HOSPITAL MANAGEMENT SYSTEM

MINI PROJECT REPORT

Submitted by

HARISH KRISHNAN P (953623205018)

BALASHANKAR R (953623205006)

MANISH VARDHAN G (953623205027)

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

RAMCO INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

RAJAPALAYAM-626 108

ANNA UNIVERSITY: CHENNAI 600 025

ABSTRACT

The Hospital Management System is a web-based application designed to streamline hospital administration tasks. It manages patient records, doctor assignments, appointments, and department details efficiently. The system replaces traditional manual methods, reducing errors and saving time through automation.

Core features include patient registration, doctor and department management, and appointment scheduling. Administrators can easily manage records and assign doctors to specific departments. The platform provides real-time access to essential information like doctor specializations and appointment timings.

Developed using HTML, CSS, PHP, and MySQL with XAMPP, it ensures secure and accurate data handling. Suitable for hospitals, clinics, and health centers, it offers a cost-effective solution to improve healthcare operations.

TABLE OF CONTENTS

1.INTRODUCTION

In today's digital era, managing healthcare operations—especially in hospitals and clinics—requires organized, accurate, and accessible systems. Hospitals regularly deal with tasks such as registering patients, managing doctors, scheduling appointments, organizing departments, and maintaining medical records. Relying on traditional manual methods for handling this data can be inefficient, error-prone, and difficult to scale as the number of patients and healthcare professionals increases.

This project, titled "Hospital Management System", is designed to simplify and automate the everyday activities involved in managing hospital operations. The goal is to develop a user-friendly, web-based, and cost-effective solution that helps administrators streamline their tasks while providing a clear and structured interface to view and manage patients, doctors, appointments, and departments.

The system supports essential functionalities such as adding and deleting patient records, assigning doctors to departments, scheduling appointments, and maintaining detailed visit histories. Additionally, it includes a view to display patient information alongside their assigned doctor and department, making it easy to monitor appointments and responsibilities. Built using HTML, CSS, PHP, and MySQL via XAMPP, the application runs entirely on a local server and can be accessed from any browser without the need for complex infrastructure.

This system is especially useful for clinics, hospitals, or health centers looking for a centralized platform to digitize their hospital operations and enhance administrative efficiency.

1.1 PROJECT DESCRIPTION

This project focuses on developing a simple, efficient, and user-friendly Hospital Management System that automates key tasks involved in running hospital operations. The system allows administrators to manage patient registrations, assign doctors, schedule appointments, and organize departments—all through a centralized interface. It also supports maintaining medical records and displaying the overall structure of hospital data.

The application enables smooth patient management, including adding new patients, updating their details, and removing entries when necessary. Doctor assignment and department categorization are streamlined, making it easy to manage hospital staff and departmental structure. Appointment scheduling includes selecting the doctor, setting appointment dates, and linking patients to their visits. Additionally, administrators can view and manage each patient's visit history and medical details.

The system includes a clean reporting view that shows each registered patient along with their assigned doctor and department, giving a complete overview of the hospital's workflow and patient distribution. Designed with a modular and scalable architecture, the project is built using PHP, MySQL, HTML, and CSS, and runs on a local server via XAMPP.

By replacing manual registers and fragmented record-keeping methods with a structured digital platform, this Hospital Management System enhances accuracy, reduces administrative workload, and ensures a more organized approach to healthcare operations. The platform is ideal for clinics, hospitals, and health centers looking to modernize their processes without requiring complex or expensive infrastructure.

2.SYSTEM REQUIREMENTS

The Hospital Management System is designed to run on a local server environment using XAMPP, which includes Apache, MySQL, and PHP. The system can be accessed through any modern web browser such as Google Chrome or Mozilla Firefox. It requires only basic hardware specifications, including a dual-core processor, 4 GB of RAM, and at least 250 GB of hard disk space for smooth operation.

The system is fully compatible with both Windows and Linux operating systems. It is recommended to use PHP version 7.0 or above to support the backend functionality, and MySQL is used for managing all database operations. The frontend is developed using HTML and CSS to provide a clean and responsive user interface.

Administrators can access the application securely through a browser interface, and the database can be managed and backed up using phpMyAdmin. The system layout is responsive and optimized for use on both desktops and laptops, ensuring ease of access and usability for various user environments.

