***Software Design Specification Document***

**For project ON**

**“Online Healthcare Management System”**

**By: Ashmit Gupta**

**Submitted To: Dr. Chandani Joshi**

**TABLE OF CONTENTS**

1. INTRODUCTION
   1. Purpose
   2. Scope
   3. Definitions, Acronyms and Abbreviations
   4. Overview
2. SYSTEM ARCHITECTURE DESIGN
   1. High-level Design Overview
   2. User Interface Issues
   3. Overview of Modules/Layers/Components
   4. Structure and Relationships
3. DATA DESIGN (ER-Diagram)
4. DETAILED DESCRIPTION OF COMPONENTS
   1. Data Flow Diagram (DFD)
   2. Activity Diagram
   3. Sequence Chart
5. USER INTERFACE DESIGN
6. TESTING ISSUES
7. APPENDICES

# Introduction

The **Online Healthcare Management System** is a web-based platform designed to transform and digitize the traditional healthcare services provided by clinics, hospitals, and medical professionals. With the rising demand for streamlined, patient-centric services, this system ensures that healthcare delivery becomes more accessible, efficient, and transparent. The platform enables patients to register, book appointments, consult doctors remotely, receive prescriptions, and maintain their medical history digitally. Doctors benefit from tools to manage appointments, access patient records, and prescribe medications securely. Administrators can monitor overall system activities, generate reports, and manage users.

* 1. **Purpose**

The purpose of this Software Requirements Specification (SRS) is to define the functionality, features, and constraints of the **Online Healthcare Management System (OHMS)**. This system is designed to serve as a comprehensive, user-friendly, and secure digital solution that addresses the operational needs of modern healthcare facilities, including hospitals, clinics, and telemedicine providers. This document establishes a mutual understanding between developers, project stakeholders, testers, and end-users regarding the system’s capabilities. It serves as a contractual foundation for development, ensuring all requirements are clearly identified, traceable, and testable throughout the software lifecycle.

* 1. **Scope**

The Online Healthcare Management System will provide:

1. Patient registration and profile management
2. Appointment scheduling and reminders
3. Doctor portal with medical record access
4. Prescription and diagnostic history management
5. Secure chat and consultation feature
6. Admin control for managing users, data, and system settings

## 1.3 Objectives and Benefits:

 Centralized patient data management

 Easy and flexible appointment scheduling

 Improved doctor-patient communication

 Reduced administrative workload

 Secure, encrypted, and role-based access

 24/7 system availability

 Scalable and adaptable for any healthcare facility

 Real-time notifications and reminders

**Definitions,Acronyms, Abbreviations**

**Definitions**

* **Patient** – End-user receiving healthcare services
* **Doctor** – Medical professional providing consultation and prescriptions
* **Admin** – System manager controlling users, roles, and settings

**Acronyms**

| **Term** | **Definition** |
| --- | --- |
| EMR | Electronic Medical Record |
| UI | User Interface |
| DBMS | Database Management System |
| OTP | One-Time Password |
| REST API | Representational State Transfer Application Programming Interface |
| JWT | JSON Web Token |
| ERD | Entity Relationship Diagram |

* 1. **Overview**

The Online Healthcare Management System is a comprehensive web-based solution aimed at streamlining healthcare services for patients, doctors, and administrators. It provides an integrated platform that enables users to manage appointments, access electronic medical records (EMR), communicate securely, and handle administrative tasks with ease. The system is designed to reduce manual workload, minimize errors, and enhance the efficiency of healthcare delivery. With features such as patient registration, doctor dashboards, real-time notifications, and data encryption, it ensures a smooth and secure healthcare experience. Accessible anytime from any internet-connected device, the system supports both small clinics and large hospitals, offering scalability

1. System Overview

2.1 System Architecture The OHMS follows a three-tier architecture:

* Presentation Layer: React.js-based front-end interface accessible via browsers.
* Business Logic Layer: Backend developed using Node.js or Python Flask.
* Data Layer: MySQL or MongoDB for structured storage of user and medical data.

2.2 Data Flow Diagrams / Block Diagrams DFD Level 0 shows user interaction with system modules like Registration, Appointment Booking, EMR Access, Prescription Management, and Notifications.

