Hospital Management System

A PROJECT REPORT

Submitted by:

RATNESH SINGH VISHEN	***{23BCS11008}***
ANSHUMAN VATSA	***{23BCS12208}***
MAYANK GUPTA	***{23BCS80335}***
KALPNA SINGH	***{23BCS11909}***

In partial fulfillment for the award of the degree of

BACHELOROFENGINEERING

IN

COMPUTER SCIENCE &ENGINEERING

Chandigarh University
November, 2025

TABLEOFCONTENTS

CHAPTER1.INTRODUCTION:-

1.1 Introduction to Project:

In the modern healthcare industry, the need for efficient and reliable management systems has become increasingly important. Hospitals handle vast amounts of information daily, including patient details, doctor schedules, medical histories, billing, and appointments. Managing all these tasks manually often leads to errors, data loss, delays, and poor coordination between departments. To overcome these challenges, the *Hospital Management System (HMS)* has been developed using **Java**, **Servlets**, **JSP**, and **XML** technologies.

The proposed system provides a digital platform to automate hospital operations and improve the overall healthcare experience for both patients and staff. It allows patients to register, book appointments, and view their medical history conveniently, while doctors can manage schedules, access patient records, and update diagnoses in real time. The admin module ensures smooth coordination by managing doctors, patients, appointments, and hospital resources efficiently.

By integrating XML for data storage and communication, the system ensures flexibility, scalability, and data integrity. Additionally, role-based access control guarantees data security and privacy. This project aims to create a user-friendly, efficient, and secure hospital management platform that streamlines hospital processes, enhances service delivery, and reduces administrative workload.

2.2 Identification to Problem:

In many hospitals, administrative and operational activities are still managed manually, which leads to numerous challenges. Patient records are often maintained in paper form, making it difficult to retrieve information quickly during emergencies. Appointment scheduling is prone to human errors and conflicts, while maintaining doctor availability and patient billing becomes time-consuming and inefficient.

Manual handling of data also increases the chances of duplication, misplacement, and unauthorized access to sensitive health information. Communication between departments such as reception, doctors, and the billing section is often slow, causing delays in treatment and reduced patient satisfaction.

Furthermore, hospitals struggle to generate accurate reports related to patient visits, resource usage, and doctor consultations. The absence of a centralized system makes monitoring hospital activities and tracking medical histories difficult.

To address these issues, there is a clear need for a **computerized Hospital Management System** that automates daily operations, maintains accurate records, ensures security of sensitive data, and enhances the overall efficiency of hospital management.

CHAPTER2. BACKGROUND STUDY:-

2.1. <u>Existing Solutions</u>

1.Commercial Hospital Systems

Many hospitals use paid software such as *Mediware*, *eHospital*, *HMS365*, and *Practo Ray*.

These systems provide modules for appointments, billing, and reports. However, they are expensive and require skilled staff to operate and maintain.

2.Open-Source Systems

Tools like *OpenMRS* and *HospitalRun* offer basic hospital management features.

They are cost-free but require technical expertise for setup and customization.

Limited XML or Java integration support in many open-source options.

3.Manual Record Systems

Many small hospitals still rely on paper-based or spreadsheet-based record keeping.

These systems are prone to data loss, duplication, and human errors. Retrieval of patient history and reports takes more time.

4.Limitations of Existing Systems
High installation and maintenance cost.
Lack of flexibility and customization.
No centralized database for all hospital modules. Weak data security and privacy handling.

5.Need for Improvement
Hospitals require an affordable, secure, and user-friendly system.
Integration with **Java, JSP, Servlets, and XML** can provide better scalability and performance.

2.2. Problem Definition:

1.Manual Process Inefficiency
Most hospitals still depend on manual record keeping and appointment handling.
This leads to errors, delays, and difficulty in managing patient data.

2.Data Management Issues
Storing patient details, doctor information, and billing records on paper or spreadsheets causes data redundancy and inconsistency. Retrieving old patient histories or reports takes unnecessary time.

3.Lack of Centralized System
No unified platform exists to manage doctors, patients, appointments, billing, and reports together.
Each department works independently, causing communication gaps.

4.Limited Accessibility
Manual systems restrict easy access to information for doctors, patients, and admins.
Patients cannot book or manage appointments online.

