User Requirements

NexCare – Hospital Administrative Operations Platform

1. Introduction

This document defines the **high-level user requirements** for the NexCare Hospital Administrative Operations Platform. The requirements describe the system services from the users' perspective and focus on *what the system should provide*, rather than how it is implemented.

These requirements are intentionally **concise and abstract**, making them suitable for a student full-stack project and aligning with IEEE SRS guidelines. Detailed functional behavior and system constraints are specified in later sections of the SRS.

---

2. User Classes

The NexCare system supports the following user classes:

- **Patients** – Book appointments, request ambulance services, make payments, and submit feedback.
- **Administrative Staff** – Manage appointments, patient check-ins, billing, bed allocation, inventory, and staff schedules.
- **Ambulance Staff** – Handle ambulance dispatch requests and update transport status.
- **Admin (Hospital Management)** – Monitor hospital operations, view dashboards, and generate reports.

---

3. User Service Requirements

UR-1 Appointment & Queue Management

The system shall allow patients to book appointments and view appointment status, while allowing administrative staff to manage schedules, queues, and cancellations.

UR-2 Feedback & Complaint Management

The system shall allow patients to submit feedback or complaints and allow administrative staff to track, review, and update their resolution status.

UR-3 Ambulance Request & Coordination

The system shall allow patients to request ambulance services and allow ambulance staff and administrative staff to monitor and update request status.

UR-4 Billing & Payments

The system shall allow patients to view bills, make online payments, and access receipts, while allowing administrative staff to manage billing information.

UR-5 Patient Check-in & Movement Tracking

The system shall allow administrative staff to record patient check-ins and track patient movement across hospital departments for operational coordination.

UR-6 Bed, Room, and Ward Allocation

The system shall allow administrative staff to view availability and allocate beds or rooms while preventing duplicate or conflicting allocations.

UR-7 Inventory & Asset Management

The system shall allow administrative staff to view and update inventory and asset information and receive alerts when inventory levels or asset conditions require attention.

UR-8 Staff Scheduling

The system shall allow administrative staff to create and manage staff schedules and identify conflicts or understaffed periods.

UR-9 Wait-Time & Bottleneck Monitoring

The system shall allow administrative staff to monitor patient wait times and identify service bottlenecks affecting hospital operations.

UR-10 Operations Dashboard & Reporting

The system shall provide hospital management with dashboards, operational reports, and audit logs to monitor overall hospital performance.

---

4. User-Visible Non-Functional Expectations

From a user perspective, the system shall: - Respond to common actions in a timely manner - Restrict access based on assigned user roles - Present information clearly and understandably to non-technical users - Protect patient and hospital information from unauthorized access - Be available during normal hospital operating hours

---

5. Standards and Constraints

- The system shall follow standard web usability and accessibility practices.
- Patient information shall be handled according to applicable privacy and confidentiality guidelines.
- The system shall support commonly used modern web browsers.
- Billing functionality shall integrate with a secure and approved payment gateway.
- Key administrative actions shall be auditable.

Functional Requirements

Hospital Administrative Operations Platform *(Non-Clinical Workflows)*

1. Scope

This document defines the functional requirements for the Hospital Administrative Operations Platform. All actors, use cases, and system integrations are derived directly from the use case diagram. Every requirement in this document pertains exclusively to non-clinical, administrative workflows — including appointment scheduling, patient check-in and movement tracking, billing, ambulance coordination, inventory management, staff scheduling, and hospital operations monitoring. The platform is designed to support, and not to replace, human decision-making in hospital administration.

---

2. Definitions

Term	Definition
HMIS	Hospital Management Information System – the core system enabling paperless hospital operations.
EMR	Electronic Medical Registration – system used for patient appointment tokens and record keeping.
Token System	A queue-based appointment numbering system integrated with EMR that calls patients in order.
Triage	A prioritisation process that classifies patients by urgency to determine appointment order.
CPR Ambulance	An ambulance unit equipped with a Cardiopulmonary Resuscitation team and advanced life-support equipment.
Pre-authorisation	Mandatory insurance approval required before a cashless treatment can be initiated.
AMC	Annual Maintenance Contract – a scheduled service agreement for biomedical equipment.
CMC	Complete Maintenance Care – a comprehensive care plan covering medical machinery upkeep.
SPOC	Single Point of Contact – the administrative coordinator in the emergency department.