2.1 HARDWARE & SOFTWARE REQUIREMENTS

Hardware Requirements

Component	Specification
Processor	Intel Core i3 or higher
RAM	4 GB minimum
Hard Disk	250 GB or more

Display	1024 × 768 resolution or higher
Input Devices	Keyboard, Mouse

Software Requirements

Software	Description
Operating System	Windows 10 / 11 or Linux
Web Server	Apache (via XAMPP)
Database	MySQL (via XAMPP)
Scripting Language	PHP
Frontend	HTML, CSS, JavaScript
Browser	Google Chrome / Mozilla Firefox
Development Tools	VS Code / Notepad++

2.2 SOFTWARE REQUIREMENTS SPECIFICATION(SRS)

2.2.1 INTRODUCTION

The Hospital Management System allows administrators to manage patients, doctors, appointments, and departmental information efficiently through a web-based interface. It supports patient registration, doctor assignment, department organization, appointment scheduling, medical record maintenance, and viewing hospital structure, helping healthcare providers streamline their workflow and record-keeping.

2.2.2 FUNCTIONAL REQUIREMENTS

• Add, update, and delete patient records
• Create and manage doctor and staff details
• Assign doctors to respective departments
• Schedule appointments between patients and doctors
• Set and update patient diagnosis and treatment status
• View patients along with their assigned doctors and room details
• Display department, doctor, and staff information
• Generate reports and summaries for patients, doctors, and appointments

2.2.3 NON-FUNCTIONAL REQUIREMENTS

• The system should be accessible through modern web browsers
• Response time for database operations should not exceed 2 seconds
• All data should be securely handled and validated
• System should be easy to back up and restore using phpMyAdmin in XAMPP
• The interface should be clean, responsive, and user-friendly
• Compatible with Windows and Linux environments using XAMPP stack

2.2.4 DATABASE TABLES

1) **Patients** – Stores patient details including name, age, gender, and contact number
2) **Appointments** – Stores appointment records including patient ID, doctor ID, appointment date, and status

3. IMPLEMENTATION

- **XAMPP:** For hosting Apache and MySQL locally
- **PHP:** For backend scripting
- **MySQL:** Database to store and retrieve data
- **HTML/CSS/JavaScript:** For frontend design

3.1 DATABASE TABLE STRUCTURES

- **Patients (patients)**
  patient_id (Primary Key)
  FullName
  age
  gender
  Contact
  AdmissionDate

- Doctors(doctors)
  DoctorID
  FullName
  Specialization
  Contact

- **Appointments (appointments)**
  appointment_id (Primary Key)
  patient_id (Foreign Key)
  doctor_id
  appointment_date
  Notes

3.2 ER DIAGRAM

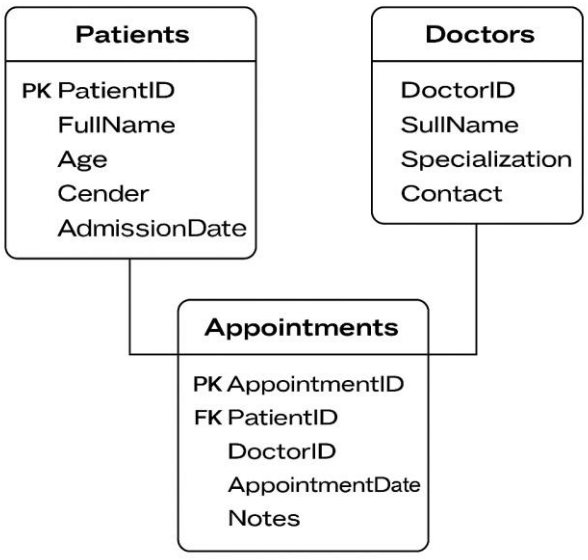


Fig.1 ER Diagram

3.3 QUERY

1. Creation of Database names hospital db.

```
CREATE DATABASE hospital_db;

USE hospital_db;
```

2. Creation of Patients Table.

```
CREATE TABLE IF NOT EXISTS Patients (
    PatientID INT AUTO_INCREMENT PRIMARY KEY,
    FullName VARCHAR(100) NOT NULL,
    Age INT,
    Gender VARCHAR(10),
    Contact VARCHAR(20),
    AdmissionDate DATE
);
```

