

DBMS REPORT

GROUP - 3

Project: Electronic Medical Record WebApplication

Name: Shah Javed Hossain
ID: 1820290

Name: Prantika Roy
ID: 2021040

Name:Munabbirul Saqlain
ID:1821013

Name: Mohammad Sufyan Rahman
ID:1920498

INTRODUCTION

In today's ever-changing healthcare market, efficient management of patient medical information is critical to providing high-quality and tailored treatment. With the introduction of technology, creative ways to improve communication, expedite healthcare procedures, and improve patient outcomes have been developed.

Our report, which is in line with this trend, centers on the launch of a web application that aims to completely transform the tracking and management of patient medical records.

In addition to centralizing patient data, this web application offers a comprehensive platform that makes it easy to access vital information including appointments, prescriptions, patient medical history and test results. Doctors, patients, hospital/clinic receptions, labs and pharmacies will have access to use the system. By addressing the issues with current record-keeping practices, the adoption of this web application seeks to promote a more effective, safe, and cooperative approach to healthcare management.

LITERATURE REVIEW

This report's Literature Review section critically reviews existing research, scholarly articles, and industry best practices to provide a thorough overview of patient record tracking systems. We intend to derive lessons, identify trends, and analyze the achievements and obstacles experienced by prior projects in similar fields by researching the literature. Key themes will be investigated, including system usability, data security, interoperability, and the influence on healthcare outcomes. We seek to develop a foundation that is both informed and inventive by rooting our work in the current body of knowledge, ensuring that our web application aligns with industry best practices.

a) Literature Review - 1

Paper 1: Barriers to Electronic Health Record System Implementation and Information Systems Resources: A Structured Review

Journal/Conference Rank: Q1

Publication Year: 2017

Reference: [?]

1.1 Summary

The paper discusses a structured literature review that aimed to identify barriers related to Electronic Health Record (EHR) readiness. The review was conducted following the Preferred Reporting Items for Systematic Reviews and Meta-Analysis (PRISMA) guidelines. The search strategy involved specific keywords related to readiness, electronic health, electronic health records, and data resources. Articles were selected based on publication date, language, availability of full text, and their content, specifically related to barriers to EHR implementation or adoption. The identified barriers were categorized into information systems resources, which include people, hardware, software, data, network, and procedures. The most common barriers included user resistance, lack of education and training, lack of IT facilities, concerns about data security, and issues related to data standards and data privacy.

1.2 Method of Identification

The barriers were identified by reviewing a collection of scholarly articles. A total of 5626 articles were selected for initial screening and eventually 38 articles met the inclusion criteria. From these articles the barriers were divided into 6 main categories: people, hardware, software, data, network and procedure.

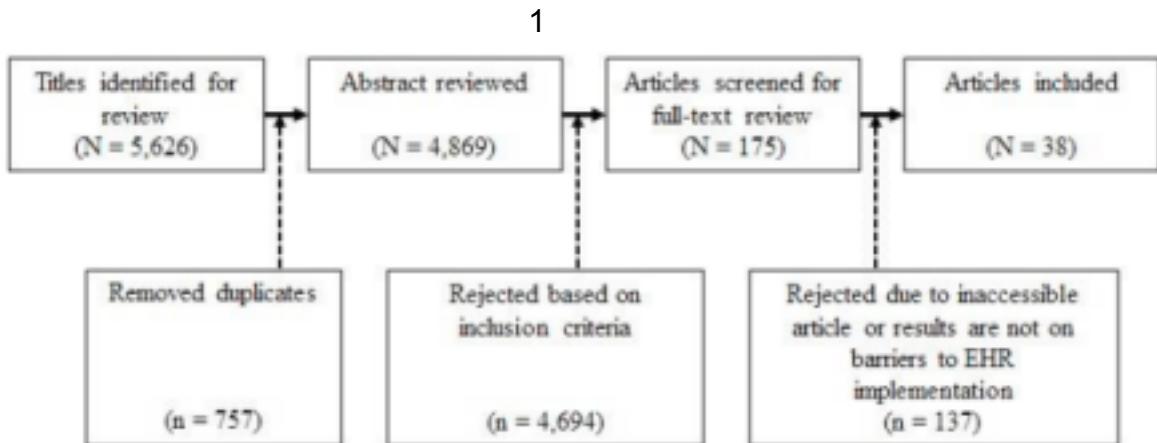


Figure 1: Flowchart of inclusion criteria

Table 4. Barriers associated with people resource

Barriers	Reference numbers	Count
1 User resistance	[6, 15, 19, 21, 24, 28, 30, 37, 40, 41, 44-47, 50]	15
2 Lack of computer skills	[23, 25, 28-30, 32, 42]	7
3 Increase of nurses and physician's workload	[6, 18, 19, 21, 24, 27, 28, 30, 36, 45, 46]	11
4 Lack of technical expertise	[25, 29, 30, 35, 37, 42, 43]	7
5 Lack of education and training	[15, 18-20, 23, 25, 26, 30, 31, 35, 36, 40, 43, 47, 48]	15
6 Inadequate staff	[16, 17, 19]	3
7 Provider or patients age	[19]	1
8 Lack of awareness of EHR/EMR and its importance	[15, 19, 20, 22, 24, 26, 29, 32, 35, 38, 40, 46, 49]	13
9 Reduces productivity and disturbs workflow	[11, 28, 29, 38]	4
10 Affects physician-patient interaction	[28, 30, 34]	3
11 Lack of healthcare providers' involvement in the design and implementation of EHRs	[21, 24, 26]	3
12 Change in culture required to embrace technology	[19, 32, 37]	3
13 Illiteracy on the part of patients	[23]	1
14 Communication among users on data entry	[30, 38]	2
15 Physicians' experience with poor products	[43]	1
16 User access limitation	[17]	1

Figure 2: Barrier parameters related to people resource

1.3 Data Parameters

parameters include elements such as publication date, language, availability of full text, content related to barriers, and the categories of information systems resources (people, hardware, software, data, network, and procedures) used to classify the identified barriers. These parameters were applied to filter and categorize the information within the dataset.

1.4 Datasets Used

The dataset in this context refers to the collection of scholarly articles that were reviewed to identify barriers to EHR implementation. This dataset includes 38 articles that met the inclusion criteria for the study.

1.4.1 Paper Link

Access the full paper at

<https://www.sciencedirect.com/science/article/pii/S1877050917329563>

b) Literature Review - 2

2 Paper 2: PatientDataChain: A Blockchain-Based Approach to Integrate Personal Health Records

Journal/Conference Rank: Q1

Publication Year: 2022

Reference: [?]

2.1 Summary

The paper introduces PatientDataChain, a blockchain-based technology designed to address the current lack of a trusted infrastructure for exchanging and storing medical data in the healthcare industry. It highlights the challenges posed by the absence of a platform

7

for monitoring patients history across the healthcare chain, which creates communication errors. PatientDataChain aims to provide an integrated solution that utilizes wearable device sensors to collect patient data from various healthcare providers. This data is then integrated into a unified Personal Health Records (PHR) system, with the patient retaining ownership of their data.

PatientDataChain is decentralized, it uses blockchain technology, ensures secure, flexible, and reliable data sharing and exchange, enhancing data confidentiality and privacy. The article describes the system's design and implementation. The key contributions of the paper include the concept of making healthcare value chain interoperable.

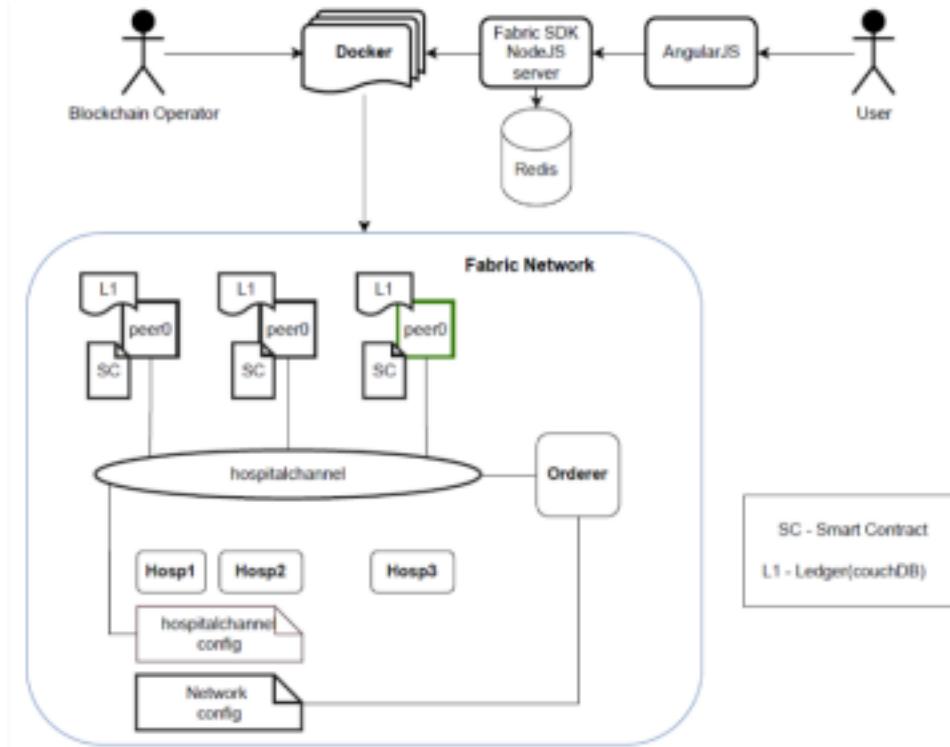


Figure 4: Software architecture diagram for Paper 1.

2.2 Software Architecture

The system uses Modex Blockchain Database (BCDB) which is a hybrid data storage system. Modex BCDB is middleware that exists between clients software and their regular database systems, this enables connection to a blockchain engine and has a similar interface as regular databases making it easier to use without additional training.

The Modex BCDB receives REST API calls directly, using GraphQL. The system can connect to multiple blockchains and database technologies. The main part of PatientDat aChain is the wallet app, functioning as the Personal Health Records (PHR) component. This app serves as a single source of truth, owned by the patient. Patients can grant read/write access to healthcare providers, enabling doctors to view their medical history and create new health records after appointments. Doctors can view patient history and the prescriptions can be stored in the database. Patients can provide access to pharmacies for viewing and even make online reservations for prescribed medications. The pharmacy notifies the patient when their order is ready for pickup. This system offers

Name	Type of System	Description
MedBlocks [56]	Electronic medical records (EMR)	P2P filesystem made immutable using distributed ledger technology, for storing medical records. Strength: Improved consensus mechanism to ensure the network is not overloaded
Secure Health Chain [57]	EMR	Stores electronic personal medical records on a blockchain and ensures that only authorized parties, who have been granted permission, can access data.
MedicalChain [58]	EMR	Decentralized platform that enables storage, exchange and use of medical data. It enables the user to give healthcare professionals access to their personal health data.
Doc.AI [59]	Generate insights from medical data	Enterprise AI platform that uses natural language processing, computer vision and blockchain to generate insights from medical data.
OmniPHR [60]	PHR	Distributed system for storing patient health data. It stores different patient datasets into different blocks on the chain
MIStore [61]	Medical Insurance Storage System	A blockchain-based medical insurance storage system

Figure 5: Review of blockchain systems in healthcare

a patient-centric approach to managing health records and facilitates secure interactions with healthcare providers and pharmacies.

