Hospital ERP System

1. Problem Statement & Objectives

Problem Statement

Small and medium-sized hospitals often face major inefficiencies in handling administrative, medical, and financial operations. Without a centralized system, hospitals rely on paper-based or disconnected digital tools, leading to:

- Redundant patient data entries
- Appointment mismanagement and scheduling conflicts
- Errors in billing and payment tracking
- Lack of visibility into employee roles and working hours
- Poor medication inventory control

These issues negatively affect operational efficiency, data accuracy, and the quality of patient care.

Objectives

The **Hospital ERP System** aims to deliver an integrated and user-friendly platform for managing all hospital operations in one place.

Project Objectives:

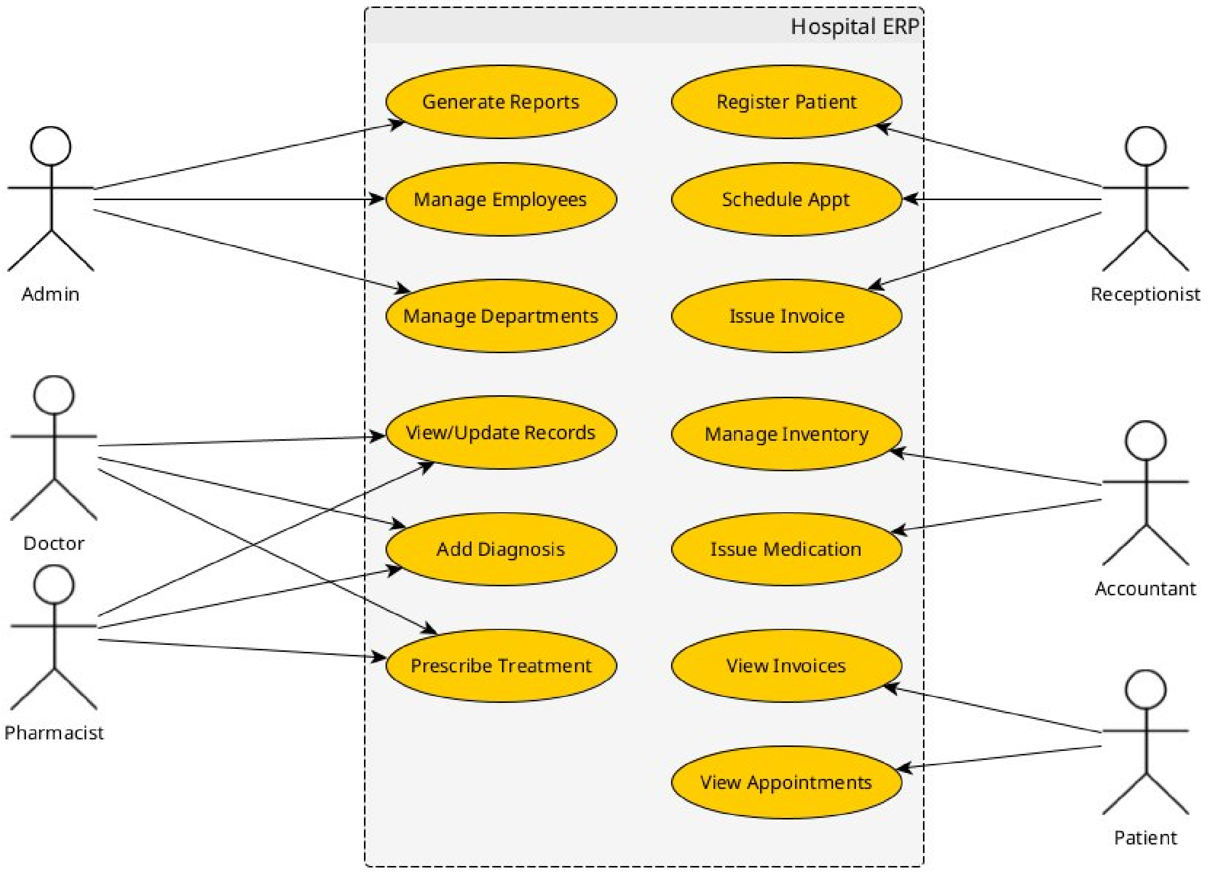
1. **Centralize hospital operations** — unify patient, billing, HR, and inventory data.
2. **Enhance service efficiency** — automate appointment booking, invoicing, and reporting.
3. **Ensure data integrity** — use MSSQL with enforced foreign key constraints.
4. **Support role-based access control** — secure sensitive medical and financial data.
5. **Provide actionable insights** — generate real-time reports for hospital administration.
6. **Deliver a scalable architecture** — built using **.NET Core MVC** and **MSSQL Server** for reliability and future expansion.

