TEXAS TECH UNIVERSITY

Lubbock, Texas

CS 5356-001

Advanced Database Management System

Fall 2023

ERSTREAMLINE

<u>Submitted by</u>

Sumit Shrestha

R11836842

<u>Submitted on</u>

December 13th, 2023

Table of Contents

Introduction *3*

Project Overview 3

Team Composition 3

Project Description *4*

Database Overview 4

Technical Specifications 6

Challenges and Solutions 7

Personal Contribution *7*

Role and Responsibilities 7

Work Process 9

Technical Implementation 10

Reflection *11*

Learning Outcomes 11

Teamwork and Collaboration 12

Challenges Faced 12

Skills Developed 12

Future Application 13

Conclusion *13*

Appendices *14*

References *26*

Introduction

Project Overview

An Emergency Room (ER) is an essential component of a community’s health which must run smoothly and have a robust infrastructure to provide quick patient care in an emergency. Nowadays real-time databases are crucial for smooth operation and community well-being in today's busy environment. In emergency rooms, overcrowding frequently results in delays in patient care. The health of the patients is therefore put at risk as emergency response times grow less efficient. There is a challenge to make an effective management of patient data, records, and medical histories in emergency rooms (ERs), especially in high-stress scenarios that we saw in COVID-19 where timely information access is essential and compliance with healthcare regulations such as the Health Insurance Portability and Accountability Act (HIPAA) requires accountability.

ERStreamline, our specialized database system, which surpasses traditional methods in its efficient data retrieval and compliance with healthcare regulations, addresses the challenges faced by emergency rooms by providing real-time data access with strong security and scalability in a critical healthcare setting. It is designed to streamline the operation of an Emergency Room which enables administrative staff to create and manage administrative staff, infrastructure, patient records billing information, Health Staff to create, manage and access the patient visit records like diagnosis, medication, lab reports, and Patients to access their visit records, keep track of medications, lab reports and billing statement.

Team Composition

We focus on individual contributions in relation to team makeup throughout the project. This report highlights the individual work, accomplishments, and specialization of, highlighting my individual contributions to the project’s success. The goal of this project is to highlight the skills and qualities that contributes to the project of the database system. Here is the list of team members with their respective roles: Bipin Chhetri (Project Leader/Frontend Developer), Nisseem Rana Magar (Project Manager/ Tester), Raj Dhaubanjar (Documentation

Specialist/Database Developer), Roshan Acharya (Database Architect/Frontend Developer) and **Sumit Shrestha (Query Developer/Backend Developer/ReactJS Developer)**.
This report highlights my specific role, highlighting the impact on project milestones and outcomes. I am the Query Developer/Backend Developer, which is one of the crucial roles in the team for our project ERStreamline. My focus on individual contributions aims to offer a comprehensive perspective that captures the distinctive qualities that make our project success as we navigate throughout this report.

Project Description

Database Overview

The ERStreamline database project is to improve the management of emergency healthcare services, with a focus on providing accurate, fast, and safe healthcare to patients, particularly in urgent situations. It is intended to address the multifarious issues encountered by emergency rooms by offering a robust infrastructure to maximize patient care through sophisticated data management.

Purpose

ERStreamline's main goal is to simplify and increase the efficiency of emergency healthcare services. In the fast-paced setting of an emergency department, when time is of the importance, it seeks to provide healthcare workers with the resources they need to make swift decisions. The system's purpose is based on real-time access to patient records, security measures, scalability, data integrity, disaster recovery.

Scope

Many different functionalities are included in ERStreamline, all of which are designed to specifically address the features or demands in the emergency healthcare ecosystem to make the emergency room operate well. To fulfill its objective, the project tackles the following crucial areas:

a) Real Time Data Access

   Real-time access to test results, medical histories, and patient records is functioned well in ERStreamline.

b) Data Security and Compliance

   Ensuring compliance with healthcare data privacy requirements (such as HIPAA) and safeguarding patient information requires the use of strong security measures. This covers encryption, audit trails, and role-based access controls.

c) Scalability

   In an emergency or other public health crisis, the system needs to be scalable to handle large amounts of patient data.

d) Data Integrity

   To avoid errors in patient treatment, ensure data correctness and consistency. Apply data validation rules and limitations.

On the other side, ERStreamline is intended to help healthcare workers streamline their work, but patients can also use it to check their test results and medicines. To maintain data security compliance and system usability, user roles and access privileges should be implemented. They are described in detail below:

- Administrative Staff

  These individuals manage non-clinical aspects of the ER. Their needs include:

  - Managing patient admission, discharge processes, billing and insurance.
  - Scheduling appointments and resources.
- Healthcare Staff

  This category includes doctors, nurses, and medical staff. Their need include:

  - Access historical patient data to monitor progress and make informed decisions.
  - Access privilege to view and modify patient records within their department or assigned patients.
- Patients

  This category includes the patients and their family member which include:

  - Access to billing statements, prescribed medications, results and reports.

All the flowcharts of the user roles are shown in Appendices.

Technical Specifications

ERStreamline ensures excellent speed, security, and scalability by utilizing state-of-the-art technology and adhering to best practices in database administration and data modeling.

Following technologies were used during the development of the project ERStreamline:

BackEnd - MongoDB, NodeJS, ExpressJS, Postman

FrontEnd – ReactJS, HTML, CSS, Tailwind

Diagrams – Draw.io, Lucid

Collaboration – GitHub, LaTeX, Teams

Among the above listed Technologies, I have used technologies for Backend Development (MongoDB, NodeJS, ExpressJS, Postman), and Frontend Development (ReactJS).

For the dataset, we took data from Mockaroo. To ensure comprehensive testing, we created a minimum of 30 unique datasets for each 15 relations.

The following describes the ERStreamline system's technical specifications:

- Database Management System (DBMS)

  We used MongoDB, a NoSQL database system for JSON documents known for its schema-less, document-oriented structure, which enables efficient handling of unstructured data with benefits such as high speed, scalability, and dynamic schema adaptation.

- Data Modeling Techniques

  To create and organize database, we have modeled Entity-Relationship (ER) showing various interactions between entities, including patients, healthcare staffs, administrative staffs.

  ER Diagram is shown in Appendices.

- Data Integrity

  Applies restrictions and guidelines for data validation to provide reliable and consistent healthcare data.

- Scalability

  Cloud-based technologies are integrated for elastic scalability, and modular architecture is used for horizontal scaling.

- Security Measures

Implements encryption, Role-Based Access Controls to ensure data security and HIPAA compliance.

Challenges and Solutions

a) Real-time Data Access:
Challenge: Latency in data access.
Solution: Optimized indexing, caching, and load balancing.

b) Data Security and Compliance:
Challenge: HIPAA compliance for patient data.
Solution: Password Encryption, Restrict Access, and Role based Access Control for Patient, Healthcare Staffs and Administrative Staffs.

c) Scalability:
Challenge: Handling high volumes of patient data.
Solution: Modular architecture, cloud integration for scalability using MongoDB.

d) Data Integrity:
Challenge: Maintaining accuracy in complex healthcare data.
Solution: Data validation rules, regular checks.

Personal Contribution

I will explain all my contribution which includes Role and Responsibilities, Work Process, and Technical Implementation in detail below.

Role and Responsibilities

My role was multiple for this project since the project is huge and we had limited time. I worked as a Query Developer/Backend Developer/ReactJS Developer. My primary focus was on developing query using MongoDB database and the server-side development using NodeJs and ExpressJs. After we finalized database design, optimization, and normalization, I was responsible for handling the database with user input or fetch data from database. Since, we have used NoSQL based database, so mostly the data that I need must be in (JavaScript Object Notation) JSON format. As a Query Developer on the ERStreamline project, my primary focus was on the MongoDB database and the server-side development using Node.js and Express.js. I was responsible for a variety of tasks that helped successfully complete our project ERStreamline. Here is the list of my responsibilities: -

a) Data Integration and Migration
   - Implemented data integration techniques to improve the flow of information within the system from the cloud.

b) Query Optimization
   - Performed performance study on database queries and improved them for efficiency.

c) NodeJS and ExpressJS Development
   - Developed server-side APIs using Node.js and Express.js.
   - Created RESTful endpoints to provide seamless communication between the frontend and the MongoDB database.

d) ReactJS Development
   - Built reusable components, interactive and dynamic UI making it easier to manage and update the frontend.
   - Integrated react components with the Node.js backend and MongoDB database.

e) API Design
   - Developed API endpoints, request-response formats, and authentication processes for integration convenience.

f) Data Security and Access Control
   - Defined and implemented access control policies to prevent unauthorized access to vital data as we have role-based data access and page access for Patient, Health Staff and Administrative Staff.

Work Process

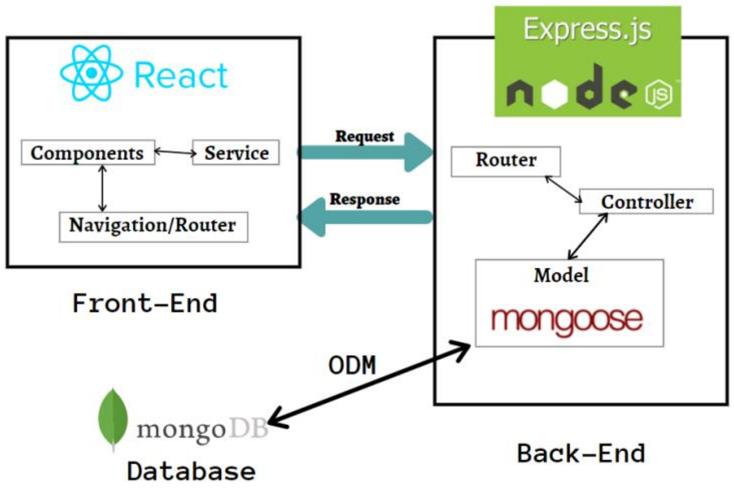


Figure 1: My workflow of ERStreamline[1]

The above diagram shows my work process in this project. As a Query Developer/Back-End Developer and ReactJS Developer, I must make sure that the connection between each component is valid. I used NodeJS, a server-side JavaScript runtime environment to run Javascript code on the server, to run ExpressJS server, which handles server-side logic and communication with mongoDB database. MongoDB is a NoSQL database for storing data in a flexible, JSON-like BSON (Binary JSON) that uses collections to store documents. Additionally, ExpressJS is a web-application framework for NodeJS that makes it easier to build robust and scalable server-side applications. It also handles HTTP requests and responses, routing, middleware, and other server-side functionalities. Finally, it aids in the creation of RESTful APIs, which makes it simple to define routes and interact with the databases.
The workflow inside the application works as follow:

a) **Client HTTP Request**: The user makes a HTTP request by interacting with the React.js frontend.
b) **ExpressJS Routes**: After receiving the request, Express.js routes the request and calls middleware.
c) **Database Interaction (MongoDB):** Express.js with the help of NodeJS interacts with MongoDB to carry out CRUD tasks.
d) **Server Processes Request:** Server-side logic, business functions, and data retrieval are handled using ExpressJS.

e) **HTTP Response to Client:** ReactJS receives a HTTP response from the server that frequently includes database data.

f) **ReactJS Update UI:** React.js uses the data it receives to dynamically refresh the user interface.