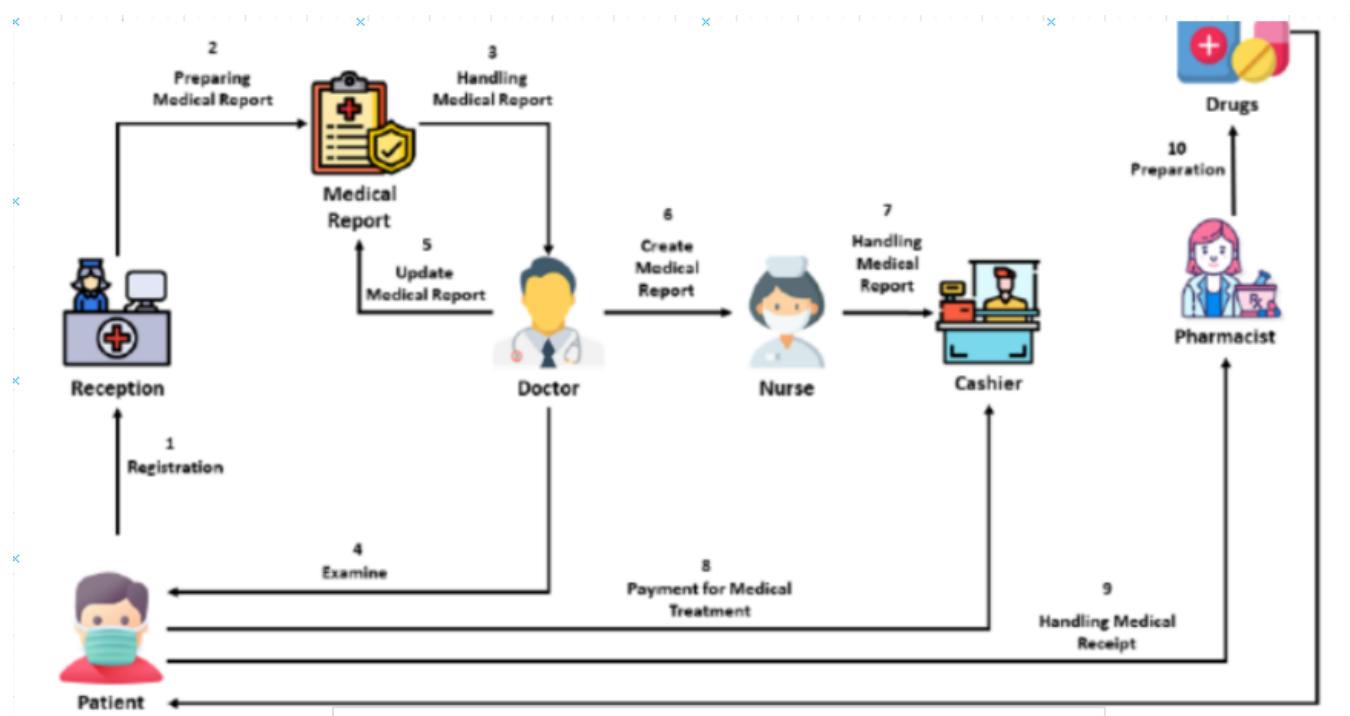
2.2.1 Paper Link

Access the full paper at <https://www.mdpi.com/1424-8220/20/22/6538>.

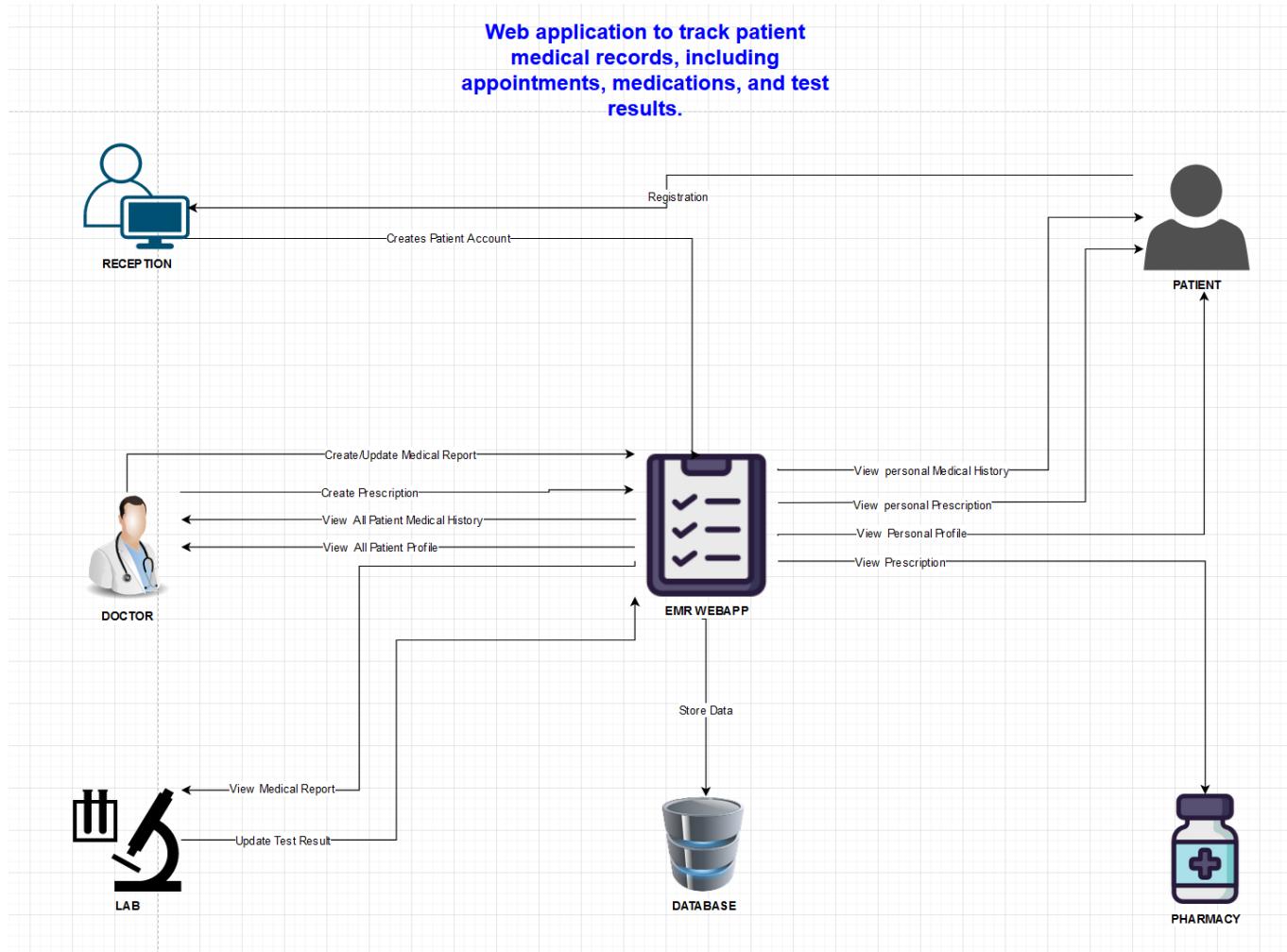
SYSTEM DESIGN

This section of the report navigates into the database management system design, illustrating the As-is Rich Picture of the existing system, To-be Rich Picture of the proposed system, Entity - Relationship Diagram, Schema, Normalized Schema and Data Dictionary that collectively contribute to the application's effectiveness.

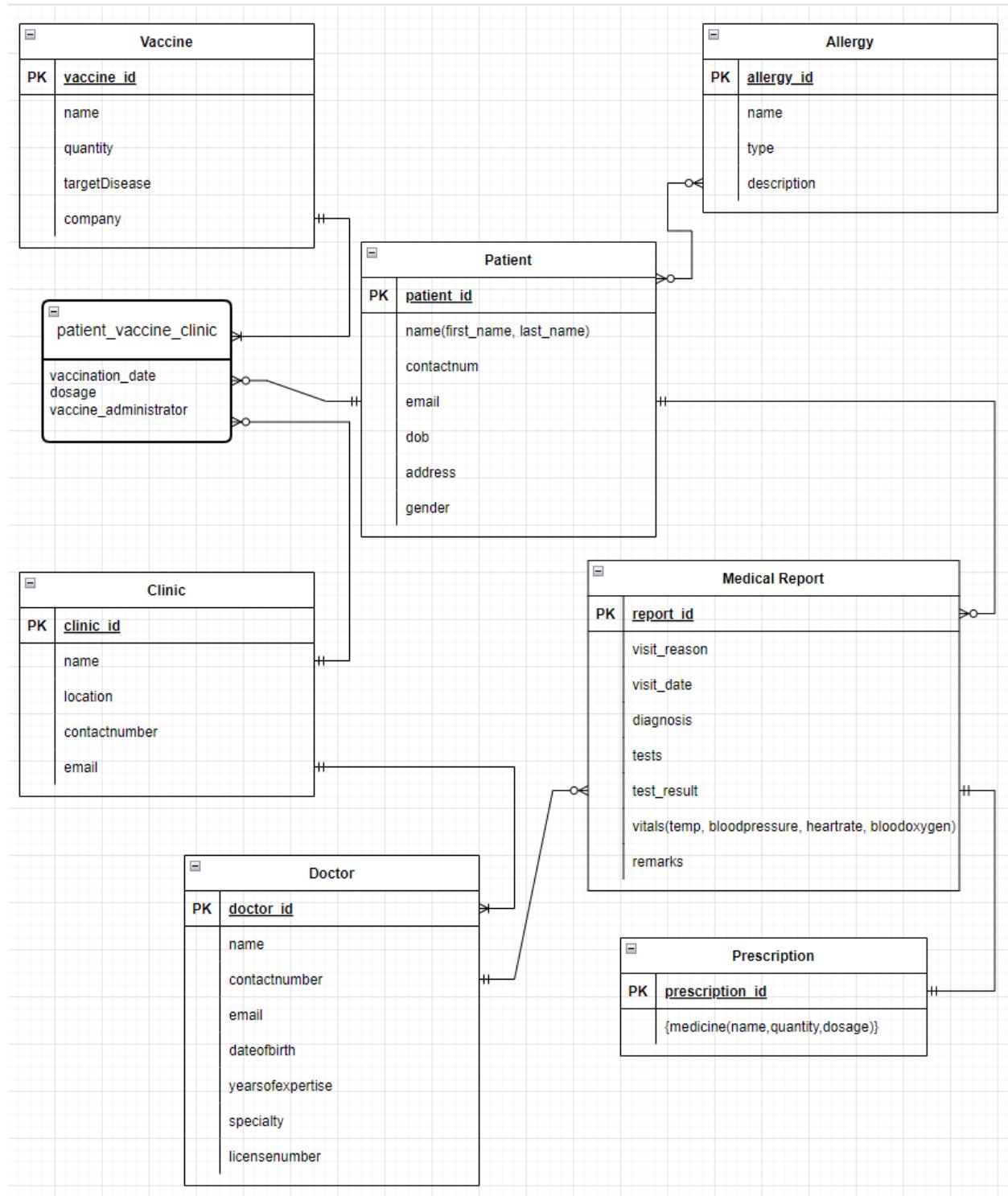
RICH PICTURE - AS IS



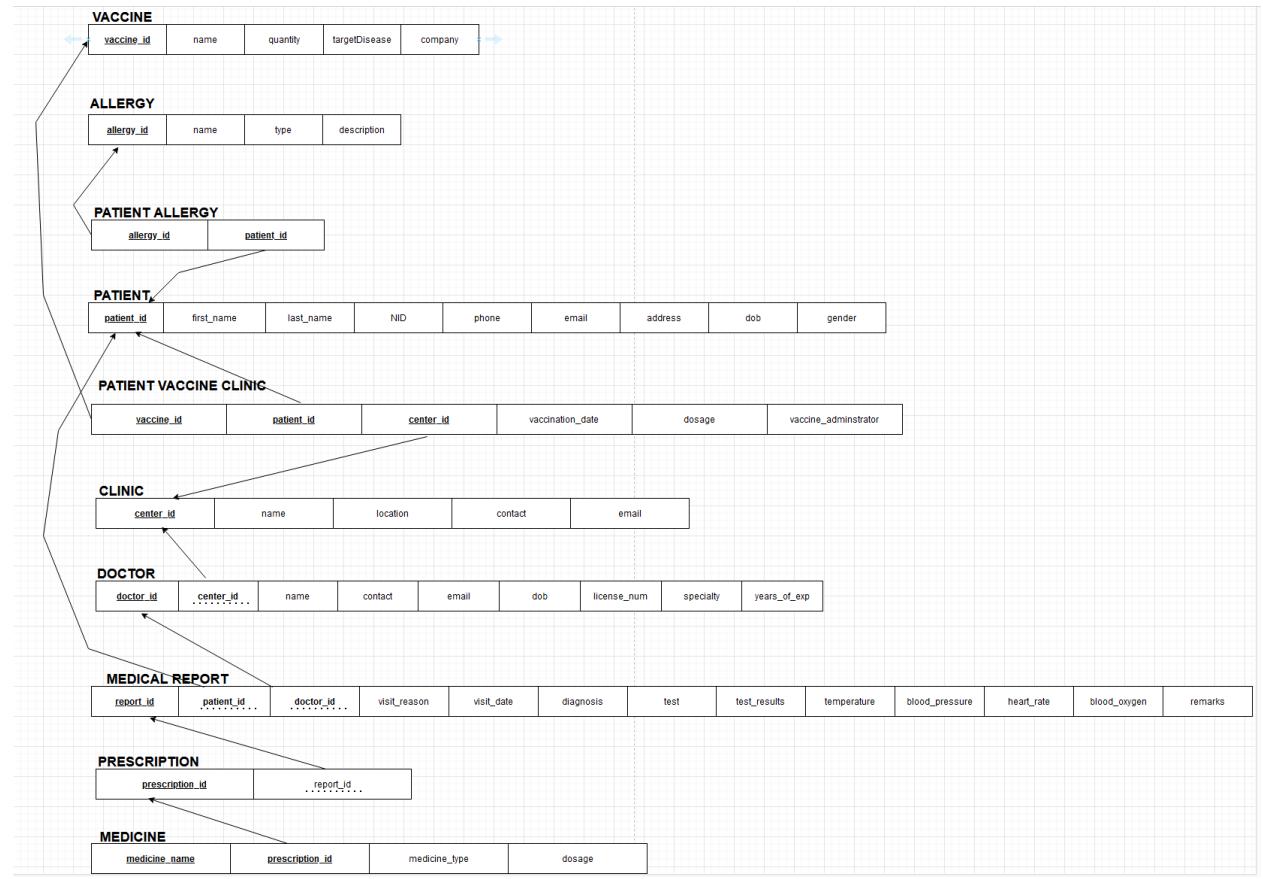
RICH PICTURE - TO BE



ERD



RELATIONSHIP SCHEMA



NORMALIZATION

ENTITY	ATTRIBUTES		ENTITY	ATTRIBUTES	
	PATIENT ID	P1		DOCTOR ID	D1
	NAME	P2		NAME	D2
	CONTACT NUM	P3		CONTACT NUMBER	D3
PATIENT	EMAIL	P4	DOCTOR	EMAIL	D4
	DOB	P5		DATE OF BIRTH	D5
	ADDRESS	P6		YEARS OF EXPERTISE	D6
	GENDER	P7		SPECIALITY	D7
				LICENSE NUMBER	D8