5.Security and Privacy Concerns
Sensitive medical data is often not protected properly. Unauthorized access or data leakage is a common risk.

6.Reporting and Monitoring Difficulties
Generating performance reports, patient statistics, and doctor consultations manually is time-consuming and inaccurate.

7.Need for Automation
There is a strong requirement for a digital solution that automates hospital tasks, improves coordination, ensures data security, and enhances patient care.

2.3. Goals and Objectives

1. Main Goal
To design and develop a **web-based Hospital Management System** using **Java, Servlets, JSP, and XML** that automates hospital operations, improves data management, and enhances communication between patients, doctors, and administrators.

2. Specific Objectives

1.Patient Management
Enable easy registration and management of patient profiles.
Store and retrieve patient medical history securely using XML files.

2.Doctor Management
Maintain doctor profiles with their specialization and availability schedules.
Allow doctors to view appointments and update patient records efficiently.

3.Appointment System
Provide patients the ability to book, cancel, or reschedule appointments online. Allow doctors to confirm or modify appointment timings.

4.Electronic Medical Records (EMR)
Digitally store diagnosis, prescriptions, and test reports for quick access. Ensure authorized users can securely access medical data.

5.Billing and Payments
Generate and manage bills for consultations, procedures, and lab tests. Keep a digital billing history in XML format for future reference.

6.Admin Module
Allow administrators to manage hospital resources, users, and reports.
Monitor system usage and ensure data integrity.

7.Reports and Analytics
Generate detailed reports on patient visits, doctor consultations, and revenue. Help management make data-driven decisions.

8.Security and Privacy
Implement role-based access for admin, doctors, and patients.
Protect sensitive medical and personal data from unauthorized access.

CHAPTER 3. DESIGN FLOW/PROCES:-

3.1 Evaluation & Selection of Specifications/Features:

FeatureOverview(Descriptive)
The features of the proposed **Hospital Management System (HMS)** have been carefully evaluated and selected based on the needs of hospital administration, patients, and doctors. The selection ensures the system is user-friendly, efficient, secure, and scalable for future enhancements.

1.Technology Evaluation:-

1.Programming Language: *Java* — chosen for platform independence, object-oriented features, and strong support for web applications.
2.Frontend: *JSP (Java Server Pages)* — allows creation of dynamic and interactive user interfaces.
3.Backend: *Servlets* — handle business logic and server-side processing efficiently.
4.Database / Data Storage: *XML* — provides a lightweight and structured way to store patient records, doctor profiles, and billing data.
5.Web Server: *Apache Tomcat* — selected for easy integration with Javabased web technologies.

2. Functional Features Selected:-

1.Patient Registration & Management
Add, view, update, and delete patient information.
Maintain XML-based digital medical records.
2.Doctor Management
Register doctors and manage their specializations. Maintain consultation schedules and availability.

3.Appointment Booking System
Allow patients to book, cancel, or reschedule appointments. Notify doctors and patients of appointment status.

4.Electronic Medical Records (EMR)
Store and access diagnosis, prescriptions, and lab results electronically. Support retrieval of previous visit data for continuity of care.

5.Billing & Payment Module
Generate bills for consultations, tests, and procedures.
Maintain billing history in XML for transparency and reference.

6.Admin Control Panel
Manage hospital resources, users, and access rights.
Generate system reports and monitor overall performance.

7.Search and Filter Options
Search patients by ID, name, or disease.
Filter doctors by specialization or availability.

8.Notifications and Alerts
Send appointment confirmations, reminders, and follow-up alerts.

9.Reports and Analytics
Generate monthly or yearly hospital performance and patient visit reports.

10.Security Features
Role-based login system for Admin, Doctor, and Patient.
Data protection through restricted access and validation mechanisms.

3. Selection Criteria :-

1.Usability: Simple interface for non-technical users.

2.Reliability: Accurate and consistent performance.

3.Scalability: Easy to expand modules in the future.

4.Maintainability: XML-based data allows easy modification and portability.

5.Security: Protection of sensitive medical information through authentication and access control.

3.2 Analysis of Featuresand Finalization Subject to Constraints:

The development of the **Hospital Management System (HMS)** requires careful analysis of each proposed feature to ensure that it meets the functional requirements, remains technically feasible, and operates efficiently within system constraints such as cost, time, and technology.