3. Creation of Doctors Table.

```
CREATE TABLE IF NOT EXISTS Doctors (
    DoctorID INT AUTO_INCREMENT PRIMARY KEY,
    FullName VARCHAR(100) NOT NULL,
    Specialization VARCHAR(100),
    Contact VARCHAR(20)
);
```

3.4 CODE

```
<?php
// Database Config
$conn = new mysqli("localhost", "root", "", "hospital_db");
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Handle Add Patient
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['add_patient'])) {
    $stmt = $conn->prepare("INSERT INTO Patients (FullName, Age, Gender, Contact, AdmissionDate) VALUES (?, ?, ?, ?, ?)");
    $stmt->bind_param("sisss", $_POST['fullname'], $_POST['age'], $_POST['gender'], $_POST['contact'], $_POST['admission']);
    $stmt->execute();
}

// Handle Add Doctor
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['add_doctor'])) {
    $stmt = $conn->prepare("INSERT INTO Doctors (FullName, Specialization, Contact) VALUES (?, ?, ?)");
    $stmt->bind_param("sss", $_POST['dname'], $_POST['specialization'], $_POST['dcontact']);
    $stmt->execute();
}

// Handle Add Appointment
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['add_appointment'])) {
    $stmt = $conn->prepare("INSERT INTO Appointments (PatientID, DoctorID, AppointmentDate, Notes) VALUES (?, ?, ?, ?)");
    $stmt->bind_param("iiss", $_POST['patient_id'], $_POST['doctor_id'], $_POST['date'], $_POST['notes']);
    $stmt->execute();
```

```
}
?>

<!DOCTYPE html>
<html>
<head>
  <title>Hospital Management System</title>
  <style>
    body {
      font-family: 'Segoe UI', sans-serif;
      background: linear-gradient(to right, #e0f7fa, #ffffff);
      margin: 0;
      padding: 0;
    }

    header, footer {
      background-color: #005792;
      color: white;
      text-align: center;
      padding: 25px 0;
      font-size: 28px;
      letter-spacing: 1px;
    }

    nav {
      background-color: #007BFF;
      display: flex;
      justify-content: center;
      padding: 10px 0;
    }

    nav a {
      color: white;
      text-decoration: none;
      margin: 0 15px;
      padding: 12px 18px;
      background-color: #0056b3;
      border-radius: 6px;
      transition: background 0.3s ease;
    }

    nav a:hover {
      background-color: #003d80;
    }

    .container {
      padding: 30px 10%;
    }
```

```
        h2 {
            color: #005792;
            margin-top: 50px;
        }

        form {
            background: #ffffff;
            padding: 25px;
            border-radius: 10px;
            margin-top: 20px;
            box-shadow: 0 0 15px
rgba(0, 0, 0, 0.05);
        }

        input[type="text"],
input[type="number"],
input[type="date"], select, textarea {
            width: 100%;
            padding: 10px;
            margin-top: 10px;
            margin-bottom: 20px;
            border: 1px solid #ccc;
            border-radius: 6px;
            box-sizing: border-box;
        }

        textarea {
            resize: vertical;
        }

        input[type="submit"] {
            background-color: #28a745;
            color: white;
            font-weight: bold;
            border: none;
            padding: 12px 20px;
            cursor: pointer;
            border-radius: 6px;
        }

        input[type="submit"]:hover {
            background-color: #218838;
        }

        table {
            width: 100%;
            border-collapse: collapse;
            background-color: white;
            margin-top: 40px;
            border-radius: 10px;
            overflow: hidden;
        }
```

```
        th, td {
            padding: 14px;
            border-bottom: 1px solid #ddd;
            text-align: center;
        }

        th {
            background-color: #007BFF;
            color: white;
        }

        tr:hover {
            background-color: #f1f1f1;
        }

        footer {
            font-size: 16px;
        }
    </style>
</head>
<body>
<header>
    Hospital Management System
</header>

<nav>
    <a href="#add_patient">Add Patient</a>
    <a href="#add_doctor">Add Doctor</a>
    <a href="#add_appointment">Book Appointment</a>
    <a href="#appointments">View Appointments</a>
</nav>

<div class="container">

    <h2 id="add_patient">🩺 Add Patient</h2>
    <form method="post">
        <input type="hidden" name="add_patient" value="1">
        Full Name: <input type="text" name="fullname" required>
        Age: <input type="number" name="age" required>
        Gender:
        <select name="gender">
            <option>Male</option>
            <option>Female</option>
            <option>Other</option>
        </select>
```

