**SYNOPSIS**

**Report on**

**Health Record Management**

**by**

Ritika Verma : 2300290140146

POOJA : 2300290140117

**Session:2024-2025 (III Semester)**

Under the supervision of

**Prof. (Dr.) / Dr. / Ms. Divya mam**

### KIET Group of Institutions, Delhi-NCR, Ghaziabad



### Department Of Computer Applications

**KIET GROUP OF INSTITUTIONS, DELHI-NCR, GHAZIABAD-201206**

( 2024 - 2025)

**INDEX**

**TABLE OF CONTENTS**

1. Introduction --
2. Project / Research Objective --
3. Hardware and Software Requirements --
4. Project Flow/ Research Methodology --
5. Project / Research Outcome --
6. Proposed Time Duration --
7. Project /References/ Bibliography

**INTRODUCTION :**

* The Health Record Management System is a web-based application designed to process of storing, managing, and retrieving patient health records in a Digital form.
* With the increasing complexity of medical data, the traditional paper-based system is becoming outdated and inefficient. This project aims to digitize patient records, offering healthcare providers easy access to patient information, enhancing the quality of care, and improving data security, fix appointment between patient and doctor.
* Healthcare providers can use this system to create, update, and manage patient information securely. The system is built with a backend using Node.js and Express.js, and it stores data in a database, which is well-suited for managing large amounts of unstructured patient data.

**Project / Research Objective :**

The objective of a Health Record Management project is to improve the way patient health information is stored, accessed, and managed within healthcare systems. This project aims to develop or enhance an efficient, secure, and user-friendly system for managing electronic health records (EHRs).

1. **Efficient Storage and Management of Health Records:**

Store patient health data in a structured, scalable, and flexible manner.

For this we use Mongoose schemas to strore patient information in structured form.

**2. RESTful API Development for CRUD Operations:**

Create a set of RESTful API endpoints that allow authorized users (e.g., healthcare providers) to perform Create, Read, Update, and Delete (CRUD) operations on patient records.

**3. Secure Data Handling:**

Protect sensitive patient information by implementing security measures such as encryption, authentication, and role-based access control .

窗体顶端

窗体底端

**Hardware and Software Requirements :**

### ****Hardware Requirements****:

1. **Servers**: Powerful servers are essential to store and manage large volumes of patient data securely.
2. **Computers and Workstations**: Healthcare providers need desktops or laptops to access and update patient records in real-time.
3. **Backup Devices and Storage**: External hard drives or cloud storage solutions are required to back up data regularly, ensuring no information is lost.
4. **Network Infrastructure**: Routers, switches, and secure internet connections are necessary to ensure the system can be accessed by healthcare workers in different locations.

### ****Software Requirements****:

1. **Operating System**: The system should run on reliable and secure operating systems like Windows, Linux, or macOS.
2. **Nodejs**: which used to manage Packages and dependencies of the project.
3. **Mongodb** : The database which store the Large amount od data on Server. Data are store in the form of Mongoose Schemas in structure formate.
4. **Express.js**:
5. A web application framework for Node.js used to build the API by install it using npm:
6. **Electronic Health Record (EHR) Application**: The Web based application that allows healthcare professionals to input, update, and retrieve patient records digitally.
7. **Security Software**: Protect sensitive patient information with the help of Mongodb because it offer encrypt data storage.

This combination of hardware and software ensures a robust, efficient, and secure health record management system.

Project Flow/ Research Methodology **:**

* **Flow of Information**:
  + User → Node.js Server (Express) → MongoDB (Data) → Response to User.
* **Step 1**: MongoDB and Node.js setup.
* **Step 2**: Create RESTful API endpoints for patient records.
* **Step 3**: Implement Mongoose schemas for patient data.
* **Step 4**: Build the front-end interface and style using CSS.
* **Step 5**: Test and validate API routes using Postman.
* **Step 6**: Secure and deploy the application.

.

**Project / Research Outcome :**

窗体顶端

窗体底端

**Centralized Digital Health Record System**

The final system provides a centralized platform where hospitals, clinics, and healthcare professionals can securely store and access patient records. This eliminates the need for paper-based records and improve quality of care.

**Efficient and Real-time Data Access**

With the implementation of **Node.js** (for the backend) and **MongoDB** (for data storage), the system offers fast and real-time access to patient records. Healthcare providers can quickly retrieve patient data, medical history, and prescriptions.

**Scalability and Performance**

The use of **MongoDB** ensures the system can scale horizontally as the number of users and data grows. This flexibility allows hospitals and clinics to scale their infrastructure based on demand without performance degradation.

**Increased Patient Satisfaction**

With faster access to care, reduced waiting times, and better communication between healthcare providers, patients experience improved satisfaction levels. Patients will appreciate the accuracy and timeliness of their diagnoses and treatments.

**Proposed Time Duration:**

Planning and Requirement Analysis (1-2 Weeks)

* Conducting stakeholder interviews, gathering requirements, and defining functional/non-functional requirements.
* Researching regulatory requirements such as HIPAA, for security and data privacy compliance.
* Defining the project scope, identifying user roles (e.g., doctor, patient, admin), and developing user stories.

2. System Design (2-3 Weeks)

* Database Design: Design the MongoDB schema for patient records, doctor profiles, appointments, medical histories, etc.
* API Design: Define RESTful API endpoints for CRUD operations (patients, doctors, appointments, medical records).
* Security Design: Plan for authentication (JWT), role-based access control, and data encryption.

3. Backend Development (3-4 Weeks)

* Set up the Node.js environment and install necessary dependencies (Express.js, Mongoose, JWT).
* Develop API endpoints for user authentication, patient record management, appointment scheduling, etc.
* Implement role-based access control (doctors, patients, admins) to restrict data access.
* Implement error handling, logging, and security mechanisms.
* Unit testing for the developed features.
* Deliverables: Completed backend API with tested endpoints, secured with authentication and access control.

4.Frontend Development (Optional) (1-2 Weeks)

For frontend we use HTML5 , CSS.

5. Testing and Quality Assurance (2-3 Weeks)

* Unit Testing: Write unit tests for backend APIs and database models using Mocha, Chai, or Jest.
* Integration Testing: Test the integration between the backend API and the MongoDB database to ensure proper data flow.
* User Acceptance Testing (UAT): Allow healthcare professionals (doctors, nurses) to test the system for usability and gather feedback.
* Security Testing: Perform security audits and penetration testing to ensure compliance with data protection laws.
* Deliverables: Fully tested and bug-free application with quality assurance reports.

6. Deployment and Launch (1-2 Weeks)

* Set up cloud hosting using platforms like Heroku, AWS, or DigitalOcean.
* Deploy the MongoDB database using a cloud solution such as MongoDB Atlas.
* Configure the environment (load balancers, SSL certificates, environment variables).
* Perform final testing in the live environment.
* Deliverables: Fully deployed and live health record management system with monitoring tools in place.

窗体顶端

**Project /References/ Bibliography**

**World Health Organization (WHO):**

Electronic Health Records: Technical Briefing. Available at: <https://www.who.int>  
This document provides an overview of the implementation of electronic health records (EHR) in healthcare systems worldwide.

https://www.w3schools.com/

<https://nodejs.org/en>

### Node.js Official Documentation

Essential for understanding the core modules and features of Node.js.

* [Express.js Guide](https://expressjs.com/)

Official documentation for Express.js, covering routing, middleware, and more

.

* https://docpulse.com/

窗体顶端

窗体底端