## **Stakeholders Overview**

| **Stakeholder** | **Role** |
| --- | --- |
| Clinic Owner/Management | Business decision-makers |
| Doctors | Service providers |
| Receptionists/Admin Staff | System operators |
| Patients | End users |
| IT Team | Developers, testers, maintainers |

**System Components (Architecture Overview)**

1. **Frontend (MVC - Razor Views)**
   * UI for Patients, Doctors, Receptionists, and Admin
   * Forms, Dashboards, Appointment Management
2. **Backend (ASP.NET MVC - C#)**
   * Business Logic Layer
   * Controllers (handle requests and logic)
   * Services (custom logic/services)
   * Repositories (handle DB access)
3. **Database (MS SQL Server)**
   * Tables for Patients, Doctors, Appointments, Medical Records, Billing, Users, Roles, etc.
4. **API (Optional but Recommended for Mobile App or Integration)**
   * RESTful APIs using ASP.NET Web API
   * Token-based authentication (JWT)
   * Enables mobile integration and external system communication
5. **Admin Section**
   * User and Role Management
   * System Settings
   * Reports and Logs
   * Data Backup/Restore
6. **Authentication & Authorization**
   * ASP.NET Identity or Custom JWT
   * Role-Based Access Control (RBAC)
   * Optional Two-Factor Authentication (2FA)

### ****Core Features of the Application****

#### **Patient Module**

* Registration and profile management
* Appointment booking
* View prescriptions and medical history
* Notification and reminders (SMS/Email)

#### **Doctor Module**

* View schedule and patient appointments
* Write prescriptions and diagnosis
* Access patient history
* Add medical records

#### **Receptionist Module**

* Manage bookings and cancellations
* Process payments
* Generate invoices

#### **Admin Module**

* CRUD users (doctors, patients, staff)
* Set roles and permissions
* View analytics and reports
* Manage inventory and billing

#### **Common Features**

* Login/Logout, Forgot Password
* Secure Session Management
* Search, Sort, Filter options
* Backup/Restore data
* Responsive Design

### ****Security and Authentication Approaches****

#### **Authentication Options**

* **ASP.NET Identity (Forms Authentication)**
  + Easy integration with MVC
  + Cookie-based sessions
* **JWT (JSON Web Token)**
  + Stateless
  + Ideal for APIs and mobile integration

#### **Authorization Options**

* **Role-Based Access Control (RBAC)**
  + Users assigned roles like Admin, Doctor, Patient
* **Claims-Based Authorization**
  + Based on user identity and attributes

#### **Security Practices**

* HTTPS for all connections
* Password hashing (e.g., BCrypt)
* SQL Injection prevention using parameterized queries/Entity Framework
* Rate limiting APIs
* CSRF and XSS protections

**Reservation Channels**

1. **Online (Web App)**
   * Patient logs into the website
   * Selects available doctor/time
   * Confirms and reserves
   * Auto-notification via Email/SMS
2. **By Phone (Manual by Receptionist)**
   * Receptionist receives a call
   * Logs in to receptionist dashboard
   * Books appointment manually on behalf of patient
   * Marks reservation as “Phone Reservation”
3. **Mobile App (optional future upgrade)**
   * Mobile app with same functionality as web
   * Calls the backend API
   * Useful for WhatsApp-like integration later
4. **Walk-In Reservation**
   * Patient visits clinic
   * Receptionist books directly in system

**Project Proposal Document**

**Title**: Clinic Management System  
**Objective**: To automate clinic operations like patient management, appointment scheduling, billing, and reports.  
**Tools**: ASP.NET MVC (C#), MS SQL Server, Entity Framework, Bootstrap  
**Target Audience**: Small to medium-sized clinics  
**Stakeholders**: Clinic Owner, Doctors, Admin Staff  
**Modules**:

* Patient Management
* Doctor Schedule
* Appointment Booking
* Medical Records
* Billing and Reports
* Admin Dashboard

## **Project Values (Benefits)**

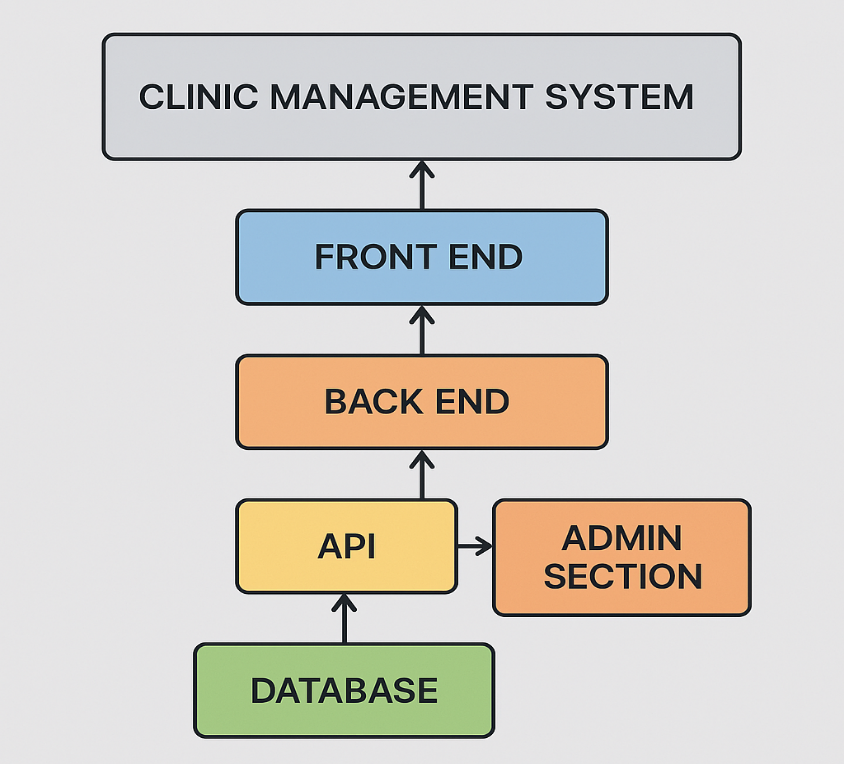
| **Value** | **Description** |
| --- | --- |
| **Efficiency** | Automates scheduling, billing, and patient management — reducing manual effort. |
| **Accuracy** | Reduces errors in patient records, prescriptions, and invoices. |
| **24/7 Accessibility** | Online booking available any time for patients and doctors. |
| **Better Patient Experience** | Shorter wait times, appointment reminders, access to history. |
| **Data Centralization** | All records (medical, financial, admin) in one secure system. |
| **Reports & Analytics** | Admins and doctors can generate performance, financial, and clinical reports. |
| **Scalability** | Easy to add more branches, doctors, or services. |
| **Security & Privacy** | Data access controlled through authentication & roles. |
| **Remote Access** | Doctors or admins can log in from anywhere securely. |
| **Compliance Ready** | Can support local data protection laws (HIPAA, GDPR equivalents). |

