

Final Doc

-Pranoy Dev

1. Requirements

1.1. Overview

The Patient Tracker System is an innovative application designed to revolutionize the management of patient information and medical records. The primary goal of this system is to introduce a seamless and efficient approach for both patients and healthcare providers to access, update, and manage patient records in real time. Embracing digital transformation, this system eradicates the reliance on manual paperwork, offering a robust, secure, and accessible platform for managing patient data. The objectives encompass facilitating real-time access, promoting user-friendly interactions, and enhancing overall healthcare efficiency. Intended for both healthcare providers and patients, the system aims to streamline workflows, improve patient care, and empower users with comprehensive insights. The project's motivations lie in addressing the limitations of traditional paper-based record systems, fostering better communication, and ultimately contributing to a more connected and patient-centric healthcare ecosystem.

1.2. Features

-> User Authentication and Access Control:

Secure login for patients and healthcare providers ensured with robust authentication methods, safeguarding patient records and regulating access based on roles and permissions, ensuring data integrity.

-> Comprehensive Patient Database Management:

Centralized storage of a wide array of patient information, including personal details, medical history, prescribed medications, diagnoses, and treatment plans for efficient and organized record-keeping and comprehensive patient care.

-> Intuitive User Interface for Healthcare Professionals:

An intuitive and user-friendly interface designed specifically for healthcare providers, enabling seamless navigation and easy interaction with patient records, optimizing healthcare delivery.

-> Data Security and Compliance Measures:

Implementation of strong data encryption, secure data transmission, and stringent compliance with healthcare data regulations to ensure patient information confidentiality, maintaining trust and legal adherence.

-> Advanced Search and Filtering Capabilities for Efficient Care:

Advanced search and filtering options allowing rapid retrieval of specific patient information, such as medications, previous checkup details, diagnostic reports, and medical history, streamlining healthcare processes for enhanced patient care.

-> Collaborative Care Planning and Notes:

Collaborative care planning features allowing healthcare providers to create and share patient notes, ensuring seamless communication and coordinated care among the healthcare team.

1.3. Functional Requirements (Use Cases)

- As a Doctor, I want to view patient profiles with medical history to prepare for consultations.
 - Action: Access patient records to review medical history, diagnosis, and prescribed medications.
- As a Doctor, I want to update doctor notes after a consultation.
 - Action: Enter new diagnoses, treatment plans, or prescription updates into the patient's record.
- As a Doctor, I want to collaborate with the patient, by viewing patient notes
 - Action: View patient notes
- As a Doctor, I want to get an outline for my day so that I can plan my work for each patient.
 - Action: View Daily Schedule
- As a Doctor, I want to get an outline for my week so that I can plan personal leaves/ work accordingly.
 - Action: View the Appointment List, which gives the list of appointments in the current week.
- As a Doctor, I want to view all patients under me.
 - Action: View the Patient list.
- As a Patient, I want to access my medical records and review my health history for better understanding.
 - Action: Log in to the patient portal to view detailed medical records, including past diagnoses, treatments, and prescribed medications.
- As a Patient, I want to view my personal information to check if there are any errors
 - Action: View "My Info" page.
- As a Patient, I want to update my personal health information or provide additional details after a healthcare appointment.
 - Action: Edit and update patient notes, which will be visible to the doctor.
- As a Patient, I want to view my appointments with the doctors for the given week.
 - Action: Go to the View Appointments section

