School Of Computer Science

UNIVERSITY OF PETROLEUM & ENERGY STUDIES, DEHRADUN-248007

Software Requirements Specification (SRS)
For
Cloud Based Emergency Health Record Access System

28-November-2024

Prepared by:
MANVITA GOEL (500094777)-BIGDATA (Non-Hons.)
AFREEN ALI (500096949)- CCVT (Non-Hons.)
HARSH VERMA (500096904)- CCVT (Non-Hons.)

Submitted to:
Dr. Shresth Gupta

Table of Contents

1.Introduction:

In the fast-paced environment of healthcare, timely access to accurate patient information is critical, especially during emergencies. Traditional health record systems often face challenges in providing immediate access to vital patient data, particularly when patients are treated outside their regular healthcare network. This can lead to delays in care, medical errors, and adverse outcome.
To address this issue, we present the **Cloud-Based Emergency Health Record Access System**. This innovative solution leverages the power of cloud technology to ensure that healthcare providers can access essential health records anywhere, anytime. By centralizing patient data in a secure, interoperable cloud platform, the system provides authorized medical personnel with real-time access to critical information such as allergies, medications, and medical history, even in the most urgent situations.

This system is designed to enhance patient safety, improve healthcare outcomes, and streamline the care process by ensuring that crucial information is always available when needed, regardless of geographic or network barriers.

1.1 Purpose of the Project:

The primary purpose of this project is to improve the accessibility and availability of critical patient health records during emergencies. By developing a cloud-based system, the project aims to ensure that healthcare providers can quickly access essential patient information—such as allergies, medications, and past medical history—regardless of the patient's location or their regular healthcare network. This system is designed to enhance the quality and speed of care, minimize medical errors, and potentially save lives during emergencies.

1.2 Target Beneficiary:

Patients:

- Patients are the primary beneficiaries, as the system ensures their critical health information is accessible during emergencies, leading to faster and more accurate medical care.

Healthcare Providers:

- Emergency room doctors, paramedics, and other healthcare professionals will benefit from quick access to essential health records, enabling them to make informed decisions in critical situations.

Healthcare Institutions:

- Hospitals, clinics, and other healthcare facilities will benefit from improved patient outcomes and reduced liability due to better-informed emergency care.

Public Health Systems:

Public health systems will benefit from the integration and interoperability of health records across different healthcare providers, leading to more efficient and coordinated care during emergencies.

1.3 Project Scope:

Development:

- **Cloud Infrastructure:** Design and implement a secure, scalable cloud platform to store and manage patient health records.
- **Data Integration:** Develop mechanisms to integrate data from various Electronic Health Record (EHR) systems into the cloud platform.
- **Access Control:** Implement robust authentication and authorization protocols to ensure that only authorized personnel can access the records.
- **User Interface:** Create user-friendly interfaces for both web and mobile platforms that allow healthcare providers to quickly retrieve relevant patient information.
- **Data Encryption and Security:** Apply end-to-end encryption and other security measures to protect sensitive health data during transmission and storage.
- **Emergency Access Protocols:** Establish protocols that allow for quick and temporary access to records during emergency situations, possibly with a time-limited access key.

Testing and Validation:

- Conduct extensive testing, including security audits and usability tests, to ensure that the system performs effectively and securely in real-world scenarios.
- Validate interoperability with existing EHR systems and other relevant healthcare IT infrastructure.

Deployment and Training:

- Deploy the system in a phased manner across various healthcare settings, including hospitals, emergency rooms, and urgent care centers.
- Provide training and support to healthcare professionals on how to use the system efficiently during emergencies.

Compliance and Legal Considerations:

- Ensure the system complies with relevant healthcare regulations and standards, such as HIPAA (Health Insurance Portability and Accountability Act) in the United States, to protect patient privacy.

Address legal and ethical issues related to emergency access to patient data.

1.4 References:

- **Case Study on EHR in Emergency Care**: Studies showing the impact of EHR systems in emergency care.

  Reference: Smith, J., & Doe, A. (2020). The Impact of Electronic Health Records on Emergency Care: A Case Study. *Journal of Emergency Medicine*, 30(4), 123-130.

- **Health Insurance Portability and Accountability Act (HIPAA): This regulation outlines the requirements for safeguarding patient health information.**

  Reference: U.S. Department of Health and Human Services. (1996). Health Insurance Portability and Accountability Act (HIPAA). Retrieved from hhs.gov.

