**AarogyaLekha**

**Project Planning :-**

Front end - React

Back end - Nodejs

Desktop Frameworks - Electron

Database - MySQL

Communication between front-end - axios

Your Health, Your History, Your Control.

AarogyaLekha is a comprehensive medical records platform designed to securely store and manage patient health information. This platform provides a centralized repository for critical medical data, including surgical history, chronic conditions, and treatment plans. By streamlining access to patient records, AarogyaLekha aims to improve healthcare efficiency and patient care.

Hardik & Sujal : Logo and Synopsis

Omkar & Dushant : Presentation and Scripting

Features:

1. Lab Test results section
2. Current Medications
3. **Medical Imaging Integration :** Allow healthcare providers to view and analyze medical images, such as X-rays and MRIs.

Problems with current system :

1. Difficulty in accessing patient data across different hospitals.
2. More efficient and secure than physical data storing.
3. Lack of real-time access to patient information

Key Features :

1. **Risk Stratification:** Identify high-risk patients and prioritize their care.

Future scope :

1. AI powered predicitve Analytics
2. AarogyaLekha for the people

Page 1-Hospital Login

1. Login

* ID
* Password

1. Sign up

* Name of hospital
* State , City
* ID
* 2 times password with eye feature

Note-Login should be done after Sign up

Page 2-Home Page

1. Logo and AarogyaLekha
2. Contact
3. Theme(Light/Dark) preferably Sun Moon Icon
4. About Us
5. Patient Login(Existing)

* Patient ID
* Submit

1. Patient SignUp(New)

* Photo , Name, Identity
* Unique ID generation
* Password
* Submit button

Page 3 - Patient Data

1)Identity

* Photo
* Name
* Age
* Address
* 2 Phone Numbers
* Blood Group

2)Medical Condition

3)Operation History

4)Current Medication

5)Lab Reports

Note-An “EDIT” button on each and every tab to update data

Also a “PRINT” button to print a pdf of this information

**Project Execution:-**

****1. Set Up Your Project:****

* Create a folder for your website files (e.g., "my\_website").

****2. Build the Basic HTML Structure:****

* Create an index.html file with the fundamental HTML tags: <html>, <head>, and <body>.
* Include a script.js file to handle the database interaction.
* Add a div element where you want the fetched data to be displayed (e.g., <div id="data-container"></div>).

****3. Choose a Server-Side Technology (not HTML/JS):****

* You'll need a server-side language like Node.js (with Express), Python (with Flask or Django), or Java (with Spring Boot) to interact with the database.

****4. Create a Server-Side Script:****

* This script will:
  + Connect to your MySQL database.
  + Fetch data from the database using a query.
  + Provide an API endpoint for your HTML page to access the data.

****5. Create a MySQL Database and Table:****

* Use a MySQL client to set up a database and a table to store your website's data.

****6. Run the Server-Side Script:****

* Start the server script using the appropriate command for your chosen technology (e.g., node server.js for Node.js).

****7. Access Your Website:****

* Open http://localhost:3000/ (or the relevant port) in your web browser to see the website.

****Key Points to Remember:****

* Security is crucial. Don't store database credentials directly in your code.
* Implement error handling on both the server and client sides.
* Sanitize user input to prevent vulnerabilities.
* Consider database optimization for larger datasets.

This simplified explanation provides a roadmap for connecting your website to a database. You'll need to delve deeper into the specific server-side technology you choose to write the necessary code.