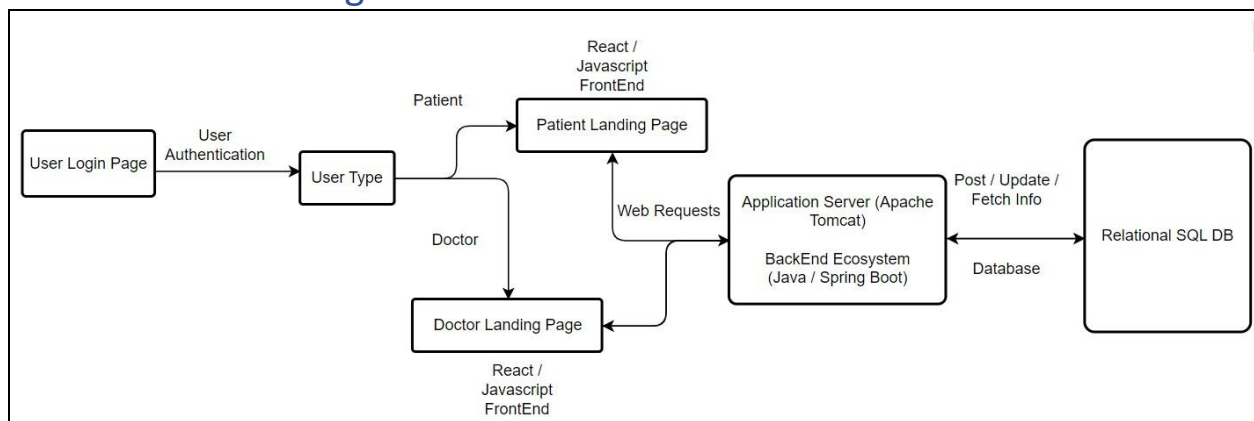
1.4. Non-Functional Requirements

- Performance:
 - To ensure system responsiveness with minimal latency for data retrieval and update actions, ensuring healthcare professionals can quickly access and update patient information during consultations, resulting in efficient patient care.
 - We performed Query Optimization to optimize database queries and used indexing to speed up data retrieval. Asynchronous processing was used for tasks that didn't need an immediate response, enhancing system responsiveness.
- Reliability:
 - Maintaining consistent and uninterrupted access to critical healthcare data and system functionality is essential for supporting continuous patient care and ensuring operational efficiency. Healthcare providers rely on this system for accurate, real-time information.
 - We Implemented robust error handling to gracefully manage unexpected situations and provide meaningful error messages.

- **Security:**
 - Implementing robust data encryption, secure data transmission, and access control is paramount to safeguard patient records. It ensures the confidentiality and integrity of sensitive patient data, preventing unauthorized access and data breaches.
 - We employed HTTPS to encrypt data during transmission and Implemented encryption for sensitive data stored in the PostgreSQL database.
- **Scalability:**
 - Designing the system architecture to handle potential future increases in the user base and data volume is crucial for accommodating the growth of the healthcare facility and ensuring that the system remains effective as the patient load increases.
- **Usability:**
 - Providing an intuitive and user-friendly interface is essential to facilitate easy navigation and use for both doctors and patients. A user-friendly design enhances the overall experience, making it more efficient and reducing the learning curve.
 - We prioritized user experience (UX) in the frontend design to make it intuitive and user-friendly and ensured the application was responsive to different screen sizes for a seamless experience on various devices.
- **Compliance:**
 - Adhering to healthcare data regulations and standards is critical to ensuring patient data privacy and legal compliance. Compliance guarantees that patient records are handled in a way that meets the necessary legal and ethical requirements, maintaining trust in the system.

2. Design

2.1. Architecture Diagram



2.2. Technology Stack with Justification

1) Frontend: JavaScript and React

JavaScript is a widely used programming language for web development supported by all modern web browsers and a vast developer ecosystem for creating interactive and dynamic user interfaces. We can implement classes also as needed to make the code structure object-oriented.

React, a JavaScript library for building user interfaces is an easy to use component driven framework. Its component-based architecture makes it easy to create reusable UI elements and manage the front end efficiently.

Major Advantages:

- > Virtual DOM feature that enhances performance by minimizing direct manipulation of the actual DOM, resulting in faster UI rendering.
- > An active community and good online documentation that contributes towards learning about the libraries, tools and resources to expedite development.
- > React supports server-side rendering, which generally speaking is beneficial for initial load times.
- > Its component reusability and data flow make it easier to maintain and scale your front end.

2) Backend: Java and Spring Boot

Java is a robust, mature, widely adopted and reliable object oriented programming language, known for its scalability, security, and extensive libraries. In a healthcare application like the Patient Tracker System, data security and scalability are paramount. Additionally, the extensive Java ecosystem provides various libraries and frameworks for efficient development. Also being an object oriented language, it allows us to structure our code keeping in important principles like modularity, reusability and other OOPS principles.

We have used Spring Boot because it simplifies the development of Java based applications. It provides a range of features that are well-suited for our project, including:

- > Ease of Development: It simplifies the development process by providing defaults and auto-configurations, which reduce the need for extensive setup.
- > Microservices Support: If our project needs to scale or extend its capabilities with microservices in the future, Spring Boot provides excellent support for building microservices.
- > Security: Spring Boot includes robust security features, including authentication and authorization, which are critical for protecting patient data.
- > Database Integration: Spring Boot makes it easy to integrate with various databases, which is essential for storing and managing patient records.
- > Community Support, Documentation: Spring Boot has a large and active community and well documented help websites, so we can find plenty of resources and support as we develop our application.
- > Compliance with Java EE Standards: Spring Boot adheres to Java EE standards, which ensures our application's compatibility with established best practices.