- **Impact of Interoperability in Healthcare**: Research papers discussing the benefits and challenges of interoperability in healthcare systems.

  Reference: Johnson, R., & Lee, M. (2021). Interoperability in Healthcare: Challenges and Opportunities. *Health Informatics Journal*, 27(3), 456-467.

2. Project Description:

2.1 Data/ Data structure:

1. Data Types:

- **Patient Identification Data:**
    - Name, date of birth, gender, address, and other identifying information.
    - Unique patient ID (possibly linked to national health ID or insurance ID).

- **Medical History:**
    - Summary of past medical conditions, surgeries, hospitalizations, and treatments.
    - Chronic conditions (e.g., diabetes, hypertension).

- **Allergies:** ○ Known allergies to medications, foods, or other substances. ○ Severity and type of allergic reactions.

- **Emergency Contact Information:** ○ Contact details for next of kin or designated emergency contacts.

- **Access Logs:**
    - Records of who accessed the health records, when, and what information was viewed.

2. Data Structure:

- **Relational Database:**

  ○ Structured data stored in tables with relationships between them (e.g., patients, medications, allergies, lab results).

2.2 SWOT Analysis:

Strengths:

- **Improved Emergency Care:** This system ensures that critical health information is available during emergencies, leading to better-informed and quicker medical decisions.
- **Accessibility:** Healthcare providers can access patient records from anywhere, ensuring continuity of care even when patients are outside their regular network.
- **Security:** With strong encryption and access control, this system protects sensitive patient data from unauthorized access.
- **Interoperability:** The use of standards like FHIR ensures that the system can integrate with existing EHR systems and healthcare IT infrastructure.

Weaknesses:

- **Data Integration Challenges:** Integrating data from different EHR systems with varying standards and formats can be complex and timeconsuming.

- **Dependency on Internet Access:** In areas with poor internet connectivity, access to the cloud-based system could be limited, potentially impacting its effectiveness during emergencies.
- **High Initial Costs:** Developing and deploying a secure and scalable cloud-based system can be expensive, requiring significant investment in infrastructure and training.
- **Privacy Concerns:** Patients and healthcare providers may have concerns about the security of sensitive health information stored in the cloud.

Opportunities:

- **Expansion to Global Markets:** The system can be adapted for use in different countries, addressing similar needs in emergency healthcare globally.
- **Integration with Emerging Technologies:** The system can be enhanced with AI and different tools of CI/CD for predictive analytics, improving decision-making in emergency care.
- **Patient Empowerment:** Providing patients with controlled access to their own health records could encourage proactive management of their health.
- **Partnerships with Healthcare Providers:** Collaborations with hospitals, clinics, and other healthcare institutions can drive adoption and further development of the system.

Threats:

- **Cybersecurity Risks:** The system could be a target for cyber-attacks, which could lead to data breaches and loss of patient trust.
- **Regulatory Changes:** Changes in healthcare regulations and data protection laws could impact how the system operates and require costly adjustments.
- **Resistance to Change:** Healthcare providers and institutions may be reluctant to adopt a new system, especially if they are accustomed to existing EHR solutions.
- **Technical Failures:** Downtime or technical issues with the cloud platform could hinder access to critical health records during emergencies.

2.3 Project Features:

Task Management:

- Create, assign, and track tasks with due dates and progress indicators.
- Integration with project timelines to manage team workflows.

Real-Time Communication:

- Video conferencing for up to 50 participants.
- Peer-to-peer screen sharing with optional annotations.

Digital Whiteboard:

- Drawing, diagramming, and brainstorming in real time.
- Persistent boards for long-term projects.

Notification System:

- Alerts for task deadlines, meeting schedules, and updates.

- Configurable settings for individual and group notifications.

User-Friendly Dashboard:

- Centralized view of project progress, team activity, and meetings.

Security Features:

- Encrypted data storage and secure communication channels using AES and TLS protocols.
- Multi-factor authentication for user login.

2.4 Design and Implementation Constraints:

Regulatory Compliance:

- **HIPAA and GDPR:** Must comply with data protection regulations like HIPAA (USA) and GDPR (EU).
- **Data Localization:** In some countries, data must be stored locally, affecting cloud storage options.

