Software Requirements Specification (SRS)

Session 2023 - 2027

Submitted by:

Umme Aymen 2023-CS-112

Supervised by:

Dr. Amjad Farooq

Course:

Web Technologies

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Hospital Management System

Document Information

Field	Details
Project Name	Hospital Management System
Version	1.0
Date	January 6, 2026
Technology Stack	MERN (MongoDB, Express.js, React.js, Node.js)

Table of Contents

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document provides a comprehensive description of the Hospital Management System. It outlines the functional and non-functional requirements for the web-based application designed to manage hospital operations including patient appointments, doctor management, and departmental organization.

1.2 Scope

The Hospital Management System is a full-stack web application that enables:

- Patients to book appointments with doctors
- Doctors to manage their appointments and schedules
- Administrators to manage users, doctors, and departments
- Review and rating system for doctors

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
MERN	MongoDB, Express.js, React.js, Node.js
API	Application Programming Interface
JWT	JSON Web Token
CRUD	Create, Read, Update, Delete
SRS	Software Requirements Specification
UI	User Interface

1.4 Technology Stack

Layer	Technology
Frontend	React.js, React Router, Axios, React Toastify
Backend	Node.js, Express.js
Database	MongoDB (Atlas)
Authentication	JWT (JSON Web Tokens), bcryptjs
Deployment	Vercel (Frontend), Render (Backend)

2. Overall Description

2.1 Product Perspective

The Hospital Management System is a standalone web application that provides an integrated platform for managing hospital operations. It serves as a bridge between patients seeking medical care and healthcare providers offering their services.

2.2 Product Functions

- User Authentication & Authorization
- Patient Appointment Booking
- Doctor Schedule Management
- Department Management
- Review & Rating System
- Admin Dashboard

2.3 User Classes and Characteristics

2.3.1 Patient

- Can register and login
- Can view departments and doctors
- Can book, view, and cancel appointments
- Can submit reviews for completed appointments
- Can update profile information

2.3.2 Doctor

- Can login with doctor credentials
- Can view assigned appointments
- Can update appointment status (confirm, complete, cancel)
- Can add prescriptions to completed appointments
- Can manage availability and schedule

2.3.3 Administrator

- Full system access

- Can manage all users (CRUD operations)
- Can manage departments (CRUD operations)
- Can manage doctors (CRUD operations)
- Can view all appointments
- Can access system statistics

2.4 Operating Environment

- **Client Side**: Modern web browsers (Chrome, Firefox, Safari, Edge)
- **Server Side**: Node.js runtime environment
- **Database**: MongoDB Atlas (Cloud)
- **Hosting**: Vercel (Frontend), Render (Backend)

2.5 Design and Implementation Constraints

- Must use MERN stack technologies
- Must implement RESTful API architecture
- Must use JWT for authentication
- Must be responsive for mobile and desktop devices

3. System Features

3.1 User Authentication System

3.1.1 Description

Secure user registration and login system with role-based access control.

3.1.2 Functional Requirements

ID	Requirement	Priority
FR-1.1	System shall allow new users to register with name, email, password, and phone	High
FR-1.2	System shall authenticate users using email and password	High
FR-1.3	System shall generate JWT token upon successful login	High
FR-1.4	System shall support three user roles: Patient, Doctor, Admin	High

FR-1.5	System shall hash passwords using bcrypt before storing	High
FR-1.6	System shall allow users to update their profile information	Medium
FR-1.7	System shall allow users to change their password	Medium

3.2 Department Management

3.2.1 Description

Management of hospital departments with associated information.

3.2.2 Functional Requirements

ID	**Requirement**	**Priority**
FR-2.1	System shall display all active departments	High
FR-2.2	System shall allow admin to create new departments	High
FR-2.3	System shall allow admin to update department details	High
FR-2.4	System shall allow admin to delete/deactivate departments	Medium
FR-2.5	System shall support department search functionality	Medium

3.3 Doctor Management

3.3.1 Description

Complete doctor profile and schedule management system.

3.3.2 Functional Requirements

ID	**Requirement**	**Priority**
FR-3.1	System shall display list of all available doctors	High
FR-3.2	System shall show doctor details (specialization, qualification, experience, fee)	High
FR-3.3	System shall filter doctors by department	High
FR-3.4	System shall allow admin to add new doctors	High
FR-3.5	System shall allow admin to update doctor information	High

FR-3.6	System shall allow doctors to set available days and time slots	High
FR-3.7	System shall display doctor ratings and reviews	Medium
FR-3.8	System shall allow searching doctors by name	Medium

3.4 Appointment Management

3.4.1 Description

Complete appointment booking and management system.

