Technical Career Education Private Limited

5th floor, Sahyadri Campus, Adyar, Mangalore 575007

Full Stack Development Skill Lab Course

PROJECT REPORT

2023 - 24

Project Title: Electronic Medical Record

Submitted by:

Vignesh D Jogi	4SF22CI117
Tejas	4SF22CI107
Sharath	4SF22CI088
Bharath Rai	4SF22CI023

Institution:

Sahyadri College of Engineering and Management

Adyar Mangalore 575007

CONTENTS

Project Overview

Project Overview

Problem Statement	Develop an Electronic Medical Record (EMR) System to streamline patient data management, enhance data accessibility, and improve the efficiency of healthcare services.		
Solution Proposed *(video Link)*	https://drive.google.com/file/d/158XYcr_TkGLRF1b-z0ENffBiIp89iyFc/view?usp=sharing		
Link to the final Challenge presentation	https://drive.google.com/file/d/1tAnifWGWR93XDLOl4tEzRDa9Fxuy1dU2/view?usp=sharing		
Link to photos/ videos drive	https://drive.google.com/file/d/158XYcr_TkGLRF1b-z0ENffBiIp89iyFc/view?usp=sharing		
Class/ Section	4A		
Team Name	Vignesh		
Team Members	Name	USN	Github Link
	Vignesh D Jogi	4SF22CI117	https://github.com/Vo-id00/Medical-appointment.git
	Tejas	4SF22CI107	https://github.com/Phiserte/assignment.git
	Sharath	4SF22CI088	https://github.com/SHARATHPOOJAR/skilllab-3/upload
	Bharath Rai	4SF22CI023	https://github.com/BharathRai

1. Introduction

- An Electronic Medical Record (EMR) system is a digital version of a patient's paper chart in a healthcare provider's office. It contains the medical and treatment history of patients in one practice, offering a comprehensive view of patient care. EMRs facilitate the management of patient information and streamline the workflow of healthcare providers by enabling quick access to medical records, reducing paperwork, and minimizing errors. These systems often include features such as medical history, diagnosis, medications, treatment plans, immunization dates, allergies, radiology images, and laboratory test results. EMRs enhance the quality of care by supporting evidence-based decision-making, improving patient safety, and enabling better coordination among healthcare professionals. They are a crucial component in the transition towards more efficient and effective healthcare delivery.

2. Problem Statement

- Develop an Electronic Medical Record (EMR) System to streamline patient data management, enhance data accessibility, and improve the efficiency of healthcare services.
- In many regions, the current manual and disparate methods of managing patient records lead to inefficiencies, data inaccuracies, and hindered patient care. An integrated EMR system will provide a centralized platform for managing patient information, ensuring data accuracy, and facilitating seamless access for authorized users.
- **Features and Requirements:**

  Doctor:

  1. Full access to all features and functionalities.
  2. Manage patient data.
  3. Generate and analyze reports.

  Admin:

  1. Access to user management functionalities.
  2. View dashboard and notifications.
  3. Manage system settings and permissions.

  Patient:

  1. Basic user functions such as viewing personal medical records.
  2. View dashboard and notifications.

  User Management:

  1. Secure login mechanisms for various user roles (e.g., admin, doctor, patient).
  2. Role-based access control to ensure data privacy and security.

3. Multi-factor authentication to enhance security.

Data Management:

1. Efficient handling and storage of patient records, medical history, and treatment plans.
2. Ensure data integrity, accuracy, and confidentiality.
3. Export and import data functionalities.

Dashboard:

1. Intuitive dashboard displaying key metrics and patient information.
2. Personalised views for different roles (doctors, admins, patients).

3. Solution

3.1 System requirements for the project

To create an Electronic Medical Record (EMR) system using React.js, you'll need a set of tools and technologies to build, manage, and deploy the application. Here are some essential tools and technologies:

1. Development Environment

- Node.js: A JavaScript runtime environment that allows you to run JavaScript on the server side and use npm (Node Package Manager) to install packages.
- npm or Yarn: Package managers to manage your project's dependencies.