Integration with Legacy Systems:

- **Interoperability Issues:** Difficulty integrating with older EHR systems that do not support modern standards like FHIR.
- **Data Migration:** Migrating existing records from disparate systems to the cloud.

Security Constraints:

- **Cybersecurity:** This system is robust against hacking, with high levels of encryption and security protocols.
- **User Access Management:** Ensuring secure, role-based access to sensitive information.

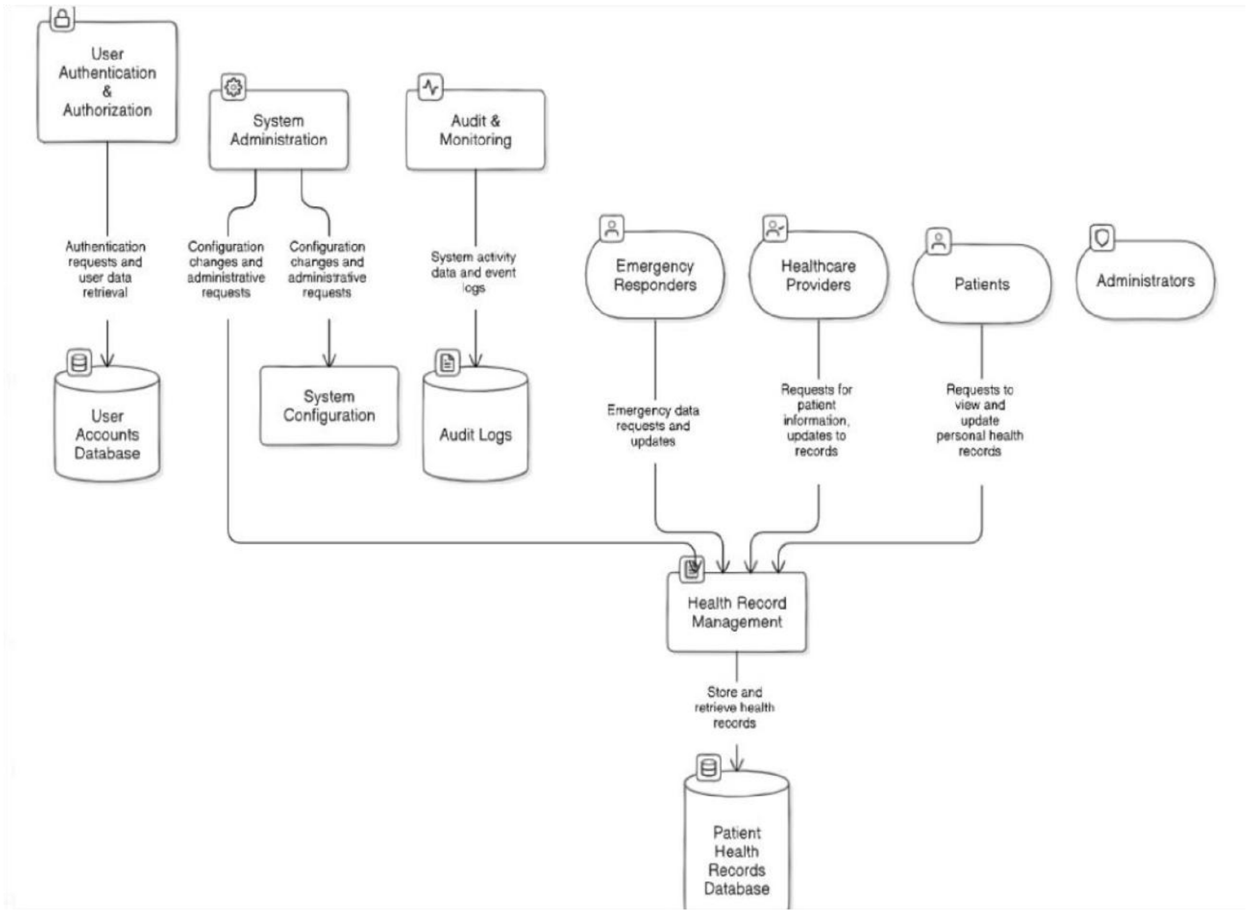
Performance Requirements:

- **Real-time Access:** This system provides rapid access to health records, especially in emergencies.
- **Scalability:** The architecture must handle increasing loads as more users and data are added.