---

3. Actors

3.1 Primary Actors

Primary actors are the human users who interact directly with the system.

Actor	Description
Patient	End user who books appointments, submits feedback, requests ambulance services, and views or pays bills through the platform.

Actor	Description
Administrative Staff	Front-office and operations personnel who manage appointments, handle patient check-in, allocate beds/rooms/wards, manage inventory and assets, and plan staff shifts.
Ambulance Staff	Personnel responsible for accepting ambulance assignments and updating real-time status of dispatched units.
Admin	Senior administrator who monitors overall hospital operations dashboards and reviews system-generated reports.

3.2 System Actors (External Integrations)

System actors represent external services that integrate with the platform to fulfill use cases.

System Actor	Role in the Platform
Notification Service	Sends automated alerts to patients and staff regarding feedback acknowledgement, appointment updates, and check-in status changes. Also receives triggers from the Wait-Time & Bottleneck Detection module.
Payment Gateway	Processes online bill payments initiated by patients and confirms transaction status back to the platform.
Triage System	Classifies incoming patients by urgency and feeds priority data into the appointment booking and management workflows.
GPS Tracking System	Provides real-time geolocation of ambulance units during dispatch and en-route travel.
Reporting System	Aggregates operational data across all modules and generates dashboards and summary reports for the Admin.

---

4. Functional Requirements

Each requirement is written as a testable statement describing what the system shall do. The Actor column identifies which primary actor initiates or is primarily involved in the use case. “System” denotes an automated process with no direct primary-actor trigger.

4.1 Triage-Based Appointment Booking

ID	Functional Requirement	Actor
FR-01	The system shall allow patients to book appointments by selecting department, date, and time slot.	Patient
FR-02	The system shall assign a unique appointment token to each confirmed booking.	System
FR-03	The system shall prevent duplicate token assignment for the same time slot.	System
FR-04	The system shall allow administrative staff to view, reschedule, or cancel appointments.	Admin. Staff
FR-05	The system shall notify patients when their token is called or appointment status changes.	System

---

4.2 Patient Check-in & Movement Tracking

ID	Functional Requirement	Actor
FR-06	The system shall allow administrative staff to register patient check-in with timestamp.	Admin. Staff

ID	Functional Requirement	Actor
FR-07	The system shall track the current department/location of a checked-in patient.	Admin. Staff
FR-08	The system shall maintain a movement history log for each patient visit.	System

---

4.3 Request Ambulance & Dispatch

ID	Functional Requirement	Actor
FR-09	The system shall allow patients or staff to submit an ambulance request with location details.	Patient / Admin. Staff
FR-10	The system shall allow ambulance staff to accept assignments and update dispatch status.	Ambulance Staff
FR-11	The system shall display real-time ambulance status and estimated arrival time.	System

---

4.4 Bed / Room / Ward Allocation

ID	Functional Requirement	Actor
FR-12	The system shall maintain real-time availability status of beds and rooms.	System
FR-13	The system shall allow administrative staff to allocate a bed or room to a patient.	Admin. Staff

ID	Functional Requirement	Actor
FR-14	The system shall prevent allocation of unavailable or blocked beds.	System

---

4.5 View & Pay Bill

ID	Functional Requirement	Actor
FR-15	The system shall display an itemized bill to the patient.	Patient
FR-16	The system shall allow patients to pay bills online through an integrated payment gateway.	Patient
FR-17	The system shall generate a receipt upon successful payment.	System

---

4.6 Inventory & Asset Management

ID	Functional Requirement	Actor
FR-18	The system shall maintain a registry of hospital assets and inventory items.	Admin. Staff
FR-19	The system shall alert administrative staff when inventory levels fall below a defined threshold.	System

