AMRITA VISHWA VIDYAPEETHAM

School of Computing

19CSE441

Full Stack Development

Hospital Management System

Group No.10

Amarnath Rao	AM.EN.U4CSE21267
Hemanth Lakshman	AM.EN.U4CSE21227
Kowshik	AM.EN.U4CSE21271
Mahesh Kumar	AM.EN.U4CSE21263

Project scope

The Hospital Management Application is a robust web-based system developed to facilitate efficient management of hospital operations. The system allows doctors and administrators to perform their respective duties using role-based dashboards. The project aims to:

- Reduce manual errors by automating hospital processes.
- Provide secure, role-specific access to sensitive data.
- Improve hospital workflows by enabling CRUD (Create, Read, Update, Delete) operations for managing patients, appointments, and medicines.

Doctor Dashboard:

- View the list of patients currently under treatment.
- Access detailed diagnostic data for individual patients.
- Perform CRUD operations for patient management.
- Manage a list of medicines, adding or updating as required.

Admin Dashboard:

- View the list of patients with non-sensitive details only.
- Manage the appointment list, with complete CRUD functionality for scheduling and updating appointments.

Technical Scope:
The application is developed using Angular 14 as the frontend framework, Spring Boot as the backend framework, and MySQL8 for data storage. It employs RESTful APIs for seamless communication between the frontend and backend.

Features and Ownership Table

Feature/Task	Description	Owner

Home Page	Displays an overview of the hospital system	Amarnath Rao
Doctor Login & Dashboard	Provides doctors access to patient and medicine management	Hemanth Lakshman
Patient Management (CRUD)	Create, update, and manage patient details	Kowshik
Medicine Management (CRUD)	Add or modify medicines for hospital inventory	Kowshik
Admin Login & Dashboard	Allows admin to view patient and manage appointment details	Mahesh Kumar
Appointment Management (CRUD)	Manage hospital appointments through the admin dashboard	Mahesh Kumar
Frontend Integration	End-to-end integration of Angular components with APIs	Hemanth Lakshman
Backend Development	Create and deploy RESTful APIs using Spring Boot	Amarnath Rao
Database Design	Design and implement relational tables in MySQL8	Mahesh Kumar
Database Integration	Integration of Spring Boot with MySQL through JPA/Hibernate	Amarnath Rao
Testing & Debugging (Frontend)	Ensure functionality and fix bugs for Angular components	Hemanth Lakshman
Testing & Debugging (Backend)	Debug APIs and ensure secure data flow in Spring Boot	Amarnath Rao
Deployment Setup	Prepare deployment scripts and ensure local server hosting	Kowshik
Real-Time Notifications	Add WebSocket support for realtime updates	Kowshik

High-Level Design

Architectural Overview

The application is built using the MVC (Model-View-Controller) architecture for modular and maintainable development.

1. Frontend
    - Developed with Angular 14.
    - Uses Angular CLI for component generation and RouterModule for dynamic routing. o Implements two-way data binding ([(ngModel)]) and reactive forms (FormBuilder). o Leverages Angular services for API calls to the backend.
2. Backend
    - Built with Spring Boot, utilizing JPA/Hibernate for database communication. o APIs follow REST principles, supporting HTTP methods: GET, POST, PUT, DELETE.
    - Implements WebSocket for real-time notifications and ensures security through CORS policies.
3. Database o MySQL8 Workbench is used for data