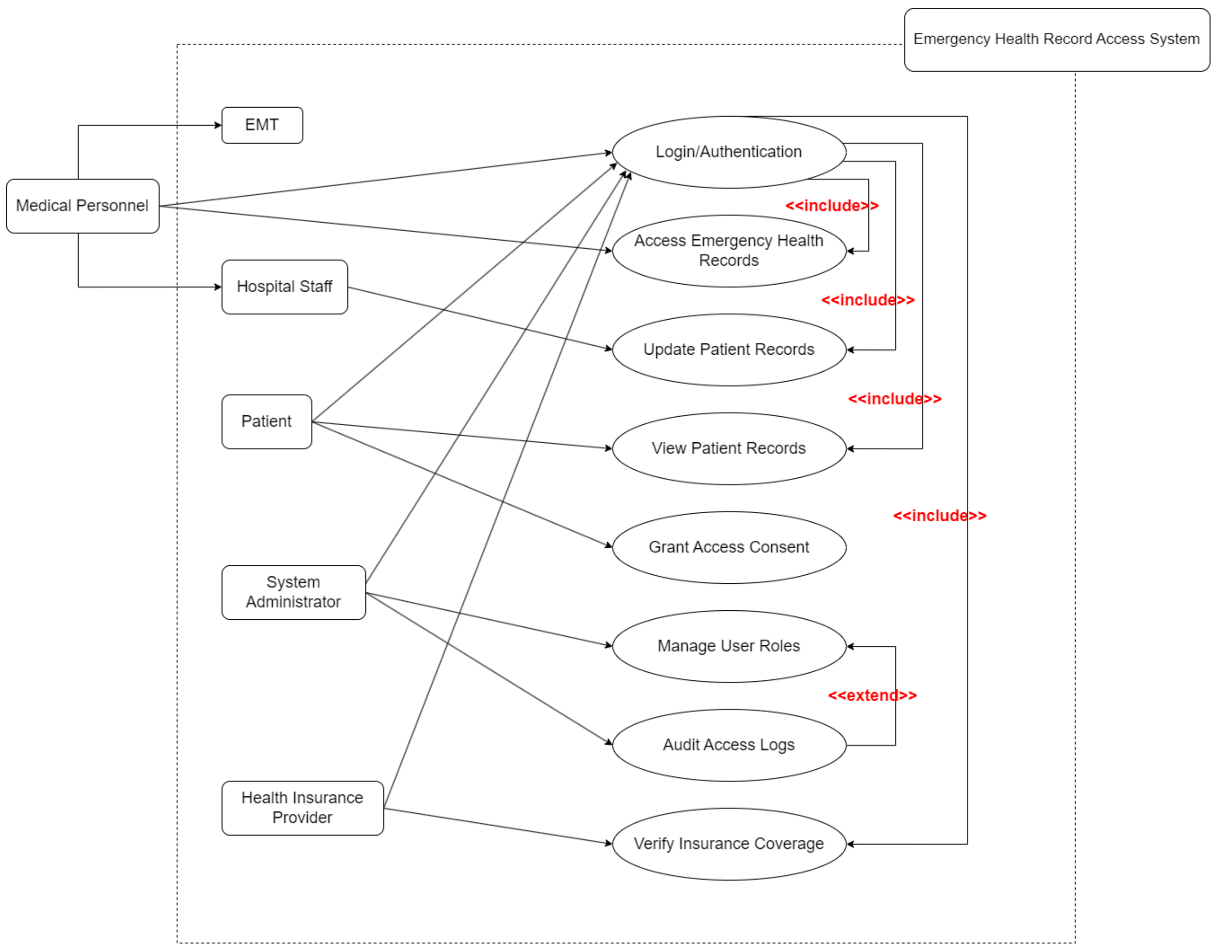
User Experience:

- **Usability:** This system will be easy to use, especially under emergency conditions.
- **Training:** Healthcare providers will require training, which could slow adoption.

