Hospital Management System – Backend

# Team Members

Mohamed Atia, Mohamed El-Gazar, Mohamed Khaled, Mohamed El-Abady, Moamen El-Sha3rawy

# Project Goal

Build a secure and scalable RESTful API to manage hospital operations such as patients, doctors, appointments, and medical records.

# Technologies and System Architecture

## Technology Stack

* ASP.NET Core (Backend Framework)
* SQL Server (Database)
* Entity Framework Core (ORM)
* JWT (Authentication)
* Postman and Swagger (Testing and Documentation)
* GitHub (Version Control)

## System Layers

* Controller: Handles HTTP requests
* Service: Contains business logic
* Repository: Handles database access
* Database: Stores persistent hospital data

# Database Schema

## Main Entities

### Patient

Name, Email, Password, Date of Birth, Contact, Address, Insurance, Emergency Contact

Related to: Appointments, Medical Records, DoctorPatients

### Doctor

Name, Email, Password, Specialty, Hourly Pay, Experience, Contact

Related to: Appointments, DoctorPatients

### Appointment

Date, Reason for Visit

Linked to: Patient ID, Doctor ID

### Medical Record

Record Date, Diagnosis, Treatment, Notes

Linked to: Patient ID

### DoctorPatient (Join Table)

Doctor ID, Patient ID, Rating

## Relationships

* One-to-Many: Patient to Appointments
* One-to-Many: Doctor to Appointments
* One-to-Many: Patient to Medical Records
* Many-to-Many: Doctor and Patient (via DoctorPatient)

# Key Features

* JWT-based authentication for doctors and patients
* Full Create, Read, Update, Delete operations for doctors and patients
* Appointment scheduling with validations
* Patient medical history management
* Data integrity enforced through model validation

# API Endpoints

## Patients

* GET api/patients
* POST api/patients
* PUT api/patients/{id}
* DELETE api/patients/{id}

## Doctors

* GET api/doctors
* POST api/doctors
* PUT api/doctors/{id}
* DELETE api/doctors/{id}

## Appointments

* GET api/appointments
* POST api/appointments
* PUT api/appointments/{id}
* DELETE api/appointments/{id}

## Medical Records

* GET api/medicalrecords
* POST api/medicalrecords
* PUT api/medicalrecords/{id}
* DELETE api/medicalrecords/{id}

# Authentication Workflow

## User Login Flow

Validate user credentials (doctor or patient)

Generate JWT token after successful authentication

## Authorization

* Protected routes are enabled using authentication and authorization middleware
* JWT token is passed in request headers for secure access

# Testing and Validation

## Tools Used

* Postman for request testing
* Swagger for auto-generated documentation and testing interface

## Tested Scenarios

* Authenticated versus unauthenticated access
* Missing or invalid fields
* Conflict resolution (e.g. overlapping appointments)