	CLINIC ID	CL1		VACCINE ID	
	NAME	CL2		NAME	
CLINIC	LOCATION	CL3	VACCINE	QUANTITY	
	CONTACT NUMBER	CL4		TARGET DISEASE	
	EMAIL	CL5		COMPANY	

	ALLERGY ID NAME TYPE	A1		REPORT ID	R1
		A2		VISIT REASON	R2
		A3		DIAGNOSIS	R3
ALLERGY	DESCRIPTION	A4	MEDICAL REPORT	TESTS	R4
	PREScription ID	PR1		VITALS	R5
PREScriptI ON	MEDICINE	PR2		REMARKS	R6

	VACCINATION DATE	PVC1
PATIENT VACCINE CLINIC	DOSAGE	PVC2
	VACCINE ADMINISTRATION	PVC3

P1 - P2,P3,P4,P5,P6,P7

D1 - D2,D3,D4,D5,D6,D7,D8

CL1- CL2,CL3,CL4,CL5

V1 - V2,V3,V4,V5

A1 - A2,A3,A4

PR1- PR2

R1 - R2,R3,R4,R5,R6,R7

PVC1- PVC2,PVC3

Normalization is a process in database design that aims to eliminate redundancy and improve data integrity by organizing data efficiently. It involves breaking down tables into smaller, related tables and establishing relationships between them. As we have properly designed our ERD, it means we have already identified and represented the relationships between entities. In an ideal scenario, a well-designed ERD should reflect a normalized structure to some

extent. The normalization process may not be necessary as ERD already exhibits characteristics of normalization, such as minimal redundancy and well-defined relationships.

DATA DICTIONARY

Table	Attribute	Data Type	Size	Required	Description
PATIENT	patient_id	INT	255	yes	Primary Key
PATIENT	first_name	VARCHAR	25	yes	first name of patient
PATIENT	last_name	VARCHAR	25	yes	last name of patient
PATIENT	NID	INT	255	yes	National Identification Number
PATIENT	phone	INT	255	yes	Contact number
PATIENT	email	VARCHAR	255	yes	Email address
PATIENT	address	VARCHAR	255	yes	Address of patient
PATIENT	DOB	DATE	Y-M-D	yes	Date of birth
PATIENT	gender	VARCHAR	8	yes	gender
PATIENT	gender	VARCHAR	8	yes	gender

Table	Attribute	Data Type	Size	Required	Description

DOCTOR	doctor_id	INT	255	yes	Primary key
DOCTOR	center_id	INT	255	yes	Foreign Key from center table
DOCTOR	name	VARCHAR	50	yes	Doctor name
DOCTOR	contact	INT	255	yes	Contact number
DOCTOR	email	VARCHAR	255	yes	Email address
DOCTOR	DOB	DATE	Y-M-D	yes	Date of birth
DOCTOR	license_num	INT	255	yes	License number of doctor
DOCTOR	specialty	VARCHAR	50	yes	Field of specialty of doctor
DOCTOR	years_of_exp	SMALLINT	255	yes	Years of experience

Table	Attribute	Data Type	Size	Required	Description
CLINIC	center_id	INT	255	yes	Primary key
CLINIC	name	VARCHAR	50	yes	Name of clinic
CLINIC	location	VARCHAR	255	yes	Location of clinic
CLINIC	contact	INT	255	yes	Contact number
CLINIC	email	VARCHAR	255	yes	Email address

Table	Attribute	Data Type	Size	Required	Description
VACCINE	vaccine_id	INT	255	yes	Primary key
VACCINE	name	VARCHAR	50	yes	Name of vaccination
VACCINE	quantity	INT	255	yes	Quantity of vaccination
VACCINE	targetDisease	VARCHAR	255	yes	Vaccination of disease

Table	Attribute	Data Type	Size	Required	Description
PATIENT VACCINE CLINIC	vaccine_id	INT	255	yes	Foreign key from VACCINE table
PATIENT VACCINE CLINIC	patient_id	INT	255	yes	Foreign key from PATIENT table
PATIENT VACCINE CLINIC	center_id	INT	255	yes	Foreign key from CLINIC table
PATIENT VACCINE CLINIC	vaccination_date	DATE	Y-M-D	yes	Date of vaccination
PATIENT VACCINE CLINIC	dosage	INT	50	yes	Dosage of vaccination
PATIENT	vaccine_	VARCHAR	255	yes	Name of vaccine

VACCINE CLINIC	administrator	R			administration
----------------	---------------	---	--	--	----------------

Table	Attribute	Data Type	Size	Required	Description
ALLERGY	allergy_id	INT	255	yes	Primary key
ALLERGY	name	VARCHAR	50	yes	Name of allergy
ALLERGY	type	VARCHAR	50	yes	Allergy type
ALLERGY	description	VARCHAR	255	yes	Description of allergy

Table	Attribute	Data Type	Size	Required	Description
PATIENT ALLERGY	allergy_id	INT	255	yes	Foreign key from ALLERGY table
PATIENT ALLERGY	patient_id	INT	255	yes	Foreign key from PATIENT table

Table	Attribute	Data Type	Size	Required	Description
MEDICAL REPORT	report_id	INT	255	yes	Medical report ID
MEDICAL REPORT	patient_id	INT	255	yes	Foreign key from PATIENT table
MEDICAL REPORT	doctor_id	INT	255	yes	Foreign key from DOCTOR table

MEDICAL REPORT	visit_reason	VARCHAR	255	yes	Reason of appointment
MEDICAL REPORT	visit_date	DATE	Y-M-D	yes	Date of appointment
MEDICAL REPORT	diagnoses	TEXT	255	yes	Diagnosis of patient
MEDICAL REPORT	test	VARCHAR	255	yes	Test given by doctor
MEDICAL REPORT	test_results	BLOB	BLOB	yes	Lab results in PDF file/format
MEDICAL REPORT	temperature	VARCHAR	56	yes	Body temperature of patient
MEDICAL REPORT	blood_pressure	INT	255	yes	Blood pressure of patient
MEDICAL REPORT	heart_rate	VARCHAR	50	yes	Heart rate of patient
MEDICAL REPORT	blood_oxygen	VARCHAR	50	yes	Blood oxygen level of patient
MEDICAL REPORT	remarks	TEXT	255	yes	Remarks made by doctor

Table	Attribute	Data Type	Size	Required	Description
PRESCRIPTION	prescription_id	INT	255	yes	Primary key
PRESCRIPTION	report_id	INT	255	yes	Foreign key from MEDICAL REPORT table

Table	Attribute	Data Type	Size	Required	Description
MEDICI	medicine_n	INT	255	yes	Medicine name

MEDICINE	prescription_name	VARCHAR	255	yes	Foreign key from PRESCRIPTION table
MEDICINE	medicine_type	VARCHAR	255	yes	Type of medicine (capsule, tablet, cream, injection, etc.)
MEDICINE	dosage	VARCHAR	255	yes	Dosage of prescribed medicine

The goal of the methodology and implementation strategy is to provide an EMR Web Application that is safe, scalable, and easy to use, while also meeting the unique requirements of healthcare providers and keeping with legal requirements. Maintaining the security of the system and keeping it in line with changing healthcare practices requires regular upgrades and maintenance.

METHODOLOGY

Project:1(React + Nodejs)

FRONTEND Methodology

1. **ReactJS:** Utilizing ReactJS for the front-end ensures a modular and efficient user interface. React's component-based architecture allows for the creation of reusable UI elements, promoting a maintainable codebase.
2. **Tailwind CSS:** Tailwind CSS is a utility-first CSS framework that facilitates rapid UI development. Its utility classes provide a pragmatic approach to styling, enabling a responsive and aesthetically pleasing design.
3. **React Charts:** Leveraging React Charts is essential for visualizing health data. The library helps in creating interactive and customizable charts, enabling users to comprehend complex medical information easily.

BACK END Methodology

1. **Node.js:** Node.js is chosen for the back-end due to its event-driven, non-blocking I/O model. This is particularly advantageous in handling concurrent requests in a medical application where responsiveness is critical.
2. **Express.js:** Express.js is a minimal and flexible Node.js web application framework, providing a robust set of features for web and mobile applications. It simplifies the development of RESTful APIs and integrates seamlessly with Node.js.
3. **MySQL Database:** MySQL is employed as the relational database management system (RDBMS) for storing and managing health records. Its reliability, scalability, and support for complex queries make it suitable for an application dealing with sensitive medical data.

Project:2 (Laravel)

FRONTEND Methodology

- **HTML:** HTML is a fundamental technology for creating and structuring content on the World Wide Web. HTML provides a standardized way to structure content on the web. This standardization ensures consistency and compatibility across different web browsers and platforms. HTML is supported by all major web browsers, making it a universal language for creating content that can be accessed by a broad audience.
- **Bootstrap:** Bootstrap is an open-source front-end framework that facilitates the development of responsive and mobile-friendly web pages. It includes pre-designed components, such as navigation bars, buttons, and forms, as well as a responsive grid system, simplifying the process of building visually appealing and consistent websites.
- **Charts:** Chart is implemented using js charting libraries in conjunction with Bootstrap. Popular charting libraries include Chart.js, D3.js, and others. These libraries allow developers to create various types of charts, such as bar charts, line charts, pie charts, etc., and they can be integrated into Bootstrap-based web projects to visualize data in a more graphical and interactive way.

BACK END Methodology

- **MVC:** MVC, or Model-View-Controller, is a software architectural pattern used in designing and organizing code in applications. It divides the application into three interconnected components: Model, View, Controller.
- **MySQL Database:** MySQL is employed as the relational database management system (RDBMS) for storing and managing health records. Its reliability, scalability, and support for complex queries make it suitable for an application dealing with sensitive medical data.

IMPLEMENTATION

Project:1(React + Nodejs)

FRONTEND Implementation

1. **Component Structure:** Design a modular component structure in React to ensure a clear separation of concerns. Components should represent different sections of the EHR, such as patient information, medical history, and diagnostic reports.
2. **User Authentication:** Implement user authentication using JWT (JSON Web Tokens) to secure access to patient records and comply with privacy regulations.
3. **Responsive Design:** Utilize Tailwind CSS to implement a responsive design, ensuring the application is accessible across various devices.
4. **Chart Integration:** Integrate React Charts to dynamically generate charts based on patient health data. Use real-time updates for live monitoring of vital signs.

BACKEND Implementation

1. **RESTful API:** Design a RESTful API using Express.js to handle requests from the front-end. Implement endpoints for CRUD operations on patient records, appointments, and other relevant data.
2. **Middleware:** Implement middleware for request validation, authentication, and authorization to ensure data security and compliance with privacy standards.

3. **Database Schema:** Create an optimized MySQL database schema to store patient records securely. Implement relationships between tables to represent associations between different entities (e.g., patients, doctors, medical reports).
4. **Data Encryption:** Implement encryption mechanisms to safeguard sensitive health data stored in the database.
5. **Error Handling:** Implement robust error handling to provide meaningful error messages to users and log errors for debugging purposes.

