

Smart Hospital Management System



Abdul Sarim Khan – (021-22-62527)

Iqra University

6/1/2025

Database and Management Systems – CCP

Hussain Bux Amur

Table of Contents

Abstract:	1
1. Introduction:	2
2. Literature Review:	3
3. Methodology:	4
3.1. Database Architecture	4
3.2. Implementing Security:	4
3.3. Stored Procedures:	4
3.4. Indexes:	4
3.5. Triggers:	4
3.6. Dashboard Construction:	5
3.7. Technologies Implemented:	5
4. Results:	5
5. Discussion:	8
6. Conclusion:	8
7. Queries:	9

Abstract:

This project aims to enhance healthcare operational processes by means of a safe and effective hospital management database system presented here in design and implementation. Among the fundamental features of the system are patient, doctor, appointment, and hospital bed management. Integration of encryption methods and role-based access control (RBAC) guarantees data confidentiality and integrity, so safeguarding private patient medical history. While a real-time dashboard created using Python and Streamlit offers simple monitoring and visualization of hospital resources, the solution uses Microsoft SQL Server with stored procedures for strong data handling. Through live data insights, the put in place system improves data security, simplifies hospital operations, and supports informed decision-making.

Smart Hospital Management System

1. Introduction:

Hospital management systems play a critical role in enhancing the efficiency and quality of healthcare delivery by automating administrative and clinical processes. Effective management of patient records, appointments, and hospital resources such as beds and medical staff is essential for improving patient care and operational workflows. This project was motivated by the need to develop a secure, automated system that safeguards sensitive patient information while providing real-time management capabilities.

The primary objectives of this project include securing patient data through encryption, implementing role-based access control to ensure data privacy, and streamlining the scheduling of appointments and allocation of beds. The system supports key hospital functions such as managing patient and doctor information, appointment bookings, and bed availability.

The scope of the system covers database design, security implementation, and the creation of a real-time dashboard for monitoring hospital resources. However, certain limitations exist, such as the absence of password hashing and the focus on core hospital operations without integration of advanced features like billing or electronic prescriptions. Future enhancements may address these limitations.

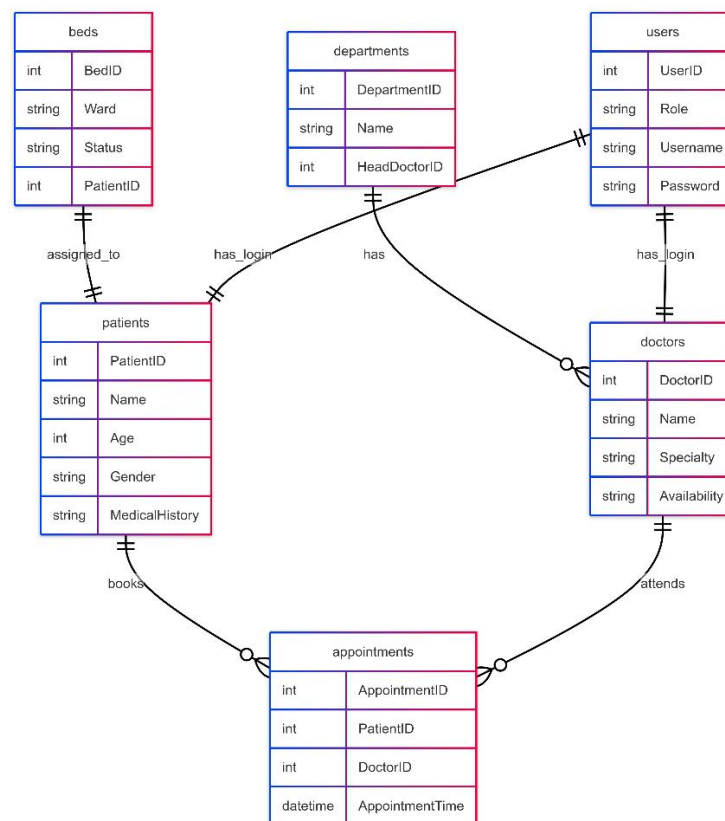


Figure 1- Entity Relational Diagram

2. Literature Review:

Smart Hospital management systems (SHMS) are critical tools in modern healthcare institutions, designed to streamline administrative, clinical, and operational processes. Over the past decades, various SHMS solutions have been developed to manage patient records, appointment scheduling, bed allocation, and staff coordination. Despite advancements, many existing systems face persistent challenges, particularly regarding data security, system scalability, and integration of real-time monitoring capabilities. Unauthorized access to sensitive medical data remains a significant concern, as breaches can lead to severe legal and ethical consequences.