Block Diagram includes:

* Patient Portal
* Doctor Dashboard
* Admin Panel
* Notification Service
* Database Server

2.3 Module Descriptions

* Authentication: Manages user login/registration using JWT.
* Appointment Module: Schedules, reschedules, or cancels appointments.
* EMR Module: Stores patient health history, diagnosis, and treatment.
* Chat Module: Real-time secure messaging.
* Admin Module: Controls system configuration and user management.

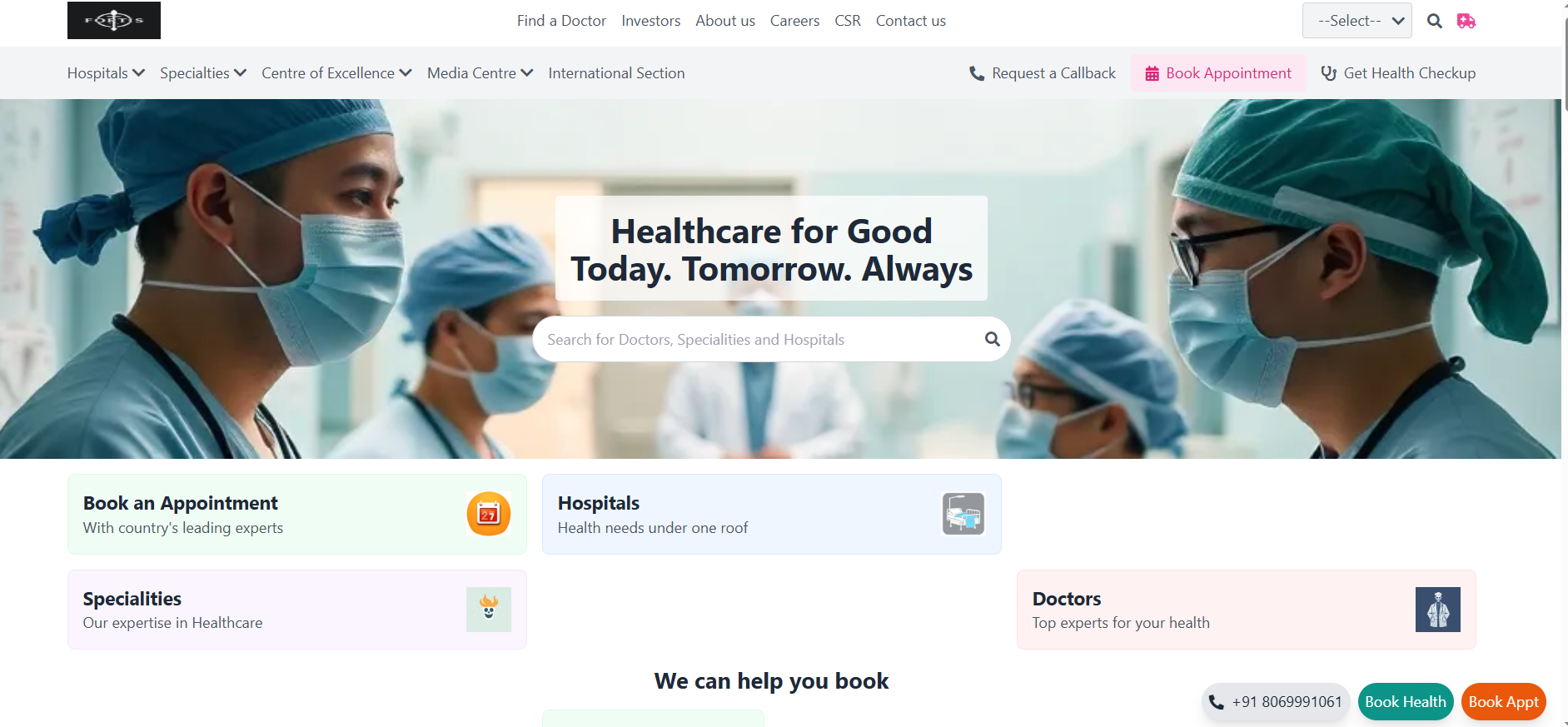
1. Detailed Design

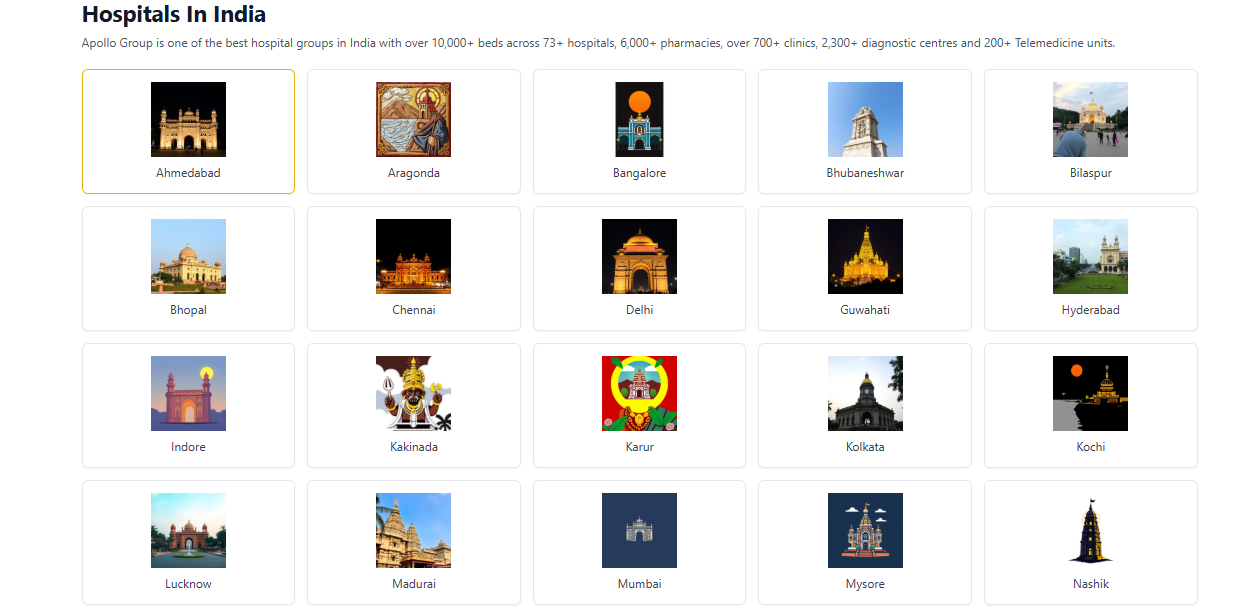
# 3) Data design (ER Diagram)

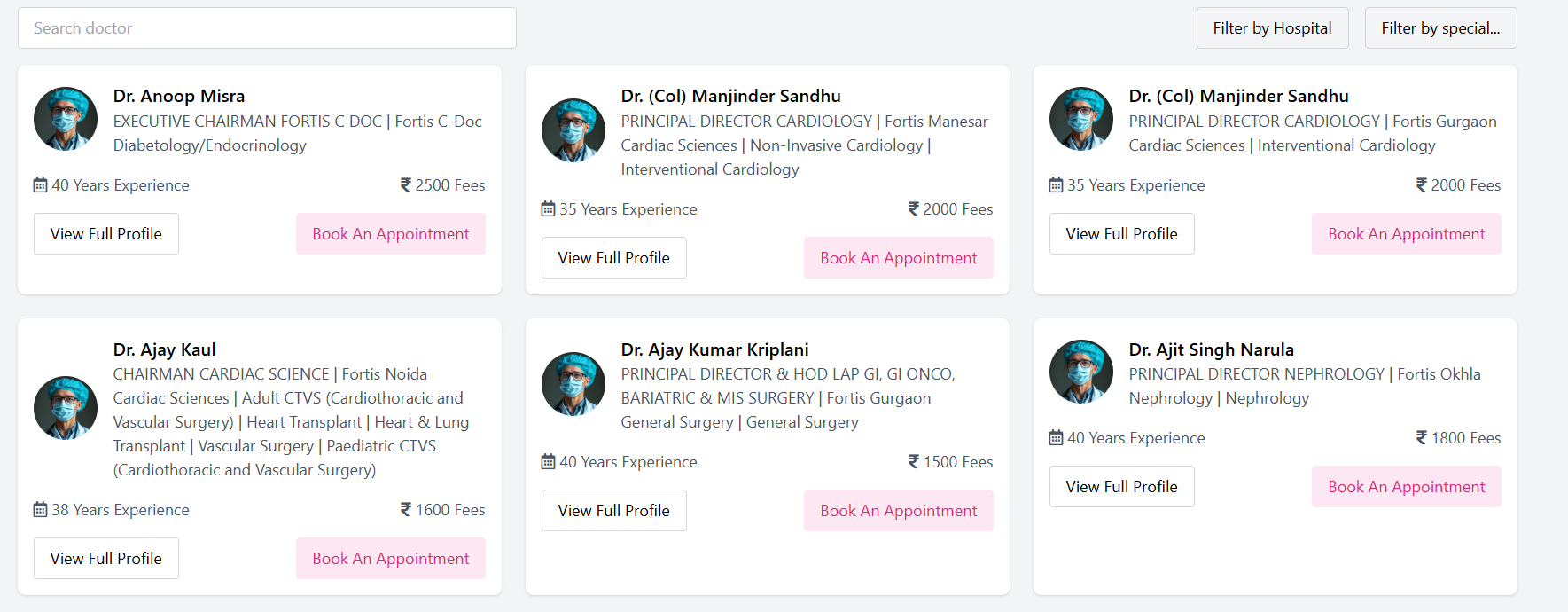
# 10 ER Diagram Examples for Hospital Management