As users communicate, this cycle is repeated, resulting in a smooth data flow between the client, server, and database.

Technical Implementation

For Backend development, I used following dependencies:

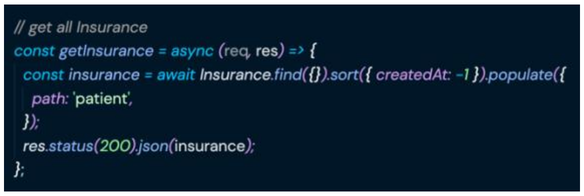
- bcryptjs: used for securing password storage library using bcrypt to hash passwords. [2]
- concurrently: used to launch the frontend and backend servers simultaneously. [3]
- cors: used for managing Cross-Origin Resource Sharing (CORS), which permits or prohibits access from a different domain to resources on a web server.
- dotenv: used to load environment variables into process.env from a.env file, making configuration setting administration easier.
- express: used to create web apps and APIs and request handling, middleware, and routing.
- jsonwebtoken: used for creation and verification of JSON Web Tokens (JWTs) for user authentication and authorization. [4]
- mongoose: used to connect MongoDB and Node.js that provides a schema-based interface to MongoDB databases.
- multer: used for managing multipart/form-data and handle file uploads (lab report). [5]
- nodemon: used to monitor file changes and restarts the Node.js server automatically. [6]

For Frontend development, I used following dependencies:

- axios: used to make asynchronous HTTP request/queries. [7]
- moment: used to parse, manipulate, and format dates. [8]
- react-router-dom: used to navigate between multiple views based on the URL.
- react-scripts: used to simplify development activities by creating a server and building.
- swiper: used to responsive slide-based interfaces. [9]

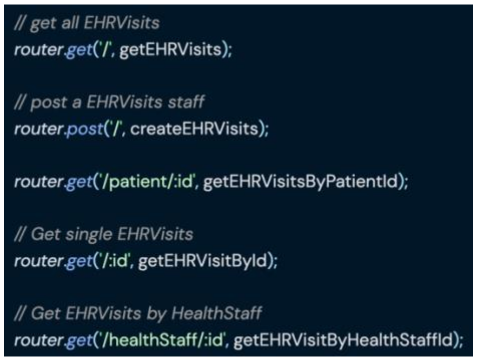
For query optimization, I focused on constructing indexes on columns that are commonly used in our queries, allowing MongoDB to discover and get data more effectively. In addition, I created covered queries in which all the fields in the query are part of an index. Also, I used sort () methods to restrict the use of cursors, particularly for read-intensive processes. These techniques contributed to a more responsive MongoDB database, assuring the performance for our application.

Some of the code snippets are shown below:



```
// get all Insurance
const getInsurance = async (req, res) => {
  const insurance = await Insurance.find({}).sort({ createdAt: -1 }).populate({
    path: 'patient',
  });
  res.status(200).json(insurance);
};
```

Figure 2: Query to show all insurance of a patient



```
// get all EHRVisits
router.get('/', getEHRVisits);

// post a EHRVisits staff
router.post('/', createEHRVisits);

router.get('/patient/:id', getEHRVisitsByPatientId);

// Get single EHRVisits
router.get('/:id', getEHRVisitById);

// Get EHRVisits by HealthStaff
router.get('/healthStaff/:id', getEHRVisitByHealthStaffId);
```

Figure 3: Route for creating different APIs.

More code snippets are shown in Appendices.

Reflection

Learning Outcomes

I have learned a lot throughout this project. I not only learned new concepts, technologies but also how to tackle in every technical and problem-solving situation. I enhanced my skills and knowledge not only from technical perspective but also personally.

From a technical perspective, this project has been a learning journey, with advanced proficiency in MongoDB, NodeJS, ExpressJS, and ReactJS. I got knowledge of designing efficient schemas, optimizing queries, and ensuring robust data security. Also, the hands-on experience in server-side development using NodeJS and ExpressJS has broadened my skills in building scalable APIs, while working with ReactJS it even enhanced my ability to build responsive and user-friendly frontends with more detail reusable components.

In terms of personal development this project developed my ability to solve problems and overcome obstacles in real time has improved and improved my soft skills. Working in a cross-functional team taught me the value of clear communication and gave me a mindset that demands constant improvement, which is why this project has been a crucial pillar of my career development.

Teamwork and Collaboration

It has been a good experience to work with the ERStreamline team, because of excellent communication and teamwork. We thoroughly communicate with each other using Microsoft Teams, Facebook Messenger, and had a weekly meeting for the updates or any difficulties and used GitHub for code collaboration. Our approach to problem-solving was improved by this lively exchange of ideas and insights, which also created a collaborative atmosphere where each team member's special talents resulted in the project's overall success.

Challenges Faced

During the development process, as my role was crucial, I technically faced a lot of challenges. It was challenging on how I can implement this problem which developed my problem-solving skills, but anyhow got through that with the discussion over the team giving insights by each member. Most challenging part was to insert data for each relation to MongoDB without any validation error. To get around this, we conducted a comprehensive study, collaborated extensively with team members to gather a range of viewpoints, and implemented a staged migration approach that made analysis and development easier down the road.

Skills Developed

I became an advanced MongoDB user, mastering the creation of effective database structures, query optimization, and strong security implementation. My understanding of server-side development has expanded thanks to my practical work with Node.js and Express.js, which has allowed me to build scalable APIs and improve system performance. I improved my frontend development skills by using ReactJS to create responsive, modular user interfaces and reusable components. I got to know and use many more backend and frontend dependencies. My skill set was further expanded by real-time data synchronization mechanisms in MongoDB, which guaranteed smooth updates and consistent data. Due to time constraint for this huge project, I also learned about the time management on how we can wrap a project like this in a favorable time.

Future Application

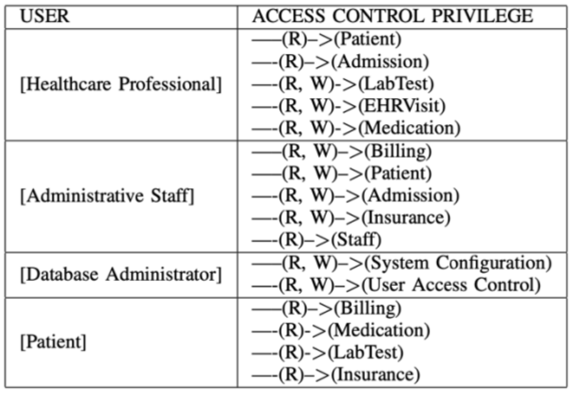
I have a wide range of experiences and abilities from the ERStreamline project that will be very useful for my professional growth and future situations. My expertise in Node.js, Express.js, and MongoDB places me in a strong development position for projects needing server-side programming and strong database administration. This is especially pertinent to the healthcare industry, since the skills developed on this project are applicable to the complexities of emergency room systems. I gained my expertise with ReactJS programming, which provides opportunities for me to contribute to the design of user-friendly, responsive interfaces and reusable components. I can now say that I am a junior full-stack developer because of my work as a Frontend and Backend developer in this project. Also, it has developed me to think outside the box for a problem and giving me ways to tackle any kind of problem in future.

Conclusion

ERStreamline project has been a great learning opportunity that has greatly expanded my knowledge of database systems and enhanced my skill set. I improved my knowledge of database design, query optimization, and server-side development by using MongoDB, NodeJS, and ExpressJS in real-world applications and enhanced more experience in ReactJS. In addition, the project developed a collaborative mindset by highlighting the value of efficient teamwork, communication, and flexibility in the face of difficulties. This experience has improved my technical skills and given me a greater understanding of the complexities of database systems, setting me up for future endeavors and career advancement in the exciting world of software development.

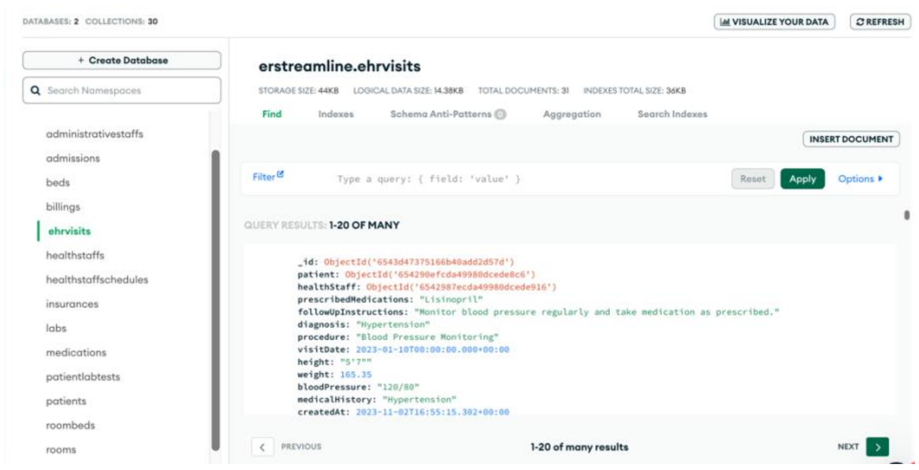
Appendices

The wireframes were created and can be found in detail here.