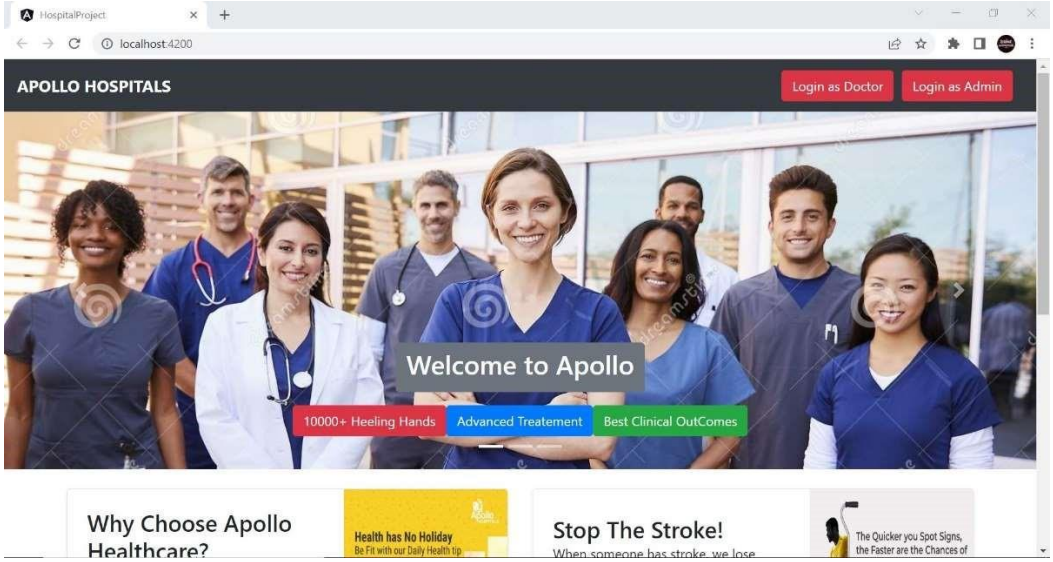
    storage.

    o Relational tables for patients, appointments, and medicines.