To address data confidentiality, encryption has become a fundamental practice in healthcare information systems. Numerous studies emphasize the use of symmetric and asymmetric encryption techniques to safeguard sensitive fields such as patient medical histories and personal identification details. Symmetric key encryption, due to its balance of security and computational efficiency, is widely adopted in database-level encryption, ensuring that sensitive data remains protected even if the database is compromised.

In addition to encryption, Role-Based Access Control (RBAC) is extensively researched and implemented in healthcare systems to restrict data access based on user roles. RBAC enhances security by ensuring that users such as doctors, nurses, administrators, and patients have access only to information relevant to their responsibilities. Literature indicates that RBAC not only reduces the risk of data leakage but also simplifies user management by aligning permissions with organizational roles. Implementations of RBAC have proven effective in complying with healthcare regulations such as HIPAA and GDPR, which mandate strict control over patient data access.

Real-time monitoring and dashboard visualization are increasingly recognized as vital components in SHMS to improve operational awareness and decision-making. Dashboards consolidate critical data such as bed occupancy rates, upcoming appointments, and patient statuses into interactive, easily interpretable formats. Several commercial and academic studies highlight the positive impact of such tools in optimizing hospital workflows, reducing patient wait times, and enabling proactive resource allocation.

This project synthesizes these well-established approaches—encryption for data security, RBAC for controlled access, and real-time dashboards for operational oversight—into a cohesive hospital management database system. The design and implementation draw upon best practices documented in recent research and healthcare IT solutions, aiming to address common shortcomings and deliver a secure, efficient, and user-friendly platform.

3. Methodology:

3.1.Database Architecture

The project is based on a relational database shaped by required hospital entities produced with an entity-relationship diagram (ERD). Important tables call for users, patients, doctors, beds, and appointments. Ensuring consistent authentication, the users table controls all system users including administrative, doctor, and patient. By means of foreign keys, the tables of patients and doctors preserve one-to-one relationships and extend user information with domain-specific traits. The beds table records ward locations and patient assignments; the appointments table links doctors and patients with scheduled times. Data integrity is enforced by primary and foreign keys, special constraints, and check restrictions—that is, role validation.

3.2.Implementing Security:

Particularly in relation to SQL Server generates symmetric keys, master keys, certificates to securely encrypt data at rest, the system uses symmetric key encryption to protect private patient data including medical histories. Transparency in encrypted and decrypted data access ensures that only authorised roles could access decrypted data within stored operations. Applying role-based access control (RBAC) by defining user roles in the users table and enforcing access rights using stored procedures helps one to do so. This design reduces data visibility and operations depending on the user's intended role, so stopping undesired access.

3.3.Stored Procedures:

Store systems record all Create, Read, Update, and Delete (CRUD) activities for beds, doctors, patients, and appointments. Through control of database interactions, this encapsulation continuously supports corporate logic and enhances security. Soft delete capability made possible by an IsActive flag in patients and doctors tables helps to deactivate records so preserving historical data and facilitating audits without actual deletion.

3.4.Indexes:

Query performance is optimized by nonclustered indexes constructed on often queried columns including AppointmentTime in the appointments table, DoctorID and PatientID foreign keys, and IsActive flags in patients and doctors tables. Quick filtering of accessible rather than occupied beds made possible by indexing on the BedStatus column enhances responsiveness in critical operations.

3.5.Triggers:

A database trigger built into the beds table keeps consistency between patient assignment and bed occupancy status. Should a patient be assigned, the trigger automatically sets the Bed Status to "Occupied" upon entry or update of a bed record; otherwise, it sets "Availability". This automation helps to reduce hand-made errors and guarantees correct real-time status tracking.

3.6. Dashboard Construction:

The system includes a real-time Python and Streamlit library monitoring dashboard. Running searches against pre-defined views and tables, the dashboard connects via the pyodbc driver to the SQL Server database. For doctors, present patients, upcoming visits, and open beds, it provides simple visuals. The interface uses responsive designs to properly present significant data, so supporting administrative decision-making.

3.7. Technologies Implemented:

While the project is built using Microsoft SQL Server for security features and database management, T-SQL stored procedures handle simple logic. The front-end dashboard built using Python's Streamlit architecture lets interactive web apps expand rapidly. The pyodbc library manages database connectivity, so allowing perfect Python and SQL Server integration.

4. Results:

Including all fundamental tables and relationships as specified in the ERD, the Smart Hospital Management System database was effectively developed and installed. Constraints and referential integrity guaranteed consistent and accurate data storage among entities including patients, doctors, visits, and beds.