2. Frontend Development

- React.js: The main JavaScript library for building the user interface.
- React Router: For handling navigation and routing within the application.
- Redux or Context API: For state management, especially if the application has complex state requirements.
- Axios or Fetch API: For making HTTP requests to interact with APIs.
- Material-UI, Ant Design, or Bootstrap: Component libraries for building a responsive and visually appealing UI.
- Form Libraries: Such as Formik or React Hook Form for managing form state and validation.

3. Backend Development

- Node.js with Express: For building the backend API that will handle data storage, retrieval, and processing.

- Database: Such as MongoDB, PostgreSQL, or MySQL for storing medical records and other data.

3.2 Flowchart of the project

1. Requirement Gathering and Planning

- **Identify Requirements**: Understand the needs of healthcare providers, including data entry, retrieval, and reporting.
- **Compliance and Security**: Ensure the system meets legal requirements, such as HIPAA, and includes security measures to protect patient data.
- **Design the System Architecture**: Define the overall structure, including frontend, backend, and database components.

2. Design Phase

- **UI/UX Design**: Create wireframes and mockups for the user interface, focusing on ease of use and accessibility.
- **Database Design**: Plan the database schema to store patient records, including tables for patients, appointments, prescriptions, etc.
- **API Design**: Design RESTful APIs for communication between the frontend and backend, specifying endpoints for data operations like create, read, update, and delete (CRUD).

3. Frontend Development

- **Set Up Development Environment**: Initialize the project using React and set up necessary tools like Webpack, Babel, and linting tools.
- **Build UI Components**: Develop reusable components for the interface, such as forms for patient data entry, tables for displaying information, and dashboards for analytics.
- **State Management**: Implement state management using Redux or Context API to handle data flow and application state.
- **Routing**: Use React Router to manage navigation between different pages or views in the application.

4. Backend Development

- **Set Up Backend Environment**: Set up Node.js and Express, and configure the server.
- **Database Integration**: Connect the backend to a database (e.g., MongoDB, PostgreSQL) and implement models for different entities (patients, doctors, records).
- **Implement API Endpoints**: Develop RESTful endpoints to handle requests from the frontend for CRUD operations.
- **Authentication and Authorization**: Implement user authentication and authorization to secure sensitive data and ensure only authorized personnel can access specific information.

5. Integration and Testing

- **Integrate Frontend and Backend**: Ensure seamless communication between the frontend and backend through API calls.
- **Testing**: Conduct unit testing, integration testing, and end-to-end testing to ensure the application functions correctly and securely.
- **User Acceptance Testing (UAT)**: Involve healthcare professionals in testing the system to gather feedback and make necessary adjustments.

3.3 Frontend

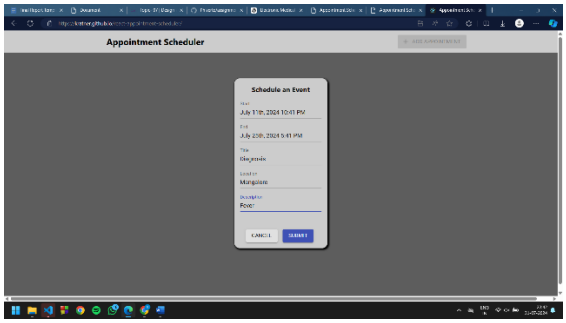
```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/css/bootstrap.min.css" integrity="sha384-
KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+d
N9+nJOZ" crossorigin="anonymous" />
   <link rel="stylesheet" href="style.css" />
   <title>Document</title>
</head>
```