Front end

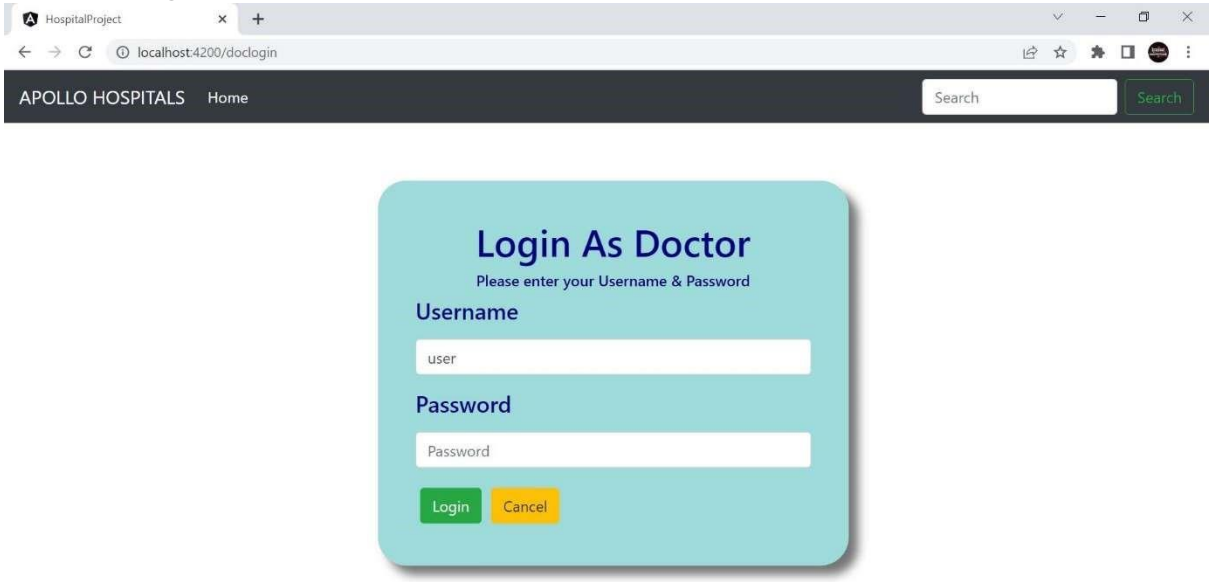
Key Components

1. Home Page



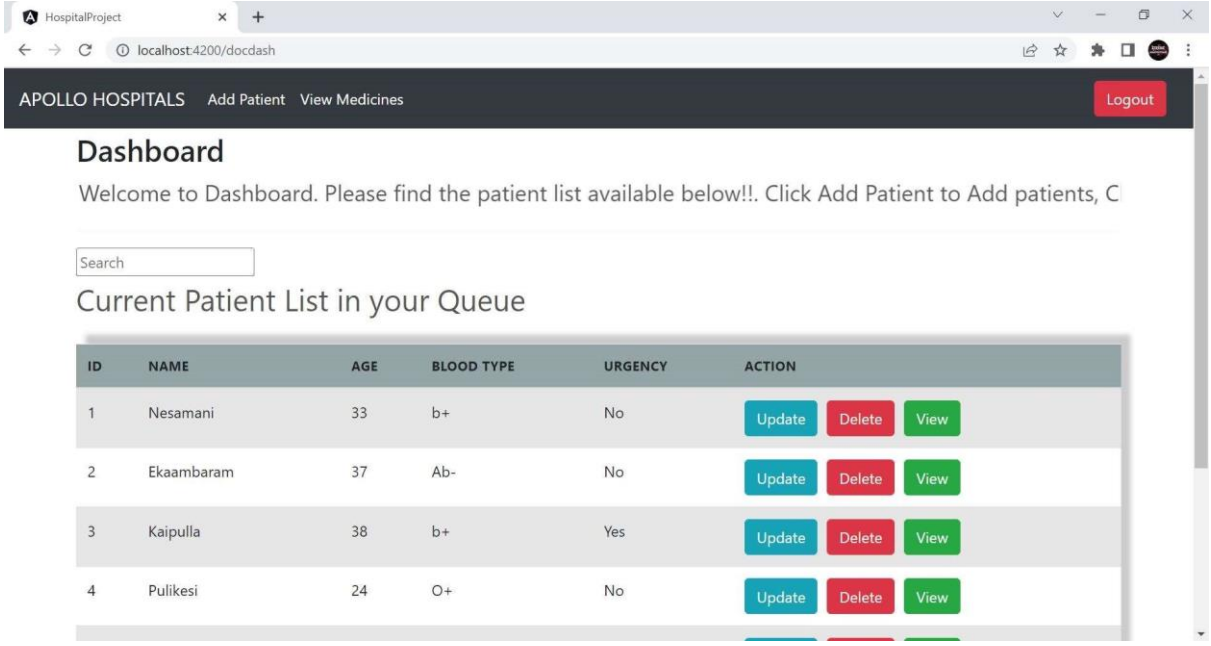
2. Doctor Login

A secure login screen with form validation for doctors.



3. Doctor Dashboard

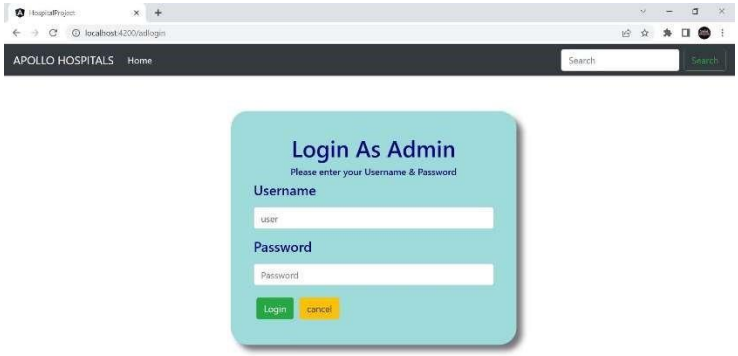
A dashboard where doctors can manage patients and medicines. Features include:

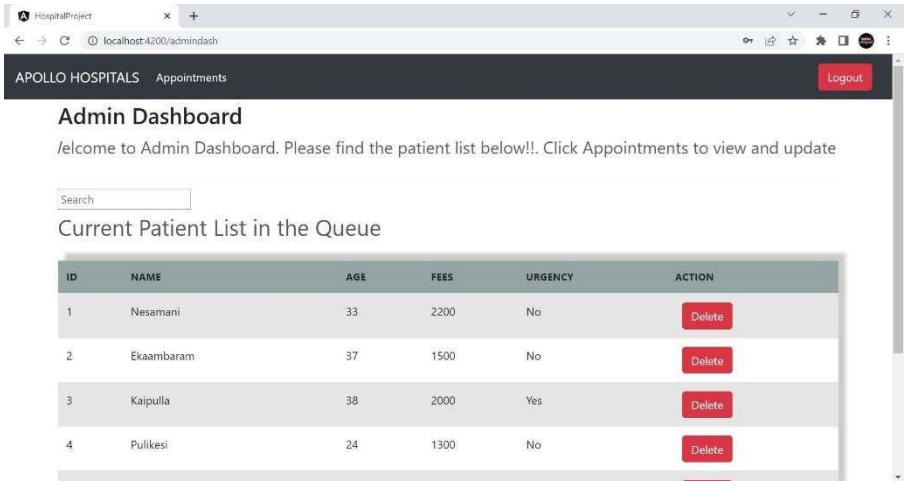


- Search by patient name (*ngFor directive).
- Add new patients using modal forms.