1. Feature Analysis:-

1.Patient Registration and Management

Feasibility: Highly feasible using Java Servlets and XML storage.

Impact: Simplifies the process of maintaining accurate and up-to-date patient data.

Constraint: XML file size management and data parsing efficiency.

2.Doctor Management Module

Feasibility: Easily implemented using form-based JSP pages and Servlet backend.

Impact: Helps organize doctor schedules and specializations effectively.

Constraint: Requires synchronization between doctor availability and appointment system.

3.Appointment Booking System

Feasibility: Achievable using JSP forms and Servlet request handling.

Impact: Reduces manual scheduling conflicts and improves patient experience.

Constraint: Must handle concurrent requests and avoid overlapping appointments.

4.Electronic Medical Records (EMR)

Feasibility: XML-based data storage can efficiently maintain patient medical histories.

Impact: Enables quick access to medical history and continuity of treatment.

Constraint: Needs proper XML structure validation and data protection mechanisms.

5.Billing and Payment Module

Feasibility: Simple to generate and manage through Servlets and XML storage.

Impact: Automates payment calculation and maintains billing transparency.

Constraint: Must ensure data accuracy and prevent unauthorized modifications.

6.Admin Panel

Feasibility: Easily implemented with JSP-Servlet integration.

Impact: Centralized control for managing doctors, patients, and reports.

Constraint: Access restricted only to authorized admin users.

7.Reports and Analytics

Feasibility: Data can be extracted and summarized from XML records.

Impact: Helps management make data-driven decisions.

Constraint: Complex data analysis may require structured database integration in the future.

8.Security and Authentication

Feasibility: Implemented using role-based access control (Admin, Doctor, Patient).

Impact: Ensures data privacy and secure user operations.

Constraint: Must implement strong session management and input validation.

2. Constraints Considered:-

1.Technical Constraints:

Use of XML limits large-scale data handling compared to SQL databases.

Requires a stable Java-supported web server like Apache Tomcat.

2.Resource Constraints:

Limited hardware resources and development time for full-scale implementation.

Manual testing instead of automated testing due to resource limitations.

3.Operational Constraints:
Users must have basic technical knowledge to use the system efficiently.
Internet access is required for online operations.

4.Security Constraints:
Sensitive patient information must be encrypted and accessible only to authorized users.

3. Finalization of Features:-

After detailed evaluation, the following features have been finalized for implementation:
1.Patient Registration and Management
2.Doctor Management
3.Appointment Booking System
4.Electronic Medical Records (EMR)
5.Billing and Payments
6.Admin Control Panel
7.Reports and Analytics
8.Notifications and Reminders
9.Role-based Security System

3.3. Design Flow/Sample Code:

Login Page:-

```
<%@   page   contentType="text/html;charset=UTF-8"   language="java"   %>
<html>
<head>
   <title>Login - Hospital Management</title>
   <link rel="stylesheet" href="css/style.css">
   <style> body { font-family: Arial,
      sans-serif;    background-color:
      #f3f6fa;  display:  flex;  justify-
      content:    center;    align-items:
      center; height: 100vh;
      }
      .form-container {
        background: #fff;
        padding: 30px; border-
        radius: 10px;
        box-shadow:   0   0   10px   rgba(0,0,0,0.1);
        width: 350px;
      } h2 { text-align: center;
      margin-bottom:    20px;
      color: #333;
      }
      .form-group      {      margin-
        bottom: 15px;
      }
      label {
        font-weight: bold;
      }
input, select { width: 100%;
      padding: 8px; margin-
      top: 5px; border: 1px
      solid #ccc;
       border-radius: 5px;
     }
     .btn { width: 100%; padding:
       10px;     background-color:
       #007bff;
       color:           white;
       border:           none;
       border-radius: 5px;
       cursor: pointer;
     }
     .btn:hover { background-color:
       #0056b3;
     }
     .error-message          {
       color: red;
       text-align: center;
     }   p   {   text-align:
     center;
```

```
        }
     </style>
  </head>
  <body>
  <div class="form-container">
     <h2>Hospital Login</h2>
     <form action="login" method="post">
        <div class="form-group">
           <label for="id">Login ID (Contact # or Name):</label>
           <input type="text" id="id" name="id" required>
 </div>
        <div class="form-group">
           <label for="password">Password:</label>
           <input      type="password"      id="password"      name="password"
required>
        </div>
        <div class="form-group">
           <label for="role">Login As:</label>
           <select id="role" name="role">
              <option value="patient">Patient</option>
              <option value="doctor">Doctor</option>
              <option value="admin">Admin</option>
           </select>
        </div>
        <button type="submit" class="btn">Login</button>
     </form>
     <%  String  error  =  (String)  request.getAttribute("error");  if
       (error != null) { %>
       <p class="error-message"><%= error %></p>
     <% } %>
     <p>Don't have an account? <a href="patientregistration.jsp">Register as
Patient</a></p>
</div>
</body>
</html>
```