Contact: <input type="text" name="contact" required>

Admission Date: <input type="date" name="admission" required>

<input type="submit" value="Add Patient">

</form>

<h2 id="add_doctor">👨‍⚕️ Add Doctor</h2>

<form method="post">

<input type="hidden" name="add_doctor" value="1">

Full Name: <input type="text" name="dname" required>

Specialization: <input type="text" name="specialization" required>

Contact: <input type="text" name="dcontact" required>

<input type="submit" value="Add Doctor">

</form>

<h2 id="add_appointment">🗓️ Book Appointment</h2>

<form method="post">

<input type="hidden" name="add_appointment" value="1">

Patient:

<select name="patient_id" required>

<option value="">-- Select Patient --</option>

<?php

$patients = $conn->query("SELECT * FROM Patients");

while ($p = $patients->fetch_assoc()) {

echo "<option value='{$p['PatientID']}'>{$p['FullName']}</option>";

}

?>

</select>

Doctor:

<select name="doctor_id" required>

<option value="">-- Select Doctor --</option>

<?php

$doctors = $conn->query("SELECT * FROM Doctors");

```
        while ($d = $doctors->fetch_assoc()) {
            echo "<option value='{$d['DoctorID']}'>{$d['FullName']}</option>";
        }
        ?>
    </select>
    Date: <input type="date" name="date" required>
    Notes: <textarea name="notes" rows="3" placeholder="Any important notes..."></textarea>
    <input type="submit" value="Add Appointment">
  </form>

  <h2 id="appointments">📋 All Appointments</h2>
  <table>
    <tr>
      <th>ID</th>
      <th>Patient</th>
      <th>Doctor</th>
      <th>Specialization</th>
      <th>Date</th>
      <th>Notes</th>
    </tr>
    <?php
    $result = $conn->query("
      SELECT
        a.AppointmentID, p.FullName AS PatientName, d.FullName AS DoctorName,
        d.Specialization, a.AppointmentDate, a.Notes
      FROM Appointments a
      JOIN Patients p ON a.PatientID = p.PatientID
      LEFT JOIN Doctors d ON a.DoctorID = d.DoctorID
      ORDER BY a.AppointmentDate DESC
    ");
    while ($row = $result->fetch_assoc()) {
      echo "<tr>
        <td>{$row['AppointmentID']}</td>
        <td>{$row['PatientName']}</td>
        <td>{$row['DoctorName']}</td>
        <td>{$row['Specialization']}</td>
        <td>{$row['AppointmentDate']}</td>
```

```
<td>{$row['Notes']}</td>
            </tr>";
        }
        ?>
    </table>
</div>
<footer>
    &copy; 2025 Hospital Management System. Designed with  for better care.
</footer>
</body>
</html>
```

4. EXPERIMENTAL RESULTS

The Hospital Management System was thoroughly tested to ensure smooth functionality across all modules and user interactions. Core operations such as patient registration and appointment booking were performed successfully without system errors or data conflicts.

Each module was validated for data integrity and usability. The system reliably handled the addition of patient records, storing details like name, age, gender, and contact. Appointment booking with patient ID, doctor ID, date, and status was executed accurately. All input forms were tested using valid and invalid data, including empty fields and incorrect formats. The system consistently displayed proper validation messages and prevented the insertion of incorrect data.

Performance tests were conducted through repeated user simulation. Operations such as registering a patient or booking an appointment were completed with an average response time of under two seconds. The MySQL backend handled relational data between the patients and appointments tables effectively, ensuring foreign key relationships were preserved and maintained during data modifications.

The web interface was tested for responsiveness and cross-browser compatibility. It performed reliably on major browsers including Google Chrome and Mozilla Firefox. The design remained user-friendly and accessible on both Windows and Linux environments using the XAMPP stack. Overall, the system demonstrated high levels of stability, efficiency, and readiness for real-world hospital or clinic deployment.