By following these methodologies and implementation details, you can ensure the development of a secure, scalable, and user-friendly Electronic Health Record web application. Regular testing, code reviews, and adherence to coding standards are crucial throughout the development process to maintain the application's integrity and security.

Project:2 (Laravel)

FRONTEND Implementation

- **Blade View:** Laravel Blade is a templating engine used in the Laravel PHP framework. It provides a concise and expressive syntax for defining views in Laravel applications. Some key features of Laravel Blade include: Inline PHP Code, Template Inheritance, Control Structures, Extending and Including Views, Blade Directives, Escaping Content.
- **User Authentication:** Laravel's built-in authentication system primarily uses sessions to manage user authentication. When a user logs in, Laravel creates a session to keep track of their authentication status. This session persists across subsequent requests, allowing the application to recognize and authenticate the user.
- **Responsive Design:** Utilize Bootstrap to implement a responsive design, ensuring the application is accessible across various devices.

- **Chart Integration:** Integrate Charts using JS scripts to dynamically generate charts based on patient health data. Use real-time updates for live monitoring of vital signs.

BACKEND Implementation

6. **Controller:** Design and implement a MVC controller method to handle requests from the front-end. Implement endpoints for CRUD operations on patient records, appointments, and other relevant data.
7. **Middleware:** Implement middleware for request validation, authentication, and authorization to ensure data security and compliance with privacy standards.
8. **Database Schema:** Create an optimized MySQL database schema to store patient records securely. Implement relationships between tables to represent associations between different entities (e.g., patients, doctors, medical reports).
9. **Data Encryption:** Implement encryption mechanisms to safeguard sensitive health data stored in the database.
10. **Error Handling:** Implement robust error handling to provide meaningful error messages to users and log errors for debugging purposes.

ARCHITECTURE

Software Architecture: Overview of 3-Tier Architecture

In the realm of software development, the choice of architecture plays a pivotal role in shaping the efficiency, scalability, and maintainability of a system. Our project adopts a 3-tier architecture, a well-established model that separates the

concerns of user interaction, business logic, and data storage into distinct layers. This design ensures a modular and scalable approach to building robust applications.

Tier 1: Presentation Layer (Client App)

The first tier, also known as the presentation layer, is where users interact with the system. In our architecture, this interaction takes place through the Client App, a user-friendly interface designed for seamless communication and intuitive user experience. The Client App is responsible for capturing user input, rendering data, and presenting information in a comprehensible manner. We're using a React+Tailwind app for our client side.

And in Laravel, Blade templates is the user interface which is coded with HTML-CSS which is responsible for capturing user input, rendering data, and presenting information in a comprehensible manner.

Tier 2: Application Layer (Backend Server)

Sitting between the Presentation Layer and the Data Layer, the Backend Server is the heart of our architecture. This layer houses the business logic, ensuring that the application's functionality is executed efficiently. When the Client App requests data or submits user input, the Backend Server processes these requests, interacts with the database, and orchestrates the flow of information. It acts as a bridge, facilitating communication between the user interface and the data storage layer. We're using a node+express framework for our back-end server.

And in Laravel, the Controller requests data or submits user input, the Backend Server processes these requests, interacts with the database, and orchestrates the flow of information. It acts as a bridge, facilitating communication between the user interface and the data storage layer. Here we are using the Controller of MVC architecture.

Tier 3: Data Layer (Database)

The final tier, the Data Layer, involves the storage and retrieval of data. Our chosen database system is queried by the Backend Server to fetch and store information. This separation of concerns ensures that data management is abstracted from the business logic, providing flexibility and scalability. We have implemented secure and efficient database interactions to guarantee the integrity and availability of the stored data. We're using mySQL for our database needs.

Interaction Flow

1. **User Interaction:** Users interact with the Client App, providing input and receiving real-time feedback.
2. **Client-Server Communication:** The Client App sends requests to the Backend Server, seeking data or initiating processes.
3. **Business Logic Execution:** The Backend Server processes requests, executes business logic, and interacts with the database as needed.
4. **Data Retrieval/Storage:** Database queries are performed to retrieve or store data, based on the requirements of the application.
5. **Data Transmission:** The retrieved data is sent back to the Client App, where it is presented to the user while in Laravel data is sent to the Controller for interaction.

This 3-tier architecture ensures a modular and scalable design, allowing for easy maintenance, updates, and future expansions. The clear separation of concerns enhances system flexibility, making it adept at handling evolving requirements while maintaining a robust foundation for optimal performance.

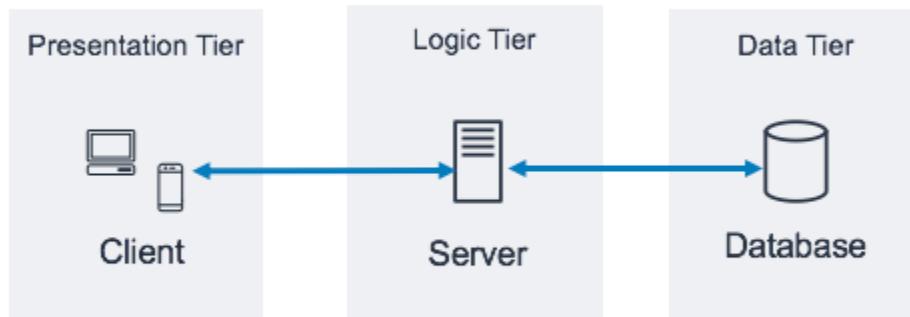
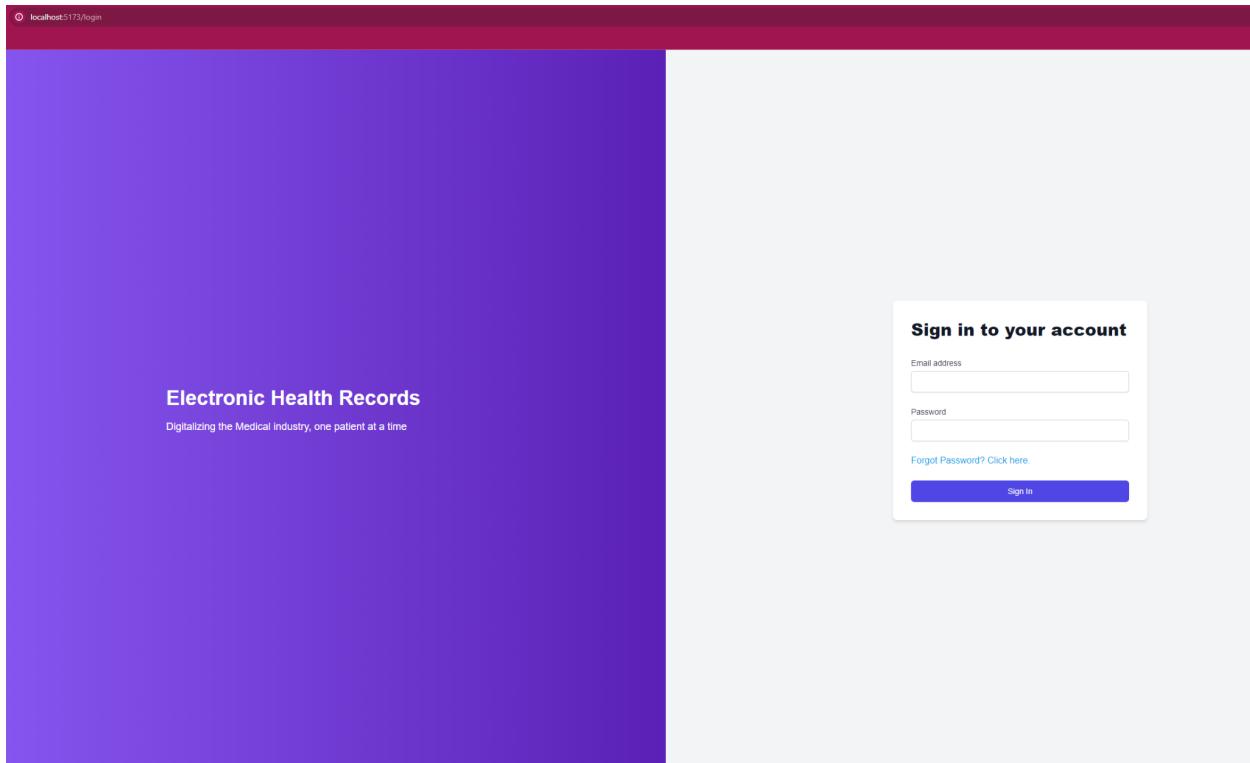


Fig: 3 Tier Architecture

INTERFACE

Project:1 (React + Nodejs)

ALL User: Login



All User Login features (Functionality, mySQL commands)

- Checks for user in database, if user doesn't exist, sends "user doesn't exist" error
- Checks for user in database, if user exists but wrong password says "invalid username or password"
- Checks for user in database, if user exists and types correct password, successfully logs in and is assigned a JWT token for future requests

SQL Statement:

```
// Authentication Login
app.post('/login/auth', async (req, res) => {
    const {email, password} = req.body;
    const sqlSearch = "SELECT * FROM user2 WHERE email = ?"
    const search_query = mysql.format(sqlSearch,[email])

    db.query (search_query, async (err, result) => {
        if (err) throw (err)

        if(result.length == 0) {
            console.log("User doesn't exist")
            res.sendStatus(404)
        }
        else {
            const hashedPassword = result[0].password
            console.log(result)

            if(await bcrypt.compare(password, hashedPassword)) {
                // GENERATE JWT TOKEN
                const accessToken = jwt.sign(
                    {
                        email: result[0].email,
                        id: result[0].id,
                        role: result[0].role
                    },
                    "accessTokenSecretKey"
                )
                console.log("Login Sucessfull!")
                res.json(accessToken)
            }
            else {
                console.log ("Incorrect Password")
                res.send("Incorrect Password")
            }
        }
    })
})
```

ALL User: Registration

User Registration

Name

Email Address

Password

Role

Register User

Already have an account? [Sign in here](#)

Features

- Ability to register user
- Available roles - admin, patient, doctor
- During registration, searches for duplicate email, if duplicate found, shows an error 'user already exists'.

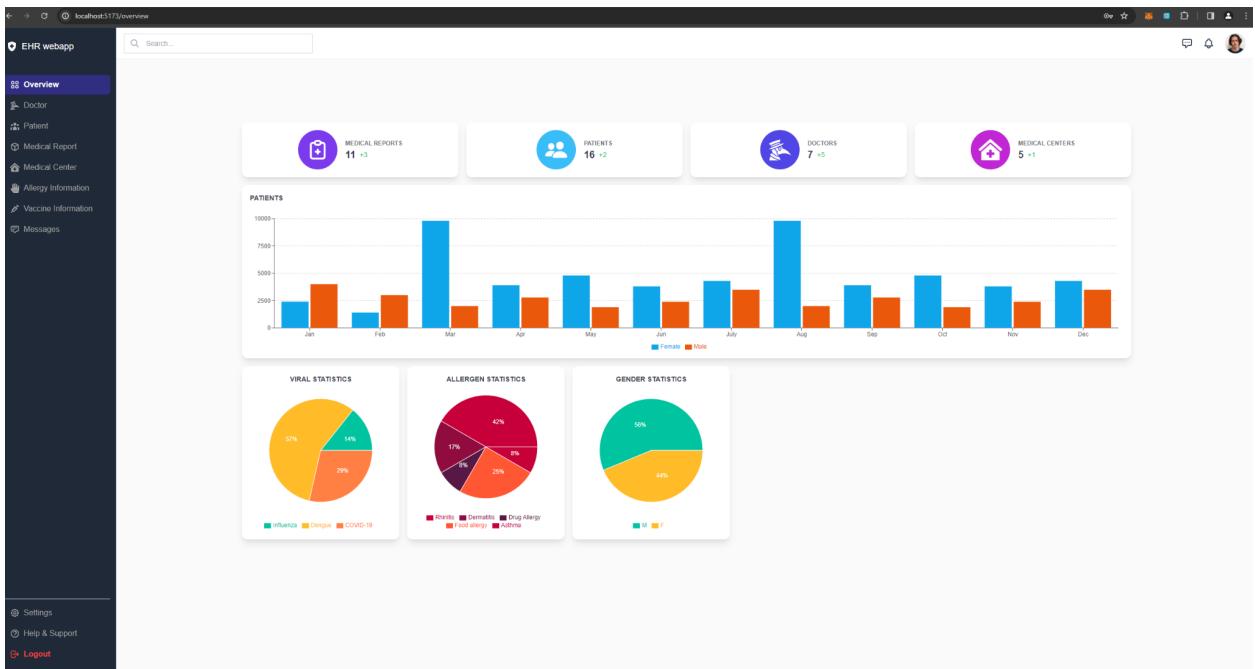
SQL Statement:

```
// REGISTER USER
app.post('/api/register_user', async (req,res) => {
    console.log(req.body);
    const name = req.body.name;
    const email = req.body.email;
    const hashedPassword = await bcrypt.hash(req.body.password,10);
    const role = req.body.role;

    // concept of escaping to prevent SQL injection used below -> msql.format
    const sqlSearch = "SELECT * FROM user2 WHERE email = ?"
    const sqlInsert = "INSERT INTO user2 (name, email, password, role) VALUES (?, ?, ?, ?)"
    const search_query = mysql.format(sqlSearch,[email])

    db.query (search_query, async (err, result) => {
        if (err) throw (err)
        console.log("----> search results")
        console.log(result.length)
        if (result.length !=0) {
            console.log("User already exists")
            res.sendStatus(409)
        }
        else {
            db.query (sqlInsert, [name, email, hashedPassword, role], (err, result) => {
                console.log(err);
                console.log("Created new user")
                console.log(result.insertId)
                res.sendStatus(201)
            })
        }
    })
    db.commit();
});
```