While developing a Java based app, we'll also need a web server or application server to run our Java application. Since you're using the Spring Boot framework for your Java application, we can take advantage of its embedded servers like Tomcat. Spring Boot simplifies the deployment process and offers flexibility in choosing the embedded server that suits our needs.

3) Database : RDBMS

We have used AWS RDS for the Database. We are using an encrypted instance of Postgres available in AWS. We used Relational Database Management Systems (RDBMS) because they are suitable for healthcare applications where structured data is important, and relationships between different data points need to be maintained. RDBMS is known for its data consistency, ACID compliance, and mature query languages.

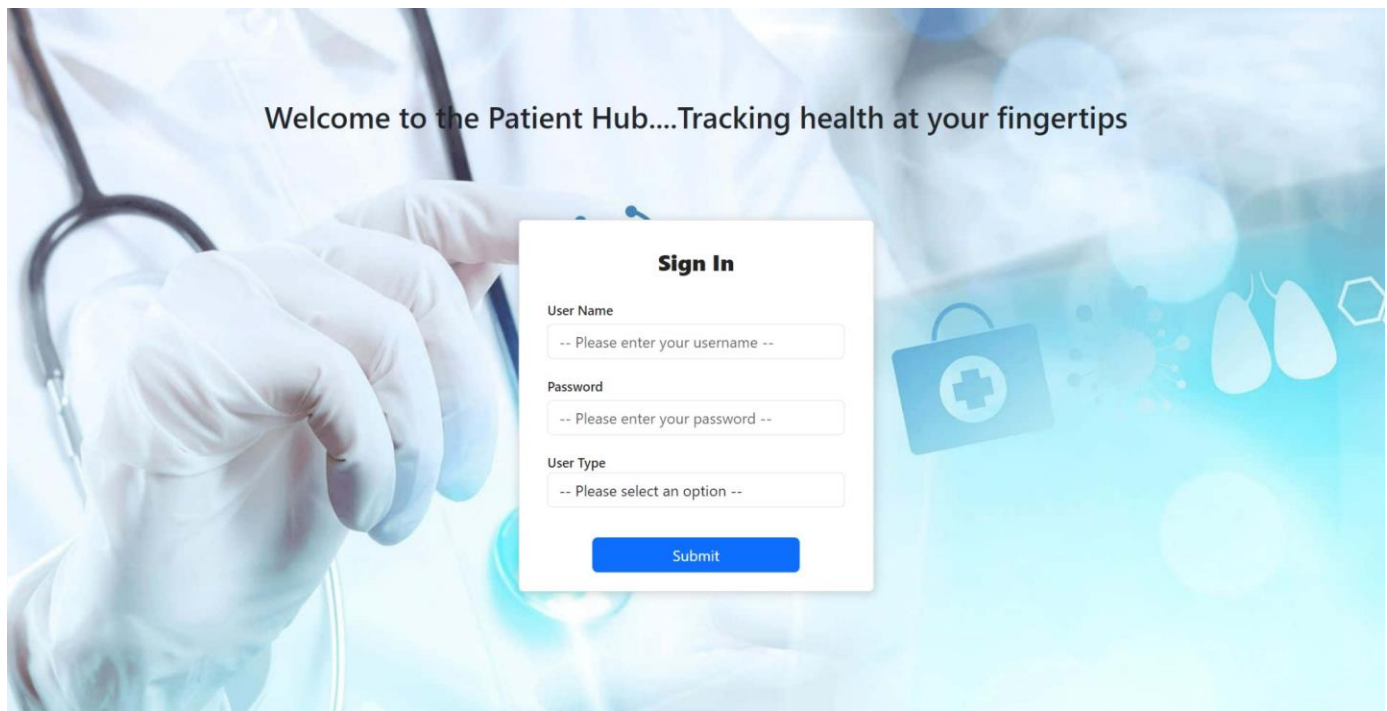
Features:

- > Structured Data: Since our patient data is highly structured and follows a well-defined schema with many relationships (e.g., patients, doctors, appointments, medical history), an RDBMS is a natural fit. SQL databases excel at maintaining data integrity and enforcing relationships between tables.
- > ACID Compliance: RDBMS systems are known for their strong ACID (Atomicity, Consistency, Isolation, Durability) compliance, making them suitable for applications where data consistency and reliability are critical, such as healthcare.
- > Complex Queries: If our application requires complex queries, reporting, and analysis of the patient data, SQL databases provide powerful querying capabilities and join operations.
- > Data Integrity: Healthcare applications demand a high level of data integrity and security. SQL databases provide robust security mechanisms and are well-suited for maintaining patient data privacy.

