**AarogyaLink**

This document provides a comprehensive overview of the "AarogyaLink" project, a full-stack web application designed to connect users with healthcare services.

**1. Project Overview**

**AarogyaLink** is a web platform designed to streamline and simplify access to healthcare information. The application enables users to analyze symptoms and find suitable hospitals based on specific criteria, while providing a dedicated panel for hospital administrators to manage their facility's data.

The project is built as a complete, end-to-end application with a clear roadmap for future enhancements.

**2. Current and Future Features**

This section details the features that are currently implemented and a list of enhancements planned for future development.

**Implemented Features (Version 1.0)**

These are the functionalities that are fully operational in the current version of the application:

* **Symptom-Based Search:** A user-friendly form allows users to input a list of symptoms.
* **Disease Analysis:** The backend processes the symptoms using a comprehensive mapping logic to provide a probable disease diagnosis.
* **Hospital Filtering:** Based on the diagnosed disease and user criteria (city, healthcare scheme, insurance), the application filters and displays a list of matching hospitals.
* **Hospital Admin Panel:** A separate page provides a form for hospital administrators to add new hospital entries to the database.
* **Hospital Data Display:** The admin panel also displays a table of all existing hospitals.
* **In-Memory Database:** The application uses a static, in-memory array (dummyHospitals.js) to store and manage hospital data, allowing for quick setup and testing.

**Planned Features (Future Development)**

The following features are planned to be added in future versions to make the application more robust and ready for production:

* **Persistent Database:** Replace the in-memory array with a real **MongoDB** database to store data permanently.
* **User Authentication:** Implement a secure login and authentication system for both users and hospital administrators.
* **Advanced Search & Filtering:** Add more nuanced filtering options (e.g., hospital ratings, availability of doctors) and an improved search algorithm.
* **Enhanced User Interface:** Upgrade the front-end design for a more modern, intuitive, and mobile-responsive user experience.
* **Dynamic Data:** Allow administrators to edit and delete existing hospital entries, not just add new ones.
* **Report Upload:** Implement actual file handling for user reports instead of the current mock file upload.

**3. Technologies Used**

* **Frontend:**
  + **React.js:** A JavaScript library for building the user interface.
  + **Axios:** A promise-based HTTP client for making API calls to the backend.
  + **React Router DOM:** For handling client-side navigation and routing between pages.
* **Backend:**
  + **Node.js:** A JavaScript runtime for running the backend server.
  + **Express.js:** A web framework for Node.js, used to build the REST API.
  + **CORS:** A Node.js middleware for enabling Cross-Origin Resource Sharing.

**4. Application Structure**

The project is organized into two main directories:

* **client/**: Contains the entire React frontend code.
* **server/**: Contains the Node.js backend code.
  + data/: Stores the dummyHospitals.js file, which acts as the in-memory database.
  + routes/: Contains the api.js file, which defines all the backend API endpoints.
  + server.js: The main entry point for the Express server.

**5. Backend API Endpoints**

The Express server exposes a set of RESTful API endpoints:

| Method | Endpoint | Description |
| --- | --- | --- |
| POST | /api/analyzeSymptoms | Takes a comma-separated string of symptoms and returns a probable disease. |
| GET | /api/hospitals | Takes query parameters (disease, city, scheme, insurance) and returns a filtered list of hospitals. |
| POST | /api/addHospital | Takes a new hospital's data and adds it to the in-memory database. |
| GET | /api/allHospitals | Returns all hospital entries, used for the admin page display. |

Export to Sheets