Data Flow Diagram (DFD)


User Authentication & Authorization
System Administration
Audit & Monitoring
Authentication requests and user data retrieval
Configuration changes and administrative requests
Configuration changes and administrative requests
System activity data and event logs
Emergency Responders
Healthcare Providers
Patients
Administrators
User Accounts Database
System Configuration
Audit Logs
Emergency data requests and updates
Requests for patient information, updates to records
Requests to view and update personal health records
Health Record Management
Store and retrieve health records
Patient Health Records Database

Use Case diagram



3. System Requirements:

3.1 Functional Requirements:

- **User Authentication**: This system supports secure user authentication mechanisms, including multi-factor authentication (MFA) for healthcare providers and system administrators.
- **Patient Data Retrieval**: Authorized users will be able to retrieve and view critical patient records, including allergies, medications, and medical history, in real-time.
- **Emergency Access**: This system allows for expedited access to patient records during emergencies, bypassing certain non-critical security checks, but still ensuring data protection.
- **Data Synchronization**: This system synchronize data with various Electronic Health Record (EHR) systems in real-time or near real-time to ensure that the most up-to-date information is available.
- **Audit Logging**: All access to patient data must be logged, including the time of access, the data retrieved, and the identity of the user.
- **Role-Based Access Control (RBAC)**: Different levels of access to be defined for users based on their roles (e.g., doctors, nurses, admin staff) to ensure that they only have access to the data necessary for their role.

3.2 Hardware Requirements:

- **Servers:**
  - High-performance cloud servers with sufficient CPU, memory, and storage to handle large volumes of data and concurrent user access.
- **Network Infrastructure:**
  - Reliable, high-speed internet connectivity to ensure uninterrupted access to the cloud system.
- **Security Devices:**
  - Firewalls, Intrusion Detection Systems (IDS), and other security appliances to protect the cloud infrastructure.

User Interface: The interface should be intuitive and user-friendly, allowing healthcare providers to quickly locate and access necessary information.

Software Interface:

The **Software Interface** defines how different components of EHR interact internally and externally to provide a cohesive experience. The focus is on seamless integration, modularity, and efficiency.

1. **Frontend Interface**:
   - Built with **React** to ensure a dynamic and responsive user interface.
   - Modular components for easy updates and maintenance.

    - Interaction with APIs through well-defined endpoints for data retrieval and updates.

2. **Backend Interface**:
    - Implements a REST API for non-real-time operations like user authentication, file uploads, and task management.
    - Backend services are built with TypeScript for better code structure and error handling.
3. **Database Interface**:
    - A powerful Python web framework used for backend development, providing a robust and scalable foundation for the EHR system.
    - Efficient querying and indexing for handling large datasets.
    - A reliable and efficient open-source relational database used for storing patient and system data securely.
4. **Error Handling and Logging**:
    - Centralized logging mechanism for tracking application errors and user activity.
    - User-friendly error messages to guide users in case of issues.
    - Regular diagnostics to monitor performance and pinpoint bottlenecks.
5. **Scalability and Performance**:
    - Server-side load balancing to ensure uninterrupted access during peak usage.
    - Modular codebase allows scaling features and adding functionalities with minimal disruptions.
6. **API Documentation**:
    - Comprehensive documentation for APIs to assist developers in extending the platform or integrating external services.

4. Non-functional Requirements:

4.1 Performance Requirements

1. **System Responsiveness:**
   - **General Operations:** The system must respond to common user actions (e.g., viewing patient records, updating data) within **1 second on average**.
   - **Complex Operations:** Tasks like generating detailed reports or querying large datasets must complete in under **5 seconds**.
2. **Scalability:**
   - **Concurrent Users:** Support up to **1,000 concurrent users** for routine activities such as viewing, editing, and querying records.
   - **Peak Load:** Dynamic scaling to accommodate up to **10,000 users** during peak periods (e.g., pandemic scenarios).
3. **Database Performance:**
   - PostgreSQL must handle up to **10,000 read and write operations per second** under normal loads, leveraging indexes and optimized queries.
   - Implement connection pooling to support high concurrency.
4. **API Throughput:**
   - REST API calls must process within **200 ms** for simple queries and **500 ms** for complex data processing.
   - Rate-limiting ensures fair use while protecting the system from API abuse.
5. **Load Handling:**
   - Stress tests simulate peak loads to validate system performance. The system must maintain at least **95% uptime** during testing.
   - Use caching mechanisms (e.g., Redis) to reduce database load for frequently accessed data.
6. **Data Throughput:**
   - File uploads/downloads (e.g., imaging files) must support **50 Mbps transfer speed** on a standard broadband connection.
   - Implement data compression techniques for large files like DICOM images to enhance performance.
7. **Error Recovery:**
   - Recovery from unexpected server crashes or errors within **15 seconds**.
   - Implement transactional logging in PostgreSQL to ensure no data loss during failures.