USER	ACCESS CONTROL PRIVILEGE
[Healthcare Professional]	—–(R)–>(Patient) —-(R)–>(Admission) —-(R, W)->(LabTest) —-(R, W)->(EHRVisit) —-(R, W)->(Medication)
[Administrative Staff]	—–(R, W)–>(Billing) —–(R, W)–>(Patient) —-(R, W)–>(Admission) —-(R, W)–>(Insurance) —-(R)–>(Staff)
[Database Administrator]	—–(R, W)–>(System Configuration) —-(R, W)–>(User Access Control)
[Patient]	—–(R)–>(Billing) —-(R)->(Medication) —-(R)->(LabTest) —-(R)–>(Insurance)

Figure 4: Access control based on User Roles



```
_id: ObjectId('6543d47375166b40add2d57d')
patient: ObjectId('654290efcda49980dcede8c6')
healthStaff: ObjectId('6542987ecda49980dcede916')
prescribedMedications: "Lisinopril"
followUpInstructions: "Monitor blood pressure regularly and take medication as prescribed."
diagnosis: "Hypertension"
procedure: "Blood Pressure Monitoring"
visitDate: 2023-01-10T00:00:00.000+00:00
height: "5'7""
weight: 165.35
bloodPressure: "120/80"
medicalHistory: "Hypertension"
createdAt: 2023-11-02T16:55:15.302+00:00
```

