**Case Study: Patient Management System**

**Problem Statement:**

**Background**

Healthcare providers require an efficient, secure, and user-friendly system to manage patient records, doctor information, and appointment scheduling. Existing manual or legacy systems often lack scalability, real-time data access, and robust validation, leading to errors, inefficiencies, and poor user experience. There is a need for a modern web-based solution to streamline these operations while ensuring data integrity, security, and compliance with healthcare standards.

**Objective**

Develop a **Patient Management System (PMS)** using **ASP.NET Core MVC** and **Entity Framework Core (EF Core)** to provide healthcare providers with a reliable platform for managing patient data, doctor details, and appointments. The system must support CRUD operations, enforce data relationships, implement validations, and ensure scalability for future enhancements.

**Requirements**

**Functional Requirements**

1. **Patient Management**:
   * Create, read, update, and delete patient records, including fields for name, date of birth, gender, email, and phone number.
   * Support searching and filtering patients by name or email.
2. **Doctor Management**:
   * Perform CRUD operations for doctors, capturing name and specialty.
3. **Appointment Management**:
   * Schedule, view, update, and cancel appointments, linking each appointment to one patient and one doctor.
   * Track appointment date/time and status (e.g., Scheduled, Completed, Cancelled).
4. **Data Relationships**:
   * Establish a one-to-many relationship between patients and appointments (one patient can have multiple appointments).
   * Establish a one-to-many relationship between doctors and appointments.
5. **Search and Filter**:
   * Allow users to search for patients and filter appointments by status or date.
6. **User Interface**:
   * Provide a responsive, intuitive UI using Bootstrap for accessibility across devices.

**Non-Functional Requirements**

1. **Data Validation**:
   * Enforce server-side and client-side validations (e.g., required fields, valid email, future appointment dates, realistic patient age).
2. **Database**:
   * Use a relational database (e.g., SQL Server) with EF Core’s code-first approach for schema management.

**Success Criteria**

* Users can efficiently manage patients, doctors, and appointments with minimal errors.
* All validations (e.g., no future birth dates, unique emails) are enforced.