**Software Requirements Specification (SRS)**

**1. Introduction**

**1.1 Purpose**

This document specifies the requirements for the Emergency Health Center Locator Systems, a platform for users to discover, interact with, and manage information about hospitals and health centers, including user authentication, notifications, and health resource management.

**1.2 Scope**

Emergency Health Center Locator System is a mobile application with a Django REST Framework backend and a React Native frontend. It allows users to:

* Register and log in
* Search for hospitals/health centers by location, specialty, and services
* View detailed information about hospitals
* Receive and manage notifications
* Upload images (e.g., hospital photos)
* Access interactive API documentation

**1.3 Definitions, Acronyms, and Abbreviations**

* **API**: Application Programming Interface
* **DRF**: Django REST Framework
* **JWT**: JSON Web Token
* **SRS**: Software Requirements Specification

**2. Overall Description**

**2.1 Product Perspective**

Emergency Health Center Locator System consists of:

* A Django backend providing RESTful APIs
* A React Native frontend for mobile/web
* API documentation via drf-spectacular (Swagger UI, ReDoc)

**2.2 Product Functions**

* User registration, login, and authentication (JWT)
* Hospital/health center search and listing
* Detailed hospital/health center profiles
* Favorites management
* Notification system (including emergency notifications)
* Image upload for hospitals
* Health check and system status endpoints
* API documentation (Swagger, ReDoc, OpenAPI schema)

**2.3 User Classes and Characteristics**

* **General Users**: Can search, and view hospitals, manage notifications.
* **Admins/Staff**: (If implemented) Can manage hospital data and notifications.

**2.4 Operating Environment**

* Backend: Python 3.x, Django 4.x, DRF, SQLite (dev).
* Frontend: React Native (Expo), web/mobile browsers
* API Docs: Accessible via browser

**2.5 Design and Implementation Constraints**

* Must use Django REST Framework for backend APIs
* JWT for authentication
* drf-spectacular for API documentation

**3. Specific Requirements**

**3.1 Functional Requirements**

**3.1.1 User Management**

* Users can register with email and password.
* Users can log in and receive JWT tokens.

**3.1.2 Hospital/Health Center Management**

* Users can search for hospitals by city, specialty, and services.
* Users can view a list of hospitals and detailed profiles.

**3.1.3 Notifications**

* Users receive notifications for viewed hospitals and system messages.
* Users can mark notifications as read and delete them.

**3.1.4 District/Location Features**

* Users can retrieve a list of available districts/cities where hospitals are located.
* The system provides geolocation support for users and hospitals (latitude/longitude).
* Users can view hospitals sorted by proximity to their current location.

**3.1.5 Filtering and Search**

* Users can filter hospitals by:
* City/district
* Services (emergency, ambulance, pharmacy, lab)
* Conditions treated
* Specialties
* Search supports partial matches and is case-insensitive.

**3.1.6 API Documentation**

API documentation is available at /api/docs/ (Swagger UI), /api/redoc/ (ReDoc), and /api/schema/ (OpenAPI JSON).

**3.1.7 Health Check**

System health and status endpoint available at /api/health/.

**3.1.8 Error Handling & Feedback**

All API endpoints return clear error messages and status codes.

The frontend displays user-friendly error and success messages.

**3.1.9 Security**

Passwords are securely hashed and never stored in plain text.

Sensitive endpoints require authentication and proper permissions.

CORS is configured to allow only trusted origins in production.

**3.1.10 Network/Server Status**

The frontend can display server/network status to the user.

Health check endpoints provide detailed system and database status.

**3.2 Non-Functional Requirements**

**Performance**: API should respond within 2 seconds for standard queries.

**Security**: All sensitive endpoints require JWT authentication.

**Reliability**: System should handle errors gracefully and log exceptions.

**Usability**: API documentation must be clear and interactive.

**4. External Interface Requirements**

**4.1 API Endpoints**

* /api/register/, /api/login/, /api/logout/
* /api/hospitals/, /api/hospitals/{id}/
* /api/favourites/
* /api/notifications/
* /api/health/
* /api/upload/
* /api/docs/, /api/redoc/, /api/schema/

**4.2 User Interface**

Mobile built with React Native, providing forms and views for all major features.

**5. Other Requirements**

The system should be easily deployable on standard cloud platforms.

Documentation should be kept up to date as endpoints or features change.

The backend includes automated tests for major endpoints and business logic.

The frontend includes basic UI and integration tests.

The frontend should follow accessibility best practices (labels, contrast, keyboard navigation).