4.1 OUTPUT SCREEN

1.Operations

1.1 Add Team1

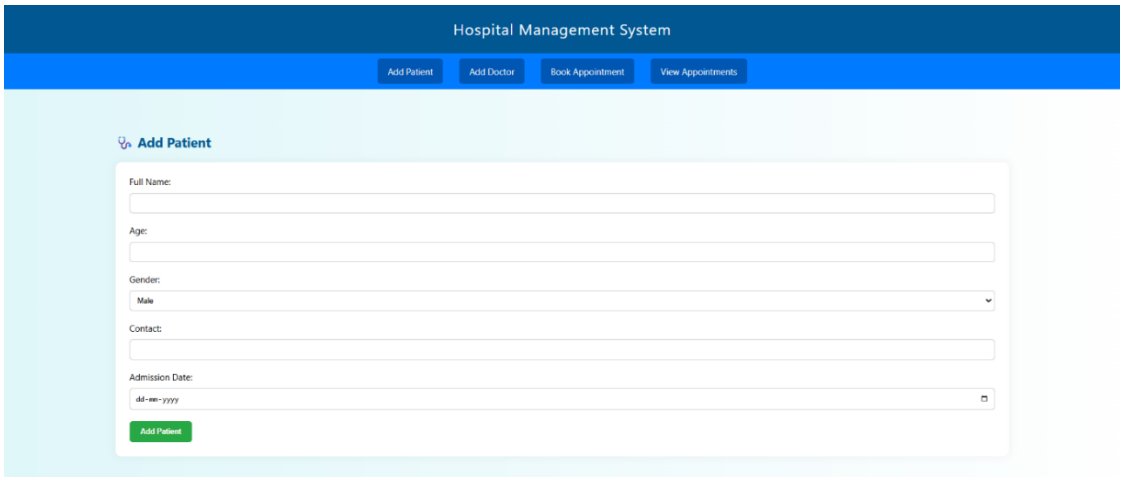


Fig.2 Add Patient

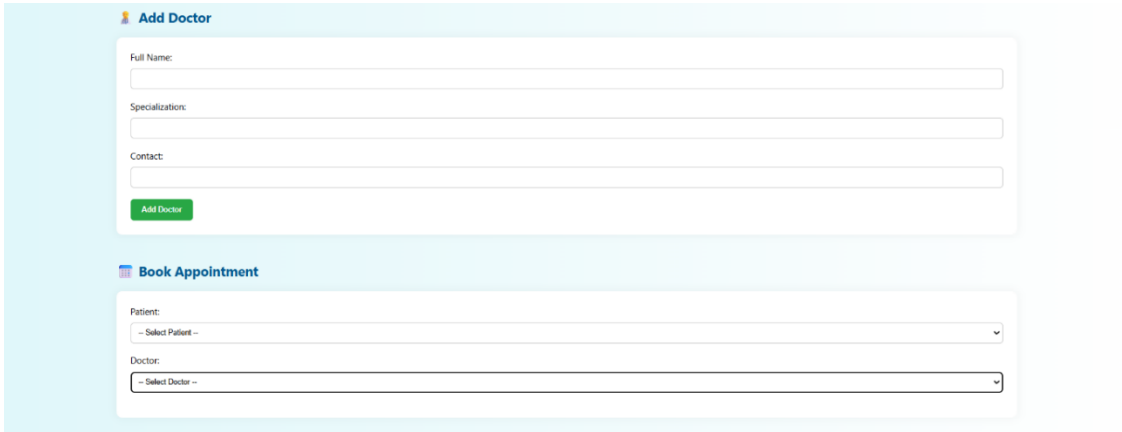