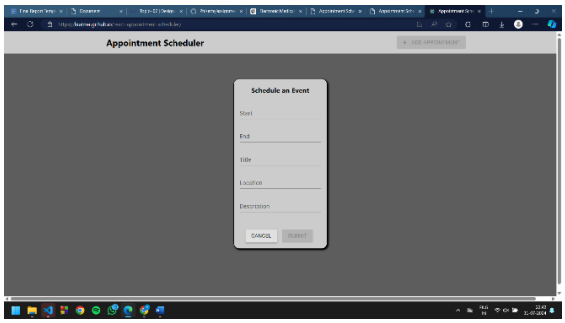
3.4 Backend

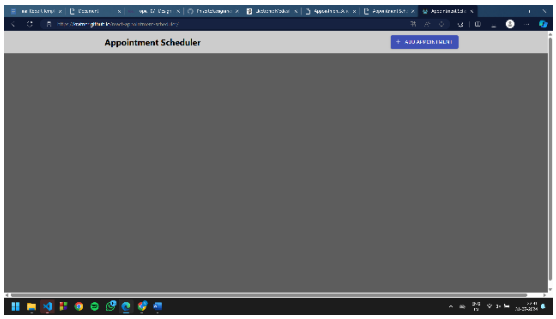
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />

    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

    <title>Appointment Scheduler</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

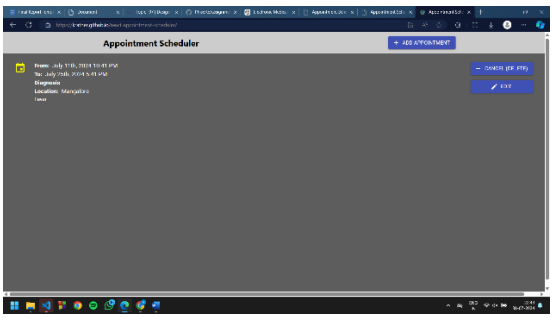
4. Conclusion/Outcome

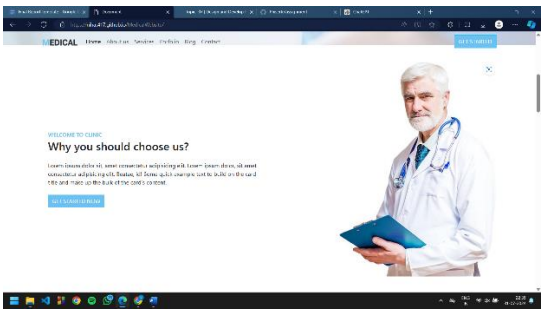
The proposed EMR system solution uses React.js to create a structured and scalable frontend. It features reusable UI components, centralized state management with Redux, and efficient data handling with Axios and Formik. Material-UI ensures a modern and responsive design. The system is designed for seamless integration with backend security measures and is easily deployable, providing a user-friendly interface that supports scalability and compliance with healthcare regulations.

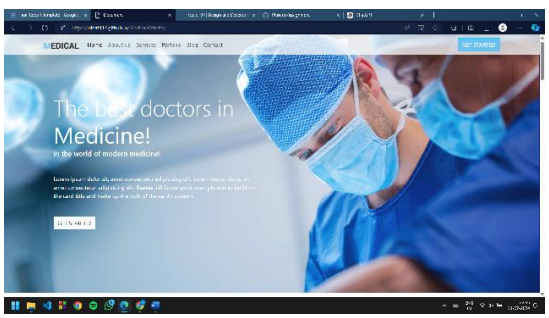


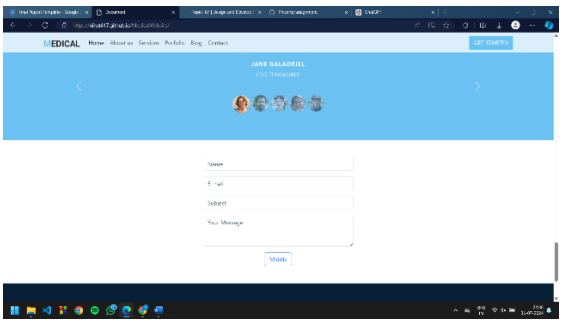












5. References

- Bootstrap
- ChatGPT
- Google
- W3Schools
- https://www.apollotelehealth.com/benefits-of-electronic-medical-
- records
- https://www.aufaitux.com/blog/healthcare
- https://digitalhealth.folio3.com/blog/what-is-emr-system/