3.4.2 Functional Requirements

ID	**Requirement**	**Priority**
FR-4.1	System shall allow patients to book appointments with doctors	High
FR-4.2	System shall show available time slots based on doctor's schedule	High
FR-4.3	System shall prevent double booking of same time slot	High
FR-4.4	System shall allow patients to view their appointments	High
FR-4.5	System shall allow patients to cancel pending/confirmed appointments	High
FR-4.6	System shall allow doctors to confirm appointments	High
FR-4.7	System shall allow doctors to mark appointments as completed	High
FR-4.8	System shall allow doctors to add prescriptions	Medium
FR-4.9	System shall support appointment types (consultation, follow-up, emergency)	Medium
FR-4.10	System shall display appointment statistics on dashboard	Medium

3.5 Review and Rating System

3.5.1 Description

Patient feedback system for completed appointments.

3.5.2 Functional Requirements

ID	Requirement	Priority
FR-5.1	System shall allow patients to rate doctors (1-5 stars)	Medium
FR-5.2	System shall allow patients to write review comments	Medium
FR-5.3	System shall only allow reviews for completed appointments	Medium
FR-5.4	System shall prevent duplicate reviews for same appointment	Medium
FR-5.5	System shall calculate and display average doctor rating	Medium
FR-5.6	System shall support anonymous reviews option	Low

3.6 Admin Dashboard

3.6.1 Description

Comprehensive administrative control panel.

3.6.2 Functional Requirements

ID	Requirement	Priority
FR-6.1	System shall display system statistics (total users, doctors, appointments)	High
FR-6.2	System shall allow admin to manage all users	High
FR-6.3	System shall allow admin to activate/deactivate user accounts	High
FR-6.4	System shall provide user role management	High
FR-6.5	System shall allow admin to view all appointments	Medium

4. External Interface Requirements

4.1 User Interfaces

4.1.1 General Requirements

- Responsive design for desktop and mobile devices
- Clean and intuitive navigation
- Consistent color scheme and branding

- Loading indicators for async operations
- Toast notifications for user feedback

4.1.2 Pages/Screens

Page	**Description**	**Access**
Home	Landing page with overview and quick links	Public
Login	User authentication form	Public
Register	New user registration form	Public
Departments	List of all hospital departments	Public
Doctors	List of all doctors with filters	Public
Doctor Detail	Individual doctor profile with booking option	Public
Book Appointment	Appointment booking form	Patient
My Appointments	Patient's appointment list	Patient
Patient Dashboard	Patient overview and stats	Patient
Doctor Dashboard	Doctor overview and appointments	Doctor
Doctor Appointments	Doctor's appointment management	Doctor
Admin Dashboard	Admin overview and statistics	Admin
Manage Users	User CRUD operations	Admin
Manage Doctors	Doctor CRUD operations	Admin
Manage Departments	Department CRUD operations	Admin
Profile	User profile management	Authenticated

4.2 Hardware Interfaces

- No specific hardware interfaces required
- Standard web-enabled devices with internet connectivity

4.3 Software Interfaces

Interface	**Description**
MongoDB Atlas	Cloud database service for data storage
Vercel	Frontend hosting and deployment
Render	Backend hosting and deployment

4.4 Communication Interfaces

- HTTPS protocol for secure communication
- RESTful API architecture
- JSON data format for API requests/responses

5. Non-Functional Requirements

5.1 Performance Requirements

ID	**Requirement**
NFR-1.1	Page load time should be less than 3 seconds
NFR-1.2	API response time should be less than 500ms
NFR-1.3	System should handle 100 concurrent users

5.2 Security Requirements

ID	**Requirement**
NFR-2.1	Passwords must be hashed using bcrypt (min 10 salt rounds)
NFR-2.2	Authentication tokens must expire after 30 days
NFR-2.3	API endpoints must be protected with JWT authentication
NFR-2.4	Role-based access control must be enforced
NFR-2.5	Sensitive data must not be logged or exposed in errors

5.3 Reliability Requirements

ID	Requirement
NFR-3.1	System uptime should be 99%
NFR-3.2	Database backups should be automated
NFR-3.3	Error handling must prevent system crashes