Admin Dashboard: Overview



Overview Features

- Displays total number of registered doctors
- Displays total number of registered patients
- Displays total number of registered hospitals
- Displays total number of registered medical reports
- Dynamic Bar chart for total number of male/female patients
- Dynamic Pie chart showing most common viruses
- Dynamic Pie chart showing most common diseases
- Dynamic Pie chart showing most common allergies

SQL Statements for PIECHART

```
309 // STATISTICS API
310 app.get('/api/grid_data', (req, res) => {
311   const sqlRetrieve = "SELECT 'Patient' AS role, COUNT(*) AS total FROM user UNION ALL SELECT 'Doctor' AS role, COUNT(*) AS total FROM doctor UNION SELECT 'Hospital' AS role, COUNT(*) AS total_count
312   db.query (sqlRetrieve, (err, data) => {
313     if(err) return res.json(err)
314     return res.json(data)
315   })
316 });
317
318 app.get('/api/virus_data', (req, res) => {
319   const sqlRetrieve = "SELECT diagnosis AS name, COUNT(*) AS value FROM medical_report WHERE diagnosis IN ('covid-19', 'dengue', 'influenza') GROUP BY diagnosis"
320   db.query (sqlRetrieve, (err, data) => {
321     if(err) return res.json(err)
322     return res.json(data)
323   })
324 });
325
326 app.get('/api/gender_data', (req, res) => {
327   const sqlRetrieve = "SELECT gender AS name, COUNT(*) AS value FROM user GROUP BY gender"
328   db.query (sqlRetrieve, (err, data) => {
329     if(err) return res.json(err)
330     return res.json(data)
331   })
332 });
333
334 app.get('/api/allergy_data', (req, res) => {
335   const sqlRetrieve = "SELECT allergy.name AS name, COUNT(*) AS value FROM patient_allergy INNER JOIN allergy ON patient_allergy.allergy_id = allergy.allergy_id GROUP BY patient_allergy.allergy_id"
336   db.query (sqlRetrieve, (err, data) => {
337     if(err) return res.json(err)
338     return res.json(data)
339   })
340 });
341
342 app.get('/api/allergy_table', (req, res) => {
343   const sqlRetrieve = "SELECT * FROM allergy"
344   db.query (sqlRetrieve, (err, data) => {
345     if(err) return res.json(err)
346     return res.json(data)
347   })
348 });

localhost:5173/patientpanel
```

Admin Dashboard: Patient Panel

ID	Customer Name	Gender	Blood Group	Date of Birth	Phone	Address
#1	Shriley A. Lape	Male	AB+	17 Feb 1999	01666040999	Cottage Grove, OR 97424
#2	Ryan Carroll	Male	A+	14 Jan 1997	0176042932	Los Angeles, CA 90017
#3	Mason Nash	Female	B+	17 May 1988	01722040932	Westminster, CA 92683
#4	Luke Parker	Female	O+	23 Oct 2003	01776040990	San Mateo, CA 94403
#5	Anthony Fry	Female	B+	09 May 2001	01776040932	San Mateo, CA 94403
#5	Ryan Carroll	Male	AB+	14 May 1992	01776040932	Los Angeles, CA 90017

Patient Panel Features

- Display a minimalist table which shows all patient data.
- Data include, patient name, gender, date of birth, address, blood group and phone number.

- Ability to add new patients via the “add new patient” button, which would navigate the user to a form which they have till fill up to add a new patient into the database.

SQL STATEMENT

```

226      //ALL patient table API
227      app.get('/api/patient_table', (req, res) => {
228          const sqlRetrieve = "SELECT * FROM user"
229          db.query(sqlRetrieve, (err, data) => {
230              if(err) return res.json(err)
231              return res.json(data)
232          })
233      });

```

Admin Dashboard: Doctor Panel

ID	Doctor Name	Gender	Specialty	License Number	email	Phone
1	Malha Hossain	F	Dermatologist	831B	malha@doctor.com	01966699992
2	Shah Hossain	M	General Medicine	912Y	shah@doctor.com	01776040932
3	Sebastian Vettel	M	Pathologist	420T	vettel@doctor.com	01776040821
4	Kimi Rakkonen	M	Gastroenterologist	490A	kimi@doctor.com	01223949213
5	Lewis Hamilton	M	Orthopedics	AK2TA	lewis@doctor.com	01776040999
6	Max Verstappen	M	Cardiologist	33MEG	max@doctor.com	01776040991
7	Lando Norris	M	ENT	B4IC	lando@doctor.com	01776040991
8						
9						

Features

- Shows all doctors
- Can register new doctor

SQL Statements:

```
//ALL doctor Table API
app.get('/api/doctor_table', (req, res) => {
  const sqlRetrieve = "SELECT * FROM doctor"
  db.query(sqlRetrieve, (err, data) => {
    if(err) return res.json(err)
    return res.json(data)
  })
});
```

Admin Dashboard: Medical Report Panel

ID	Patient Name	Doctor Name	Appointment Date	Diagnosis
#1	Shelly A. Lape	Dr. Batman	17 Feb 2019	Viral Fever
#7	Ryan Carroll	Dr. Iron Man	14 Jan 2021	Sore throat
#2	Mason Nash	Dr. Shontu Montu	17 May 2022	Hypertensive Thyroid
#3	Luke Parkin	Dr. Shontu Montu	23 Oct 2010	High cholesterol
#4	Anthony Fry	Dr. Kento Pottu	09 May 2023	Inflammation of lungs
#5	Ryan Carroll	Dr. Sheikh Hasina	14 May 2023	Blocked artery

Patient Panel Features

- Display a minimalist table which shows all medical report data. Data include, patient name, doctor name, appointment date, appointment reason, diagnosis, lab results, lab test, prescription, remarks.
- Ability to add a new medical report via the “add new medical report” button, which would navigate the user to a form which they have till fill up to add a new medical report into the database.

SQL STATEMENT

```

278 //ALL REPORT API
279 app.get('/api/report_table', (req, res) => {
280   const sqlRetrieve = "SELECT medical_report.report_id,user.name AS patient_name,doctor.name AS doctor_name,medical_report.diagnosis,medical_report.visit_date FROM medical_report INNER JOIN user ON
281   db.query (sqlRetrieve, (err, data) => {
282     if(err) return res.json(err)
283     return res.json(data)
284   })
285 });

```

Admin Dashboard: User Profile

PATIENT DETAILS

ID:	1
Name:	Shah Javed
Gender:	M
Blood Group:	AB+
Date of Birth:	29 Nov 2001
Email:	shah@patient.com

ALLERGY DETAILS

Rhinitis
Dermatitis
Drug Allergy
Food allergy
Asthma

ADD ALLERGY

VACCINE DETAILS

Vaccine Name	Date
Influenza Vaccine	29 Dec 1999
Chicken Pox Vaccine	01 Jan 2000
Hepatitis A	23 May 2000

ADD MEDICAL REPORT for Patient

Individual Patient Medical Reports

Report ID	Patient Name	Doctor Name	Appointment Date	Diagnosis
#1	Shah Javed	Dr. Shah Hossain	29 Nov 2022	Influenza
#5	Shah Javed	Dr. Martha Hossain	10 Oct 2019	COVID-19
#6	Shah Javed	Dr. Kari Räkinen	06 Dec 2023	Heartburn
#12	Shah Javed	Dr. Sebastian Vittel	14 Dec 2023	sd

User Profile Feature:

- Shows individual patient details
- Show patient allergy details
- Shows patient vaccine details
- Can add allergy to patient
- Can add vaccine to patient
- Patient based medical report list
- Can add patient based medical report

SQL STATEMENTS

```

//Input ALLERGY TO PATIENT
app.post('/api/add_allergy/:id', (req, res) => {
    console.log(req.body);
    const patient_id = req.params.id
    const allergy_id = req.body.allergy_id

    const sqlInsert = "INSERT INTO patient_allergy (patient_id, allergy_id) VALUES (?,?)"
    db.query(sqlInsert, [patient_id, allergy_id], (err, result) =>{
        console.log(err);
    })
    db.commit();
});

```

```

//Input VACCINATION TO PATIENT
app.post('/api/add_vaccine/:id', (req, res) => {
    console.log(req.body);
    const patient_id = req.params.id
    const vaccine_id = req.body.vaccine_id
    const vaccination_date = req.body.date
    const vaccine_administrator = req.body.vaccine_administrator

    const sqlInsert = "INSERT INTO patient_vaccine_clinic (patient_id, vaccine_id, vaccination_date, vaccine_administrator) VALUES (?,?,?,?)"
    db.query(sqlInsert, [patient_id, vaccine_id, vaccination_date, vaccine_administrator], (err, result) =>{
        console.log(err);
    })
    db.commit();
});

```

```

//INDIVIDUAL MEDICAL REPORT API (ALL report for specific patient)
app.get('/api/patient/medical_report/:id', (req, res) => {
    const patient_id = req.params.id

    const sqlRetrieve = "SELECT medical_report.report_id, user.name AS patient_name, doctor.name AS doctor_name, medical_report.diagnosis, medical_report.visit_date FROM medical_report INNER JOIN user
db.query (sqlRetrieve, [patient_id], (err, data) => {
    if(err) return res.json(err)
    return res.json(data)
})
});

267 //PATIENT ALLERGY API
268 app.get('/api/patient/allergy/:id', (req, res) => {
269     const patient_id = req.params.id
270
271     const sqlRetrieve = "SELECT allergy.allergy_id, allergy.name AS allergy_name, user.patient_id AS patient_id, user.name AS patient_name FROM patient_allergy INNER JOIN allergy ON patient_allergy.all
272     db.query (sqlRetrieve, [patient_id], (err, data) => {
273         if(err) return res.json(err)
274         return res.json(data)
275     })
276 });

```

```
//Individual Patient profile API
app.get(`/api/patient/:id` , (req, res) => {
    const patient_id = req.params.id

    const sqlRetrieve = "SELECT * FROM user WHERE patient_id= ?"
    db.query (sqlRetrieve, [patient_id], (err, data) => {
        if(err) return res.json(err)
        return res.json(data)
    })
});
```

Admin Dashboard: In-depth Medical Report

The screenshot displays the Admin Dashboard interface for an in-depth medical report. The top navigation bar shows the URL as `localhost:5173/report/1`. The left sidebar menu includes options like Overview, Doctor, Patient, Medical Report, Medical Center, Allergy Information, Vaccine Information, and Messages. The main content area is titled "Medical Report ID: 1" and shows the appointment date as "29 Nov 2022" and reason as "fever".