2.3. UI Mockup

Showcasing user interface designs and layouts for the system, emphasizing doctor and patient interactions.

Login Page:



Failed Auth:

Welcome to the Patient Hub...Tracking health at your fingertips

Sign In

User Name

Password

User Type

Invalid Credentials !!!...Please try again

Success Auth Patient User -> Patient Landing UI:

Patient Portal

My Info

Appointments

Logout

The main content area of the Patient Portal features a large, light blue-tinted background image of a doctor's hands in white gloves, one holding a stethoscope. Overlaid on this image are several white medical icons: a stethoscope, a plus sign, a pill, a first aid kit, and a pair of lungs. The overall design is clean and professional, typical of a healthcare application.

Patient View My Info -> Update Patient Notes provision :

Patient Portal

My Info

Appointments

Logout

PATIENT NAME

Pranoy Dev

PATIENT AGE

25

PATIENT ID

11

MEDICAL HISTORY

Immunity Disorder, Cough and Cold

CURRENT MEDICATIONS

Herbal Immunity Tonic

PREVIOUS DIAGNOSIS

Mild Immunity Disorder, Cough and Cold

DOCTOR NOTES

Regular Exercise and Healthy Diet

PATIENT NOTES

Following prescription....Cough and Cold seem to reduce

Save Info

Patient View Appointments:

Patient Portal

My Info

Appointments

Logout

Friday : 15/12/2023

Time Slot : 9AM - 10AM

Doctor Name : Yash

Time Slot : 12PM - 1PM

Doctor Name : Megha

Saturday : 16/12/2023

Time Slot : 9AM - 10AM

Doctor Name : Alexis

Time Slot : 1PM - 2PM

Doctor Name : Barbatos

Sunday : 17/12/2023

Time Slot : 9AM - 10AM

Doctor Name : Vaishnavi

Time Slot : 1PM - 2PM

Doctor Name : Aditya

Success Auth Doctor User -> Doctor Landing Page:



Doctor Daily Schedule View -> With Patient View Record Capability:



Doctor Appointment List View -> With Patient View Record Capability:

Doctor Portal

Daily Schedule

Appointments

Patients

Logout

Friday : 15/12/2023

Time Slot : 9AM - 10AM	Patient Name : Pranoy Dev	View Record
Time Slot : 12PM - 1PM	Patient Name : Mikhael	View Record
Time Slot : 1PM - 2PM	Patient Name : Xiao Hang	View Record
Time Slot : 3PM - 4PM	Patient Name : Sunil Sharma	View Record
Time Slot : 5PM - 6PM	Patient Name : Alexander	View Record

Saturday : 16/12/2023

Time Slot : 9AM - 10AM	Patient Name : Amey Sharma	View Record
Time Slot : 1PM - 2PM	Patient Name : Nishi	View Record

Thursday : 17/12/2023

Time Slot : 9AM - 10AM	Patient Name : Vaishnavi	View Record
------------------------	--------------------------	-----------------------------

Doctor Patient List View -> With Patient View Record Capability:

Doctor Portal

Daily Schedule

Appointments

Patients

Logout

-- Please search using Name --

Patient Name : Megha Sharma	View Record
Patient Name : Pranoy Dev	View Record
Patient Name : Aditya Patil	View Record

Doctor Patient List View -> With Search Capability:

Doctor Portal

Daily Schedule

Appointments

Patients

Logout

Pra

Patient Name : Pranoy Dev

View Record



Doctor View Patient Info -> With update Doctor Notes Capability

Doctor Portal

Daily Schedule

Appointments

Patients

Logout

PATIENT NAME

Pranoy Dev

PATIENT AGE

25

PATIENT ID

11

MEDICAL HISTORY

Immunity Disorder, Cough and Cold

CURRENT MEDICATIONS

Herbal Immunity Tonic

PREVIOUS DIAGNOSIS

Mild Immunity Disorder, Cough and Cold

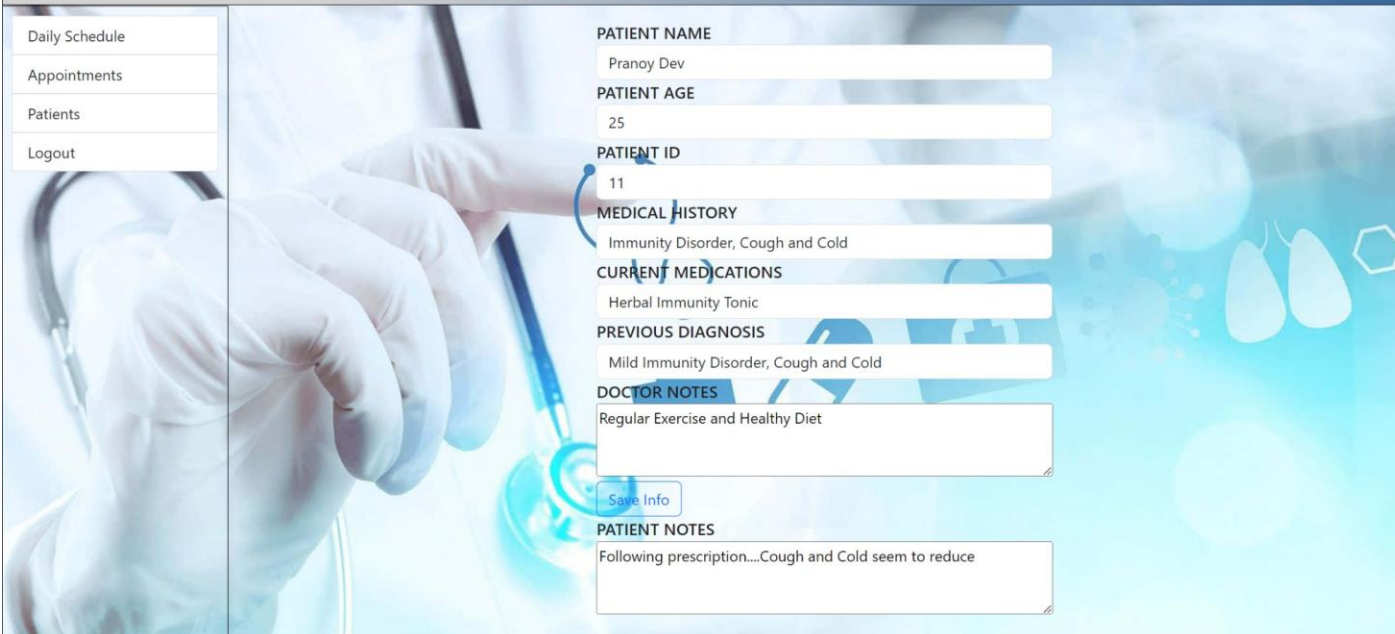
DOCTOR NOTES

Regular Exercise and Healthy Diet

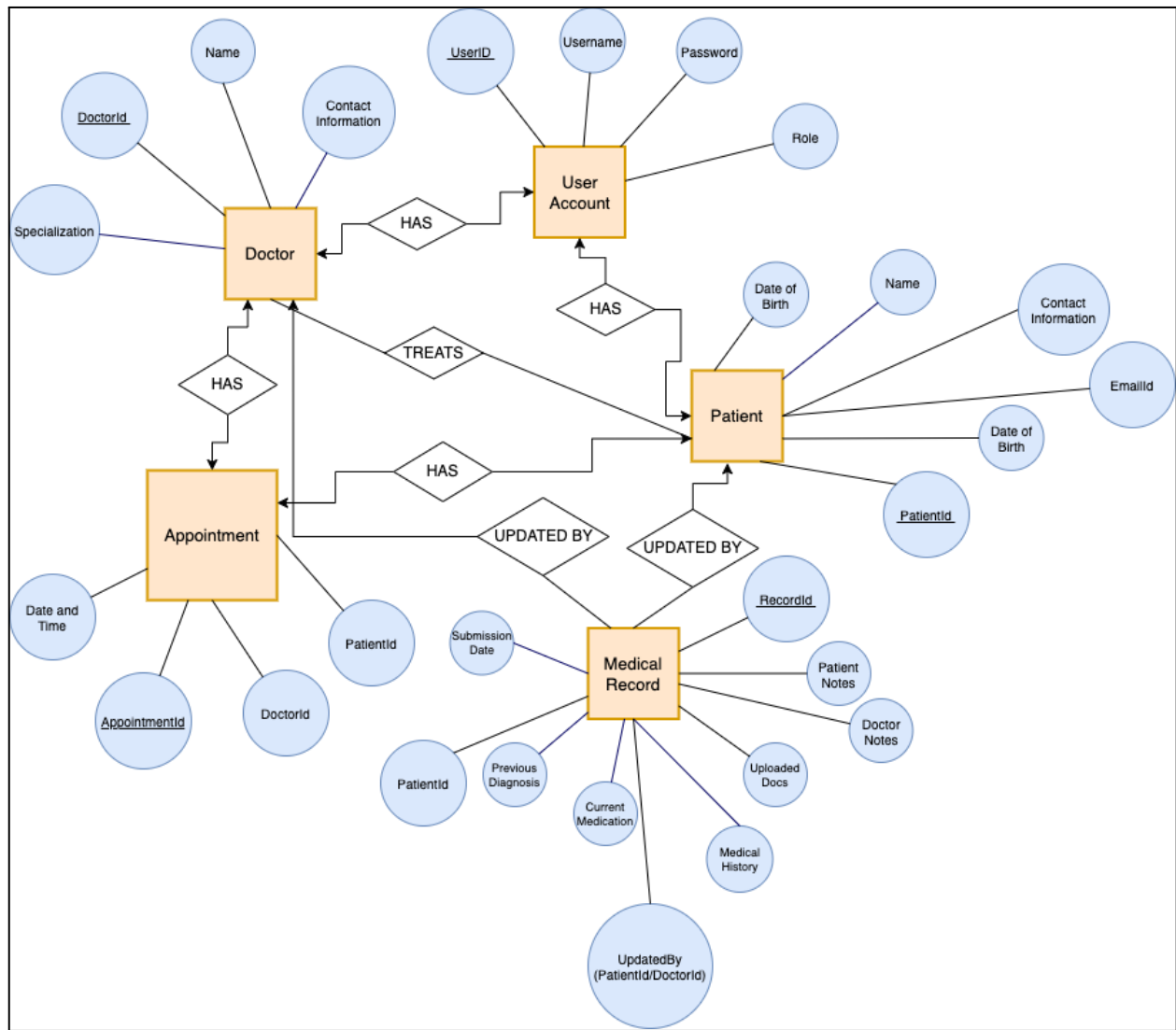
Save Info

PATIENT NOTES

Following prescription....Cough and Cold seem to reduce



2.4. Data Model



PATIENT DB SCHEMA:

```

patient_portal=> \d patient
      Table "public.patient"
      Column          | Type          | Collation | Nullable | Default
      -----|-----|-----|-----|-----
      patientid       | integer       |           | not null | nextval('patient_patientid_seq'::regclass)
      dateofbirth     | date         |           |          |
      name            | character varying(100) |           | not null |
      contactinformation | character varying(100) |           |          |
      email           | character varying(100) |           |          |
      userid          | integer       |           |          |
  Indexes:
    "patient_pkey" PRIMARY KEY, btree (patientid)
  Foreign-key constraints:
    "patient_userid_fkey" FOREIGN KEY (userid) REFERENCES useraccount(userid)
  Referenced by:
    TABLE "appointment" CONSTRAINT "appointment_patientid_fkey" FOREIGN KEY (patientid) REFERENCES patient(patientid)
