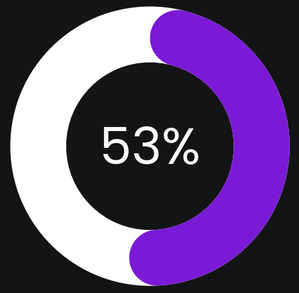
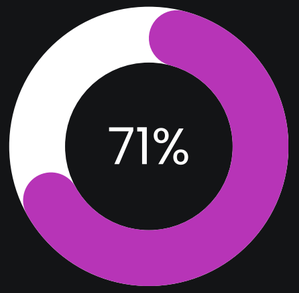
Hospital Management System

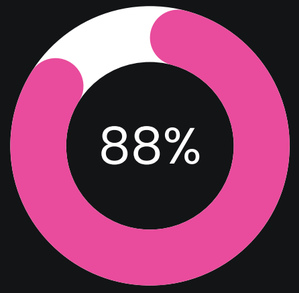
Tech Stack



Spring Boot (3.4.0)



H2 Database



JPA (Java Persistence API)

Patient APIs

- Patient APIs:

POST /api/patients

- Create new patient records
- Store patient demographics
- Generate unique patient IDs

```
Example Request:
{
    "name": "John Doe",
    "age": 30,
    "gender": "Male",
    "address": "123 Main St",
    "contact": "123-456-7890"
}

Response:
{
    "success": true,
    "message": "Patient added successfully",
    "data": {
        "id": 1,
        ...
    }
}
```

```java
package com.hospital.management.controller;

import com.hospital.management.dto.ApiResponse;
import com.hospital.management.entity.Patient;
import com.hospital.management.service.impl.PatientServiceImpl;
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.*;
import java.util.List;

@RestController
@RequestMapping("/api/patients")
@RequiredArgsConstructor
public class PatientController {
    private final PatientServiceImpl patientService;

    @GetMapping
    public ApiResponse<List<Patient>> getAllPatients() {
        List<Patient> patients = patientService.getAllPatients();
        return ApiResponse.success("Patients retrieved successfully", patients);
    }

    @GetMapping("/{id}")
    public ApiResponse<Patient> getPatientById(@PathVariable Long id) {
        Patient patient = patientService.getPatientById(id);
        return ApiResponse.success("Patient retrieved successfully", patient);
    }

    @PostMapping
    public ApiResponse<Patient> createPatient(@RequestBody Patient patient) {
        Patient savedPatient = patientService.createPatient(patient);
        return ApiResponse.success("Patient added successfully", savedPatient);
    }

    @PutMapping("/{id}")
    public ApiResponse<Patient> updatePatient(@PathVariable Long id, @RequestBody Patient patient) {
        Patient updatedPatient = patientService.updatePatient(id, patient);
        return ApiResponse.success("Patient updated successfully", updatedPatient);
    }

    @DeleteMapping("/{id}")
    public ApiResponse<Void> deletePatient(@PathVariable Long id) {
        patientService.deletePatient(id);
        return ApiResponse.success("Patient deleted successfully", null);
    }
}
```

```java
package com.hospital.management.controller;

import com.hospital.management.dto.ApiResponse;
import com.hospital.management.entity.Doctor;
import com.hospital.management.service.impl.DoctorServiceImpl;
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.*;
import java.util.List;

@RestController
@RequestMapping("/api/doctors")
@RequiredArgsConstructor
public class DoctorController {
    private final DoctorServiceImpl doctorService;

    @GetMapping
    public ApiResponse<List<Doctor>> getAllDoctors() {
        List<Doctor> doctors = doctorService.getAllDoctors();
        return ApiResponse.success("Doctors retrieved successfully", doctors);
    }

    @GetMapping("/{id}")
    public ApiResponse<Doctor> getDoctorById(@PathVariable Long id) {
        Doctor doctor = doctorService.getDoctorById(id);
        return ApiResponse.success("Doctor retrieved successfully", doctor);
    }

    @PostMapping
    public ApiResponse<Doctor> createDoctor(@RequestBody Doctor doctor) {
        Doctor savedDoctor = doctorService.createDoctor(doctor);
        return ApiResponse.success("Doctor added successfully", savedDoctor);
    }

    @PutMapping("/{id}")
    public ApiResponse<Doctor> updateDoctor(@PathVariable Long id, @RequestBody Doctor doctor) {
        Doctor updatedDoctor = doctorService.updateDoctor(id, doctor);
        return ApiResponse.success("Doctor updated successfully", updatedDoctor);
    }

    @DeleteMapping("/{id}")
    public ApiResponse<Void> deleteDoctor(@PathVariable Long id) {
        doctorService.deleteDoctor(id);
        return ApiResponse.success("Doctor deleted successfully", null);
    }
}
```

Doctor APIs

- Doctor APIs:

POST /api/doctors

- Register new doctors
- Track specializations
- Manage doctor availability

```
Example Request:
{
    "name": "Dr. Smith",
    "specialization": "Cardiology",
    "experience": 10,
    "contact": "987-654-3210"
}

Response:
{
    "success": true,
    "message": "Doctor added successfully",
    "data": {
        "id": 1,
        ...
    }
}
```

```java
package com.hospital.management.controller;

import com.hospital.management.dto.ApiResponse;
import com.hospital.management.entity.Department;
import com.hospital.management.service.impl.DepartmentServiceImpl;
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.*;
import java.util.List;

@RestController
@RequestMapping("/api/departments")
@RequiredArgsConstructor
public class DepartmentController {
    private final DepartmentServiceImpl departmentService;

    @GetMapping
    public ApiResponse<List<Department>> getAllDepartments() {
        List<Department> departments = departmentService.getAllDepartments();
        return ApiResponse.success("Departments retrieved successfully", departments);
    }

    @GetMapping("/{id}")
    public ApiResponse<Department> getDepartmentById(@PathVariable Long id) {
        Department department = departmentService.getDepartmentById(id);
        return ApiResponse.success("Department retrieved successfully", department);
    }

    @PostMapping
    public ApiResponse<Department> createDepartment(@RequestBody Department department) {
        Department savedDepartment = departmentService.createDepartment(department);
        return ApiResponse.success("Department added successfully", savedDepartment);
    }

    @PutMapping("/{id}")
    public ApiResponse<Department> updateDepartment(@PathVariable Long id, @RequestBody Department department) {
        Department updatedDepartment = departmentService.updateDepartment(id, department);
        return ApiResponse.success("Department updated successfully", updatedDepartment);
    }

    @DeleteMapping("/{id}")
    public ApiResponse<Void> deleteDepartment(@PathVariable Long id) {
        departmentService.deleteDepartment(id);
        return ApiResponse.success("Department deleted successfully", null);
    }
}
```

Department APIs

- POST /api/departments

- Create new departments
- Assign department head
- Set department structure

```
Example Request:
{
    "name": "Cardiology",
    "headOfDepartment": "Dr. Smith"
}

Response:
{
    "success": true,
    "message": "Department created successfully",
    "data": {
        "id": 1,
        "name": "Cardiology",
        "headOfDepartment": "Dr. Smith"
    }
}
```

Staff APIs

- POST /api/staff

- Register new staff
- Assign to department
- Set staff roles

```
Example Request:
{
    "name": "Jane Smith",
    "role": "Nurse",
    "department": {
        "id": 1
    }
}

Response:
{
    "success": true,
    "message": "Staff member added successfully",
    "data": {
        "id": 1,
        "name": "Jane Smith",
        "role": "Nurse",
        "department": {
            "id": 1,
            "name": "Cardiology"
        }
    }
}
```

```java
package com.hospital.management.controller;

import com.hospital.management.dto.ApiResponse;
import com.hospital.management.entity.Staff;
import com.hospital.management.service.impl.StaffServiceImpl;
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.*;
import java.util.List;

@RestController
@RequestMapping("/api/staff")
@RequiredArgsConstructor
public class StaffController {
    private final StaffServiceImpl staffService;

    @GetMapping
    public ApiResponse<List<Staff>> getAllStaff() {
        List<Staff> staffList = staffService.getAllStaff();
        return ApiResponse.success("Staff members retrieved successfully", staffList);
    }

    @GetMapping("/{id}")
    public ApiResponse<Staff> getStaffById(@PathVariable Long id) {
        Staff staff = staffService.getStaffById(id);
        return ApiResponse.success("Staff member retrieved successfully", staff);
    }

    @GetMapping("/department/{departmentId}")
    public ApiResponse<List<Staff>> getStaffByDepartmentId(@PathVariable Long departmentId) {
        List<Staff> staffList = staffService.getStaffByDepartmentId(departmentId);
        return ApiResponse.success("Department staff retrieved successfully", staffList);
    }

    @PostMapping
    public ApiResponse<Staff> createStaff(@RequestBody Staff staff) {
        Staff savedStaff = staffService.createStaff(staff);
        return ApiResponse.success("Staff member added successfully", savedStaff);
    }

    @PutMapping("/{id}")
    public ApiResponse<Staff> updateStaff(@PathVariable Long id, @RequestBody Staff staff) {
        Staff updatedStaff = staffService.updateStaff(id, staff);
        return ApiResponse.success("Staff member updated successfully", updatedStaff);
    }

    @DeleteMapping("/{id}")
    public ApiResponse<Void> deleteStaff(@PathVariable Long id) {
        staffService.deleteStaff(id);
        return ApiResponse.success("Staff member deleted successfully", null);
    }
}
```