---

4.7 Staff Shift & Availability Management

ID	Functional Requirement	Actor
FR-20	The system shall allow administrative staff to create and modify staff duty rosters.	Admin. Staff
FR-21	The system shall allow staff leave requests to be recorded and reviewed.	Admin. Staff

---

4.8 Monitor Hospital Operations & Reports

ID	Functional Requirement	Actor
FR-22	The system shall provide an operational dashboard for administrators.	Admin
FR-23	The system shall generate summary reports on appointments, beds, ambulances, and inventory.	System
FR-24	The system shall maintain an audit trail of critical administrative actions.	System

---

4.9 Feedback & Complaint Management

ID	Functional Requirement	Actor
FR-25	The system shall allow patients to submit feedback or complaints related to hospital administrative services.	Patient

ID	Functional Requirement	Actor
FR-26	The system shall generate a unique reference ID for each submitted feedback or complaint.	System
FR-27	The system shall allow administrative staff to view, update, and close feedback or complaint records.	Admin. Staff

---

5. Summary

Category	FR Count
Appointment & Queue Management	5
Patient Check-in & Movement	3
Ambulance Operations	3
Bed / Room Allocation	3
Billing & Payments	3
Inventory Management	2
Staff Management	2
Monitoring & Reports	3
Feedback & Complaint Management	3
Total Functional Requirements	**27**

Non-Functional Requirements

Hospital Administrative Operations Platform

1. Scope

This document defines the non-functional requirements (NFRs) for the Hospital Administrative Operations Platform student project. These requirements have been streamlined and scoped appropriately for a full-stack development project with realistic testing and validation constraints.

The NFRs focus on essential system qualities that can be implemented, demonstrated, and tested within an academic project timeline, while maintaining professional development standards. All overly ambitious enterprise-grade requirements have been removed or scaled to student-project-appropriate levels.

2. Definitions

Term	Definition
Response Time	The duration between a user action and the system's visible response.
Session	A period of continuous user activity from login to logout or timeout.
Concurrent Users	The number of users actively using the system at the same time.
Functional Module	A distinct feature set (e.g., Appointment Booking, Billing, Ambulance Tracking).
API	Application Programming Interface – the contract for communication between system components.

3. NFR Categories and Rationale

The requirements are organized into six essential categories suitable for a student project:

Category	Rationale
Performance	Ensures the system responds quickly enough for practical use during demos and testing. Critical for appointment booking and real-time ambulance tracking.
Security	Protects patient data and ensures proper access control. Essential for any healthcare-related system, even in academic contexts.
Usability	Makes the system learnable and usable by teaching assistants, instructors, and demo users without extensive training.
Reliability	Ensures the system works consistently during testing, demos, and evaluation periods without frequent crashes.
Maintainability	Enables team members to understand, modify, and extend the codebase. Critical for iterative development and bug fixes.

Category	Rationale
Testability	Ensures the system can be properly tested and validated within project constraints.

---

4. Non-Functional Requirements

Each requirement is testable and measurable within the constraints of a student project.

4.1 Performance Requirements

ID	Requirement
NFR-01	The system shall respond to any user action (login, booking, form submission) within **3 seconds** under normal load (up to 10 concurrent users).
NFR-02	The system shall display appointment availability or queue status within **2 seconds** of a user request.
NFR-03	The system shall complete bill payment processing and display confirmation within **5 seconds** using a sandbox payment gateway.

Rationale: These cover all performance expectations required for demos and evaluations without load-testing complexity.

4.2 Security Requirements

ID	Requirement
NFR-04	The system shall implement **Role-Based Access Control (RBAC)** for Patient, Administrative Staff, and Admin roles.
NFR-05	The system shall require authenticated login before allowing access to any protected functionality.
NFR-06	User passwords shall be securely hashed (e.g., bcrypt) and never stored in plain text.
NFR-07	The system shall automatically log out users after **30 minutes of inactivity**.
NFR-08	The system shall validate and sanitize all user inputs to prevent SQL injection and XSS attacks.