2. Use Case Diagram & Descriptions

System Actors

Actor	Role / Responsibility
Admin	Manages users, departments, and system-wide settings
Doctor	Accesses patient medical records, adds diagnoses, and manages treatments
Receptionist	Registers patients, schedules appointments, and issues invoices
Pharmacist	Manages medication inventory and dispensing
Accountant	Handles invoices, payments, and generates financial reports
Patient	(Optional) Views appointment details or invoices via patient portal

Use Case Diagram



Use Case Descriptions

Use Case	Primary Actor	Description
Register Patient	Receptionist	Create a new patient record with personal and contact details.
Schedule Appointment	Receptionist	Assign a doctor and time for patient consultation.
Record Diagnosis	Doctor	Add or update patient medical diagnosis and link treatment.
Manage Medication Inventory	Pharmacist	Track stock, update quantities, and monitor expiry.
Generate Invoice	Receptionist / Accountant	Create invoices for medical services and medication.
Manage Employees	Admin	Add, edit, or remove hospital employees and define roles.
Generate Reports	Admin / Accountant	Generate daily, monthly, or annual reports for management.

3. Functional & Non-Functional Requirements

Functional Requirements

Module	Requirement
Patient Management	Add, edit, and retrieve patient records with unique ID and linked medical history.
Appointment Management	Schedule, update, cancel, and view appointments between patients and doctors.
Medical Records	Store diagnoses and treatments for each patient, linked with doctor and date.
Billing System	Generate invoices, calculate totals, track payments, and link services and medications.
Inventory Management	Maintain a list of medications, quantities, costs, and expiry dates.
Employee Management	Manage hospital staff with roles, departments, and schedules.
Reports	Generate summaries (e.g., total patients, revenue, medication usage).

Non-Functional Requirements

Category	Description
Performance	The system must respond to user actions within 2 seconds under normal load (≤100 concurrent users).
Scalability	Designed to support future modules (e.g., lab, radiology) with minimal restructuring.
Security	Implement **ASP.NET Identity** for authentication and **role-based authorization** for different staff roles.
Data Integrity	MSSQL Server enforces referential integrity via foreign key relationships defined in the ERD.
Availability	The system should operate with at least **99% uptime** during business hours.
Maintainability	The **MVC pattern** separates business logic from UI for easier debugging and updates.
Usability	The system should have an intuitive and clean interface using **Razor Pages** and **Bootstrap**.
Backup & Recovery	MSSQL database backups scheduled daily with recovery plans for critical data loss scenarios.
Compatibility	Compatible with modern browsers and deployable on Windows Server environments.

4. Software Architecture

Architecture Overview

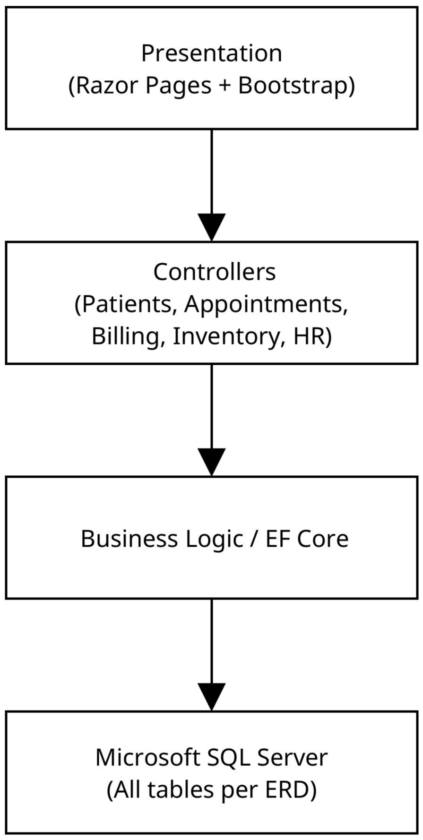
The **Hospital ERP System** is designed using the **Model–View–Controller (MVC)** architecture pattern within the **.NET Core Framework**.
It provides a clear separation of concerns, improving maintainability, testability, and scalability.

High-Level Component Architecture

Layer	Description	Technologies

Presentation Layer (View)	User interface for hospital staff to interact with the system — built using **Razor Views** with Bootstrap for responsive design.	Razor Pages, HTML, CSS, Bootstrap
Controller Layer (Business Logic)	Processes user requests, interacts with models, and returns appropriate views or JSON responses.	ASP.NET Core Controllers
Model Layer (Data Access)	Represents entities and manages data access through **Entity Framework Core**.	EF Core, LINQ
Database Layer	Stores all hospital data in a relational format according to the ERD.	Microsoft SQL Server
Authentication & Authorization	Manages user login, roles, and permissions.	ASP.NET Core Identity
Reporting & Analytics	Generates reports and analytics dashboards for admins and accountants.	RDLC / FastReport.NET / Razor Reports

Architecture Diagram



Deployment Details

- **Framework:** .NET Core MVC 8.0
- **Database:** Microsoft SQL Server (MSSQL)

- **Hosting:** IIS or Azure App Service
- **ORM:** Entity Framework Core (Code-First or Database-First approach)
- **Authentication:** ASP.NET Core Identity with Role-based Authorization
- **Reporting:** RDLC / FastReport.NET integrated with Controllers