Figure 5: MongoDB EHRVisit Collection

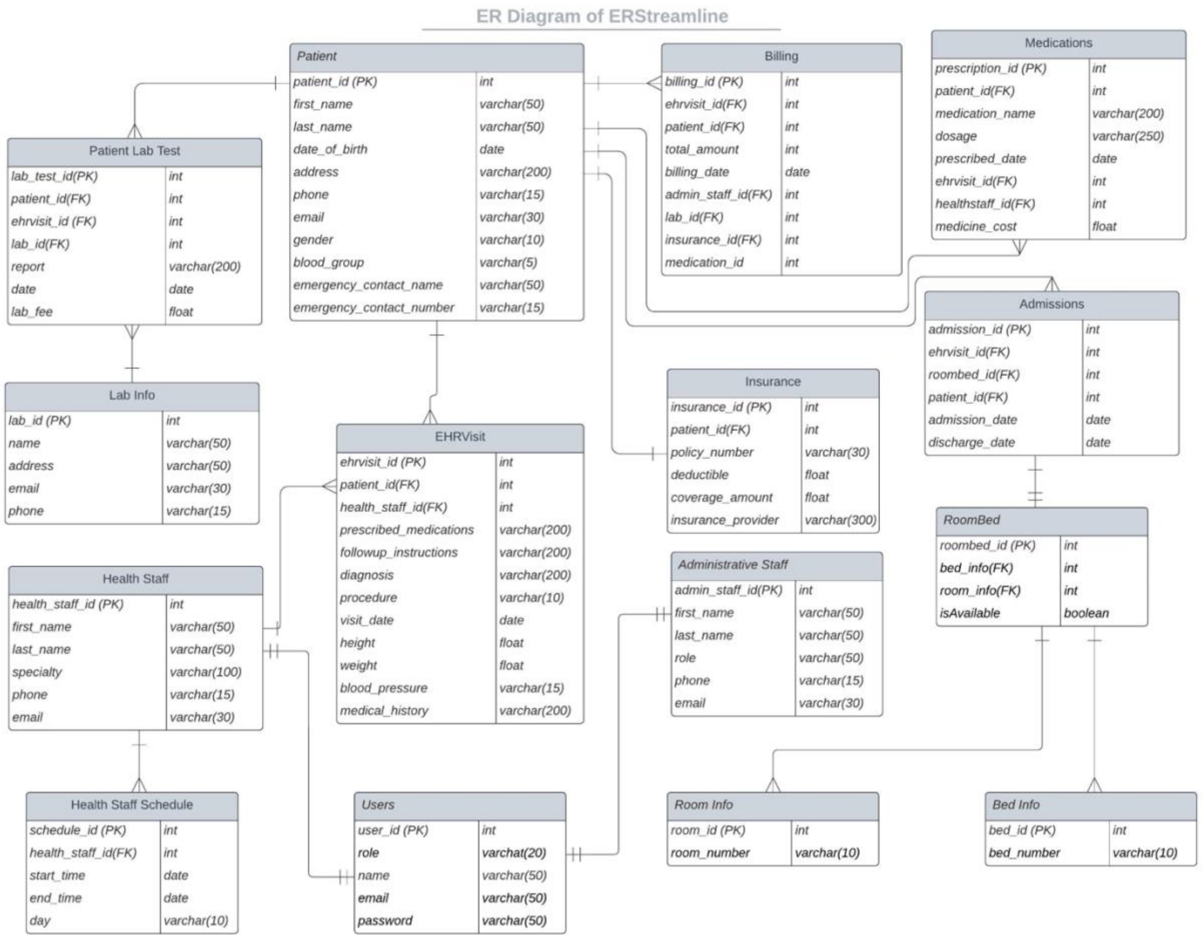


Figure 6: Entity Relationship (ER) Diagram of ERStreamline

Figure 7: Logo of ERStreamline

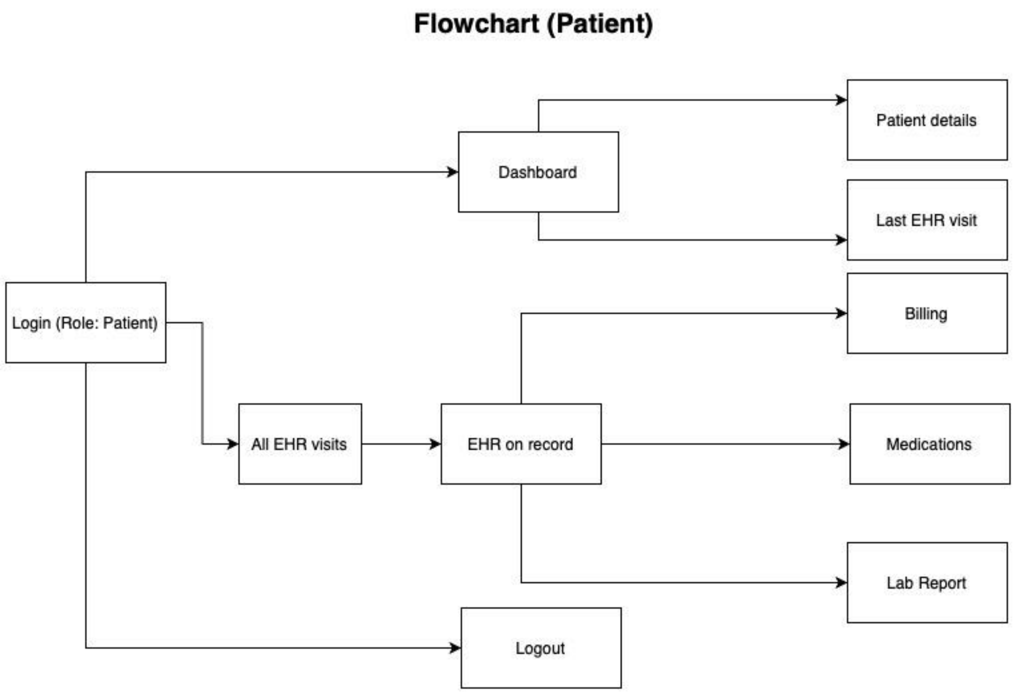


Figure 8: Patient User Flow Diagram

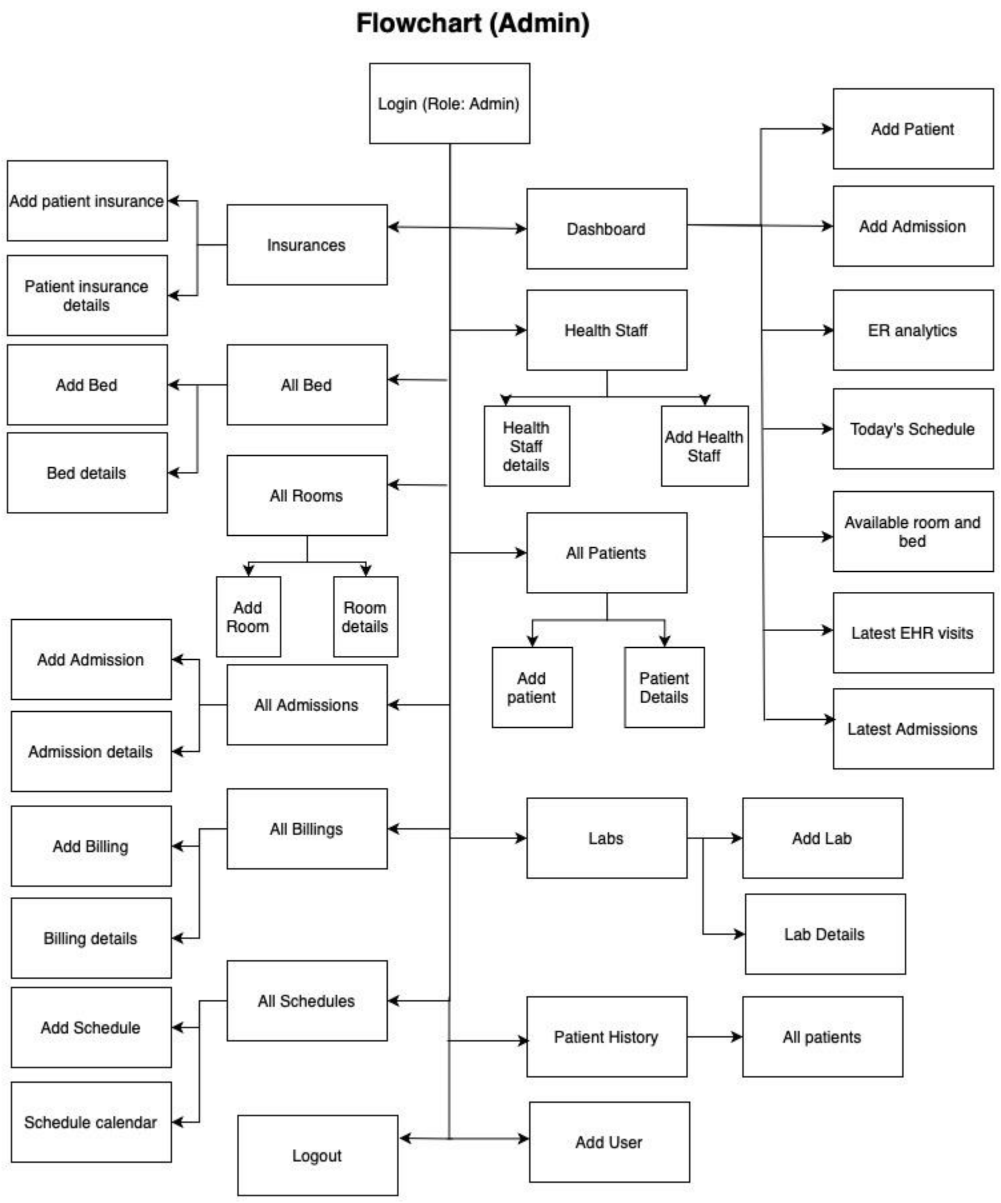


Figure 9: Admin Staff User Flow Diagram

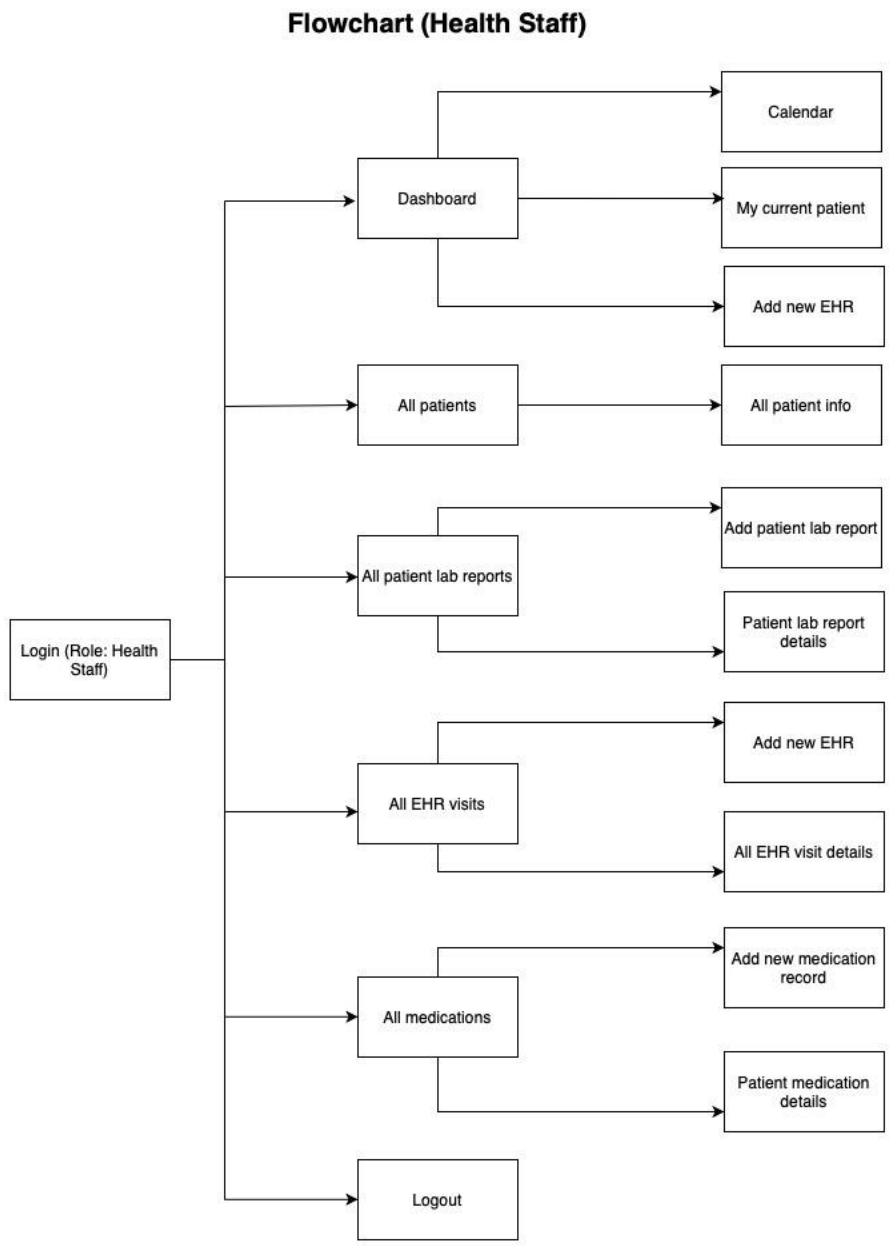
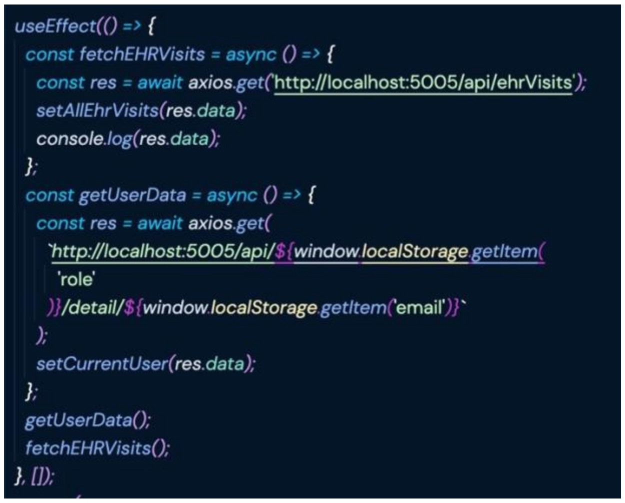
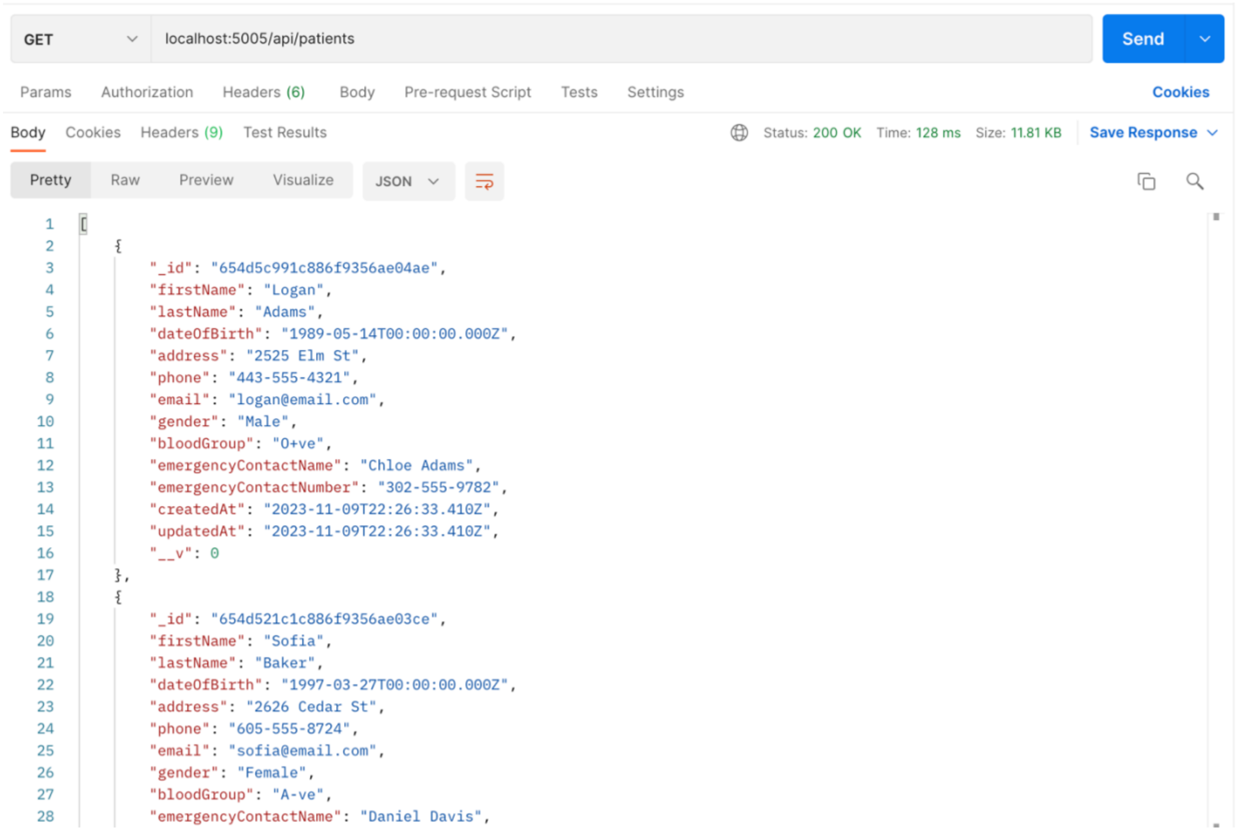


Figure 10: Health Staff User Flow Diagram



```
useEffect(() => {
  const fetchEHRVisits = async () => {
    const res = await axios.get('http://localhost:5005/api/ehrVisits');
    setAllEhrVisits(res.data);
    console.log(res.data);
  };
  const getUserData = async () => {
    const res = await axios.get(
      `http://localhost:5005/api/${window.localStorage.getItem(
        'role'
      )}/detail/${window.localStorage.getItem('email')}`
    );
    setCurrentUser(res.data);
  };
  getUserData();
  fetchEHRVisits();
}, []);
```

