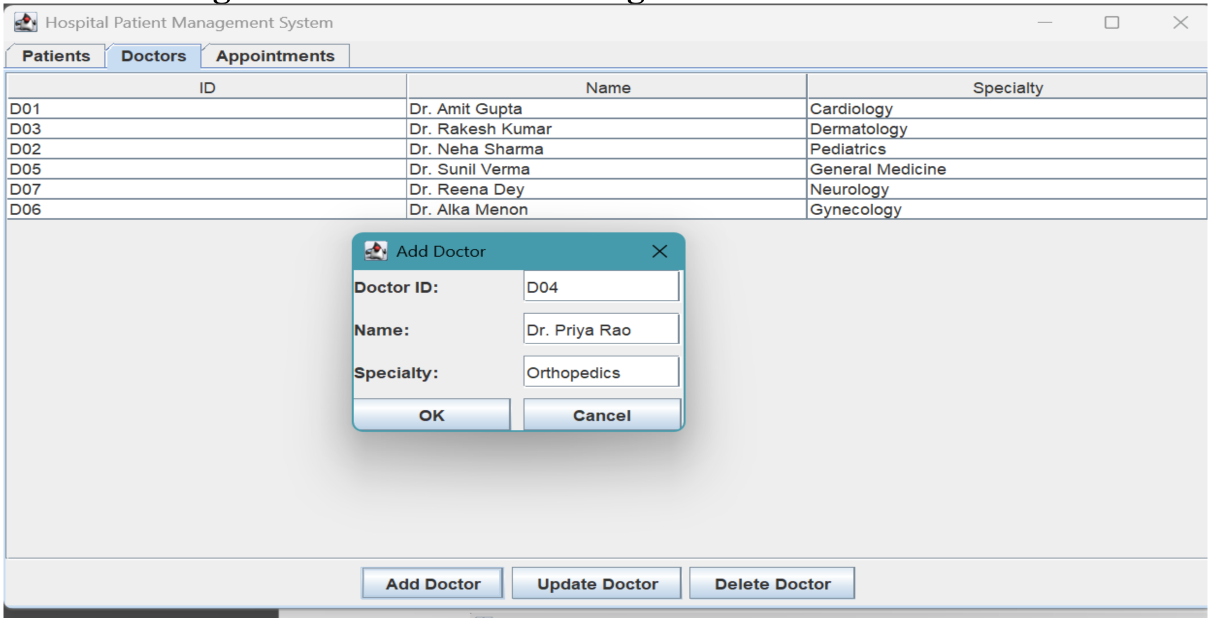
- **Main Dashboard Patient Dashboard and registration form**



ID	Name	Age	Contact
01	Mani Shankar Jha	19	9162539231
02	Pakhi Agrawal	18	6204563371
03	Samridhi Sahi	18	8931802764
04	Shreya Singh	25	9876543210
05	Ananya Sharma	28	9991234567
06	Suresh Yadav	37	7060504030
07	Sumit Kumar	19	6343665367

Patient ID:	08
Name:	Rohan Jha
Age:	19
Contact:	9162539231

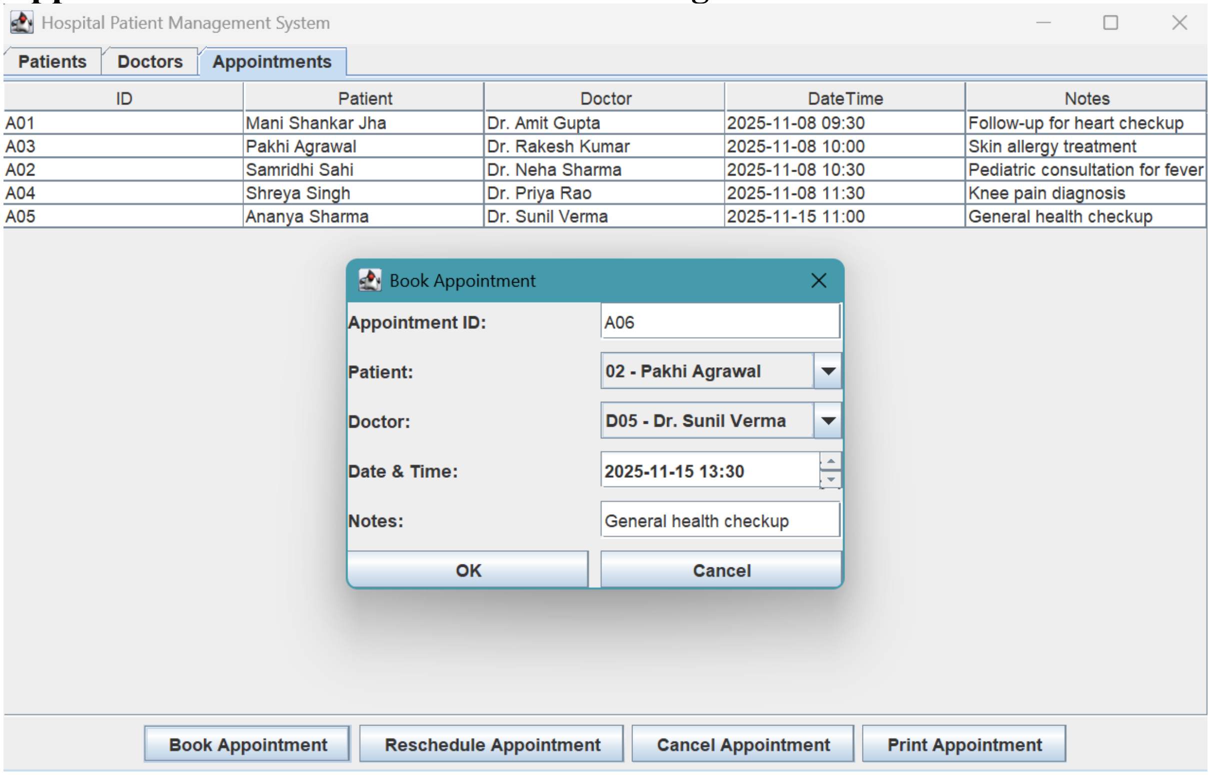
- **Doctor management Dashboard and registration form**