# 

**User Interface**

****

****

****

3.1 Appointment Management Module

* Description: Enables patients to schedule and manage appointments.
* Inputs: Patient ID, preferred date/time, doctor selection.
* Outputs: Confirmation status, notification.
* Algorithms: IF doctor is available THEN Schedule appointment; Send confirmation to both; ELSE Suggest next available slot.
* Data Structures: appointment\_id, patient\_id, doctor\_id, date, time, status.

3.2 EMR Module

* Description: Manages storage and retrieval of medical records.
* Inputs: Doctor notes, lab reports, prescriptions.
* Outputs: Patient medical history view.
* Algorithms: On doctor update: Save diagnosis and treatment to patient record; Generate timestamp.
* Data Structures: emr\_id, patient\_id, doctor\_id, diagnosis, prescriptions, timestamp.

3.3 User Authentication Module

* Description: Secure login/logout and session handling.
* Inputs: Username/email, password.
* Outputs: JWT session token, user role-based access.
* Algorithm: Authenticate credentials; Generate JWT; Redirect to respective dashboard.

3.4 Chat/Consultation Module

* Description: Real-time chat between doctor and patient.
* Inputs: Sender ID, receiver ID, message.
* Outputs: Message delivery confirmation, notifications.
* Technology: WebSocket/Firebase.

1. Interface Design

4.1 User Interface Design

* Screens: Login, Registration, Dashboards, Appointment Booking, EMR View, Chat Window.
* Navigation: Sidebar and top-nav based routing.
* Wireframes: Provided separately via design tools (Figma).

4.2 Hardware Interface

* Clients: Android/iOS devices, desktops.
* Servers: Linux VPS, 8-core CPU, 16 GB RAM, SSD storage.

4.3 Software Interface

* Frontend: React.js/Angular.
* Backend: Node.js/Flask.
* APIs: REST API using Express/FastAPI.
* Database: MySQL/PostgreSQL/MongoDB.

4.4 Communication Interface

* HTTP/HTTPS for API calls.
* WebSocket for chat.
* Email/SMS via third-party APIs (SendGrid, Twilio).

1. Data Design

5.1 Data Models

* ER Diagram includes entities: Patient, Doctor, Admin, Appointment, EMR, Prescription.
* Relationships: One patient - many appointments; One doctor - many EMRs; Admin manages all users.

5.2 Database Design

* Tables: Users, Appointments, EMR, Prescriptions, Messages.
* Example (Appointments Table): Fields: id, patient\_id, doctor\_id, date\_time, status.

1. Security & Performance Considerations

* HTTPS encryption for data in transit.
* AES encryption for data at rest.
* JWT authentication.
* Load balancing and caching for performance.
* Auto-scaling for traffic handling.

1. Design Constraints

* Web-only access in initial phase.
* Requires stable internet.
* Responsive design mandatory.
* Integration with external APIs for notifications.

1. Appendix Glossary:

* EMR: Patient record
* JWT: Security token
* API: Interface for connecting software modules

References:

* IEEE 1016 standard
* React.js docs
* Node.js/Express docs

Assumptions:

* All users will have internet access.
* Doctors are verified before approval.
* Admin will manage system maintenance and upgrades.

.

# 8) Appendices

**Website:**https://exonum.com/health-care-syatem

* https:/[/www.dock.io/post/blockchain-verification](http://www.dock.io/post/blockchain-verification)

**Books/Tutorials:**

* Tutorial : https://www.youtube.com/watch?v=ls1Noh2b5Ms&t=0s