## **Project Risks (with Mitigation)**

| **Risk** | **Impact** | **Likelihood** | **Mitigation Strategy** |
| --- | --- | --- | --- |
| **Data Breach** | High | Medium | Implement strong authentication, HTTPS, password hashing, role-based access control |
| **Feature Scope Creep** | Medium | High | Stick to agreed MVP (Minimum Viable Product); manage change through approval workflow |
| **Technical Complexity** | Medium | Medium | Use modular design (MVC, layered architecture); build in phases |
| **User Resistance to New System** | Medium | High | Offer training and documentation; phased rollout |
| **Downtime or Data Loss** | High | Medium | Regular backups, disaster recovery plan, cloud-hosted DB if needed |
| **Performance Issues Under Load** | Medium | Low | Optimize DB queries, use caching, asynchronous processing for notifications |
| **Poor Internet Connectivity (for Online Use)** | Low | Medium | Allow offline queueing (future upgrade), enable phone/walk-in reservations |
| **Regulatory Non-compliance** | High | Low | Understand legal requirements, enforce data access rules, audit logs |
| **Mobile Device Compatibility (future app)** | Medium | Low | Design APIs now to support future mobile integration |
| **Staff Shortage or Turnover (during implementation)** | Low | Medium | Cross-train team, have good documentation for handovers |

### ****Feasibility Study****

| **Aspect** | **Analysis** |
| --- | --- |
| **Technical** | Feasible using .NET MVC and SQL Server |
| **Operational** | Will ease clinic workload and improve efficiency |
| **Economic** | Initial investment in development; cost-effective in long term |
| **Legal** | Must comply with data privacy laws (e.g., HIPAA or local equivalent) |
| **Schedule** | Estimated development time: 4-6 months |



## Work Breakdown Structure (WBS) – Clinic Management System

### ****Level 1: Project Phases****

1. **Initiation**
2. **Planning**
3. **Design**
4. **Development**
5. **Testing**
6. **Deployment**
7. **Training & Handover**
8. **Maintenance**
9. **Detailed WBS**

| **WBS Code** | **Task** | **Phase** |
| --- | --- | --- |
| 1.1 | Define goals and scope | Initiation |
| 1.2 | Feasibility study and risk assessment | Initiation |
| 2.1 | Stakeholder analysis | Planning |
| 2.2 | Requirement gathering | Planning |
| 2.3 | Project charter & approval | Planning |
| 2.4 | Resource and budget planning | Planning |
| 3.1 | System architecture design | Design |
| 3.2 | UI/UX wireframes (Admin, Doctor, Patient) | Design |
| 3.3 | Database schema design (ERD) | Design |
| 4.1 | Backend API Development (.NET MVC + SQL) | Development |
| 4.2 | Frontend – Admin Dashboard (Razor/MVC) | Development |
| 4.3 | Frontend – Doctor Portal | Development |
| 4.4 | Frontend – Patient Portal (Booking) | Development |
| 4.5 | Role-based Authentication & Permissions | Development |
| 4.6 | Appointment Scheduling Module | Development |
| 4.7 | Billing & Invoice Module | Development |
| 4.8 | Notifications (Email/SMS) | Development |
| 5.1 | Unit testing | Testing |
| 5.2 | System integration testing | Testing |
| 5.3 | User Acceptance Testing (UAT) | Testing |
| 6.1 | Hosting & deployment setup | Deployment |
| 6.2 | Live environment go-live | Deployment |
| 7.1 | Admin & Staff Training | Training |
| 7.2 | Documentation delivery | Training |
| 8.1 | Bug fixes and updates | Maintenance |
| 8.2 | Support & feedback integration | Maintenance |

## Gantt Timeline (8-Week Plan – Sample)

| **Week** | **Main Activities** |
| --- | --- |
| Week 1 | Initiation + Planning (WBS 1–2) |
| Week 2 | Design UI, DB Schema (WBS 3) |
| Week 3–5 | Development Phase (WBS 4) |
| Week 6 | Testing Phase (WBS 5) |
| Week 7 | Deployment + Training (WBS 6–7) |
| Week 8 | Feedback, Maintenance Begins (WBS 8) |

