?? Software Requirements Specification (SRS)

?? 1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to define a clear and detailed understanding of the functional and non-functional requirements for the MediSphere – Smart Hospital Ecosystem. This system is being developed to digitize and automate the essential operations of a modern hospital, thereby enhancing the efficiency, accuracy, and accessibility of healthcare services.
The MediSphere system aims to provide a centralized, web-based solution that facilitates the management of patient data, appointment scheduling, doctor-patient interactions, medical records, billing, pharmacy stock, laboratory services, and administrative analytics. The platform supports different user roles such as administrators, doctors, nurses, lab technicians, receptionists, and patients, with each having access only to the functionalities relevant to their role.
One of the key goals is to reduce manual paperwork, minimize human error, and improve overall patient care through real-time access to accurate information. The system also incorporates modern advancements such as telemedicine consultations, AI-powered symptom checkers(optional), and emergency alert mechanisms, making it highly adaptable to current healthcare needs.

1.2 Scope

MediSphere – Smart Hospital Ecosystem is a comprehensive hospital management solution designed to streamline and digitize all major operations of a healthcare institution. It is intended to support the day-to-day functions of hospital staff, including administrators, doctors, nurses, lab technicians, and receptionists, while also offering features for patients to engage directly with their healthcare providers.

The system automates key workflows such as patient registration, appointment scheduling, medical history tracking, billing, pharmacy stock management, laboratory services, and internal communication, resulting in reduced manual work, fewer errors, and improved coordination across departments.
What sets MediSphere apart is its integration of advanced, next-generation features that make it suitable for smart and modern hospitals:
Real-Time Emergency Services: Includes features like ambulance request tracking and critical case alerts to doctors and staff, ensuring quick responses in emergencies.
Analytics & Dashboards: Offers powerful administrative tools to visualize and monitor key performance indicators such as revenue, patient inflow, top health issues, and resource utilization.
Teleconsultation (Optional): Facilitates remote consultations through secure video/audio calls and chat, allowing patients to connect with doctors without physical visits.
AI-Driven Symptom Checker (Optional): Provides patients with preliminary health insights based on their entered symptoms using artificial intelligence algorithms.
MediSphere supports role-based access, ensuring that each user type (e.g., doctor, nurse, admin, patient) interacts with a tailored interface and set of functionalities relevant to their responsibilities. Its modular structure allows flexibility for hospitals to enable or disable certain features depending on their needs.
The ultimate goal of MediSphere is to create a smart, efficient, and patient-centric ecosystem for hospitals, bringing together technology and healthcare to deliver better service, faster processes, and improved medical outcomes.

1.3 Intended Audience

Developers & Engineers
Hospital Administrators
Final-Year Project Evaluators
QA Testers
Patients (End-Users)

Overall Description (Java-Based System)
2.1 Product Perspective
MediSphere is a Java-based hospital management system designed to handle all essential hospital activities in one place. It simplifies patient registration, appointment booking, billing, medicine stock(optional), lab tests(optional), and emergency services. It also includes some modern features and online doctor consultations.

2.2 Product Functions
Role-based user login (Admin, Doctor, Nurse, Patient, Receptionist)
Patient registration and health record management
Appointment booking and doctor scheduling
Billing and pharmacy inventory control
Emergency services (ambulance requests, alerts)
Admin dashboards and reports

2.3 User Classes and Characteristics
Admin: Full control over users, settings, and data
Doctor: Access to patient details, prescriptions, and schedule
Nurse/Lab Technician: Access to patient tests and reports
Patient: Book appointments, view reports, and health info
Receptionist: Handles appointment scheduling and patient queues
2.4 Operating Environment
Programming Language: Java
Frameworks/GUI:java swing
Database: MySQL
IDE used:IntelliJ IDE
Runs on: Desktop or laptop with Java-supported browsers
Dependencies: JDBC (for DB connectivity)
2.5 Design Constraints
User-friendly and responsive interface
Secure role-based login system
Real-time data for appointments and alerts
Easily extendable for future features like wearable device integration

? 3. System Features & Requirements
3.1 User Authentication Module
Secure login/logout functionalities
Role-based access control
3.2 Patient Module
Registration and profile management
Appointment scheduling
Access to medical history
Health dashboard overview
3.3 Doctor Module
Schedule management
Patient consultation records
3.4 Appointment System
Real-time scheduling and rescheduling
Emergency appointment prioritization
3.5 Emergency Services
Ambulance request handling
Emergency patient prioritization
3.6 Analytics and Reports
Patient admission trends

Financial summaries

4.Software Interface

The system interfaces with various external or internal software systems to enhance functionality:

Software	Purpose	Type
MySQL Database	Stores all hospital records	Internal Interface
JDBC (Java Database Connectivity)	Connects Java app to MySQL database	Internal
PDF/Excel Libraries	Exports reports and data	Internal Utility

? 5. Performance Requirements

This section defines how the MediSphere Hospital Management System should perform under various conditions. It outlines both static and dynamic performance expectations to ensure the system runs efficiently and meets real-time needs within the hospital environment.

5.1 Static Performance Requirements

These requirements define the system’s performance attributes that are not directly related to execution speed, but are still critical for overall system efficiency and reliability.

5.2 Dynamic Performance Requirements

These define how the system behaves during execution in real-time environments under normal and peak usage.

?? 6. Non-Functional Requirements

Security: Data encryption, secure authentication
Performance: Quick response times (<2 seconds for critical operations)
Scalability: Support for concurrent users
Reliability: High system uptime (99.9%)
Usability: Intuitive user interface
Maintainability: Modular codebase for easy updates