PATIENT DETAILS:

- ID: 1
- Name: Shah Javed
- Gender: M
- Blood Group: AB+
- Date of Birth: 29 Nov 2001
- Email: shah@patient.com

DOCTOR DETAILS:

- ID: 2
- Doctor Name: Shah Hossain
- Specialty: General Medicine
- Hospital: Apollo Hospitals

VITALS:

- Temperature: 101
- Blood Pressure: 90/60 mmHg
- Heart Rate: 88 BPM
- Blood Oxygen: 98%

PRESCRIPTION:

Medicine	Dosage
Napa	1+1+1 for 7 days
Paracetamol	1+0+0 for 7 days
Sergel	1+0+1 for 30 days
Vikao	1+0+0 for 60 days

Diagnosis: Influenza

Remarks: drink plenty fluids

Test: RDT
Test Result:

In depth Medical Report Feature

- More in depth details of the medical report.
- Shows vitals of patient for given medical report
- Shows prescription for the given medical report
- Shows doctor details of the given medical report
- Shows patient details of the given medical report
- Ability to add more medicine to prescription

SQL Statement

```
//Input pres TO med report
app.post('/api/add_pres/:id', (req, res) => {
    console.log(req.body);
    const prescription_id = req.params.id
    const medicine_name = req.body.medicine_name
    const dosage = req.body.dosage

    const sqlInsert = "INSERT INTO medicine (medicine_name, prescription_id, dosage) VALUES (?, ?, ?)"
    db.query(sqlInsert, [medicine_name, prescription_id, dosage], (err, result) =>{
        console.log(err);
    })
    db.commit();
});

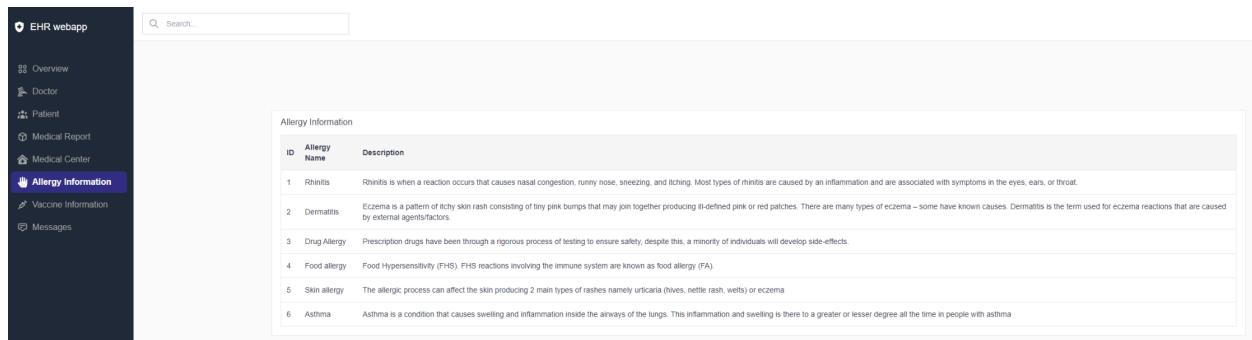
// Individual Report DETAILS API
app.get('/api/report/:id', (req, res) => {
    const report_id = req.params.id

    const sqlRetrieve = "SELECT medical_report.report_id, user.patient_id AS patient_id, user.gender, user.bloodgroup, user.email, user.dob, user.name AS patient_name, doctor.doctor_id AS doctor_id, doctor.special"
    db.query(sqlRetrieve, [report_id], (err, data) => {
        if(err) return res.json(err)
        return res.json(data)
    })
});

// PRESCRIPTION for REPORT API
app.get('/api/report/prescription/:id', (req, res) => {
    const report_id = req.params.id

    const sqlRetrieve = "SELECT medical_report.report_id, prescription.prescription_id, medicine.medicine_name, medicine.dosage FROM medical_report INNER JOIN prescription ON medical_report.report_id = prescription.prescription_id"
    db.query(sqlRetrieve, [report_id], (err, data) => {
        if(err) return res.json(err)
        return res.json(data)
    })
});
```

Allergy Database Page



The screenshot shows the EHR webapp interface. On the left, there is a sidebar with navigation links: Overview, Doctor, Patient, Medical Report, Medical Center, and Allergy Information (which is currently selected). A search bar is located at the top right. The main content area is titled 'Allergy Information' and displays a table with six rows of allergy data:

ID	Allergy Name	Description
1	Rhinitis	Rhinitis is when a reaction occurs that causes nasal congestion, runny nose, sneezing, and itching. Most types of rhinitis are caused by an inflammation and are associated with symptoms in the eyes, ears, or throat.
2	Dermatitis	Eczema is a pattern of itchy skin rash consisting of tiny pink bumps that may join together producing ill-defined pink or red patches. There are many types of eczema – some have known causes. Dermatitis is the term used for eczema reactions that are caused by external agents/factors.
3	Drug Allergy	Prescription drugs have been through a rigorous process of testing to ensure safety, despite this, a minority of individuals will develop side-effects.
4	Food allergy	Food Hypersensitivity (FHS) FHS reactions involving the immune system are known as food allergy (FA).
5	Skin allergy	The allergic process can affect the skin producing 2 main types of rashes namely urticaria (hives, nettle rash, welts) or eczema
6	Asthma	Asthma is a condition that causes swelling and inflammation inside the airways of the lungs. This inflammation and swelling is there to a greater or lesser degree all the time in people with asthma

Features

- Shows allergy information from existing database

SQL Statement

Registration: Patient

The screenshot shows a web application interface for patient registration. The left side features a large blue gradient background with the text "Electronic Health Records" and "Patient Registration Dashboard". Below this, there is a placeholder text: "Enter patient details to onboard a patient". The right side contains a form titled "Patient Registration". The form includes fields for Email address, Password, First Name, Last Name, National Identification Number, Phone, Address, Gender (with options for Male and Female), and Date of Birth (in mm/dd/yyyy format). A "Register Patient" button is located at the bottom right of the form.

localhost:5173/register/patient

Patient Registration

Email address

Password

First Name

Last Name

National Identification Number

Phone

Address

Gender
 Male
 Female

Date of Birth
mm/dd/yyyy

Register Patient

Registration Patient Features(Functionality, mySQL commands)

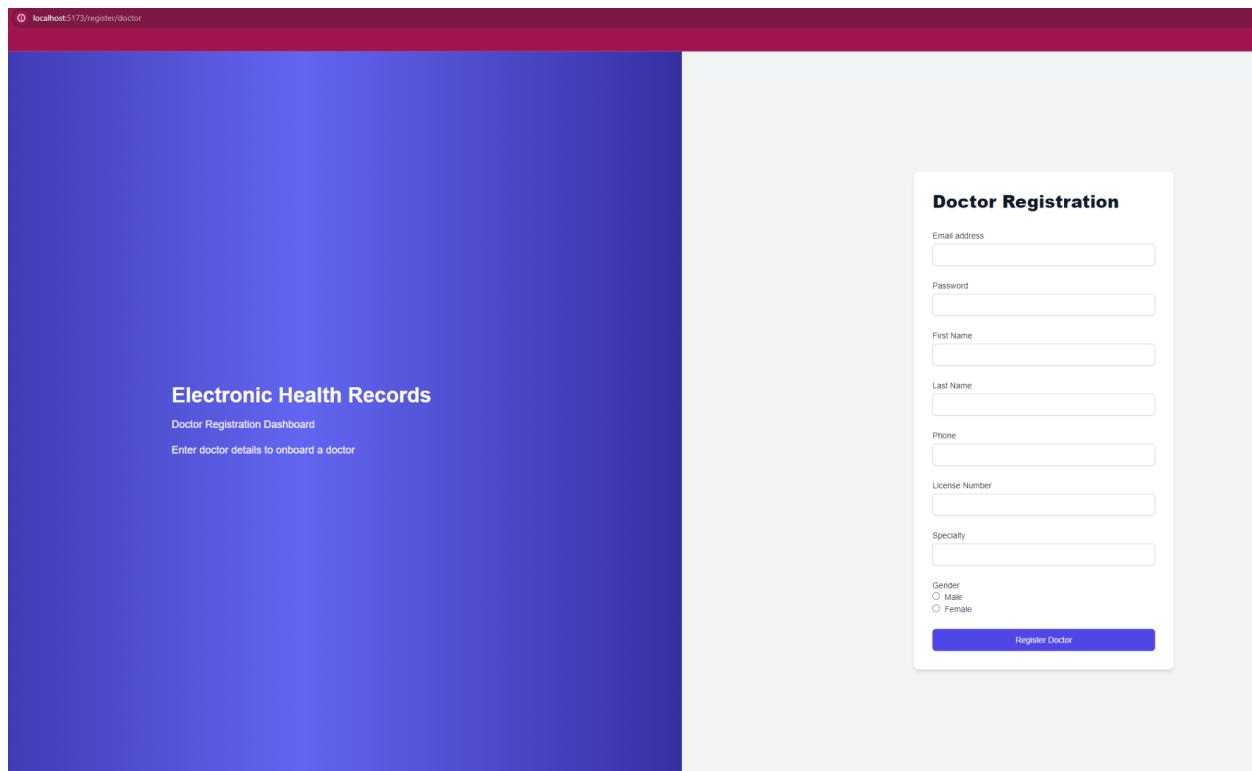
- Ability to register new patients into the database.

SQL STATEMENT

```
// INPUT PATIENT INFO
app.post('/api/register_patient', (req, res) => {
  console.log(req.body);
  const name = req.body.patientName
  const email = req.body.patientEmail
  const password = req.body.password
  const nid = req.body.nid
  const phone = req.body.phone
  const address = req.body.address
  const gender = req.body.gender
  const bloodgroup = req.body.bloodgroup
  const dob = req.body.date

  const sqlInsert = "INSERT INTO user (name, email, password, nid, phone, address, gender, bloodgroup, dob) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)"
  db.query(sqlInsert, [name, email, password, nid, phone, address, gender, bloodgroup, dob], (err, result) =>{
    console.log(err);
  })
  db.commit();
});
```

Registration: Doctor



Registration Doctor Features (Functionality, mySQL commands)

- Input form to add new doctor into the database.

SQL Statement

```
//Input DOCTOR info
app.post('/api/register_doctor', (req, res) => {
    console.log(req.body);
    const name = req.body.name
    const email = req.body.email
    const password = req.body.password
    const phone = req.body.phone
    const gender = req.body.gender
    const license_num = req.body.license
    const specialty = req.body.specialty

    const sqlInsert = "INSERT INTO doctor (name, email, password, phone, gender, license_num, specialty) VALUES (?,?,?,?,?,?,?)"
    db.query(sqlInsert, [name, email, password, phone, gender, license_num, specialty], (err, result) =>{
        console.log(err);
    })
    db.commit();
});
```

Create: Medical Report

The screenshot shows a web application interface for creating a medical report. On the left, there's a sidebar with a dark blue gradient background. It features the title "Electronic Medical Records" and a placeholder text "Enter health details to create a report". On the right, the main content area is titled "Create Medical Report". It contains several input fields:

- Appointment Date: A date input field with a calendar icon.
- Appointment Reason: A text input field.
- Diagnosis: A text input field.
- Lab Test: A text input field.
- Lab Test Result: A file input field with a "Choose file" button and a note "No file chosen".
- Remarks: A text input field.
- Prescription: A text input field.
- Vitals:
 - Temperature: A text input field.
 - Blood Pressure: A text input field.
 - Blood Oxygen: A text input field.
 - Heart Rate: A text input field.
- Patient ID: A text input field.
- Doctor ID: A text input field.

At the bottom right of the form is a purple "Create Medical Report" button.

Medical report creation Details (Functionality, mySQL commands)

- Input form to make new Medical Report

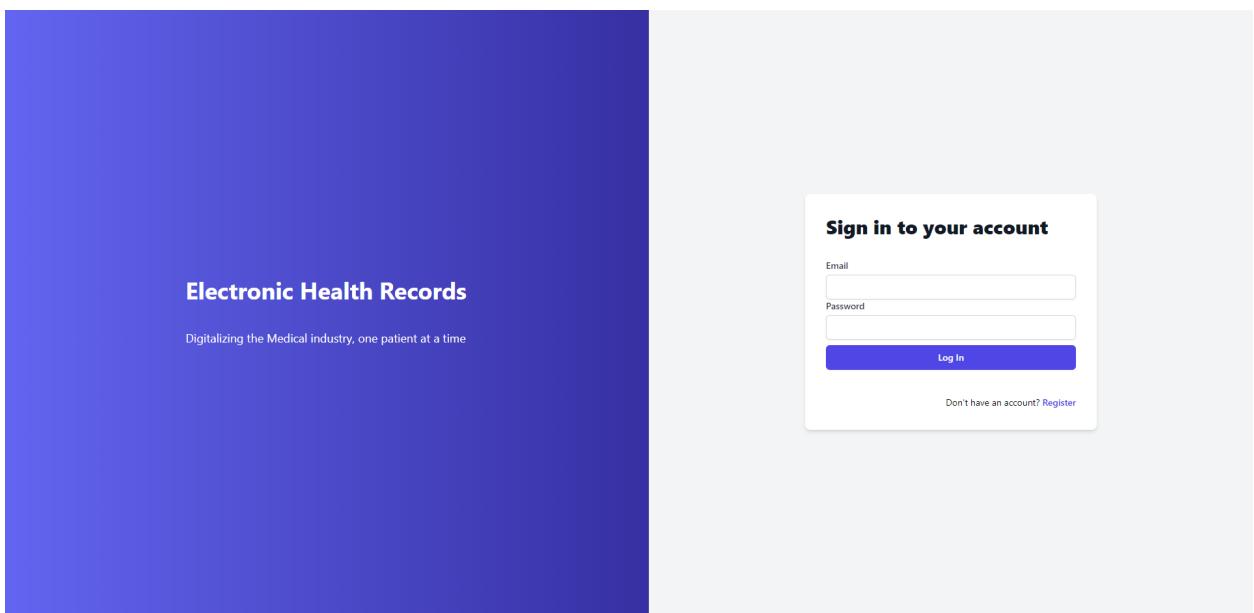
SQL Statement:

```
//Input MEDICAL REPORT
app.post('/api/register_report', (req, res) => {
  console.log(req.body);
  const patient_id = req.body.patient_id
  const visit_date = req.body.visit_date
  const visit_reason = req.body.visit_reason
  const diagnosis = req.body.diagnosis
  const test = req.body.test
  const test_result = req.body.test_result
  const temperature = req.body.temperature
  const blood_pressure = req.body.blood_pressure
  const blood_oxygen = req.body.blood_oxygen
  const heart_rate = req.body.heart_rate
  const doctor_id = req.body.doctor_id
  const remarks = req.body.remarks

  const sqlInsert = "INSERT INTO medical_report (patient_id, doctor_id, visit_date, visit_reason, diagnosis, test, test_result, temperature, blood_pressure, blood_oxygen, heart_rate, remarks ) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?"
  db.query(sqlInsert, [patient_id, doctor_id, visit_date, visit_reason, diagnosis, test, test_result, temperature, blood_pressure, blood_oxygen, heart_rate, remarks], (err, result) =>{
    console.log(err);
  })
  db.commit();
});
```