4.2 Security Requirements

1. **Data Encryption:**
   - **Data at Rest:** Encrypt using **AES-256** to protect patient information in PostgreSQL.
   - **Data in Transit:** Enforce **TLS 1.3** for all communications via REST APIs.
2. **Authentication and Authorization:**
   - Use **OAuth 2.0** and **JWT tokens** for API authentication.
   - **Multi-Factor Authentication (MFA)** required for all logins, particularly for users accessing sensitive data.
3. **Role-Based Access Control (RBAC):**
   - Define user roles such as **Admin**, **Physician**, **Nurse**, and **Patient**, with granular permissions.

    - Log all access attempts and actions for auditing purposes.
4. **Compliance:**
    - Adhere to **HIPAA**, **GDPR**, and **CCPA** standards for managing patient data.
    - Regular security audits to ensure compliance with regulatory frameworks.
5. **Vulnerability Mitigation:**
    - Frequent **penetration testing** to detect vulnerabilities in the React front-end and REST APIs.
    - Apply security patches to dependencies used in both the back-end and front-end frameworks.
6. **Secure Backup and Disaster Recovery:**
    - Daily backups with **encrypted storage** and retention for **90 days**.
    - Recovery time objective (RTO): **30 minutes**; Recovery point objective (RPO): **5 minutes**.

4.3 Software Quality Attributes

1. **Reliability:**
    - Ensure **99.9% uptime**, translating to no more than **8.76 hours of downtime per year**.
    - Failover strategies include redundant PostgreSQL clusters and load balancers.
2. **Usability:**
    - Intuitive UI built with React, adhering to **WCAG 2.1** accessibility standards.
    - Comprehensive training materials and guided walkthroughs for new users.
3. **Interoperability:**
    - Support for **FHIR** and **HL7 standards** to integrate with external healthcare systems.
    - REST APIs allow seamless integration with third-party analytics tools and patient portals.
4. **Maintainability:**
    - Modular codebase with reusable components in React and clear documentation for REST APIs.
    - CI/CD pipelines for automated testing and deployment.
5. **Scalability:**
    - Horizontal scaling of PostgreSQL instances and React front-end to handle increasing load.
    - Kubernetes-based orchestration for dynamic scaling.
6. **Portability:**
    - Optimized for all modern browsers (Chrome, Firefox, Edge, Safari).
    - Responsive design for seamless use on desktops, tablets, and smartphones.
7. **Performance:**
    - Optimize PostgreSQL queries using indexing and caching strategies.
    - Utilize React's lazy loading and state management for faster front-end rendering.
8. **Security:**
    - Implement intrusion detection systems (IDS) and real-time monitoring for threats.
    - Secure all API endpoints with robust authentication mechanisms.
9. **Efficiency:**
    - Lightweight React components and optimized PostgreSQL queries minimize resource usage.
    - Background tasks like ETL operations run during off-peak hours to conserve

system resources.

Glossary

The following terms and abbreviations are used throughout the document for the EHR system:

1. **EHR (Electronic Health Record):**
   A digital system designed to store, retrieve, and manage patient medical records, providing seamless access to patient history for healthcare providers.
2. **PostgreSQL:**
   An open-source relational database system used for managing and storing patient records, appointment schedules, and medical data.
3. **React:**
   A JavaScript library used for building the user interface of the EHR system, ensuring fast and interactive web applications.
4. **REST API (Representational State Transfer Application Programming Interface):**
   A protocol used for communication between the front-end and back-end systems, enabling secure and efficient data exchange.
5. **FHIR (Fast Healthcare Interoperability Resources):**
   A standard framework for exchanging healthcare information electronically to ensure interoperability between EHR systems.
6. **RBAC (Role-Based Access Control):**
   A method for restricting access based on a user's role, ensuring that only authorized personnel can view or modify sensitive patient data.
7. **AES-256 (Advanced Encryption Standard):**
   A high-security encryption standard used to safeguard stored and transmitted patient data in the EHR system.
8. **HL7 (Health Level Seven):**
   A set of international standards for transferring clinical and administrative healthcare data.
9. **HIPAA (Health Insurance Portability and Accountability Act):**
   U.S. legislation that ensures the confidentiality and security of healthcare data in digital systems.
10. **TLS (Transport Layer Security):**
    A cryptographic protocol that secures communication over networks, used for encrypting data transmitted via REST APIs.
11. **OAuth2:**
    An authentication protocol enabling secure access to the EHR system through external platforms like Google or Microsoft accounts.
12. **DICOM (Digital Imaging and Communications in Medicine):**
    A standard for managing and sharing medical imaging data, such as X-rays and MRIs, used within the EHR system.