Figure 11: Fetching data using ReactJS axios for all EHRVisits

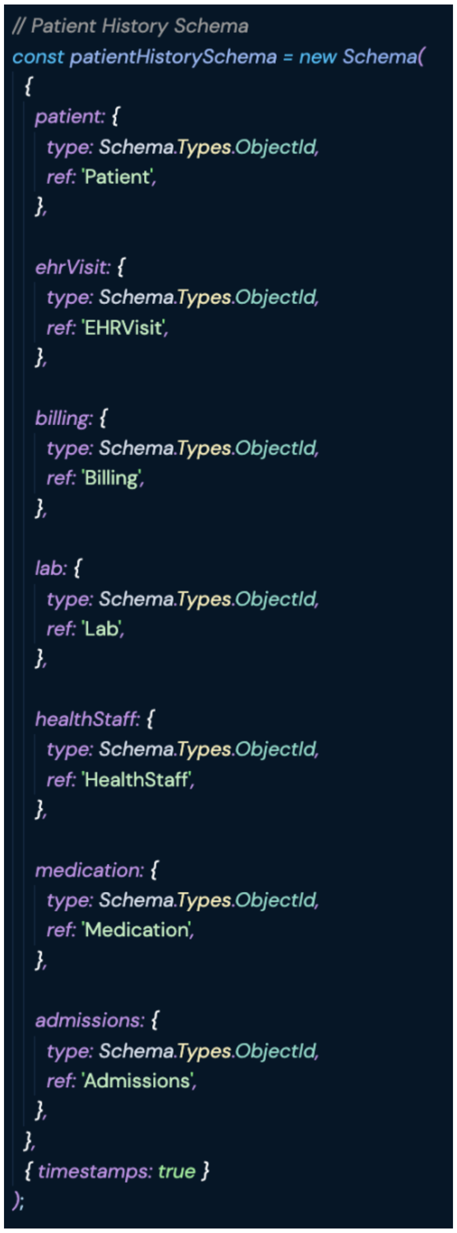


GET localhost:5005/api/patients

Status: 200 OK Time: 128 ms Size: 11.81 KB

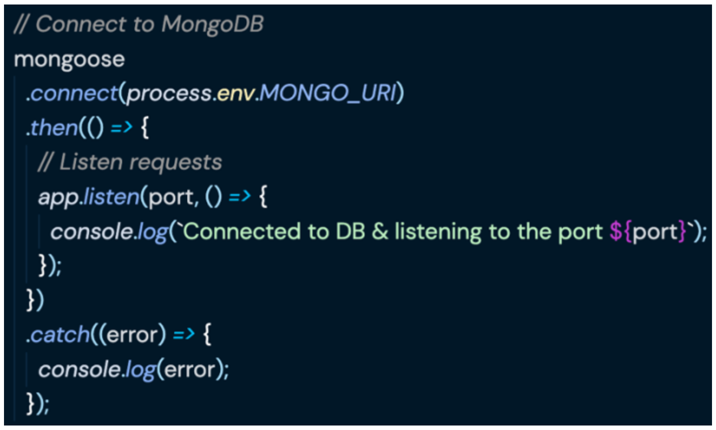
```
[
    {
        "_id": "654d5c991c886f9356ae04ae",
        "firstName": "Logan",
        "lastName": "Adams",
        "dateOfBirth": "1989-05-14T00:00:00.000Z",
        "address": "2525 Elm St",
        "phone": "443-555-4321",
        "email": "logan@email.com",
        "gender": "Male",
        "bloodGroup": "O+ve",
        "emergencyContactName": "Chloe Adams",
        "emergencyContactNumber": "302-555-9782",
        "createdAt": "2023-11-09T22:26:33.410Z",
        "updatedAt": "2023-11-09T22:26:33.410Z",
        "__v": 0
    },
    {
        "_id": "654d521c1c886f9356ae03ce",
        "firstName": "Sofia",
        "lastName": "Baker",
        "dateOfBirth": "1997-03-27T00:00:00.000Z",
        "address": "2626 Cedar St",
        "phone": "605-555-8724",
        "email": "sofia@email.com",
        "gender": "Female",
        "bloodGroup": "A-ve",
        "emergencyContactName": "Daniel Davis",
```

Figure 12: Use of Postman for fetching all patients.



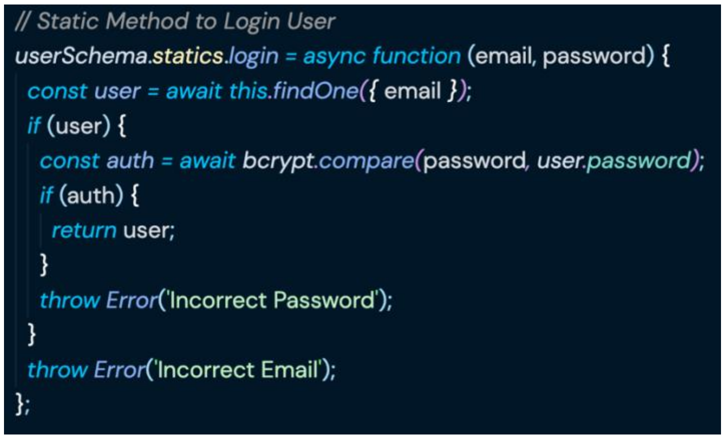
```
// Patient History Schema
const patientHistorySchema = new Schema(
  {
    patient: {
      type: Schema.Types.ObjectId,
      ref: 'Patient',
    },

    ehrVisit: {
      type: Schema.Types.ObjectId,
      ref: 'EHRVisit',
    },

    billing: {
      type: Schema.Types.ObjectId,
      ref: 'Billing',
    },

    lab: {
      type: Schema.Types.ObjectId,
      ref: 'Lab',
    },

    healthStaff: {
      type: Schema.Types.ObjectId,
      ref: 'HealthStaff',
    },

    medication: {
      type: Schema.Types.ObjectId,
      ref: 'Medication',
    },

    admissions: {
      type: Schema.Types.ObjectId,
      ref: 'Admissions',
    },
  },
  { timestamps: true }
);
```

Figure 14: Schema for Patient History in NodeJS.



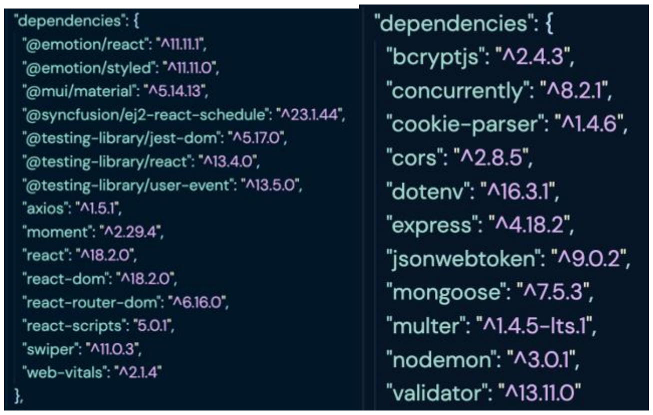
```
// Connect to MongoDB
mongoose
  .connect(process.env.MONGO_URI)
  .then(() => {
    // Listen requests
    app.listen(port, () => {
      console.log(`Connected to DB & listening to the port ${port}`);
    });
  })
  .catch((error) => {
    console.log(error);
  });
```

Figure 13: Connecting NodeJS and MongoDB using Mongoose.



```
// Static Method to Login User
userSchema.statics.login = async function (email, password) {
  const user = await this.findOne({ email });
  if (user) {
    const auth = await bcrypt.compare(password, user.password);
    if (auth) {
      return user;
    }
    throw Error('Incorrect Password');
  }
  throw Error('Incorrect Email');
};
```

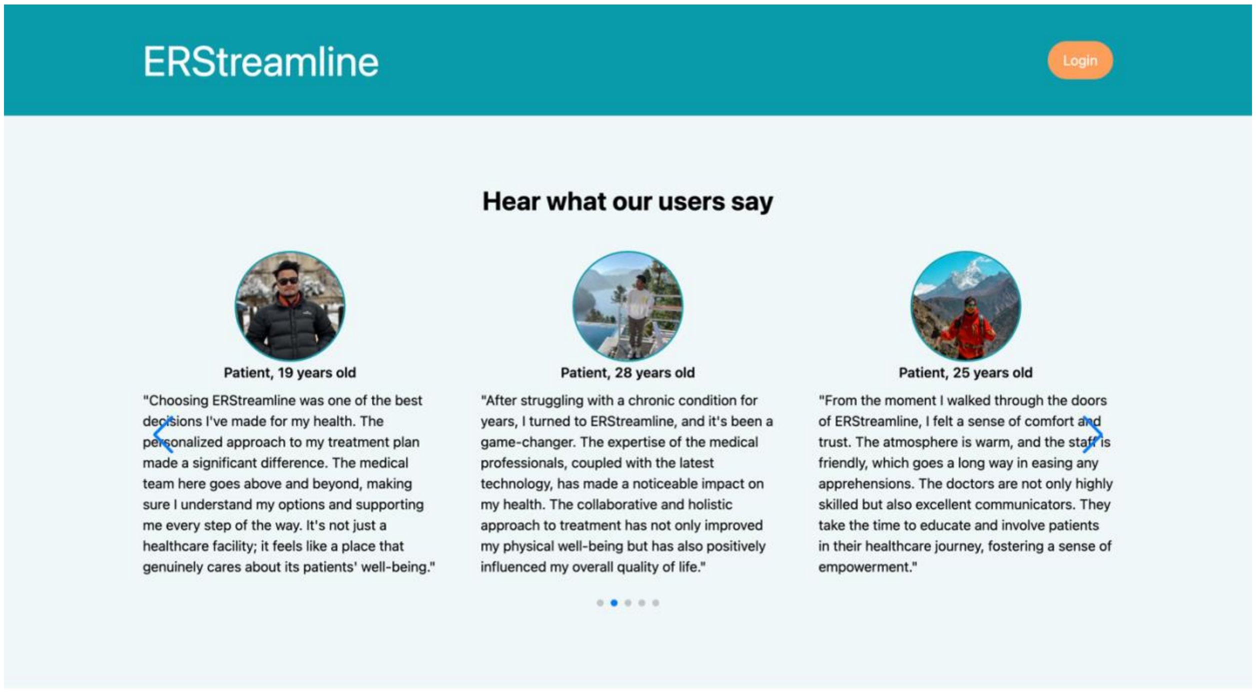
Figure 15: Use of bcrypt[2] to encrypt password.



```
"dependencies": {
  "@emotion/react": "^11.11.1",
  "@emotion/styled": "^11.11.0",
  "@mui/material": "^5.14.13",
  "@syncfusion/ej2-react-schedule": "^23.1.44",
  "@testing-library/jest-dom": "^5.17.0",
  "@testing-library/react": "^13.4.0",
  "@testing-library/user-event": "^13.5.0",
  "axios": "^1.5.1",
  "moment": "^2.29.4",
  "react": "^18.2.0",
  "react-dom": "^18.2.0",
  "react-router-dom": "^6.16.0",
  "react-scripts": "5.0.1",
  "swiper": "^11.0.3",
  "web-vitals": "^2.1.4"
},
```

```
"dependencies": {
  "bcryptjs": "^2.4.3",
  "concurrently": "^8.2.1",
  "cookie-parser": "^1.4.6",
  "cors": "^2.8.5",
  "dotenv": "^16.3.1",
  "express": "^4.18.2",
  "jsonwebtoken": "^9.0.2",
  "mongoose": "^7.5.3",
  "multer": "^1.4.5-lts.1",
  "nodemon": "^3.0.1",
  "validator": "^13.11.0"
```

Figure 16: Frontend (Left) and Backend(Right) Dependencies for ERStreamline


ERStreamline

Hear what our users say

Patient, 19 years old

"Choosing ERStreamline was one of the best decisions I've made for my health. The personalized approach to my treatment plan made a significant difference. The medical team here goes above and beyond, making sure I understand my options and supporting me every step of the way. It's not just a healthcare facility; it feels like a place that genuinely cares about its patients' well-being."

