Hospital Management System – Project Report

Title Page

Project Title: Hospital Management System

Student Name: Nitish bisht

Roll Number: 23BCA10641

Course / Department: BCA

Abstract

The Hospital Management System (HMS) is designed to simplify and automate the core operations of a healthcare facility. The objective of this project is to develop a structured system that manages patient records, doctor information, appointments, billing, and notifications. This system uses relational database design, Laravel migrations, and RESTful APIs to create a scalable and efficient management solution. The outcome of this project is a functional system that improves accuracy, reduces manual work, and enhances communication through automated email/SMS notifications.

Introduction

Healthcare institutions deal with large amounts of data involving patients, doctors, appointments, and billing. Traditional paper-based management leads to errors, data loss, and inefficiency. With technological advancements, hospitals require a digital system that ensures fast access to information and reliable record management.

The Hospital Management System aims to fulfill these needs by providing a centralized digital platform. This project is motivated by the need to improve workflow efficiency, minimize human error, and ensure secure access to vital medical information.

The main objectives include designing a relational database, building secure APIs using Laravel, and ensuring seamless communication between modules.

Problem Statement / Objectives

Problem Statement

Hospitals face challenges in maintaining accurate patient records, scheduling appointments, tracking doctor availability, and generating billing information. Manual processes are time-consuming and prone to mistakes.

Objectives

- Develop a system to manage patient data, doctor profiles, and appointments.
- Create RESTful APIs using Laravel for smooth data communication.
- Implement relational database design and Laravel migrations.
- Automate doctor–patient assignment.
- Generate real-time reports for administration.
- Implement email/SMS notifications for appointments and updates.

System Design / Approach

The system follows a modular, API-driven architectural .

System Architecture

- **Frontend:** Any web or mobile interface (optional).
- **Backend:** Laravel RESTful API.
- **Database:** MySQL (Relational model).
- **Notification Services:** Email (SMTP), SMS API.

Database Design / ER Diagram Components

- Patients table
- Doctors table
- Appointments table
- Bills table
- Notifications table

Key relationships include:

- One-to-Many: Doctor → Appointments
- One-to-Many: Patient → Appointments

One-to-One: Appointment → Bill

Flowcharts / Logic

- **User Flow:** Registration → Data Entry →

  Appointment Booking → Doctor Assignment →

  Billing → Notification

- **System Flow:** API Request → Controller → Model →

  Validation → Database → Response

Implementation

This project is built using Laravel, following MVC architecture and modular coding principles.

Key Modules

1. **Patient Module** – CRUD operations using Laravel controllers & migrations.
2. **Doctor Module** – Specialization, schedule, and assignment features.
3. **Appointment Module** – REST APIs for booking, updating, and notifications.
4. **Billing Module** – Auto-generated billing after appointment completion.
5. **Notification Module** – Email/SMS alerts triggered via Laravel Events.

Sample Code Snippets

Laravel Migration Example:

```
Schema::create('patients', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->integer('age');
    $table->string('gender');
    $table->string('phone');
    $table->timestamps();
});
```

Appointment Booking API Example:

```
public function store(Request $req) {
    $appointment = Appointment::create([
        'patient_id' => $req->patient_id,
        'doctor_id' => $req->doctor_id,
        'date' => $req->date,
        'time' => $req->time,
    ]);
    event(new AppointmentBooked($appointment));
    return response()->json(['success' => true]);
}
```

Results / Testing

Testing Methods Used:

- Unit testing (Laravel PHPUnit)
- API testing with Postman
- Manual functional testing: appointments, billing, notifications

Screenshots / Output Samples:

- Patient registration form
- Doctor list and availability
- Appointment confirmation screen
- Billing invoice
- Email/SMS notification proof

Testing Outcomes:

- All CRUD operations performed successfully.
- APIs responded within acceptable time limits.
- Notifications were delivered correctly.
- Database handled records efficiently.

Conclusion

The Hospital Management System successfully automates essential hospital operations, including patient management, doctor assignment, appointment scheduling, and billing. The use of Laravel migrations and REST APIs ensures scalability and maintainability. Notifications enhance patient engagement and reduce communication gaps. Future improvements may include AI-based appointment prediction, real-time doctor tracking, mobile app integration, and insurance claim automation.

References

1. Laravel Official Documentation – nnhhttps://laravel.com/docs
2. MySQL Database Design Guides
3. RESTful API Best Practices
4. SMS Gateway API Documentation
5. Healthcare IT Standards and Articles