Fig.3 Add Doctor

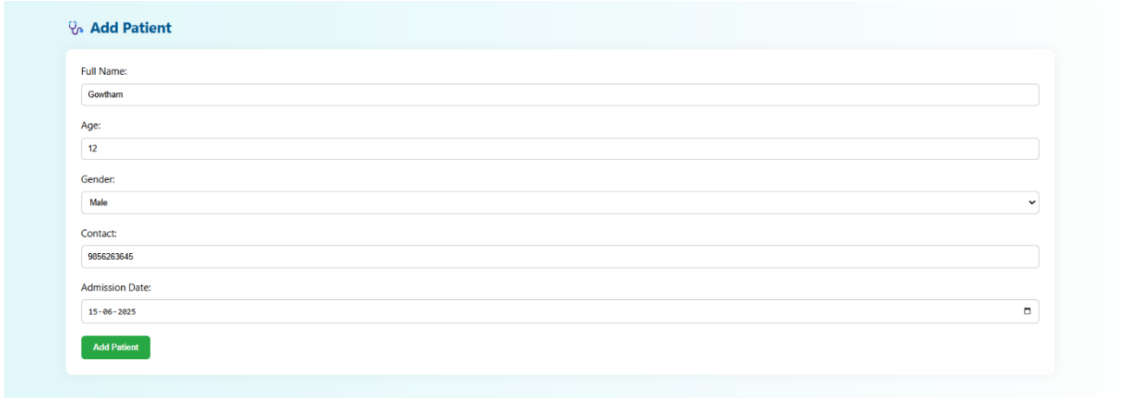


Fig.4 Add Patient Details

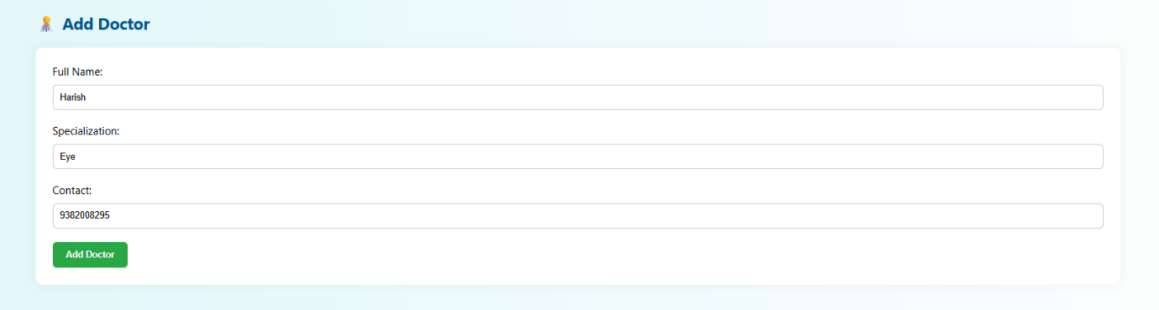


Fig.5 Add Doctor Details

