🏥 Hospital Management System — ASP.NET Core Microservices Project

# 👥 Group Members:

• Moin Arif  
• Manal  
• Hussain Nasir

# 🛠️ Tech Stack:

• Frontend: ASP.NET Core MVC (HospitalMVC\_UI)  
• Backend Microservices:  
 - PatientService — CRUD for Patients  
 - AppointmentService — CRUD for Appointments  
• Database: MySQL (1 DB per service)  
• ORM: Entity Framework Core  
• API Testing: Postman + Swagger  
• CI/CD: (Planned with Jenkins)

# 🗂️ Project Structure:

**HospitalManagementSystem/**

**── HospitalMVC\_UI/ # Frontend (MVC UI)**│ ├── Controllers/ # PatientController.cs, AppointmentController.cs, HomeController.cs  
│ ├── Models/ # Patient.cs, Appointment.cs, ErrorViewModel.cs  
│ └── Views/   
│ ├── Home/ # Index.cshtml, Privacy.cshtml  
│ ├── Patient/ # Create, Edit, Delete, Index views  
│ └── Appointment/ # Create, Edit, Delete, Index views  
│ └── Program.cs # UI Entry Point

**── PatientService/ # Microservice 1**│ ├── Controllers/ # PatientController.cs  
│ ├── Models/ # Patient.cs, PatientDbContext.cs  
│ ├── Factory/ # PatientDbContextFactory.cs  
│ ├── Migrations/ # EF Core migrations  
│ └── Program.cs # API Entry Point

**── AppointmentService/ # Microservice 2**│ ├── Controllers/ # AppointmentController.cs  
│ ├── Models/ # Appointment.cs, AppointmentDbContext.cs  
│ ├── Factory/ # AppointmentDbContextFactory.cs  
│ ├── Migrations/ # EF Core migrations  
│ └── Program.cs # API Entry Point

**── HospitalSystem.sln # Solution File**  
├── README.md # This file  
├── .gitignore # For Visual Studio  
├── PatientDB.sql , AppointmentDB.sql # SQL Workbrench for both services  
└── Postman\_Collection.json # API test cases

# ▶️ How to Run the Project:

**📦 Prerequisites:**  
• .NET SDK 7+  
• Visual Studio 2022+  
• MySQL Server 8+  
• Postman

**1️⃣ Run PatientService:**cd PatientService  
- Update appsettings.json with your MySQL connection string  
- Run DB migration: dotnet ef database update  
- Start service: dotnet run

**2️⃣ Run AppointmentService:**  
cd AppointmentService  
- Update appsettings.json with your MySQL connection string  
- Run DB migration: dotnet ef database update  
- Start service: dotnet run

**3️⃣ Run HospitalMVC\_UI:**  
cd HospitalMVC\_UI  
- Make sure base URLs for APIs are correct in controller logic  
- Start frontend: dotnet run  
Then open browser: https://localhost:xxxx/

# 🌐 API Endpoints Summary:

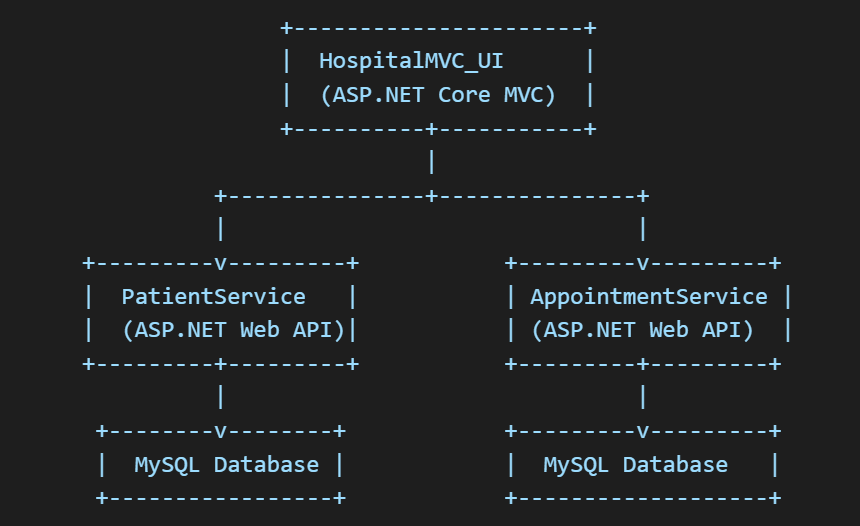
**📁 PatientService API:**  
| Method | Endpoint | Description |  
| ------ | ----------------- | -------------------- |  
| GET | /api/patient | Get all patients |  
| GET | /api/patient/{id} | Get patient by ID |  
| POST | /api/patient | Add new patient |  
| PUT | /api/patient/{id} | Update patient by ID |  
| DELETE | /api/patient/{id} | Delete patient by ID |

**📁 AppointmentService API:**  
| Method | Endpoint | Description |  
| ------ | --------------------- | ------------------------ |  
| GET | /api/appointment | Get all appointments |  
| GET | /api/appointment/{id} | Get appointment by ID |  
| POST | /api/appointment | Create new appointment |  
| PUT | /api/appointment/{id} | Update appointment by ID |  
| DELETE | /api/appointment/{id} | Delete appointment by ID |

# 🧪 Postman Test Cases:

**✅ PatientService:**  
1. Create Patient (Valid)  
2. Get Patient (Valid ID)  
3. Get Patient (Invalid ID)  
4. Create Patient (Missing Fields)  
5. Delete Patient (Invalid ID)  
 **✅ AppointmentService:**  
1. Create Appointment (Valid)  
2. Get Appointment (Valid ID)  
3. Get Appointment (Invalid ID)  
4. Create Appointment (Missing Date)  
5. Delete Appointment (Invalid ID)

# 🧭 Architecture Diagram



# 📦 Additional Files (Included in Repo)

• ✅ README.md (this file)  
• ✅ .gitignore (for Visual Studio)  
• ✅ PatientDB.sql , AppointmentDB.sql (MySQL Workbrench)  
• ✅ Postman\_Collection.json (for test automation)