5.4 Usability Requirements

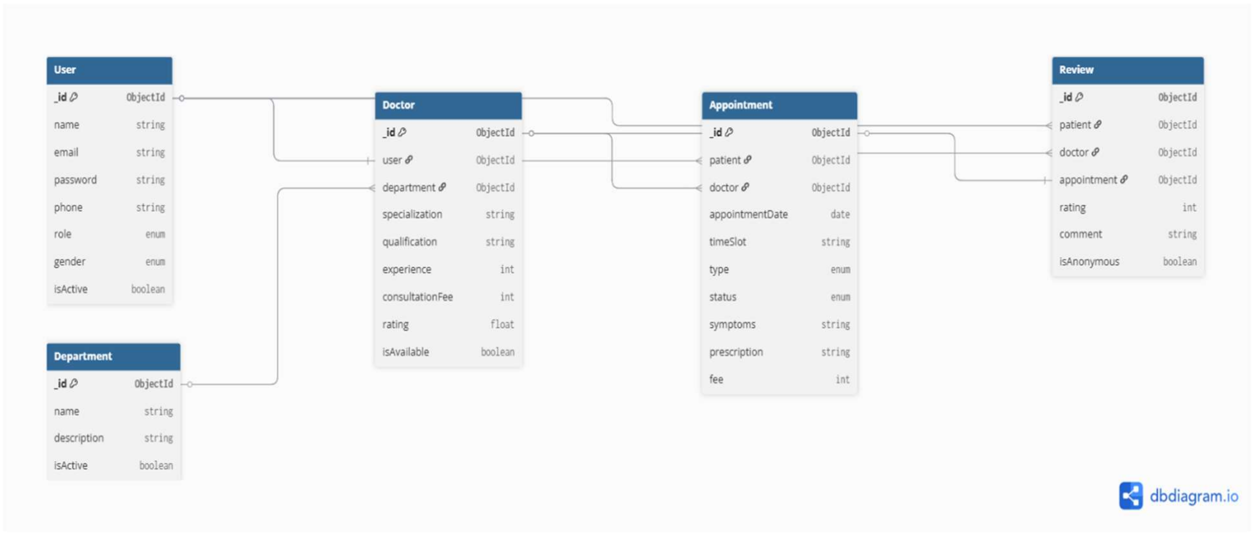
ID	Requirement
NFR-4.1	UI must be responsive across devices
NFR-4.2	Forms must have proper validation messages
NFR-4.3	Navigation must be intuitive and consistent

5.5 Scalability Requirements

ID	Requirement
NFR-5.1	Database should support horizontal scaling
NFR-5.2	Application should be containerization-ready

6. Database Design

6.1 Entity Relationship Diagram



6.2 Data Models

6.2.1 User Model

```
{
  name: String (required, max 50),
  email: String (required, unique),
  password: String (required, min 6, hashed),
  phone: String (max 20),
  role: Enum ['patient', 'doctor', 'admin'],
  gender: Enum ['male', 'female', 'other'],
  dateOfBirth: Date,
  address: {
    street: String,
    city: String,
    state: String,
    zipCode: String,
    country: String
  },
  profileImage: String,
  isActive: Boolean (default: true),
  timestamps: true
}
```

6.2.2 Doctor Model

```
{
  user: ObjectId (ref: User, required),
  department: ObjectId (ref: Department, required),
  specialization: String (required, max 100),
  qualification: String (required, max 200),
  experience: Number (required, min 0),
  consultationFee: Number (required, min 0),
  bio: String (max 1000),
  availableDays: [Enum weekdays],
  availableSlots: [{
    startTime: String,
    endTime: String
  }],
  rating: Number (1-5, default 4.5),
  totalReviews: Number (default 0),
  isAvailable: Boolean (default true),
  timestamps: true
}
```

6.2.3 Department Model

```
{
  name: String (required, unique, max 100),
  description: String (required, max 500),
  image: String (default: 'default-department.png'),
  isActive: Boolean (default: true),
  timestamps: true
}
```

6.2.4 Appointment Model

```
{
  patient: ObjectId (ref: User, required),
  doctor: ObjectId (ref: Doctor, required),
  appointmentDate: Date (required),
  timeSlot: {
    startTime: String (required),
    endTime: String (required)
  },
  type: Enum ['consultation', 'follow-up', 'emergency'],
  status: Enum ['pending', 'confirmed', 'completed', 'cancelled', 'no-show'],
  symptoms: String (max 500),
  prescription: String (max 2000),
  notes: String (max 1000),
  fee: Number (required),
  cancellationReason: String,
  timestamps: true
}
```