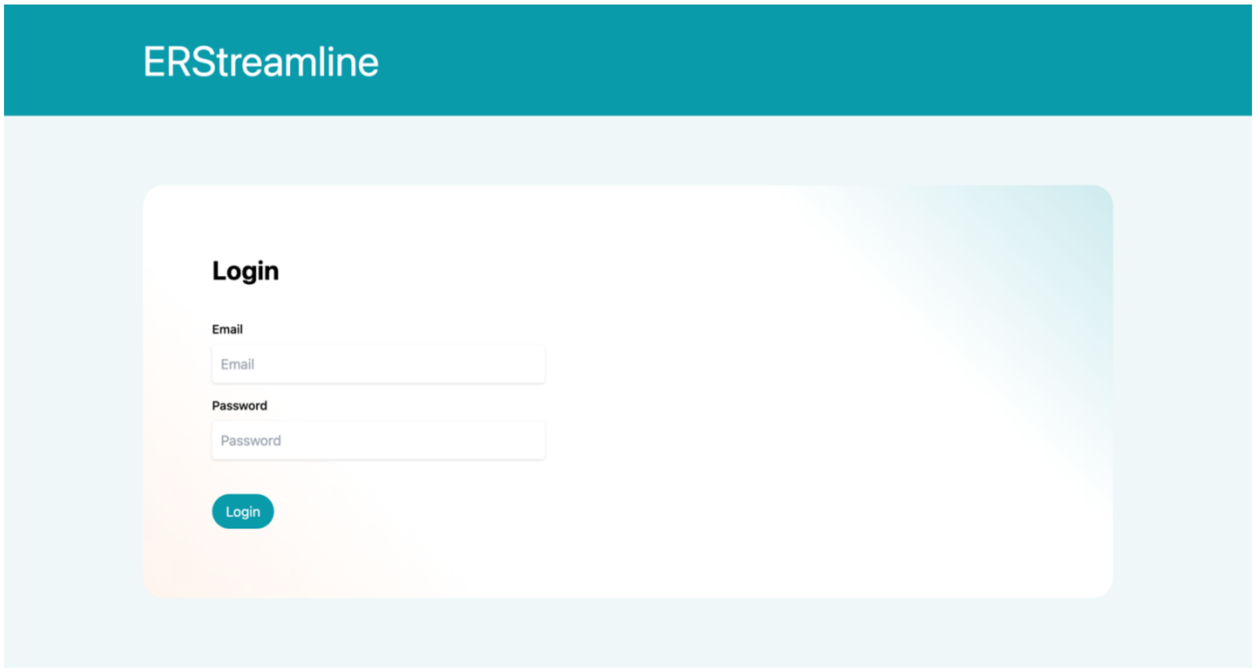
Patient, 28 years old

"After struggling with a chronic condition for years, I turned to ERStreamline, and it's been a game-changer. The expertise of the medical professionals, coupled with the latest technology, has made a noticeable impact on my health. The collaborative and holistic approach to treatment has not only improved my physical well-being but has also positively influenced my overall quality of life."

Patient, 25 years old

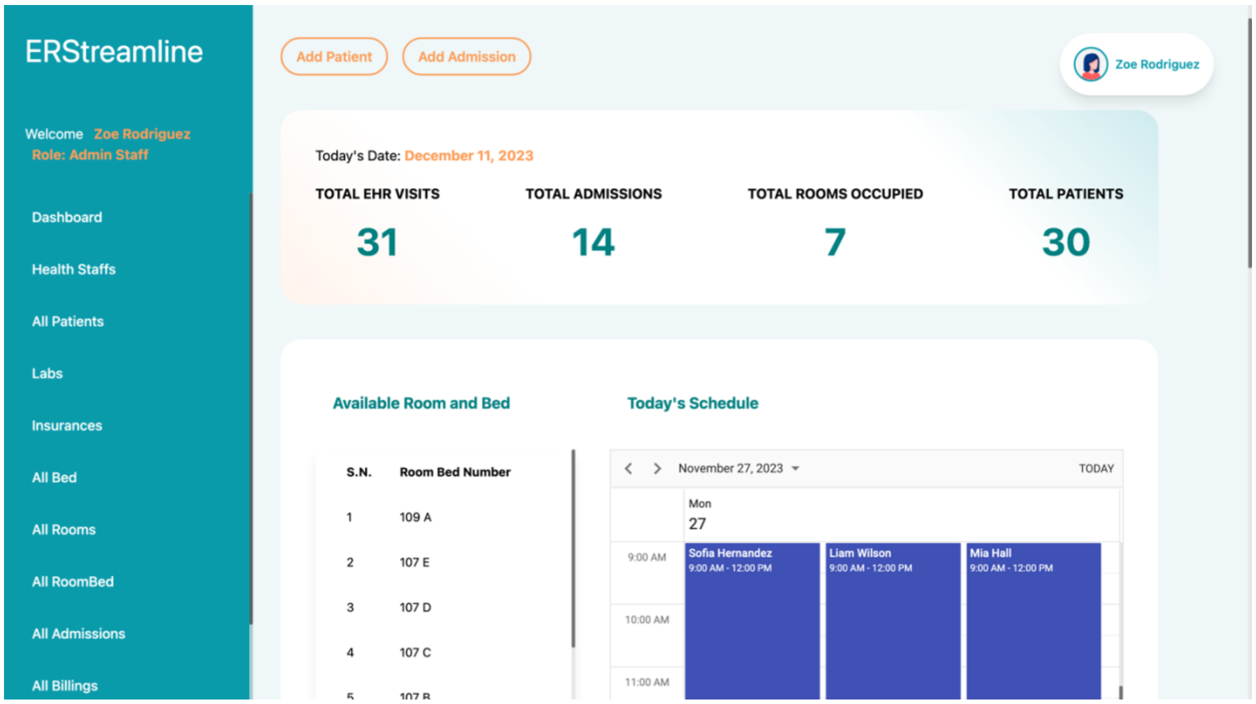
"From the moment I walked through the doors of ERStreamline, I felt a sense of comfort and trust. The atmosphere is warm, and the staff is friendly, which goes a long way in easing any apprehensions. The doctors are not only highly skilled but also excellent communicators. They take the time to educate and involve patients in their healthcare journey, fostering a sense of empowerment."

Figure 17: Landing Page after the project executes.


ERStreamline

Login

Figure 18: Login Page


ERStreamline

Add Patient | Add Admission | Zoe Rodriguez

Welcome Zoe Rodriguez
Role: Admin Staff

- Dashboard
- Health Staffs
- All Patients
- Labs
- Insurances
- All Bed
- All Rooms
- All RoomBed
- All Admissions
- All Billings

Today's Date: December 11, 2023

TOTAL EHR VISITS	TOTAL ADMISSIONS	TOTAL ROOMS OCCUPIED	TOTAL PATIENTS
31	14	7	30

Available Room and Bed

S.N.	Room Bed Number
1	109 A
2	107 E
3	107 D
4	107 C
5	107 B

Today's Schedule

November 27, 2023 TODAY

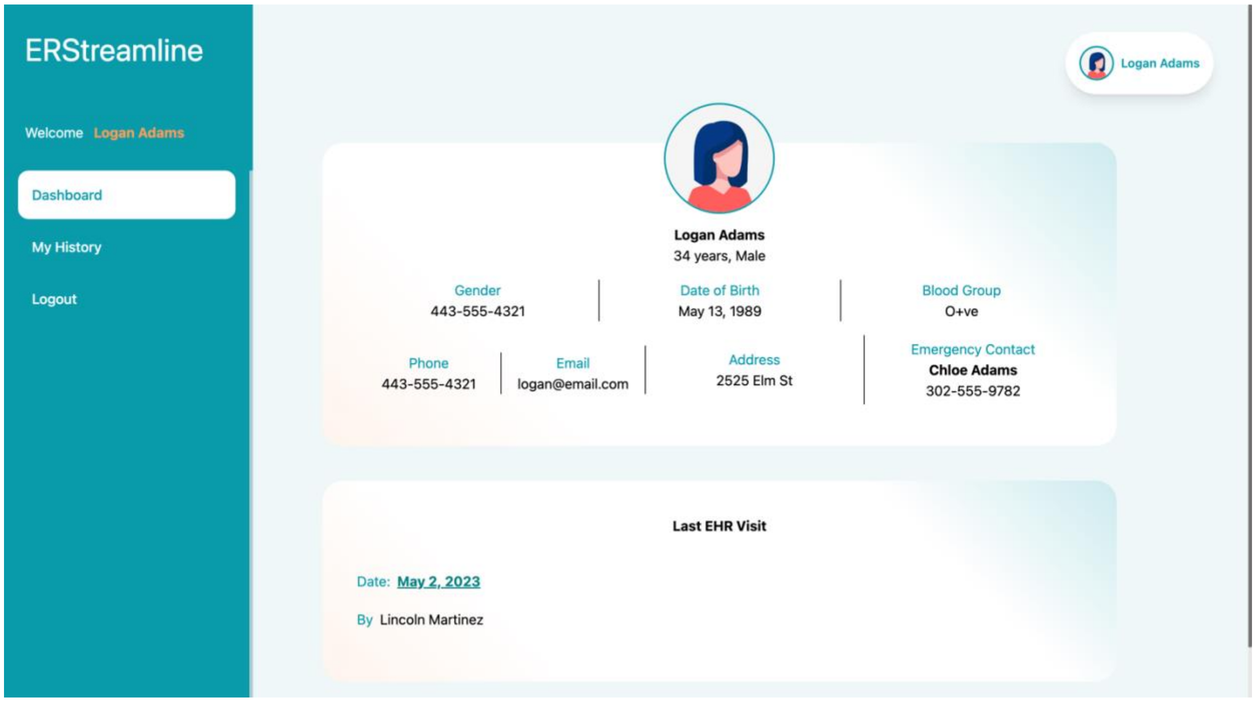
Mon 27

9:00 AM — Sofia Hernandez 9:00 AM - 12:00 PM; Liam Wilson 9:00 AM - 12:00 PM; Mia Hall 9:00 AM - 12:00 PM