Project:2 (Laravel)

ALL User: Login



All User Login features (Functionality, mySQL commands)

- Checks for user in database, if user doesn't exist, sends "invalid login credentials" error
- Checks for user in database, if user exists but wrong password says "invalid login credentials"

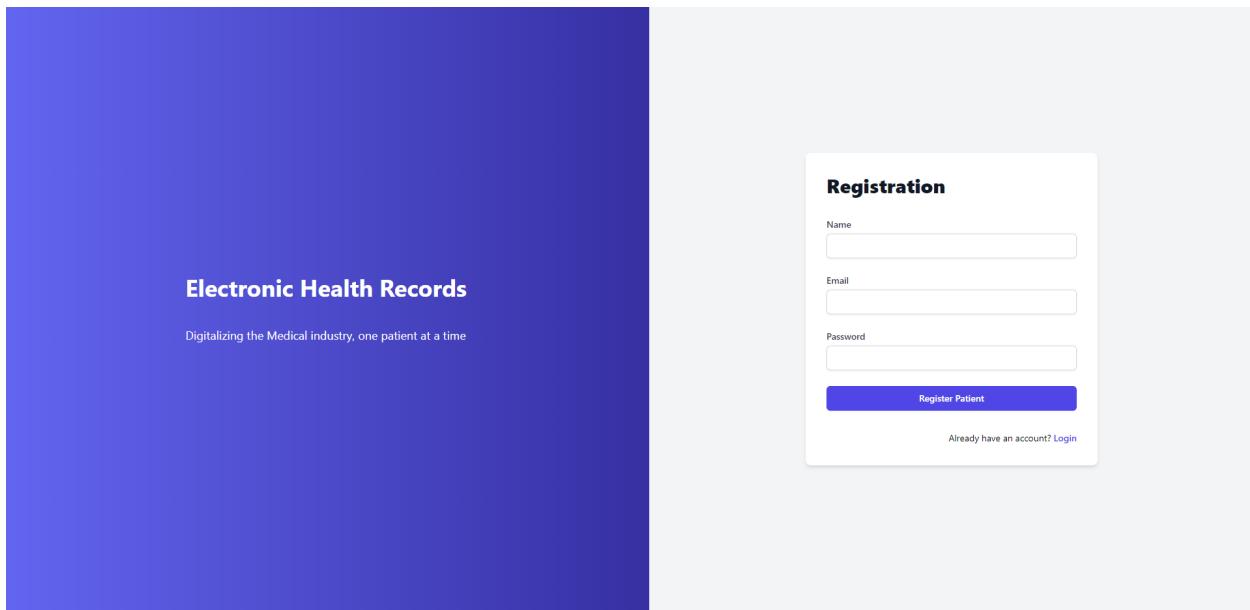
- Checks for user in database, if user exists and types correct password, successfully logs in and generates a session() for future requests

Backend Code:

```

1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use App\Models\User;
7  use Illuminate\Support\Facades\Auth;
8  use Illuminate\Support\Facades\Hash;
9  use App\Providers\RouteServiceProvider;
10
11 class LoginController extends Controller
12 {
13     public function login()
14     {
15         return view ('auth.login');
16     }
17
18     public function login_post(Request $request)
19     {
20         $success = auth()->attempt([
21             'email' => request('email'),
22             'password' => request('password')
23         ]);
24
25         if($success)
26         {
27             $request->session()->regenerate();
28             return redirect()->to(RouteServiceProvider::HOME);
29         }
30         return back()->with('loginError', 'Invalid login details');
31         // return back()->withErrors([
32         //     'email' => 'The Credentials do not match.'
33         // ])->onlyInput('email');
34     }
35 }
```

ALL User: Registration



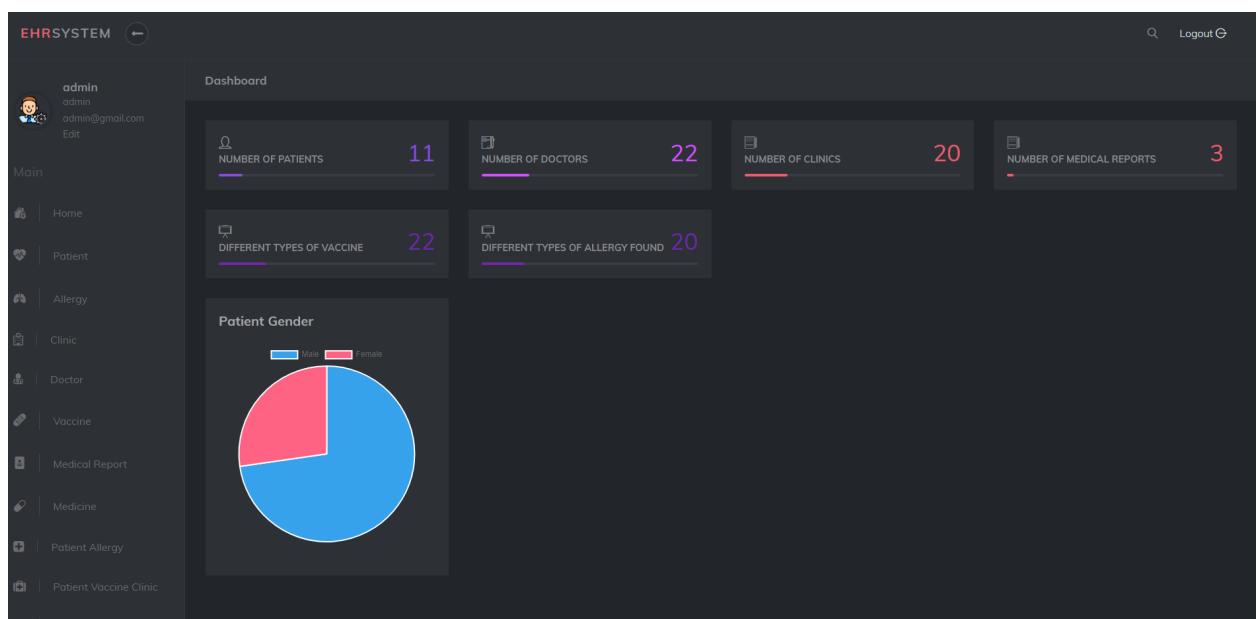
The screenshot shows a registration form titled "Registration". It contains three input fields: "Name", "Email", and "Password", each with a corresponding text input box. Below the password field is a blue button labeled "Register Patient". At the bottom right of the form is a link "Already have an account? [Login](#)". To the left of the registration form is a large blue rectangular area containing the "Electronic Health Records" logo and the tagline "Digitalizing the Medical industry, one patient at a time".

- Ability to register user
- Available roles - admin, patient, doctor
- During registration, searches for duplicate email, if duplicate found, shows an error 'user already exists'.

Backend Code:

```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use App\Models\User;
7 use Illuminate\Support\Facades\Auth;
8 use Illuminate\Support\Facades\Hash;
9 use App\Providers\RouteServiceProvider;
10
11 class RegisterController extends Controller
12 {
13     public function register()
14     {
15         return view ('auth.register');
16     }
17
18     public function register_post()
19     {
20         $user = User::create([
21             'name' => request('name'),
22             'email' => request('email'),
23             'password' => Hash::make(request('password'))
24         ]);
25
26         request()->validate([
27             'name' => ['required', 'string' , 'max:255'],
28             'email' => ['required', 'email', 'max:255'],
29             'password' => ['required', 'string', 'min:8', 'confirmed']
30         ]);
31
32         Auth::login($user);
33
34         return redirect()->to(RouteServiceProvider::HOME);
35     }
36 }
37
```

Admin Dashboard: Overview



Code of PIECHART:

```
<div class="col-lg-3">
    <div class="pie-chart chart block">
        <div class="title"><strong>Patient Gender</strong></div>
        <div class="pie-chart chart margin-bottom-sm">
            <canvas id="piePatientGender"></canvas>
        </div>
    </div>
</div>

<?php
    $patientmale = DB::table('patients')->where('gender', 'male')->count();
    $patientfemale = DB::table('patients')->where('gender', 'female')->count();
?>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script>
    var ctx = document.getElementById('piePatientGender').getContext('2d');
    var piePatientGender = new Chart(ctx, {
        type: 'pie',
        data: {
            labels: ['Male', 'Female'],
            datasets: [{
                data: [<?php echo $patientmale; ?>, <?php echo $patientfemale; ?>],
                backgroundColor: ['#36a2eb', '#ff6384']
            }]
        },
        options: {
            responsive: true,
            maintainAspectRatio: false
        }
    });
</script>
```

Admin Dashboard: Patient Panel

The screenshot shows the Admin Dashboard with the 'Patient' panel selected. On the left, there's a sidebar with icons for Home, Patient, Allergy, Clinic, Doctor, Vaccine, Medical Report, Medicine, Patient Allergy, Patient Vaccine Clinic, and Prescription. The main area has a header 'Patient Info' with a 'Logout' button. Below is a table with 19 rows of patient data. The columns are: #, First Name, Last Name, NID, Phone, Email, Address, Date Of Birth, Gender, Blood Group, and Action (Edit/Delete). The data includes various names like Javed Hoeger, Tyler Little, Bernadine Steuber, Modesta Brokus, Nedra Simonis, Sommie Hill, Korey Ankunding, Mariah Pacocha, and their respective details.

#	First Name	Last Name	NID	Phone	Email	Address	Date Of Birth	Gender	Blood Group	Action
11	Javed	Hoeger	47	123456789	hudson.emmanuel@nienow.com	5275 Antwon Center Suite 001 Jacynthstad, ID 20201	2000-01-01	male	AB+	Edit/Delete
12	Tyler	Little	225804329	123456789	christiansen.catalina@schaden.biz	471 Friesen Harbor Apt. 552 Lake Pasquale, UT 44429-8691	2000-01-01	female	A+	Edit/Delete
13	Bernadine	Steuber	25	123456789	ihammes@hill.net	6229 Doris Prairie Apt. 144 East Mauricio, OK 20079	2000-01-01	male	O+	Edit/Delete
14	Modesta	Brokus	3258	123456789	price.taya@stamm.com	497 Bayer Lane Boyletown, MO 82065	2000-01-01	male	AB-	Edit/Delete
15	Nedra	Simonis	379451	123456789	oro49@pfeffer.org	9320 Albert Island Apt. 063 East Keyon, DE 83036-2261	2000-01-01	male	O+	Edit/Delete
16	Sommie	Hill	17783094	123456789	phomenick@torphy.net	7638 Schiller Crossroad Mikemouth, HI 76195	2000-01-01	female	B-	Edit/Delete
17	Korey	Ankunding	139220	123456789	eleanore20@funk.net	482 Callie Mews Port Arely, MA 61729	2000-01-01	male	O-	Edit/Delete
19	Mariah	Pacocha	57566605	123456789	patient@gmail.com	100 Weimann Roads Apt. 907 Emelyland, VA 48925-2639	2000-01-01	male	A+	Edit/Delete

Patient Panel Features

- Display a minimalist table which shows all patient data.
- Data include, patient name, gender, date of birth, address, blood group and phone number.
- Ability to add new patients via the “add new patient” button, which would navigate the user to a form which they have till fill up to add a new patient into the database.