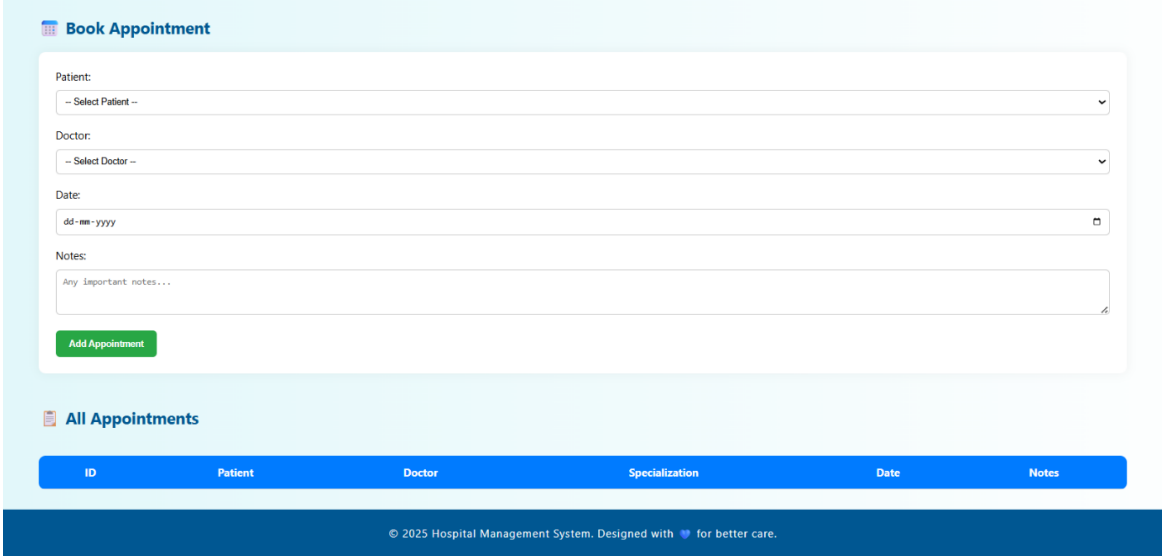


Fig.6 Book Appointment

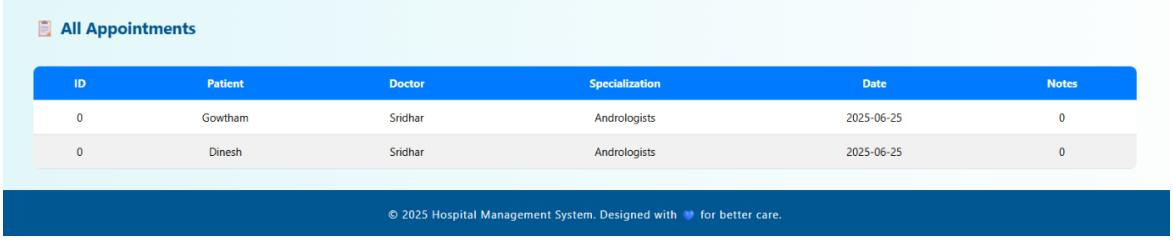
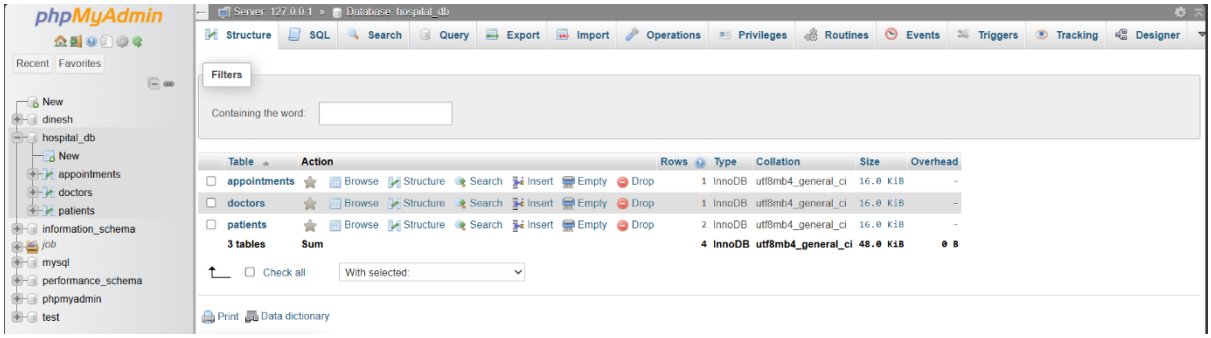