    TABLE "medicalrecord" CONSTRAINT "fk1fcjudqj5m2t0k3lqaov4omjr" FOREIGN KEY (patientid) REFERENCES patient(patientid)
  
```

DOCTOR DB SCHEMA:

```
patient_portal=> \d doctor
```

Column	Type	Collation	Nullable	Default
doctorid	integer		not null	nextval('doctor_doctorid_seq'::regclass)
userid	integer			
name	character varying(100)		not null	
contactinformation	character varying(100)			
specialization	character varying(50)			

Indexes:

"doctor_pkey" PRIMARY KEY, btree (doctorid)

Foreign-key constraints:

"doctor_userid_fkey" FOREIGN KEY (userid) REFERENCES useraccount(userid)

Referenced by:

TABLE "appointment" CONSTRAINT "appointment_doctorid_fkey" FOREIGN KEY (doctorid) REFERENCES doctor(doctorid)

APPOINTMENT DB SCHEMA:

```
patient_portal=> \d appointment
```

Column	Type	Collation	Nullable	Default
appointmentid	integer		not null	nextval('appointment_appointmentid_seq'::regclass)
datetime	timestamp without time zone		not null	
doctorid	integer			
patientid	integer			

Indexes:

"appointment_pkey" PRIMARY KEY, btree (appointmentid)

Foreign-key constraints:

"appointment_doctorid_fkey" FOREIGN KEY (doctorid) REFERENCES doctor(doctorid)

"appointment_patientid_fkey" FOREIGN KEY (patientid) REFERENCES patient(patientid)

USER ACCOUNT DB SCHEMA:

```
patient_portal=> \d useraccount
```

Column	Type	Collation	Nullable	Default
userid	integer		not null	nextval('useraccount_userid_seq'::regclass)
username	character varying(50)		not null	
password	character varying(100)		not null	
role	character varying(20)		not null	