With symmetric key cryptography applied to sensitive patient medical histories, data security mechanisms were essentially combined. By enforcing access limitations, Role-Based Access Control (RBAC) let only authorized users view or alter protected data, so preserving patient privacy.

Stored procedures were used extensively to test CRUD operations, proving consistent creation, retrieval, updating, and soft deletion of records throughout all modules. Correctly updating bed occupancy status depending on patient assignment, the bed status trigger function did not require human intervention.

The real-time dashboard offered dynamic views of active patients and doctors, forthcoming appointments, and accessible beds. Effective monitoring and management of hospital resources made possible by this visualization allowed. By greatly improving query performance, indexing on important columns accelerated data retrieval and enhanced responsiveness during peak operations.

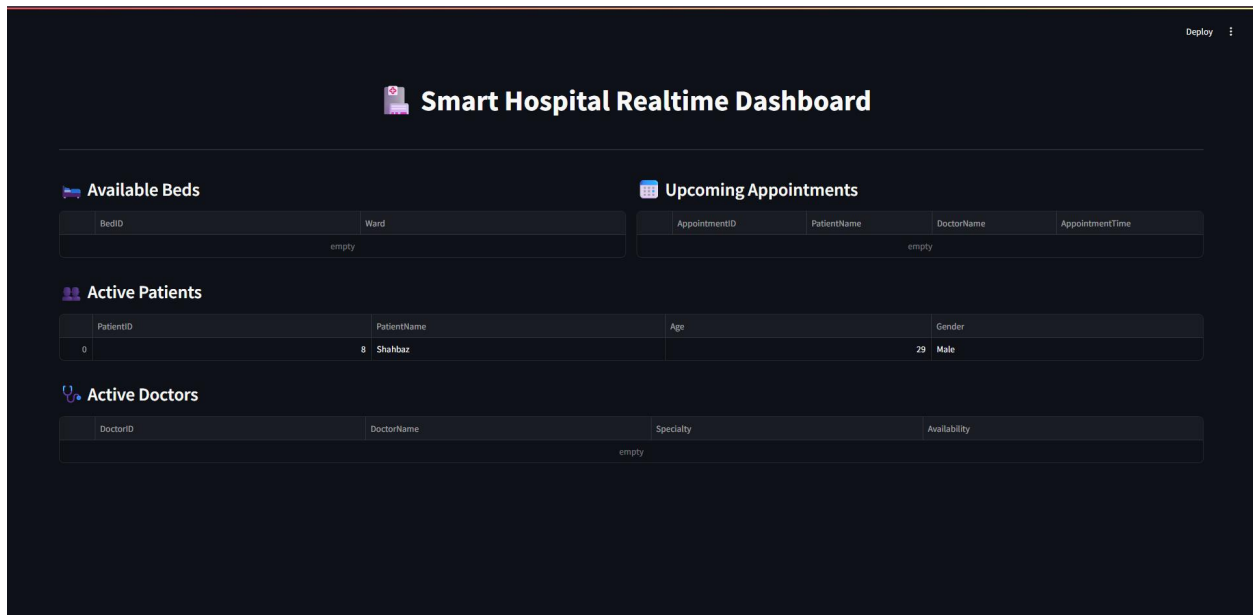


Figure 2- Snapshot of Realtime Dashboard

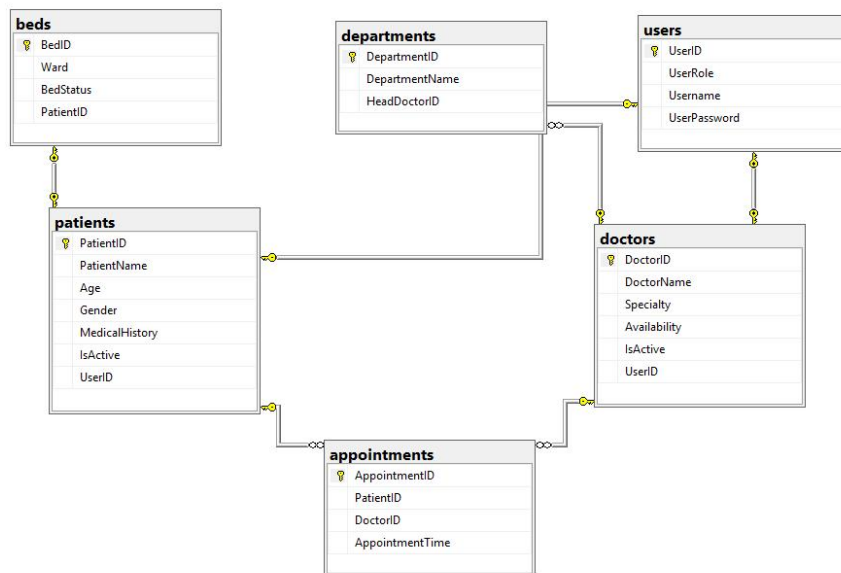


Figure 3- Final Entity Relationship Diagram

```

-- Ensure you are in the correct database
USE Smart_Hospital_Management_System;
GO

-- Step 1: Create Master Key (only once)
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Strong@MasterKey2025';
GO

-- Step 2: Create Certificate
CREATE CERTIFICATE Cert_HealthData
WITH SUBJECT = 'Health Data Encryption Certificate';
GO

-- Step 3: Create Symmetric Key
CREATE SYMMETRIC KEY SymKey_Health
WITH ALGORITHM = AES_256
ENCRYPTION BY CERTIFICATE Cert_HealthData;
GO

```

Figure 4- Query Creating Digital Certificate and Symmetric Key

```

USE [Smart_Hospital_Management_System]
GO
ALTER PROCEDURE [dbo].[GetActivePatientsByUserRole]
    @UserID INT
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @UserRole NVARCHAR(50);
    SELECT @UserRole = UserRole FROM users WHERE UserID = @UserID;
    IF @UserRole IN ('admin', 'doctor')
    BEGIN
        -- Open symmetric key before decryption
        OPEN SYMMETRIC KEY SymKey_Health
        DECRYPTION BY CERTIFICATE Cert_HealthData;
        SELECT
            p.PatientID,
            p.PatientName,
            p.Age,
            p.Gender,
            CONVERT(NVARCHAR(MAX), DECRYPTBYKEY(p.MedicalHistory)) AS MedicalHistory,
            p.IsActive
        FROM
            patients p
        WHERE
            p.IsActive = 1;
        -- Close key after use
        CLOSE SYMMETRIC KEY SymKey_Health;
    END
    ELSE
    BEGIN
        PRINT 'Access Denied: Insufficient permissions.';
    END
END;
GO

```

Figure 5- Query Implementing Role Based Access Control

5. Discussion:

Among the major advantages of the put in place Smart Hospital Management System are improved data security by means of encryption and role-based access control, so safeguarding private patient data from illegal access. The system automates important hospital tasks including patient records, appointment scheduling, and bed allocation, so simplifying management. Furthermore, the real-time dashboard offers insightful operational information that helps to guide quick and wise decisions.

Many difficulties arose during development. Using encryption inside SQL Server calls for careful key and certificate handling to maintain data confidentiality without compromising performance. Managing symmetric keys and guaranteeing safe key access brought complexity. Designing and implementing RBAC via stored procedures called for careful role validation and testing to stop data leaks or privilege escalation.





















The system has constraints even with its strengths. Passwords represent a vital area for future improvement since they are kept in plain text instead of using safe hash techniques. Furthermore, although appropriate for medium-sized hospital operations, the system may need architectural changes to scale effectively for bigger institutions or to interface with outside systems including billing or electronic health records.

Using SQL Server capabilities and Python-based visualization, this system stresses strong data security and modular design unlike standard hospital management solutions. Although commercial products usually have more features and integrations, this project shows a targeted and efficient approach to main hospital management issues with a solid security basis.

6. Conclusion:

Integrating encryption to safeguard private patient data and role-based access control to enforce rigorous data privacy, this project effectively designed and built a secure and efficient hospital management database system. Using stored procedures simplified database operations while still upholding consistent business logic and security. Using Python and Streamlit, the real-time dashboard created provides an easy interface for tracking important hospital resources including beds, appointments, and active personnel, so facilitating better operational control and decision-making. All things considered, the system emphasizes the critical need of strong data security in healthcare settings by showing how contemporary database technologies and access limits might protect patient privacy while improving hospital management effectiveness.

7. Queries:

 Views Creation Schema.sql	 Using Key & Certificate.sql	 Update Patient Procedure (Altered).:	 Read Active Patients Procedure ( Read Active Patients (Altered).sq	 RBAC Schema.sql
 RBAC Schema (Altered).sql	 Patient Procedures Schema.sql	 Patient Add.sql	 Index Creation Schema.sql	 Execution Queries.sql	 Doctors Procedures Schema.sql
 Database Schema.sql	 Creating Key & Certificate.sql	 Bed Trigger Schema.sql	 Bed Procedures Schema.sql	 Appointments Procedures Schema.	 Add Patient Procedure (Altered).:
 SMHS.bak	 SMHS-Dashboard.p y				