Backend Code to view data:

```
public function patient()
{
    $patient = Patient::all();
    return view('admin.pages.patient', compact('patient'));
}
```

Admin Dashboard: Doctor Panel

The screenshot shows the Admin Dashboard with the title "EHR SYSTEM". On the left, there is a sidebar titled "Main" with various navigation options: Home, Patient, Allergy, Clinic, Doctor, Vaccine, Medical Report, Medicine, Patient Allergy, and Patient Vaccine Clinic. The "Doctor" option is selected. The main content area is titled "Doctor List" and contains a table with 12 rows of doctor data. The columns are: #, Center ID, Doctor Name, Contact, Email, Date Of Birth, License Number, Years of Experience, and Action. Each row includes an "Edit/ Delete" link. The data in the table is as follows:

#	Center ID	Doctor Name	Contact	Email	Date Of Birth	License Number	Years of Experience	Action
1	6	Cristina Tillman DVM	8	tillman@gmail.com	1970-05-20	76100310	45	Edit/ Delete
2	8	Misty Stamm	95609662	doctor@gmail.com	2002-01-10	183025	329	Edit/ Delete
3	6	Gayle Stark	44	alexane.zulau@example.net	1987-07-10	1570298	905273138	Edit/ Delete
4	4	Dr. Justyn Graham PhD	25137	ojohnson@example.com	1985-05-02	351923334	544170030	Edit/ Delete
5	9	Aubrey Kemmer DDS	8	sawayn.dave@example.org	1978-12-07	10352	49211	Edit/ Delete
6	1	Sim Russel V	67	hassie86@example.com	1998-10-05	3	691	Edit/ Delete
7	3	Furman Howell	822365	tremblay.eugene@example.com	2000-01-21	5604	726617	Edit/ Delete
8	2	Keira Ward Sr.	23072130	dedric34@example.com	1978-12-08	122640418	6	Edit/ Delete
9	1	Hollis Bernier	30383883	schumm.cynthia@example.net	1973-10-06	90467910	319	Edit/ Delete
10	8	Matteo Gusikowski	626433	vhills@example.net	1992-06-29	8874	9951	Edit/ Delete
11	4	Glenda Klein III	9164	fgleichner@example.net	1971-12-11	7	788041022	Edit/ Delete
12	7	Dr. Laverna Lowe	2	trinity99@example.org	2021-05-30	70462002	266	Edit/ Delete

Features

- Shows all doctors
- Can register new doctor

Backend to view data:

```
public function doctor()
{
    $doctor = Doctor::all();
    $clinic = Clinic::all();
    $cl = Clinic::all();
    return view('admin.pages.doctor', compact('clinic', 'doctor', 'cl'));
}
```

Admin Dashboard: Medical Report Panel

#	ID	Patient Name	Doctor ID	Doctor Name	Visit Reason	Visit Date	Diagnosis	Tests	Test Result	Temperature	Blood Pressure	Heart Rate	Blood Oxygen	Remarks	Action
1	19	Pacocha	16	Erwin Dore	Dairrhoea	2023-12-01	Push up	Xray	condition of dysentery	98.4	100	100	98	Will die soon	Edit / Delete
2	12	Little	2	Misty Stomm	sadss	2021-10-06	sodss	sadss	asfasd	45	64	75	30	fgdg	Edit / Delete
3	11	Hoeger	2	Misty Stomm	headblast	2024-01-07	rc cola	mid	very gd	90	90	90	90	nice	Edit / Delete

Patient Panel Features

- Display a minimalist table which shows all medical report data. Data include, patient name, doctor name, appointment date, appointment reason, diagnosis, lab results, lab test, prescription, remarks.
- Ability to add a new medical report via the “add new medical report” button, which would navigate the user to a form which they have till fill up to add a new medical report into the database

Backend to view data:

```
public function medical_report()
{
    $patient = Patient::all();
    $pt = Patient::all();
    $doctor = Doctor::all();
    $doc = Doctor::all();
    $medicalreport = DB::table('medical_reports')
        ->join('patients','medical_reports.patient_id','=','patients.id')
        ->join('doctors','medical_reports.doctor_id','=','doctors.id')
        ->select('medical_reports.*','patients.last_name','doctors.doctor_name','doctors.email','patients.email')
        ->get();
    return view('admin.pages.medicalreport', compact('patient', 'doctor', 'medicalreport', 'pt', 'doc'));
}
```

Admin Dashboard: In-depth Medical Report

The screenshot displays the Admin Dashboard of the EHR System. On the left, a sidebar menu lists various modules: Main, Home, Allergy, Clinic, Doctor, Vaccine, Medical Report, Medicine, Patient Allergy, Patient Vaccine Clinic, and Prescription. The 'Medical Report' option is currently selected. The main content area is divided into three sections: 'PATIENT DETAILS' (ID:19, Name:patient), 'ALLERGIES' (impedit, cumque, saepe), and 'PATIENT DETAILS' (patient). Below these sections, a table titled 'Medical Reports' lists three entries:

Report ID	Patient Name	Doctor Name	Visit Date	Visit Reason	Action
1	Pacocha	Erwin Dare	2023-12-01	Dairrhoea	Download
2	Little	Misty Stamm	2021-10-06	sadss	Download
3	Hoeger	Misty Stamm	2024-01-07	headblast	Download

User Profile Feature:

- Shows individual patient details
- Show patient allergy details
- Shows patient vaccine details
- Can add allergy to patient
- Can add vaccine to patient
- Patient based medical report list
- Can add patient based medical report

Backend to view data:

```
public function patientdata($id)
{
    $pd = DB::table('patients')
        ->join('patient_allergies', 'patients.id', '=', 'patient_allergies.patient_id')
        ->join('allergies', 'patient_allergies.allergy_id', '=', 'allergies.id')
        ->select('patients.*', 'patient_allergies.*', 'allergies.allergynname', 'allergies.type')
        ->where('patients.id', '=', $id)
        ->get();

    $pd1 = $pd;

    $mlr = DB::table('medical_reports')
        ->join('patients', 'medical_reports.patient_id', '=', 'patients.id')
        ->join('doctors', 'medical_reports.doctor_id', '=', 'doctors.id')
        ->select('medical_reports.*', 'patients.last_name', 'doctors.doctor_name', 'doctors.email', 'patients.email')
        ->get();

    return view('admin.pages.patientdata', compact('mlr', 'pd', 'pd1'));
}
```

Allergy Page

#	Allergy Name	Type	Description	Action
1	sol	alia	Dolore corrupti consequatur vitae dolore omnis ut ut. Aut voluptatem velit officia molestias ipsam accusantium voluptas. Asperiores inventore aliquam ut id animi.	
2	non	explicabo	Eum quam unde quia nemo. Ilo qui dolorem molestiae sunt ex. Minus nam quo quidem accusantium.	
3	impedit	et	Est ea voluptate iste enim. Unde in molestiae dolores eligendi labore. Quisquam corporis consequatur iure iure nostrum aut. Dolores quaerat dolorum vitae otque error.	
4	veritatis	exercitationem	Sope dolore aspernatur sit beatiae nostrum architecto. Nihil nesciunt consequatur repudiandae unde voluptatem eum. Voluptatem autem iusto et nostrum dolorem est odit. Ex omnis et aut molestias quos.	
5	autem	id	Alias aut enim beatiae quod quibusdam. Nobis aut molestiae reprehenderit autem atque. Maxime autem facere rerum expedita rerum culpa id. Itaque impedit velit fuga dicta.	
6	cumque	doloribus	Ut sed exercitationem vel fuga qui vitae. Natus beatiae totam in reiciendis asperiores.	
7	qui	ut	Aliquid enim laboriosam et omnis nostrum ut distinctio. Praesentium optio iure non magni eum unde fugiat. Velit voluptatem sint operiam qui aliquid non. Aspernatur et corporis error iaudantium non.	
8	commodi	eligendi	Deleniti non dicta labore nesciunt. Eum ut voluptatem quas est officia. Fugit asperiores rem unde explicabo sunt repellat.	
9	ducimus	molestiae	Illum error soluta voluptas. Quoerat magni quidem repellendus magni. Et nisi totam dicta asperiores earum libero tenetur magnam.	

Features

- Shows allergy information from existing database

Backend to view data:

```
public function allergy()
{
    $allergy = Allergy::all();
    return view('admin.pages.allergy', compact('allergy'));
}
```

Create Allergy Data:

The screenshot shows the EHR System interface. On the left, there's a sidebar with a user profile for 'admin' and links to 'Home', 'Patient', 'Allergy', 'Clinic', 'Doctor', 'Vaccine', 'Medical Report', 'Medicine', 'Patient Allergy', and 'Patient Vaccine Clinic'. The 'Allergy' link is selected. In the center, there's a table titled 'Allergy' with columns: #, Name, Type, and Description. The table contains 9 rows of data. A modal window titled 'Add allergy data' is open over the table. It has three input fields: 'Allergy Name' (placeholder 'Write Allergy Name'), 'Type' (placeholder 'Write Allergy Type'), and 'Description' (placeholder 'Write description details'). Below these fields are a 'Submit' button and a 'Close' button. The background table also has a 'Action' column with various icons.

#	Name	Type	Description	Action
1	sol	alia	Dolore cor animi.	... molestias ipsam accusantium voluptas. Asperiores inventore aliquam ut id quidem accusantium.
2	non	explicabo	Eum quoniam et atque error	corporis consequatur iure labore nostrum aut. Dolores quaerat dolorum vitae
3	impedit	et	Est eo volu tate dolorem est odit. Ex omnis et aut molestias quos.	... dolorem est odit. Ex omnis et aut molestias quos.
4	veritatis	exercitationem	Saepe dolore aspernatur sit beatae nostrum architecto. Nihil nesciunt consequatur repudiandae unde voluptatem eum. Voluptatem autem iusto et nostrum dolorem est odit. Ex omnis et aut molestias quos.	... a molestias ipsam accusantium voluptas. Asperiores inventore aliquam ut id quidem accusantium.
5	autem	id	Alias aut enim beatae quod quibusdam. Nobis aut molestiae reprehenderit autem atque. Maxime autem facere rerum expedita rerum culpa id. Itaque impedit velit fuga dicta.	... corporis consequatur iure labore nostrum aut. Dolores quaerat dolorum vitae
6	cumque	doloribus	Ut sed exercitationem vel fuga qui vitae. Natus beatae totam in reiciendis asperiores.	... a molestias ipsam accusantium voluptas. Asperiores inventore aliquam ut id quidem accusantium.
7	qui	ut	Aliquid enim laboriosam et omnis nostrum ut distinctio. Praesentium optio iure non magni eum unde fugiat. Velit voluptatem sint operiorum qui aliquid non. Aspernatur et corporis error laudantium non.	... corporis consequatur iure labore nostrum aut. Dolores quaerat dolorum vitae
8	commodi	eligendi	Deleniti non dicta labore nesciunt. Eum ut voluptatem quis est officia. Fugit asperiores rem unde explicabo sunt repellat.	... a molestias ipsam accusantium voluptas. Asperiores inventore aliquam ut id quidem accusantium.
9	ducimus	molestiae	Illum error soluta voluptas. Quaderat magni quidem repellendus magni. Et nisi totam dicta asperiores earum libero tenetur magnam.	... a molestias ipsam accusantium voluptas. Asperiores inventore aliquam ut id quidem accusantium.

Features

- Can add more Allergy Data.

Backend:

```
public function allergy_submit(Request $request)
{
    $allergy = new Allergy();

    $allergy->allergyname = $request->name;
    $allergy->type = $request->type;
    $allergy->description = $request->description;

    $allergy->save();

    return redirect()->back();
}
```

RESULT ANALYSIS

During our research for EHR webapps, we learned a lot about how the electronic health record systems work. We learned how patient-doctor interaction works. When we started to design the web application, we got to learn more in depth, the database concepts like ERD, normalization, schema and data dictionary. When we got to the programming, we learned authentication and CRUD operations and SQL queries. We had many challenges on the way. Overall it was a very good learning experience.

CONCLUSION & FUTURE WORK

- Integration of the EHR webapp directly with hospitals and clinics where these bodies can directly pull data from the server. We can use the NID of the patients to uniquely identify users across medical institutes.
- Integration with laboratories and pharmacies to check for patient prescription and tests and update them from individual platforms
- Enhanced UX/UI experience for all stakeholders
- Integration of Advanced Analytics, real time analytics to show data on epidemic/pandemic
- Development of a mobile application

- Enhance security