Analysis Model:

The EHR system architecture and components are designed to provide secure, scalable, and efficient healthcare record management and collaboration. Below is an outline of the system's analysis model:

1. System Overview

- The EHR system is a web-based platform enabling healthcare providers to manage patient records, appointments, prescriptions, and imaging data.
- It operates on a client-server model, where React-based clients interact with PostgreSQL databases via REST APIs.

2. Components Overview

Frontend (Client-Side):

- **Frameworks Used:**
    - Built with **React** for fast rendering and a responsive UI.
    - State management achieved through React's Context API or Redux for larger applications.
- **Features:**
    - Secure login and patient search.
    - Interactive charts for patient vitals, appointment management, and secure messaging.
    - Mobile-friendly design for healthcare providers on the move.

Backend (Server-Side):

- **Frameworks Used:**
    - REST APIs built using Node.js and TypeScript.
    - PostgreSQL for data storage with features like indexing for faster queries.
- **Responsibilities:**
    - Handles user authentication and authorization (OAuth2, RBAC).
    - Manages medical data storage and ensures compliance with standards like HIPAA.
    - Provides endpoints for patient data retrieval, medical imaging, and billing information.

3. Data Flow Model

1. **User Authentication:**
    - Users log in using credentials or OAuth2, which validates their identity and assigns roles via RBAC.
2. **Patient Records:**
    - Requests for patient records are processed via REST APIs, retrieving encrypted data from PostgreSQL.
3. **Imaging Integration:**

    - DICOM images are stored in PostgreSQL or cloud storage, accessible through APIs for diagnostic review.
4. **Real-Time Updates:**
    - Appointment changes, lab results, and medication updates are synchronized across users in real-time.

4. Security Model

- **Encryption:**
    - **TLS** secures data in transit.
    - **AES-256** ensures data at rest is protected.
- **Role-Based Access Control:**
    - Sensitive actions (e.g., modifying records) restricted based on user roles (e.g., doctor, nurse, admin).
- **Compliance:**
    - System adheres to **HIPAA**, **GDPR**, and other regional healthcare standards.
- **Auditing:**
    - Logs all access attempts, changes to patient data, and administrative actions for accountability.

Appendix C: Issues List

1. **Performance Issues**
    - **Query Optimization:**
      Some database queries for large patient datasets take longer than expected.
        - **Solution:** Optimize PostgreSQL queries with indexing and partitioning.
    - **Peak Load Management:**
      Delays in data retrieval during high user loads.
        - **Solution:** Implement connection pooling and load balancing.
2. **Security Issues**
    - **API Vulnerabilities:**
      REST APIs are vulnerable to brute-force attacks.
        - **Solution:** Use rate-limiting and implement robust authentication mechanisms.
    - **Data Leakage Risks:**
      Potential risks in medical imaging uploads.
        - **Solution:** Validate and sanitize all file inputs and monitor file access logs.
3. **Usability Issues**
    - **Overwhelming UI for New Users:**
      The UI is complex for first-time users.
        - **Solution:** Add onboarding walkthroughs, tooltips, and video tutorials.
    - **Limited Mobile Optimization:**
      Some features do not work seamlessly on mobile devices.
        - **Solution:** Refactor UI components to prioritize mobile performance.
4. **Integration Issues**
    - **FHIR Standard Implementation:**
      Delays in ensuring complete FHIR compatibility for third-party interoperability.
        - **Solution:** Incremental updates for full compliance with FHIR specifications.

5. **Future Enhancements**
    - **AI Assistance:**
      Introduce predictive analytics for patient care recommendations.
        - **Solution:** Plan machine learning model integration using historical data.
    - **Dedicated Mobile App:**
      Build native mobile applications for iOS and Android.
        - **Solution:** Begin development with core functionalities like scheduling and messaging.