Appointment APIs

```java
package com.hospital.management.controller;

import com.hospital.management.dto.AppointmentResponseDTO;
import com.hospital.management.dto.DoctorDTO;
import com.hospital.management.dto.PatientDTO;
import com.hospital.management.entity.Appointment;
import com.hospital.management.service.impl.AppointmentServiceImpl;
import com.hospital.management.dto.ApiResponse;
import lombok.RequiredArgsConstructor;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.*;
import java.util.List;
import java.util.stream.Collectors;

@RestController
@RequestMapping("/api/appointments")
@RequiredArgsConstructor
public class AppointmentController {
    private final AppointmentServiceImpl appointmentService;

    private AppointmentResponseDTO convertToDTO(Appointment appointment) {
        AppointmentResponseDTO dto = new AppointmentResponseDTO();
        dto.setId(appointment.getId());
        dto.setAppointmentDate(appointment.getAppointmentDate());
        dto.setStatus(appointment.getStatus());

        DoctorDTO doctorDTO = new DoctorDTO();
        doctorDTO.setId(appointment.getDoctor().getId());
        doctorDTO.setName(appointment.getDoctor().getName());
        doctorDTO.setSpecialization(appointment.getDoctor().getSpecialization());
        doctorDTO.setContact(appointment.getDoctor().getContact());
        dto.setDoctor(doctorDTO);

        PatientDTO patientDTO = new PatientDTO();
        patientDTO.setId(appointment.getPatient().getId());
        patientDTO.setName(appointment.getPatient().getName());
        patientDTO.setContact(appointment.getPatient().getContact());
        dto.setPatient(patientDTO);

        return dto;
    }

    @GetMapping
    @PreAuthorize("hasAnyRole('ADMIN', 'USER')")
    public ApiResponse<List<AppointmentResponseDTO>> getAllAppointments() {
        List<AppointmentResponseDTO> appointments = appointmentService.getAllAppointments()
            .stream()
            .map(this::convertToDTO)
            .collect(Collectors.toList());
        return ApiResponse.success("Appointments retrieved successfully", appointments);
    }

    @GetMapping("/{id}")
    @PreAuthorize("hasAnyRole('ADMIN', 'USER')")
    public ApiResponse<AppointmentResponseDTO> getAppointmentById(@PathVariable Long id) {
        Appointment appointment = appointmentService.getAppointmentById(id);
        return ApiResponse.success("Appointment retrieved successfully", convertToDTO(appointment));
    }
```

- *POST /api/appointments*

- *Schedule new appointments*
- *Link patient and doctor*
- *Set appointment time*

```
Example Request:
{
    "appointmentDate": "2024-03-25T10:30:00",
    "doctor": {
        "id": 1
    },
    "patient": {
        "id": 1
    }
}

Response:
{
    "success": true,
    "message": "Appointment scheduled successfully",
    "data": {
        "id": 1,
        "appointmentDate": "2024-03-25T10:30:00",
        "status": "SCHEDULED",
        "doctor": {
            "id": 1,
            "name": "Dr. Smith"
        },
        "patient": {
            "id": 1,
            "name": "John Doe"
        }
    }
```

Error Responses

- Appointment Conflict:

```
{
    "success": false,
    "message": "Doctor has another appointment scheduled between 10:30 AM and 11:00
    "data": null
}
```

- Resource Not Found:

```
{
    "success": false,
    "message": "Doctor not found with id: 1",
    "data": null
}
```

- Validation Error:

```
{
    "success": false,
    "message": "Appointment time must be in the future",
    "data": null
}
```

- Authentication Error:

```
{
    "success": false,
    "message": "Access denied. Please login with appropriate credentials.",
    "data": null
}
```

- Conflict Prevention:

- Time slot overlap check
- Buffer time management
- Real-time availability update

```java
    @Bean
    public UserDetailsService userDetailsService(BCryptPasswordEncoder bCryptPasswordEncode
        InMemoryUserDetailsManager manager = new InMemoryUserDetailsManager();

        UserDetails admin = User.builder()
            .username("admin")
            .password(bCryptPasswordEncoder.encode("admin123"))
            .roles("ADMIN")
            .build();

        UserDetails user = User.builder()
            .username("user")
            .password(bCryptPasswordEncoder.encode("user123"))
            .roles("USER")
            .build();

        manager.createUser(admin);
        manager.createUser(user);

        return manager;
    }
}
```

```
- Authentication:
  - Basic Auth implementation
  - Username/Password validation
  - JWT token support

- Role-Based Access:
  ADMIN:
    - Full access to all endpoints
    - Staff management
    - Department management

  USER:
    - View doctors/departments
    - Book appointments
    - View own appointments
```

Security Config

Understanding Hospital Management System Relationships

Many-to-One

Patient & Appointments :

- One patient can book MANY appointments (like multiple visits over time)
-
- But each appointment can belong to only ONE patient (obviously, an appointment can't be for multiple patients)

One-to-Many

Department & Staff :

- ONE department can have MANY staff members (like a team)
- Each staff member can belong to only ONE department (can't work in multiple departments)

Thank You

Arreyan Hamid
Siddharth
Surya
Hemanth
Pramod
Vishnu