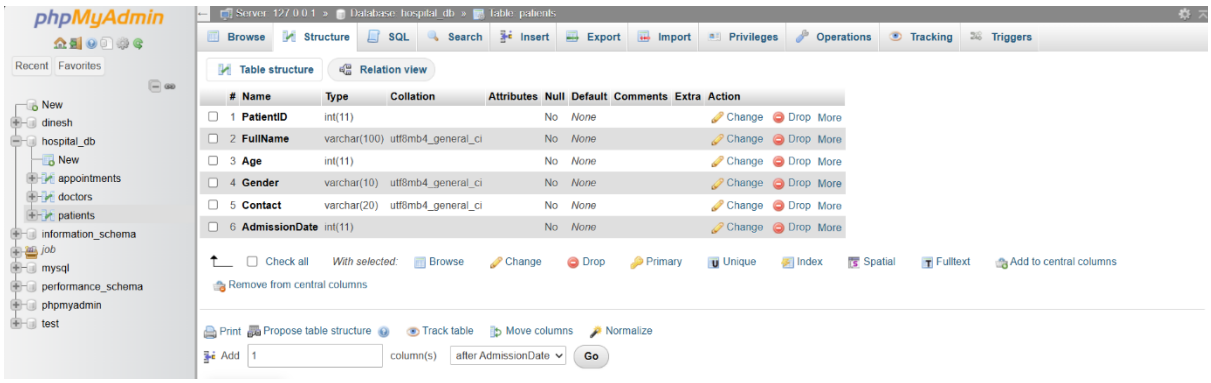
Fig.7 All Appointments

2. TABLES

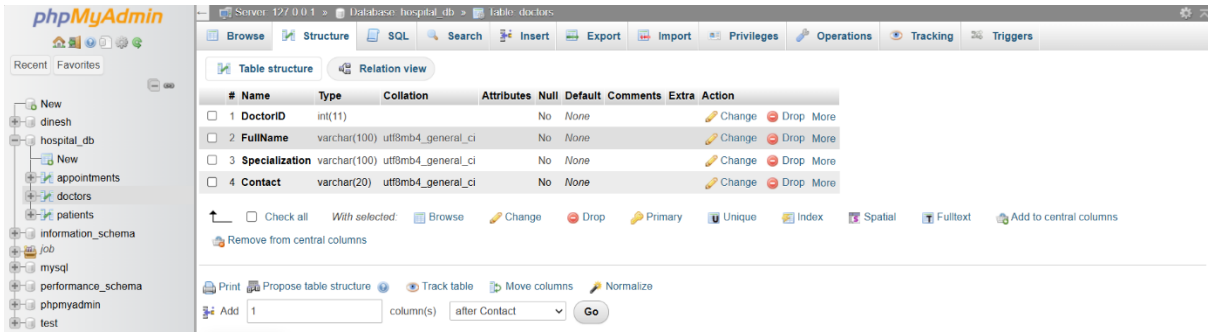
2.1 Category



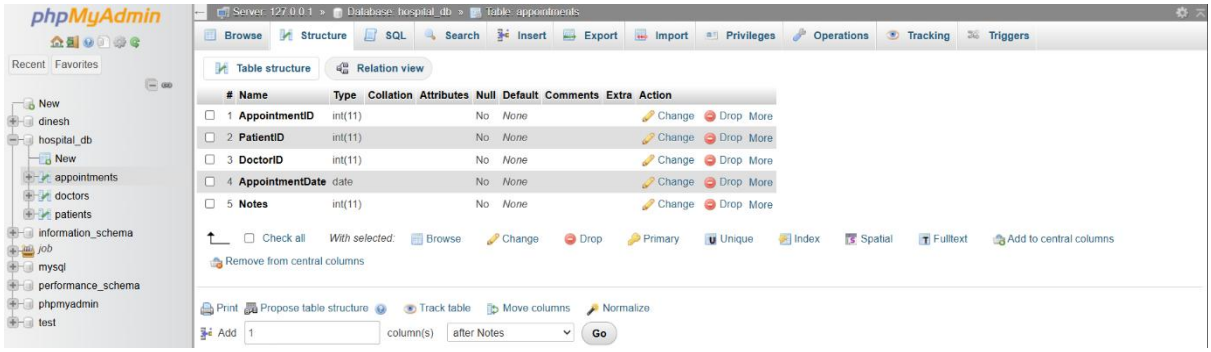
2.2 Patients



2.3 Doctors



2.4 Appointments



5.CONCLUSION

The Hospital Management System provides a reliable and efficient digital solution for managing hospital-related activities. By replacing traditional manual processes, the system improves accuracy, reduces administrative burden, and minimizes the risk of human error. It enables healthcare administrators to seamlessly register patients and schedule appointments through a centralized web-based interface.

The system's interface is clean and user-friendly, making it accessible even to users with minimal technical knowledge. Designed with scalability and modularity in mind, it supports future upgrades such as doctor management, treatment tracking, room assignments, or integration with SMS/email notification services. The backend database ensures strong relational integrity, maintaining consistent and organized records between patients and appointments.

Overall, this project successfully fulfills its objective of streamlining hospital operations. It offers a practical, easy-to-use platform for clinics, health centers, and hospitals to manage patient data and appointment scheduling efficiently—enhancing the overall administrative experience for staff and users alike.

6.REFERENCES

1. PHP Documentation – https://www.php.net/docs.php
2. MySQL Reference Manual – https://dev.mysql.com/doc/
3. W3Schools – HTML, CSS, and JavaScript Tutorials – https://www.w3schools.com
4. XAMPP Installation Guide – https://www.apachefriends.org/index.html
5. Stack Overflow – Technical Support Forum – https://stackoverflow.com
6. ER Diagram Examples – TutorialsPoint and GeeksforGeeks
7. Open Source Projects on GitHub – Library Management System examples