ID	Name	Specialty
D01	Dr. Amit Gupta	Cardiology
D03	Dr. Rakesh Kumar	Dermatology
D02	Dr. Neha Sharma	Pediatrics
D05	Dr. Sunil Verma	General Medicine
D07	Dr. Reena Dey	Neurology
D06	Dr. Alka Menon	Gynecology

Doctor ID:	D04
Name:	Dr. Priya Rao
Specialty:	Orthopedics

- **Appointment schedulers Dashboard and registration form**



ID	Patient	Doctor	DateTime	Notes
A01	Mani Shankar Jha	Dr. Amit Gupta	2025-11-08 09:30	Follow-up for heart checkup
A03	Pakhi Agrawal	Dr. Rakesh Kumar	2025-11-08 10:00	Skin allergy treatment
A02	Samridhi Sahi	Dr. Neha Sharma	2025-11-08 10:30	Pediatric consultation for fever
A04	Shreya Singh	Dr. Priya Rao	2025-11-08 11:30	Knee pain diagnosis
A05	Ananya Sharma	Dr. Sunil Verma	2025-11-15 11:00	General health checkup

Appointment ID:	A06
Patient:	02 - Pakhi Agrawal
Doctor:	D05 - Dr. Sunil Verma
Date & Time:	2025-11-15 13:30
Notes:	General health checkup

B. Sample Code:

Include important functions/methods for adding, deleting, searching records and scheduling appointments.

```
// Simplified Patient class
class Patient implements Serializable {
   private String id, name, contact;
   private int age;
   public Patient(String id, String name, int age, String contact) {
      this.id = id; this.name = name; this.age = age; this.contact =
contact;
   }
   // Getters and setters omitted for brevity
}
// Simplified Doctor class
class Doctor implements Serializable {
   private String id, name, specialty;
   public Doctor(String id, String name, String specialty) {
      this.id = id; this.name = name; this.specialty = specialty;
   }
   // Getters and setters omitted for brevity
}
```

```
// Simplified Appointment class
class Appointment implements Serializable {
   private String id;
   private Patient patient;
   private Doctor doctor;
   private LocalDateTime dateTime;
   private String notes;
   public Appointment(String id, Patient patient, Doctor doctor,
LocalDateTime dateTime, String notes) {
      this.id = id; this.patient = patient; this.doctor = doctor;
this.dateTime = dateTime; this.notes = notes;
   }
   // Getters and setters omitted for brevity
}
// In HospitalPatientManagementSystem class:
List<Patient> patients = new ArrayList<>();
Map<String, Doctor> doctors = new HashMap<>();
List<Appointment> appointments = new ArrayList<>();
// Add Patient
void addPatient(String id, String name, int age, String contact) {
   patients.add(new Patient(id, name, age, contact));
   saveData(); refreshTables();
}
// Delete Patient by ID
void deletePatient(String id) {
   patients.removeIf(p -> p.getId().equals(id));
   saveData(); refreshTables();
}
// Search Patient by ID (returns Patient or null)
Patient searchPatient(String id) {
   return patients.stream().filter(p ->
p.getId().equals(id)).findFirst().orElse(null);
}
// Add Doctor
void addDoctor(String id, String name, String specialty) {
   doctors.put(id, new Doctor(id, name, specialty));
   saveData(); refreshTables();
}
// Delete Doctor by ID
```

```
        void deleteDoctor(String id) {
            doctors.remove(id);
            saveData(); refreshTables();
        }
        // Search Doctor by ID
        Doctor searchDoctor(String id) {
            return doctors.get(id);
        }
        // Schedule Appointment
        void bookAppointment(String id, String patientId, String doctorId,
LocalDateTime dateTime, String notes) {
            Patient patient = searchPatient(patientId);
            Doctor doctor = searchDoctor(doctorId);
            if (patient != null && doctor != null) {
                appointments.add(new Appointment(id, patient, doctor,
dateTime, notes));
                saveData(); refreshTables();
            }
        }
        // Cancel Appointment by ID
        void cancelAppointment(String id) {
            appointments.removeIf(a -> a.getId().equals(id));
            saveData(); refreshTables();
        }
        // Methods saveData() and refreshTables() handle data persistence
and UI updates (not shown here)
```

Code Explanation: -

- The code is a **Java Swing** application for managing a hospital’s **patients, doctors**, and **appointments**.
- It has three main entity classes: **Patient, Doctor, and Appointment**, all **Serializable** for data persistence.
- GUI consists of three tabs—Patients, Doctors, and Appointments—each with tables showing records and buttons for CRUD operations.
- Dialog forms collect user input for adding or updating patients, doctors, and booking or rescheduling appointments.
- Uses JTable with DefaultTableModel to display and refresh data dynamically in the interface.
- Patients and doctors are stored in a list and a map respectively; appointments are stored in a list.
- Data is saved and loaded from serialized files (.ser) for persistence between runs.
- Appointment printing feature generates a PDF with appointment details using the iText library.
- Event listeners handle button actions to add, update, delete, book, reschedule, cancel, and print data.
- The application starts with a main method that displays the GUI safely using SwingUtilities.invokeLater.