10:00 AM

11:00 AM

Figure 19: Admin Dashboard Page


ERStreamline

Logan Adams

Welcome Logan Adams

- Dashboard
- My History
- Logout

Logan Adams
34 years, Male

Gender	Date of Birth	Blood Group
443-555-4321	May 13, 1989	O+ve

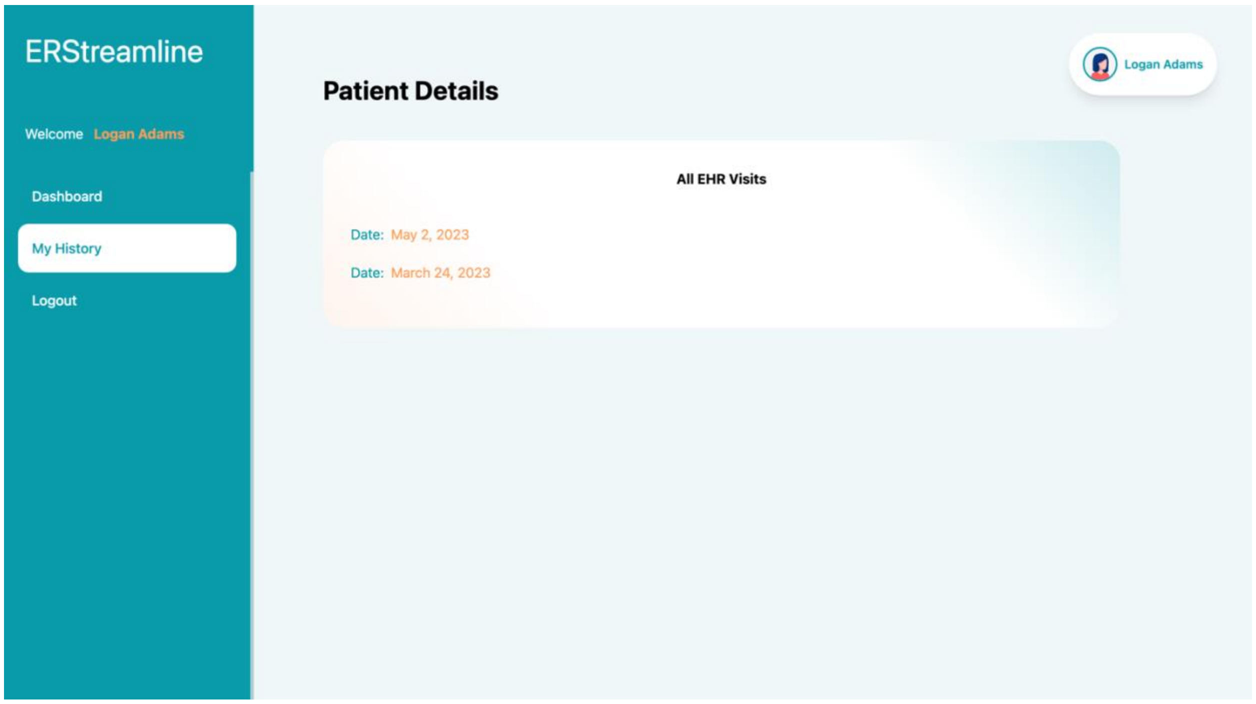
Phone	Email	Address	Emergency Contact
443-555-4321	logan@email.com	2525 Elm St	**Chloe Adams** 302-555-9782

Last EHR Visit

Date: May 2, 2023

By Lincoln Martinez

Figure 20: Patient Dashboard Page



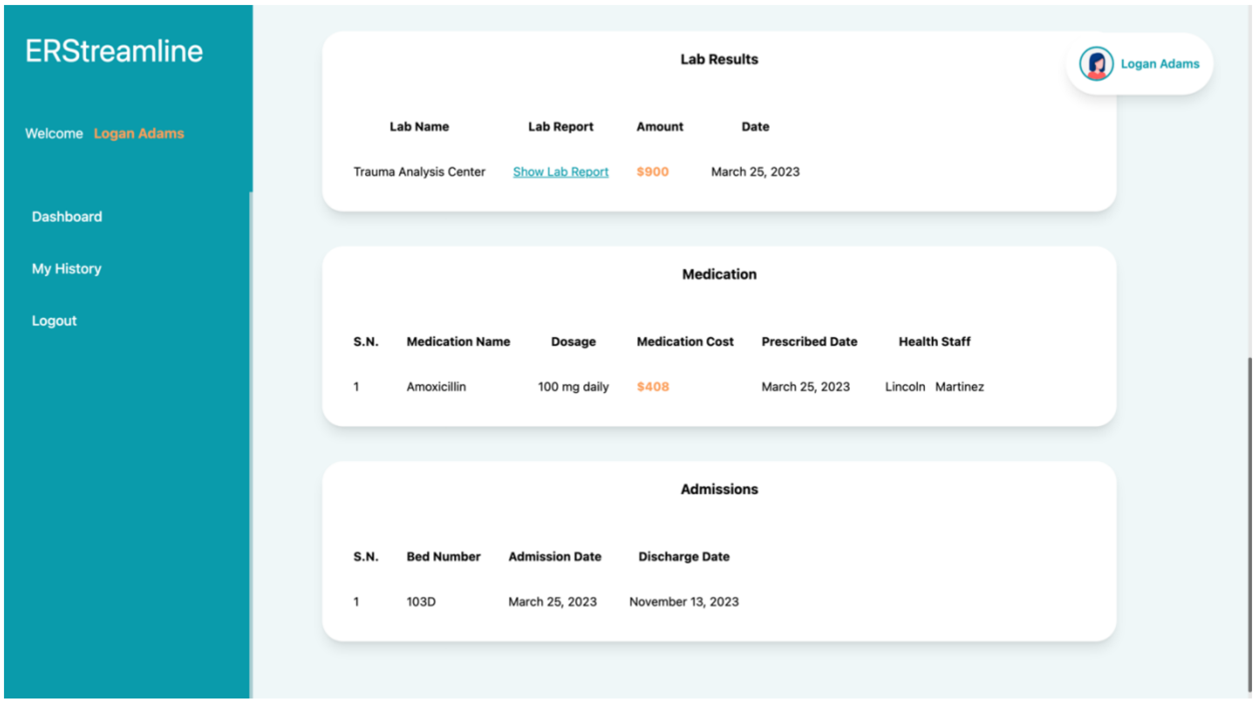
Patient Details

All EHR Visits

Date: May 2, 2023

Date: March 24, 2023

Figure 21: Patient (My EHRVisit History) Page



Lab Results

Lab Name	Lab Report	Amount	Date
Trauma Analysis Center	Show Lab Report	$900	March 25, 2023

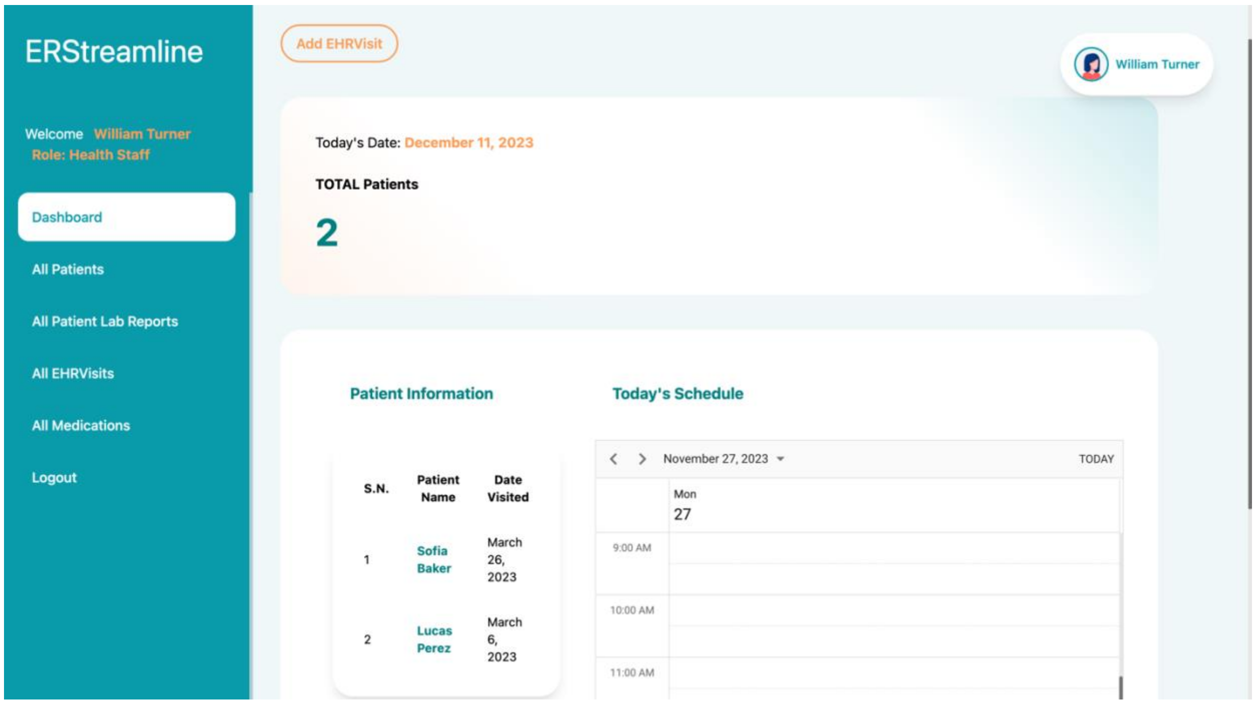
Medication

S.N.	Medication Name	Dosage	Medication Cost	Prescribed Date	Health Staff
1	Amoxicillin	100 mg daily	$408	March 25, 2023	Lincoln Martinez

Admissions

S.N.	Bed Number	Admission Date	Discharge Date
1	103D	March 25, 2023	November 13, 2023