Indexes:

"useraccount_pkey" PRIMARY KEY, btree (userid)

Referenced by:

TABLE "doctor" CONSTRAINT "doctor_userid_fkey" FOREIGN KEY (userid) REFERENCES useraccount(userid)

TABLE "patient" CONSTRAINT "patient_userid_fkey" FOREIGN KEY (userid) REFERENCES useraccount(userid)

MEDICAL RECORD SCHEMA:

```
patient_portal=> \d medicalrecord
```

Column	Type	Collation	Nullable	Default
recordid	integer		not null	
currentmedication	character varying(255)			
doctornotes	character varying(255)			
medicalhistory	character varying(255)			
patientid	integer		not null	
patientnotes	character varying(255)			
previousdiagnosis	character varying(255)			
submissiondate	timestamp without time zone			
updatedby	integer		not null	
uploadedddocs	bytea			

Indexes:

"medicalrecord_pkey" PRIMARY KEY, btree (recordid)

Foreign-key constraints:

"fk1fcjudqj5m2t0k3lqaov4omjzr" FOREIGN KEY (patientid) REFERENCES patient(patientid)

3. Implementation

Coding Process:

- > Requirements Analysis: Understanding and documenting the requirements of your Patient Tracker System, including user stories, use cases and requirements.
- > Design: Creating a high-level system architecture, database schema, UI mockup designs, Flow Diagrams, ER Diagrams etc
- > Development: Implementing the system according to the design using the chosen technologies; following best coding practices, maintaining code quality and conducting code reviews.
- > Testing: Performing unit testing, full flow testing and user acceptance testing to ensure the system functions as expected.
- > Deployment: Deploying the application and is ready to use.

Methodologies:

- > Agile: Using an Agile dev methodology, such as Scrum to promote collaboration, adaptability and incremental changes.
- > Version Control: Using version control systems like Git to manage code and collaborate effectively within our team.
- > Continuous Integration and Continuous Deployment (CI/CD): Implement proper CI/CD pipelines to automate testing and deployment processes, ensuring code changes are integrated smoothly into the production environment.
- > Documentation: Maintaining comprehensive documentation for code, system architecture and user guides.

Challenges:

- > Data Security: Ensuring the security of patient data via data encryption and access controls.
- > Scalability: As the system grows, we may face challenges related to scalability, hence keeping the design simple and designing the system to be horizontally scalable to handle a larger number of users and patient records.
- > User Training: Training end users on how to use the system effectively. Hence, developing user-friendly and intuitive interfaces and providing comprehensive training materials and documentations.

3.1. Version Control and Collaboration

Our team extensively employed Git for version control throughout the development of the Patient Tracker System. To organize our collaborative efforts, we established a GitHub repository where we systematically divided tasks among team members. Team members regularly performed 'git pull' to synchronize their local repositories with the latest changes from the remote repository. This ensured that everyone was working on the most up-to-date code. When conflicts arose during the merging process, we collaboratively resolved them, ensuring that the final codebase remained coherent and free of inconsistencies. The disciplined use of Git and the structured branching strategy contributed to a seamless and collaborative development process for the Patient Tracker System.

3.2. Coding Standards and Practices

Throughout the development of our Patient Tracker System, our team adhered to a set of rigorous coding standards and best practices to ensure consistency, maintainability, and readability in both the React frontend and Spring Boot backend. Following the MVC architecture, our React components were organized into clear, modular structures, promoting a separation of concerns and ease of maintenance. We adopted meaningful naming conventions for variables, functions, and classes, enhancing code clarity and minimizing ambiguity. Comprehensive documentation was a cornerstone of our approach, with every component, function, and API endpoint meticulously documented

to facilitate understanding for both current and future developers. This commitment to well-documented code extended to the backend as well, ensuring that the Spring Boot controllers, services, and models were comprehensively explained.

3.3. Security and Risks

Threats: This includes unauthorized access, user credentials theft and data theft.

Vulnerabilities: Vulnerabilities can arise from weak authentication, insufficient access controls or unencrypted storage.

Prevention Measures in the backend:

- > Passwords are encrypted and then stored in the database to prevent passwords from being guessed.
- > Encrypt data in the database before storing, we used AWS data encryption. RDS uses AWS KMS for key management, allowing you to control and manage encryption keys.
- > We only used the HTTPS protocol. It uses SSL/TLS protocols to encrypt the communication between the client and the server, preventing attackers from reading or tampering with the transmitted data.
- > We have used the JWT for authorization, this allows us to only give access to users with the correct roles. The doctor cannot access patient data, and the patient cannot access doctor data.
- > Train staff and users in data security best practices.