Add Appointment.java:package com.hospital.servlets;

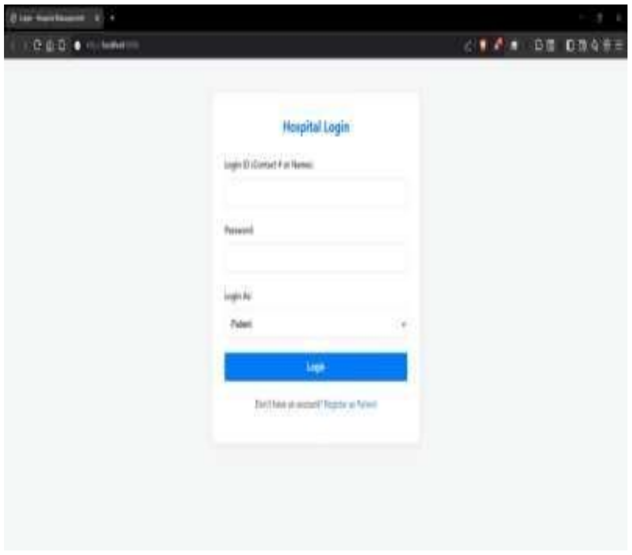
```
import com.hospital.dao.AppointmentDAO;
import com.hospital.model.Appointment;
import com.hospital.model.Patient; import
javax.servlet.ServletException; import
javax.servlet.annotation.WebServlet;
import javax.servlet.http.*; import
java.io.IOException; import
java.util.UUID;

@WebServlet("/addAppointment") public class
AddAppointmentServlet extends HttpServlet { private
AppointmentDAO appointmentDAO;

    @Override
    public void init() {
        appointmentDAO = new AppointmentDAO();
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        String doctorId = request.getParameter("doctorId"); String date
        = request.getParameter("date");

        HttpSession session = request.getSession(false); Patient patient
        = (Patient) session.getAttribute("user");

        if (patient == null) {
            response.sendRedirect("index.jsp");
            return;
        }
        String appointmentId = "appt_" + UUID.randomUUID().toString().
substring(0, 8);
        String patientId = patient.getId();
        String status = "Booked";

        Appointment appointment = new Appointment(appointmentId, patientId,
doctorId, date, status); appointmentDAO.addAppointment(appointment);

        request.setAttribute("appMessage", "Appointment booked successfully!");
        request.getRequestDispatcher("patient-dashboard.jsp").forward(request,
response);
    }
}
```