6.2.5 Review Model

```
{
  patient: ObjectId (ref: User, required),
  doctor: ObjectId (ref: Doctor, required),
  appointment: ObjectId (ref: Appointment, required),
  rating: Number (1-5, required),
  comment: String (max 500),
  isAnonymous: Boolean (default: false),
  timestamps: true
}
```

7. API Specifications

7.1 Base URL

- **Development**: http://localhost:5000/api
- **Production**: https://wt-mern-application-hospital-management.onrender.com/api

7.2 Authentication Endpoints

Method	**Endpoint**	**Description**	**Access**
POST	/auth/register	Register new user	Public
POST	/auth/login	Login user	Public
GET	/auth/me	Get current user	Private
PUT	/auth/updatedetails	Update user details	Private
PUT	/auth/updatepassword	Update password	Private

7.3 User Endpoints (Admin)

Method	**Endpoint**	**Description**	**Access**
GET	/users	Get all users	Admin
GET	/users/:id	Get user by ID	Admin
POST	/users	Create user	Admin
PUT	/users/:id	Update user	Admin
DELETE	/users/:id	Delete user	Admin
PUT	/users/:id/toggle-status	Toggle user status	Admin

7.4 Department Endpoints

Method	**Endpoint**	**Description**	**Access**
GET	/departments	Get all departments	Public
GET	/departments/:id	Get department by ID	Public
POST	/departments	Create department	Admin
PUT	/departments/:id	Update department	Admin
DELETE	/departments/:id	Delete department	Admin

7.5 Doctor Endpoints

Method	Endpoint	Description	Access
GET	/doctors	Get all doctors	Public
GET	/doctors/:id	Get doctor by ID	Public
GET	/doctors/department/:id	Get doctors by department	Public
GET	/doctors/user/:userId	Get doctor by user ID	Private
POST	/doctors	Create doctor	Admin
PUT	/doctors/:id	Update doctor	Admin/Doctor
DELETE	/doctors/:id	Delete doctor	Admin
PUT	/doctors/:id/availability	Update availability	Doctor

7.6 Appointment Endpoints

Method	Endpoint	Description	Access
GET	/appointments	Get appointments	Private
GET	/appointments/:id	Get appointment by ID	Private
POST	/appointments	Create appointment	Patient
PUT	/appointments/:id	Update appointment	Doctor
PUT	/appointments/:id/status	Update status	Doctor
PUT	/appointments/:id/cancel	Cancel appointment	Patient/Doctor
GET	/appointments/available-slots/:doctorId/:date	Get available slots	Private
GET	/appointments/stats	Get statistics	Admin

7.7 Review Endpoints

Method	Endpoint	Description	Access
GET	/reviews/doctor/:doctorId	Get doctor reviews	Public
POST	/reviews	Create review	Patient
PUT	/reviews/:id	Update review	Patient
DELETE	/reviews/:id	Delete review	Patient/Admin

7.8 API Response Format

Success Response

```
{
  "success": true,
  "data": { },
  "count": 10,
  "total": 100,
  "totalPages": 10,
  "currentPage": 1
}
```

Error Response

```
{
  "success": false,
  "message": "Error description",
  "error": "Detailed error (development only)"
}
```

8. Appendix

8.1 Test Accounts

Role	**Email**	**Password**
Admin	admin@healthcare.com	admin123
Doctor	dr.ahmed@healthcare.com	doctor123
Patient	patient@healthcare.com	patient123

8.2 Deployment URLs

Service	**URL**
Frontend (Vercel)	https://wt-mern-application-hospital-management-system.vercel.app
Backend (Render)	https://wt-mern-application-hospital-management.onrender.com
GitHub Repository	https://github.com/ummeaymen499/WT-Mern-Application-Hospital-Management-System

8.3 Environment Variables

Backend (.env)

```
NODE_ENV=production

PORT=5000

MONGO_URI=<mongodb_connection_string>

JWT_SECRET=<jwt_secret_key>

JWT_EXPIRE=30d

CLIENT_URL=<frontend_url>
```

Frontend (.env)

```
REACT_APP_API_URL=<backend_api_url>
```