4. Admin Login & Dashboard

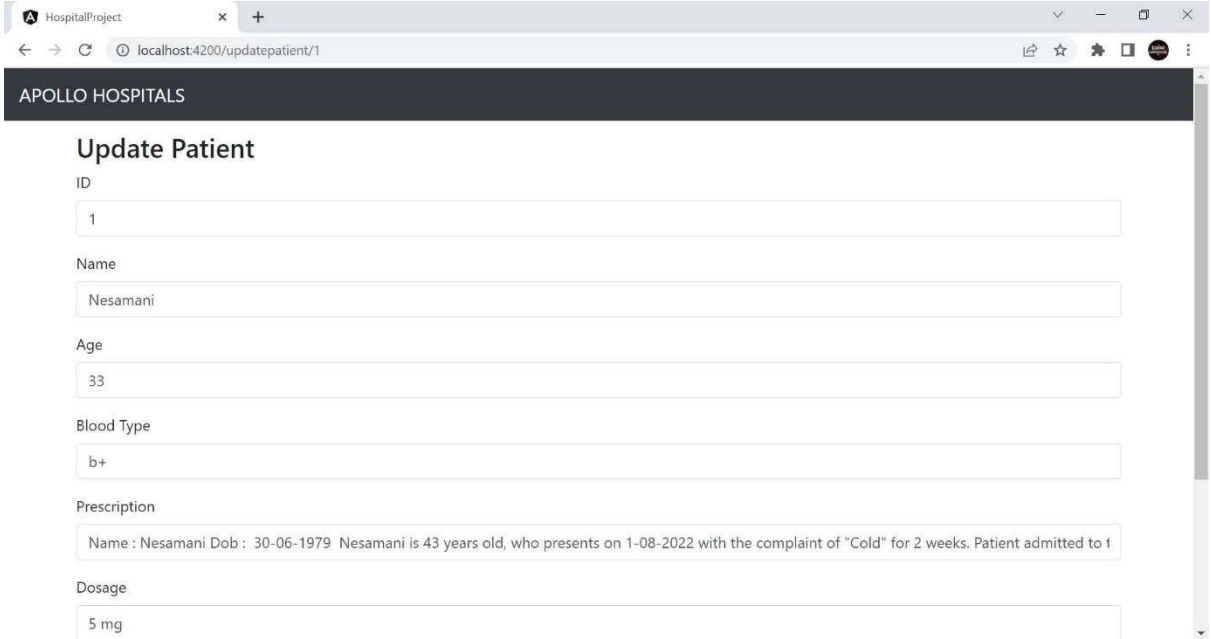
A dedicated dashboard for administrators to manage appointments.
Features include:



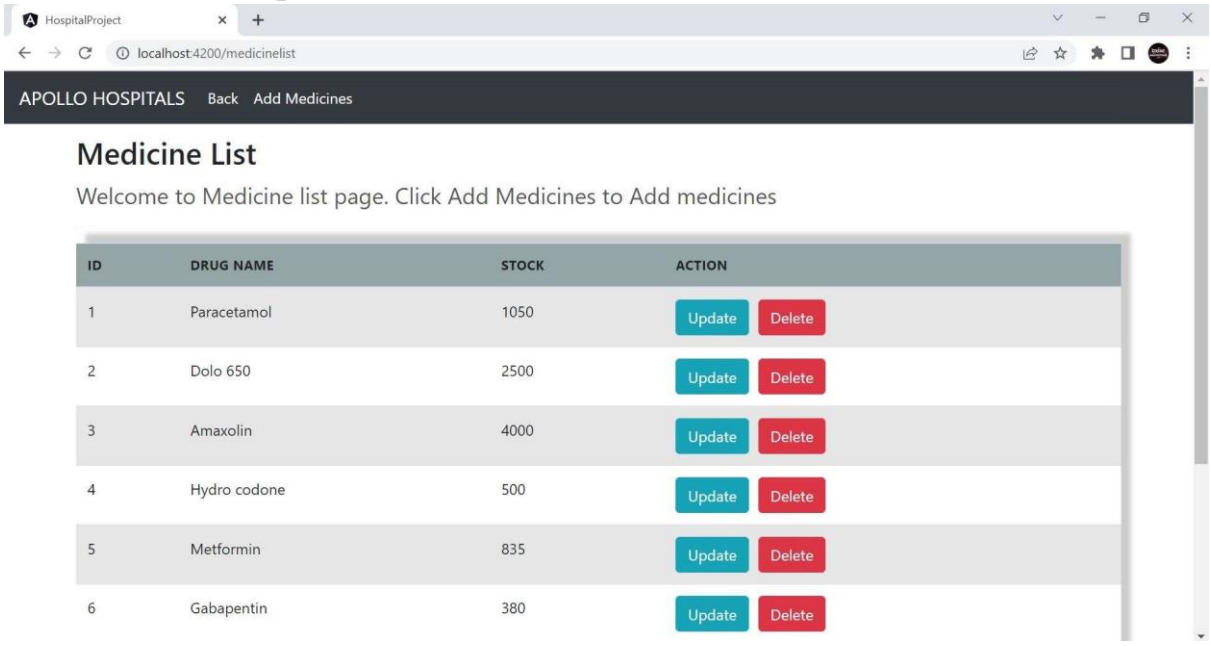


Lists for upcoming and past appointments.

5. Updating Patient Details:



6. Medicine List component:



Code Snippets:
Adminlogin.html

```html
<nav style="margin:auto" class="navbar navbar-expand-lg bg-dark">

    <a style="color: white;" class="navbar-brand" href="#">APOLLO HOSPITALS</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a style="color: white;"class="nav-link active" aria-current="page" href="javascript:void(0);" routerLink="/home">Home<
        </li>



      </ul>
      <form class="d-flex" role="search">
        <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
        <button class="btn btn-outline-success" type="submit">Search</button>
      </form>
    </div>

  </nav>


  <div class="container">

    <div class="row">


      <div class="col-md-6">
          <div class="card">

              <div class="text-center">
                  <h1>Login As Admin</h1>
                  <h6>Please enter your Username & Password</h6>
```

command 'black-box.openLiveDoc' not found

appointment-list.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { Appointment } from '../appointment';
import { AppointmentService } from '../appointment.service';

Qodo Gen: Options | Test this class
@Component({
  selector: 'app-appointment-list',
  templateUrl: './appointment-list.component.html',
  styleUrls: ['./appointment-list.component.css']
})
export class AppointmentListComponent implements OnInit {

  appointments: Appointment[];

  constructor(private appointmentService: AppointmentService,
    private router: Router) { }

Qodo Gen: Options | Test this method
  ngOnInit(): void {

    this.getAppointments();
  }

Qodo Gen: Options | Test this method
  private getAppointments() {
    this.appointmentService.getAppointmentslist().subscribe(data => {this.appointments = data;
      });
  }

Qodo Gen: Options | Test this method
  deleteAppointment(id: number){
    this.appointmentService.deleteAppointment(id).subscribe( data => {
      console.log(data);
      this.getAppointments();
    })
  }
```

Back end

Database Schema

1. Medicine Table:

Table: medicines

Columns:

id	bigint AI PK
drug_name	varchar(255)
stock	varchar(255)

2. Patient Table

Table: patientdb

Columns:

id	bigint AI PK
age	varchar(255)
blood_group	varchar(255)
dose	varchar(255)
fees	varchar(255)
first_name	varchar(255)
prescription	varchar(255)
urgency	varchar(255)

3. Doctor Table

Table: doctor

Columns:

doct_id	bigint PK
doc_name	varchar(255)
doct_age	int
doct_gender	varchar(255)
doct_specialist	varchar(255)
number_ofpatient_attneded	bigint

4. Appointment Table

Table: appointment

Columns:

id	bigint PK
age	varchar(255)
name	varchar(255)
number	varchar(255)
symptoms	varchar(255)

Data base tables

1. Medicine Table:

```
1 • select * from medicines;
```

Result Grid

id	drug_name	stock
1	Paracetamol	50
2	Ketoconazole	23
3	Thyroxin	30
4	Citrizine	34
NULL	NULL	NULL

2. Patient Table

```
1• select * from patientdb;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Con

id	age	blood_group	dose	fees	first_name	prescription	urgency
1	25	B+	Single dose	1000	John	Antibiotics	High
2	32	B+	Double dose	1500	Jane	Painkiller	Low
3	45	O+	Triple dose	1000	John	Antibiotics	High
4	29	AB+	Single dose	1500	Jane	Painkiller	Low
5	38	A+	Double dose	1000	1000	Antibiotics	High
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

3. Doctor Table

```
1• select * from doctor;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Co

doct_id	doc_name	doct_age	doct_gender	doct_specialist	number_ofpatient_attneded
1	Dr. John Smith	45	Male	Cardiologist	1200
2	Dr. Emily Carter	38	Female	Neurologist	950
3	Dr. Alan Walker	50	Male	Orthopedic	1100
4	Dr. Sarah Johnson	42	Female	Pediatrician	800
5	Dr. Michael Brown	55	Male	Dermatologist	700
NULL	NULL	NULL	NULL	NULL	NULL

4. Appointment Table

```
1• select * from appointment;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

id	age	name	number	symptoms
1	21	Hemanth	67452365	autism
NULL	NULL	NULL	NULL	NULL

Snippets of the code to show updates in the data base:

```
1   import { Component, OnInit } from '@angular/core';
2   import { ActivatedRoute, Router } from '@angular/router';
3   import { MedicineService } from '../medicine.service';
4   import { Medicine } from '../medicine';
5
6
    Qodo Gen: Options | Test this class
7   @Component({
8     selector: 'app-update-medicine',
9     templateUrl: './update-medicine.component.html',
10    styleUrls: ['./update-medicine.component.css']
11  })
12  export class UpdateMedicineComponent implements OnInit {
13
14    id: number;
15    medicine: Medicine = new Medicine();
16    constructor(private medicineService: MedicineService,
17      private route: ActivatedRoute,
18      private router: Router) { }
19
      Qodo Gen: Options | Test this method
20    ngOnInit(): void {
21      this.id = this.route.snapshot.params['id'];
22      this.medicineService.getMedicineById(this.id).subscribe(data => {
23        this.medicine = data;
24      }
25      , error => console.log(error));
26    }
27
      Qodo Gen: Options | Test this method
28    onSubmit() {
29      this.medicineService.updateMedicine(this.id, this.medicine).subscribe(data => {
30        this.goToMedicineList();
31      }
32      , error => console.log(error));
33
34    }
35
      Qodo Gen: Options | Test this method
36    goToMedicineList() {
37      this.router.navigate(['/medicinelist']);
38    }
39
40  }D
```

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { Patient } from '../patient';
import { PatientService } from '../patient.service';

Qodo Gen: Options | Test this class
@Component({
  selector: 'app-createpatient',
  templateUrl: './createpatient.component.html',
  styleUrls: ['./createpatient.component.css']
})
export class CreatepatientComponent implements OnInit {

  patient: Patient = new Patient();
  constructor(private patientService: PatientService,
    private router: Router) { }

  ngOnInit(): void {
  }

  Qodo Gen: Options | Test this method
  savePatient() {
    this.patientService.createPatient(this.patient).subscribe(data => {
      console.log(data);
      this.goToPatientList();
  },
  error => console.log(error));
}
  Qodo Gen: Options | Test this method
  goToPatientList() {
    this.router.navigate(['/docdash']);


  }

  Qodo Gen: Options | Test this method
  onSubmit() {
    console.log(this.patient);
    this.savePatient();
  }

}
```

Links:-

Github Link: github.com/Amarnath-Rao/Hospital-Management-System

Live Link: hospital-management-system-ten-nu.vercel.app