<u>Login.java:</u>package

```
com.hospital.servlets;

import com.hospital.dao.*; import
com.hospital.model.*; import
javax.servlet.*;
import javax.servlet.annotation.WebServlet;
import  javax.servlet.http.*;  import
java.io.IOException;

@WebServlet("/login")
public class LoginServlet extends HttpServlet { private
    PatientDAO patientDAO = new PatientDAO(); private
    DoctorDAO doctorDAO = new DoctorDAO(); @Override
    protected void doPost(HttpServletRequest req,
    HttpServletResponse res) throws ServletException,
    IOException {

         String id = req.getParameter("id");
         String pass = req.getParameter("password");
    String   role   =   req.getParameter("role");   HttpSession
    session = req.getSession();

    switch  (role)  {  case  "admin":  if  ("admin".equals(id)  &&
        "admin".equals(pass))     {     session.setAttribute("user",
        "admin");        session.setAttribute("role",        "admin");
        res.sendRedirect("admin-panel.jsp");
            } else showError(req, res, "Invalid Admin credentials"); break;

        case "patient":
            Patient p = patientDAO.getPatientByContact(id); if (p
            !=     null     &&     p.getPassword().equals(pass))     {
            session.setAttribute("user",                              p);
            session.setAttribute("role",                      "patient");
            res.sendRedirect("patient-dashboard.jsp");
            } else showError(req, res, "Invalid Patient credentials");

        case "doctor":
            Doctor d = doctorDAO.getDoctorByName(id); if (d
            !=     null     &&     d.getPassword().equals(pass))     {
            session.setAttribute("user",                           d);
            session.setAttribute("role",                    "doctor");
            res.sendRedirect("doctor-dashboard.jsp");
            } else showError(req, res, "Invalid Doctor credentials"); break;

        default:
            showError(req, res, "Invalid role selected");
    }
}
```

CHAPTER4. RESULTS ANALYSIS AND VALIDATION: -

4.1. <u>Implementation of solution:</u>



Hospital Login

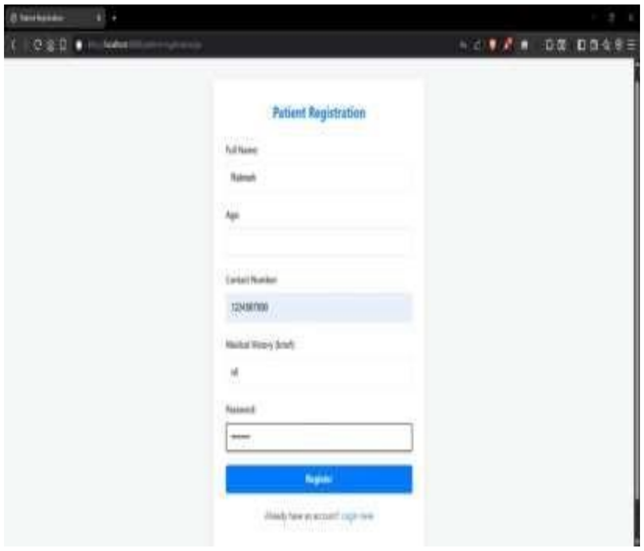
Login ID (Contact # or Name)