HIPAA Compliance:

The Health Insurance Portability and Accountability Act (HIPAA) is a U.S. federal law that establishes national standards for the protection of patients' health information. It sets rules and safeguards to ensure the confidentiality, integrity, and availability of healthcare data.

Relevance: HIPAA compliance is crucial for healthcare-related applications, such as the Patient Tracker System, as it deals with sensitive patient information. Non-compliance can lead to legal consequences and reputational damage.

Steps and Protocols for Compliance:

- > Implement strong access controls to ensure that only authorized personnel can access patient data.
- > Encrypt patient data both in transit and at rest to protect against unauthorized access.
- > Conduct regular risk assessments and vulnerability assessments to identify and mitigate security risks.
- > Develop and enforce policies and procedures for handling patient data securely.
- > Train all end users on HIPAA compliance and data security best practices, proper documentation.
- > Have a formal incident response plan in place to address data breaches and security incidents.

4. Evaluation

4.1 Evaluation of Functional Requirements

Unit tests were meticulously crafted for each service module within the Patient Tracker System, leveraging JUnit and Mockito for seamless and effective testing. Utilizing JUnit assertions, such as assertEquals, and harnessing the power of Mockito's mocking capabilities, the test cases rigorously validated various service functionalities. From basic CRUD operations to intricate business logic, the test suite ensured the reliability and accuracy of critical services, including patient information retrieval, appointment management, and secure user authentication.

In the front-end development, we employed Jest along with React Testing Library to conduct thorough testing of individual components. Leveraging the power of the render and expect commands, we executed comprehensive tests to assess both the responsiveness and visibility of these components on the user screen.

This meticulous testing approach not only established a solid foundation for code reliability but also facilitated ongoing maintenance and evolution of the system in alignment with specified business requirements.

4.2 Evaluation of Non-functional Requirements

For the evaluation of non-functional aspects, our team employed a multi-faceted approach to ensure the system's robustness and user satisfaction. We prioritized usability through peer reviews and assessments to gather valuable feedback on the system's user interface and overall user experience. By soliciting input from actual users, we gained insights into their preferences, pain points, and suggestions for improvement. This iterative feedback loop allowed us to fine-tune the system's usability, making it more intuitive and user-friendly.

5. Discussion [Limitations & Future Plans]

5.1. Challenges and Limitations (e.g., time constraints, integration, debugging, teamwork)

One notable shortcoming in the Patient Tracker System is the absence of real-time notifications, a feature that was originally planned but remained unimplemented due to time constraints. The lack of this functionality limits the system's ability to provide immediate alerts to healthcare providers for new patient submissions or updates to existing records, potentially affecting the responsiveness of healthcare interventions. However, recognizing the importance of security and reliability in handling sensitive patient data, the team decided to prioritize and allocate more resources to reinforce the system's security measures and user experience. This decision ensured a smooth and reliable user experience and data handling. Future iterations of the project will aim to strike a better balance, prioritizing the implementation of real-time notifications while continuing to uphold stringent security and responsiveness standards for the Patient Tracker System.

5.2. Future Development Plans

Looking ahead, our vision for the future of the Patient Tracker System includes the integration of real-time notifications as a pivotal enhancement. We envision implementing a robust notification system that provides healthcare providers with immediate alerts for new patient submissions or updates to existing records. This feature aims to facilitate timely responses, ensuring that healthcare professionals stay informed and can promptly address critical changes in patient status.

5.3. Ethical and Societal Implications

In the development of the Patient Tracker System, we placed a paramount emphasis on ethical considerations and the societal impact of our project. Security was a primary concern, and we implemented robust measures to safeguard patient data, ensuring compliance with privacy regulations and maintaining the highest standards of confidentiality. We took proactive steps to prevent unauthorized access and secure sensitive information, recognising the critical importance of patient privacy in the healthcare domain. Additionally, we prioritized accessibility features within our user interface to ensure that the application caters to a diverse range of users, including those with varying abilities. This commitment to accessibility aligns with the societal principle of inclusivity and underscores our dedication to making healthcare information universally accessible. Moreover, we recognized the potential impact of our system on healthcare efficiency, aiming to contribute positively to the overall quality of patient care and treatment outcomes. By consistently evaluating our development decisions through an ethical lens, we sought to create a Patient Tracker System that not only adheres to ethical standards but also positively contributes to the broader societal landscape.