Figure 22: Patient: EHRVisit Detail Page



Today's Date: December 11, 2023

TOTAL Patients

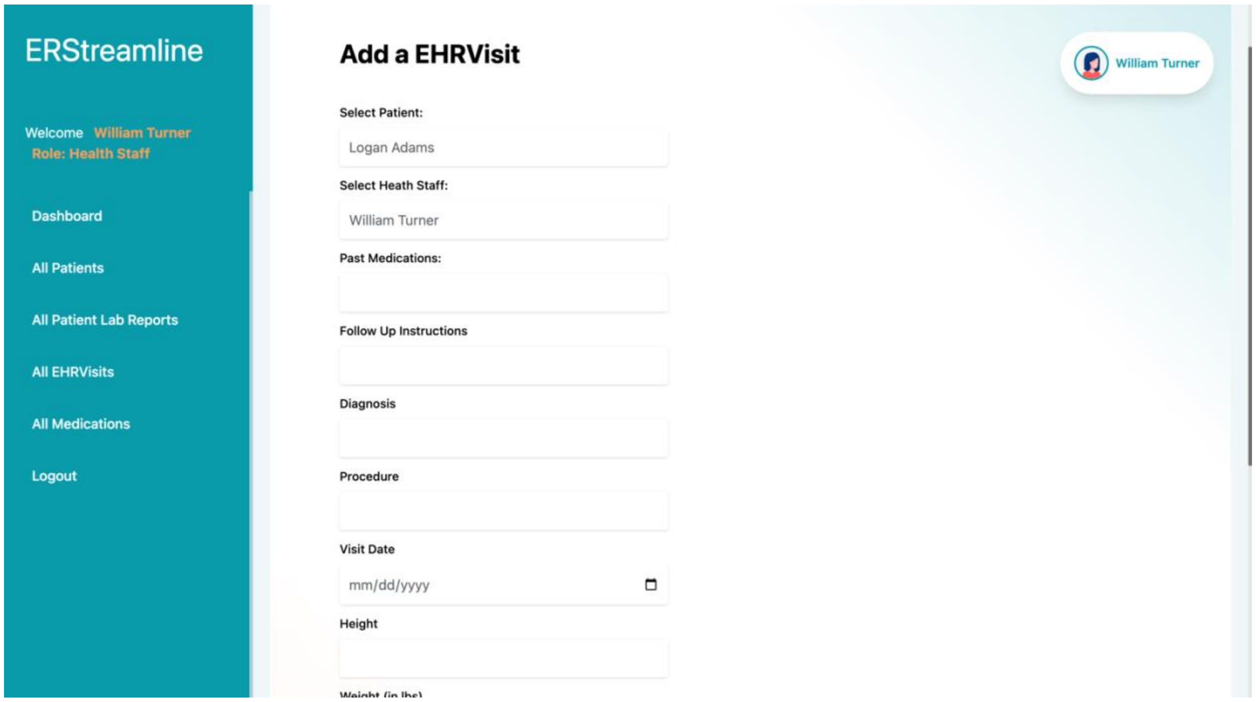
2

Patient Information

S.N.	Patient Name	Date Visited
1	Sofia Baker	March 26, 2023
2	Lucas Perez	March 6, 2023

Today's Schedule

Figure 23: Health Staff Dashboard Page



Add a EHRVisit

Select Patient: Logan Adams

Select Heath Staff: William Turner

Past Medications:

Follow Up Instructions

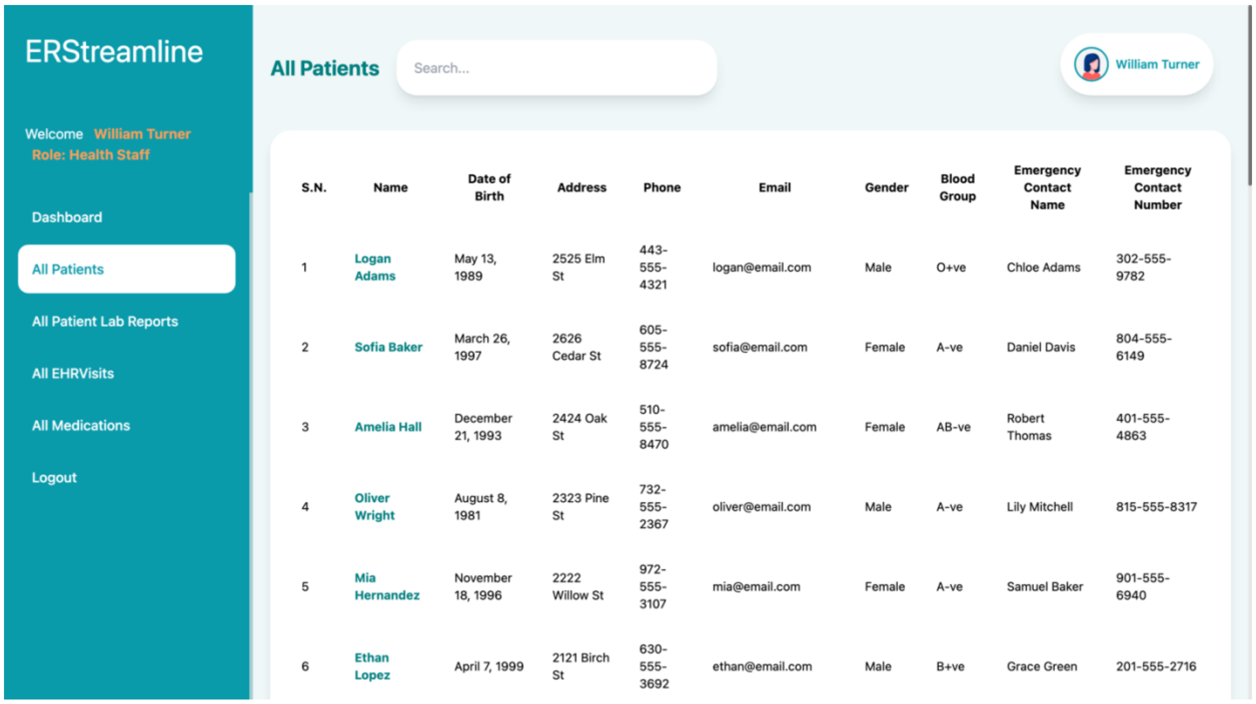
Diagnosis

Procedure

Visit Date mm/dd/yyyy

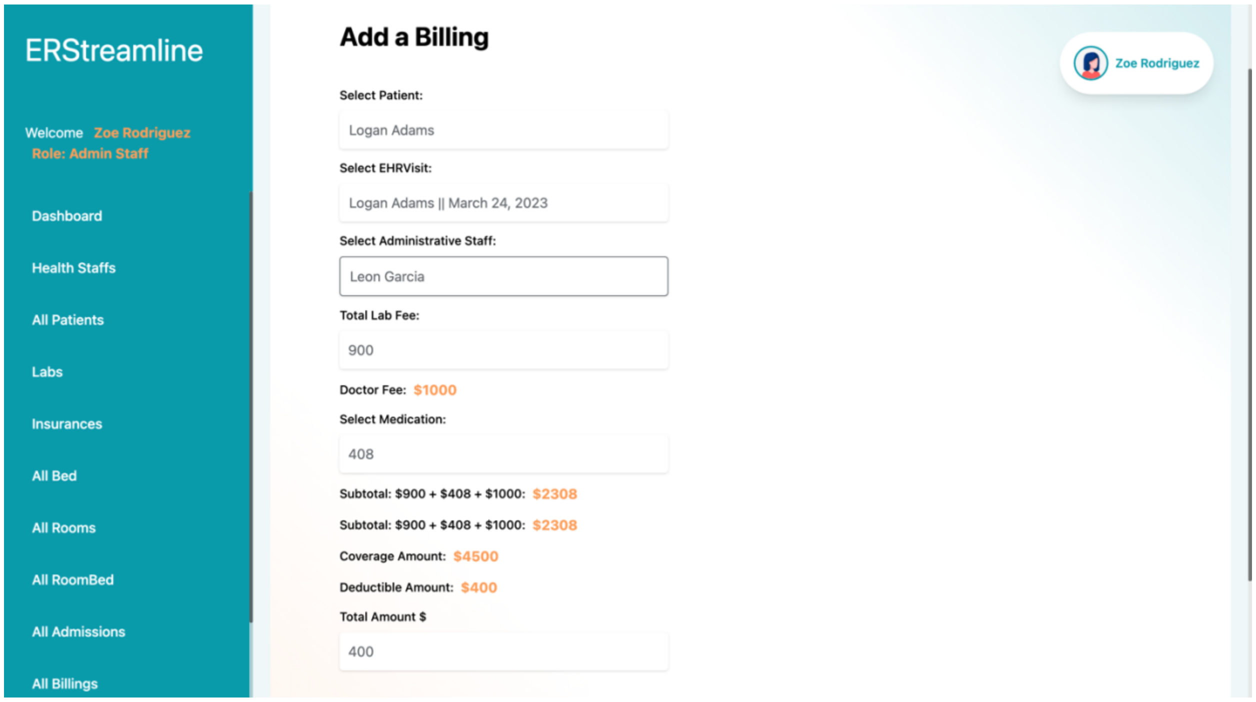
Height

Figure 24: Health Staff - Add EHRVisit Form



S.N.	Name	Date of Birth	Address	Phone	Email	Gender	Blood Group	Emergency Contact Name	Emergency Contact Number
1	Logan Adams	May 13, 1989	2525 Elm St	443-555-4321	logan@email.com	Male	O+ve	Chloe Adams	302-555-9782
2	Sofia Baker	March 26, 1997	2626 Cedar St	605-555-8724	sofia@email.com	Female	A-ve	Daniel Davis	804-555-6149
3	Amelia Hall	December 21, 1993	2424 Oak St	510-555-8470	amelia@email.com	Female	AB-ve	Robert Thomas	401-555-4863
4	Oliver Wright	August 8, 1981	2323 Pine St	732-555-2367	oliver@email.com	Male	A-ve	Lily Mitchell	815-555-8317
5	Mia Hernandez	November 18, 1996	2222 Willow St	972-555-3107	mia@email.com	Female	A-ve	Samuel Baker	901-555-6940
6	Ethan Lopez	April 7, 1999	2121 Birch St	630-555-3692	ethan@email.com	Male	B+ve	Grace Green	201-555-2716

Figure 25: Health Staff: All Patients Page



Add a Billing

Select Patient: Logan Adams

Select EHRVisit: Logan Adams || March 24, 2023

Select Administrative Staff: Leon Garcia

Total Lab Fee: 900

Doctor Fee: $1000

Select Medication: 408

Subtotal: $900 + $408 + $1000: $2308

Subtotal: $900 + $408 + $1000: $2308

Coverage Amount: $4500

Deductible Amount: $400

Total Amount $ 400

Figure 26: Admin Staff: Add Billing Page which calculates the automatic insurance coverage amount and deductible.

References

1) Mdarif. “Mdarif/Support-Desk: A Fullstack Mern Support Ticket System with Redux Toolkit.” *GitHub*, github.com/mdarif/support-desk. 2023. 8
2) “Bcryptjs.” *Npm*, www.npmjs.com/package/bcryptjs. 2023. 9
3) “Concurrently.” *Npm*, www.npmjs.com/package/concurrently. 2023. 9
4) auth0.com. “Jwt.Io.” *JSON Web Tokens*, jwt.io/. 2023. 10
5) “Multer.” *Npm*, www.npmjs.com/package/multer. 2023. 10
6) “Nodemon Reload, Automatically.” *Nodemon*, nodemon.io/. 2023. 10
7) PaulHalliday. “How to Use Axios with React.” *DigitalOcean*, DigitalOcean, 2 Dec. 2021, www.digitalocean.com/community/tutorials/react-axios-react. 10
8) “Black Lives Matter.” *Moment.Js | Home*, momentjs.com/. 2023. 10
9) “The Most Modern Mobile Touch Slider.” *Swiper*, swiperjs.com/react. 2023. 10