Password

Login As: Patient

Login

Don't have an account? Register as Patient



Patient Registration

Full Name: Ratnesh

Age

Contact Number: 1234567890

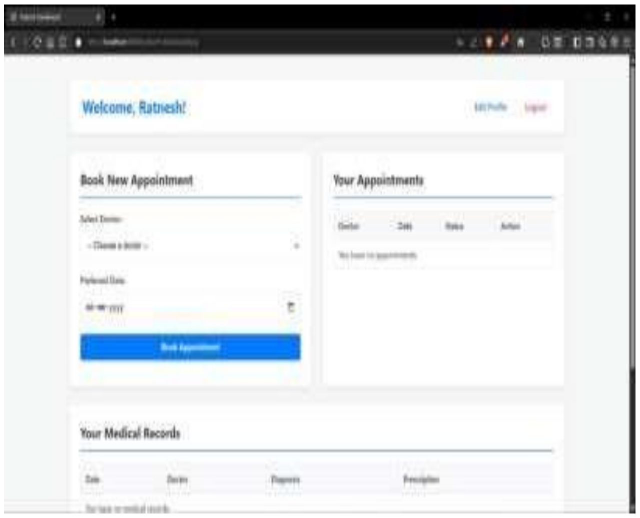
Medical History (brief): nil

Password: ••••••

Register

Already have an account? Login here

login page patient registration



Welcome, Ratnesh! Edit Profile Logout

Book New Appointment

Select Doctor: -- Choose a doctor --

Preferred Date: dd-mm-yyyy

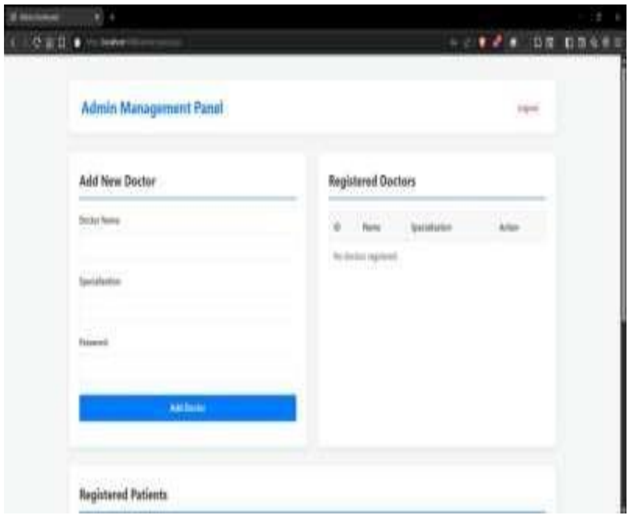
Book Appointment

Your Appointments

Doctor	Date	Status	Action
You have no appointments.			

Your Medical Records

Date	Doctor	Diagnosis	Prescription
You have no medical records.			



Admin Management Panel Logout

Add New Doctor

Doctor Name

Specialization

Password

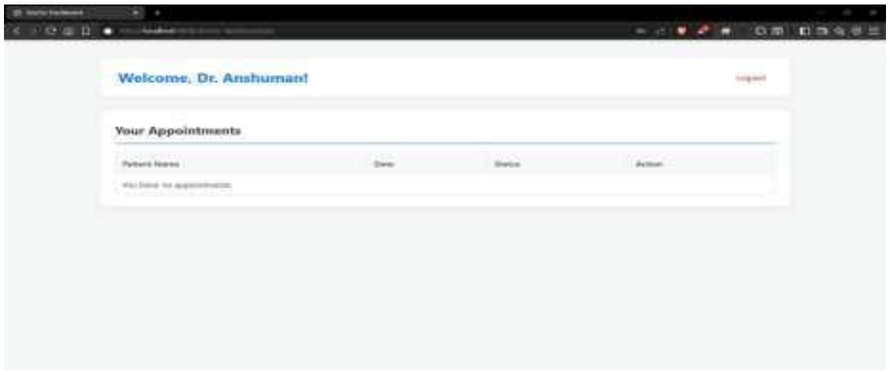
Add Doctor

Registered Doctors

ID	Name	Specialization	Action
No doctors registered.			

Registered Patients

patient dasboard admin dashboard to add credential



Welcome, Dr. Anshuman! Logout

Your Appointments

Patient Name	Date	Status	Action
You have no appointments.			

docter dasboard

CHAPTER5. CONCULSION AND FUTURE WORK:-

5.1. Conclusion:

The **Hospital Management System** successfully automates and streamlines hospital operations by integrating various processes such as patient registration, doctor management, appointment scheduling, billing, and record maintenance into a single digital platform. It minimizes manual effort, reduces errors, and enhances data accuracy and accessibility.

Through features like XML-based data storage, secure login for different roles (admin, doctor, and patient), and efficient handling of medical records, the system improves the overall efficiency and quality of healthcare services. It ensures that both patients and hospital staff can easily manage appointments, view reports, and maintain confidentiality of sensitive information.

Overall, this project demonstrates how technology can transform hospital management by providing a **reliable, user-friendly, and secure solution** that supports better decision-making and patient care.

5.2. Future Work:

Although the current Hospital Management System covers core functionalities like patient registration, appointment booking, doctor management, and billing, there is still room for future enhancement. Some possible improvements include:

1.Online Payment Integration – Adding secure payment gateways for consultation and billing through UPI, credit/debit cards, or net banking.

2.Mobile Application Support – Developing an Android/iOS app to allow patients and doctors to access the system on the go.

3.Automated Notifications – Integrating SMS or email alerts for appointment reminders, test reports, and billing updates.

4.AI-based Diagnosis Support – Implementing AI tools to suggest possible diagnoses or treatments based on patient symptoms and history.

5.Cloud Data Storage – Shifting from local XML storage to cloud-based databases for scalability and real-time access.

6.Telemedicine Module – Adding video consultation features for remote treatment and follow-ups.

7.Data Analytics Dashboard – Providing advanced analytics to track hospital performance, patient trends, and doctor workload.

8.Multi-language Support – Making the system accessible to users in different regional languages for wider usability.