4.3 Usability Requirements

ID	Requirement
NFR-09	The system shall provide **role-specific dashboards** showing only relevant data and actions.
NFR-10	All user forms shall include clear labels, validation messages, and meaningful error feedback.
NFR-11	The system shall confirm all destructive actions (delete, cancel, discharge) before execution.
NFR-12	A new user shall be able to complete core tasks using only README instructions, without formal training.

Rationale: These ensure evaluator-friendly demos and good UX without advanced UI research requirements.

4.4 Reliability Requirements

ID	Requirement
NFR-13	The system shall prevent duplicate operations such as double booking of appointments or beds.
NFR-14	The system shall handle invalid inputs or system errors gracefully without crashing or exposing stack traces.
NFR-15	All critical operations shall maintain data consistency using basic database transactions.

4.5 Maintainability & Testability Requirements

ID	Requirement
NFR-16	The system shall follow a **clear separation of concerns** between frontend, backend, and database layers.
NFR-17	The project repository shall include a README with setup instructions and execution steps.
NFR-18	Backend APIs shall be independently testable using tools such as Postman.
NFR-19	The system shall include seed data to support repeatable testing and demonstration.

Rationale: These are essential for grading, collaboration, and reproducibility.

4.6 Summary

Category	Requirement IDs	Count
Performance	NFR-01 to NFR-03	3
Security	NFR-04 to NFR-08	5
Usability	NFR-09 to NFR-12	4
Reliability	NFR-13 to NFR-15	3
Maintainability & Testability	NFR-16 to NFR-19	4
Total		**19**

Testing and Validation Guide

This section provides practical guidance on how to test and validate each NFR category within project constraints.

1. Performance Testing

Tools: Browser Developer Tools (Network tab), Apache JMeter (basic load testing), or custom scripts

Method: 1. Use browser Developer Tools to measure response times for key actions 2. Create 10 concurrent user sessions and verify no degradation 3. Document response times in a test report with screenshots

2. Security Testing

Tools: Manual testing, database inspection, browser DevTools

Method: 1. Verify RBAC by attempting to access restricted pages with different user roles 2. Check database to confirm passwords are hashed, not plain text 3. Test session timeout by leaving browser idle for 30 minutes 4. Attempt SQL injection in input fields (e.g., username: `admin' OR '1'='1`)

3. Usability Testing

Tools: Peer review, instructor evaluation

Method: 1. Give system to a peer who hasn't seen it before with only README instructions 2. Ask them to complete 5 core tasks and time how long it takes 3. Test on different screen sizes (desktop, tablet) to verify responsiveness 4. Intentionally trigger errors and verify messages are user-friendly

---

4. Reliability Testing

Tools: Load testing scripts, log files

Method: 1. Run the system continuously for 4 hours with periodic test actions 2. Monitor memory usage to detect leaks (should remain stable) 3. Test duplicate appointment booking by two users simultaneously 4. Review error logs to confirm all errors are properly logged

---

5. Maintainability Validation

Tools: Code review, architecture documentation

Method: 1. Review code structure to verify clear separation of frontend/backend/database 2. Check API endpoints follow RESTful naming conventions 3. Verify README includes all necessary setup and run instructions 4. Have another team member attempt to add a new feature to assess code clarity

---

6. Testability Validation

Tools: Postman, Jest/Mocha/PyTest (depending on stack), database seed scripts

Method: 1. Test all API endpoints independently using Postman or curl 2. Run database seed script to verify test data loads correctly 3. Execute automated tests (if implemented) and document results 4. Verify external integrations can be tested in sandbox/mock mode

---

UML Diagrams

1. Use Case Diagram

2. Activity Diagrams

2.1 Triage-Based Appointment Booking, Rescheduling and Cancelling

2.2 Billing and Payment processing

2.3 Feedback and complaint handling

2.4 Inventory & Asset Stock Management

2.5 Ambulance Request, Assignment & Tracking

2.6 Monitor Hospital Operations & Reports

2.7 Patient Check-In & Movement Tracking

2.8 Bed/Room/Ward Allocation Process

3 Sequence diagram