**Unit Testing Requirements for the Hospital Management System (HMS)**

**1. Functional Testing Areas**

Functional testing ensures that all features perform as expected. Below are the key website features and functions to test:

**1.1. Patient Management:**

* **Registration/Login**: Validate that users (patients) can register and log in successfully.
* **Appointment Booking**: Ensure patients can search for doctors by specialty and book appointments.
* **Medical Records**: Check if patients can view their medical history and doctor notes.
* **Billing**: Verify patients can see their bills and payment options.

**1.2. Doctor Management:**

* **Dashboard Access**: Ensure doctors can log in to their personalized dashboards.
* **Appointment Management**: Test if doctors can view, accept, or cancel appointments.
* **Medical Record Input**: Verify doctors can input and update patient records.

**1.3. Admin Panel:**

* **User Management**: Validate the ability to manage patient and doctor accounts.
* **Appointment Overview**: Ensure admins can oversee all appointments.
* **Staff Scheduling**: Test scheduling and shift management functionalities.
* **System Monitoring**: Ensure the system logs critical data (e.g., user activity, appointments).

**2. Non-Functional Testing Areas**

Non-functional testing focuses on the performance, security, and usability of the system.

**2.1. Performance:**

* **Load Testing**: Ensure the system can handle a large number of concurrent users (patients, doctors, and staff).
* **Response Time**: Validate that the system responds quickly, especially for appointment bookings and dashboard updates.

**2.2. Security:**

* **Data Encryption**: Verify that sensitive data (medical records, billing) is encrypted both at rest and in transit.
* **Role-Based Access**: Ensure that access to features is restricted based on user roles (admin, doctor, patient).
* **Session Management**: Test for proper session expiration and secure login mechanisms.

**2.3. Usability:**

* **UI Consistency**: Ensure the user interface is intuitive and consistent across different pages.
* **Cross-Browser Compatibility**: Validate that the website works well across different browsers (Chrome, Firefox, Edge).
* **Localization**: Ensure the site supports multiple languages, if applicable.

**2.4. Scalability:**

* **Database Scaling**: Test the database for its ability to scale as the number of users increases.
* **Component Isolation**: Ensure that each component of the system (appointments, records, billing) functions independently and can scale without affecting other modules.

**3. Environment Setup for Testing**

**3.1. Development Tools:**

* **Visual Studio**: Ensure the latest version of Visual Studio is installed with all necessary extensions for C# and ASP.NET Core.
* **SQL Server Management Studio**: Set up for managing and testing the database.
* **Testing Frameworks**:
  + **NUnit/xUnit** for unit testing.
  + **Moq** for mocking dependencies.
  + **Selenium** for automated UI testing (optional).

**3.2. Staging Environment:**

* Set up a staging environment that mirrors the production environment.
* The environment should include:
  + ASP.NET MVC and Core for backend services.
  + SQL Server database.
  + Access to simulated patient, doctor, and admin accounts for testing.

**3.3. Test Data:**

* Prepare a set of test data, including user profiles (patients, doctors, Nurses), appointment schedules, medical records, and billing information.

**4. Website Features for Testing**

* **User Registration/Login**: Check registration with validation (correct/incorrect input).
* **Forgot Password**: Test password reset functionality.
* **Appointment Scheduling**: Ensure that appointment times are properly calculated, conflicts are avoided, and real-time availability is displayed.
* **Medical History Display**: Verify accurate display of patient medical history.
* **Billing Integration**: Test multiple billing methods (credit card, insurance, etc.).