Project Description and Detailed Document for Hackathon Submission

Project Title: Secure Healthcare Application with Real-Time Monitoring and Alerts

1. Introduction

1.1. Problem Statement and Project Overview

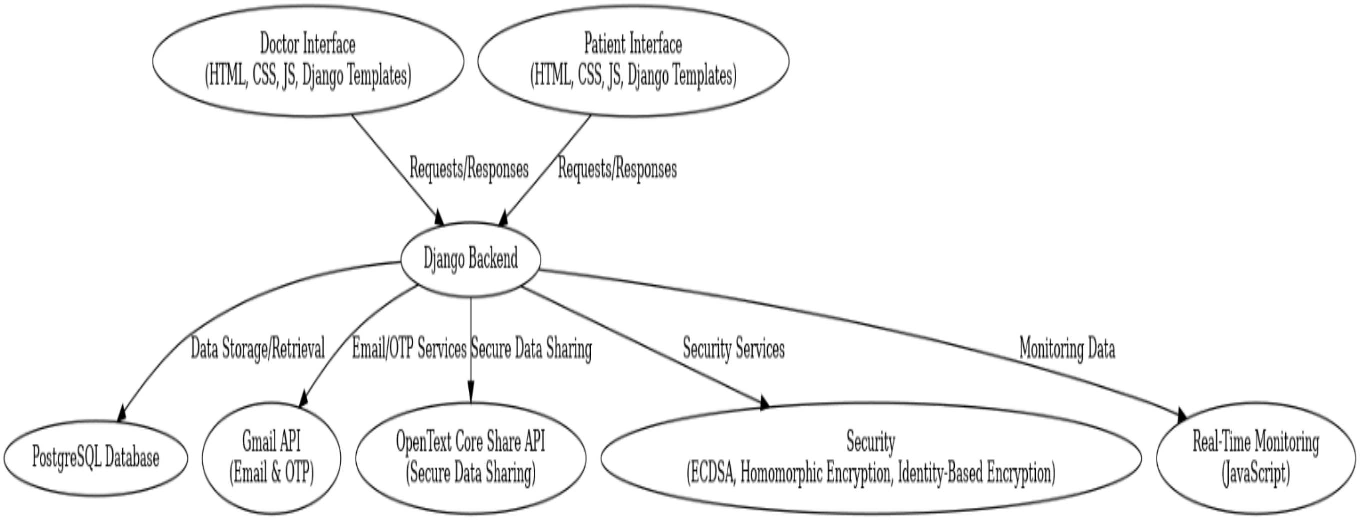
The healthcare sector faces several challenges in data management, including data availability and access, data quality and integrity, data security and privacy, and real-time data processing and analysis. This healthcare application aims to address these challenges by providing a secure and efficient platform for managing patient prescriptions and monitoring vital signs. The application ensures data privacy and integrity through advanced cryptographic techniques, real-time alerts for abnormal vital signs, and seamless communication between patients and doctors.

1.2. Key Features

Doctor and Patient Interface: Separate portals for doctors and patients to manage and view prescriptions.
Real-Time Monitoring: Continuous monitoring of patient vitals.
Alert System: Email alerts for abnormal vital signs or device malfunctions.
Security Measures: Use of ECDSA, homomorphic encryption, and identity-based encryption.
Authentication: Email-based login authentication.
Data Sharing: Secure sharing of educational videos, medical documents, and reports using OpenText Core Share API.

2. System Architecture

2.1. Architecture Diagram



2.2. Components Overview

Frontend: Developed using HTML, CSS, JavaScript, and Django templates.
Backend: Django framework for handling business logic and interactions.
Database: PostgreSQL for storing user data, prescriptions, and vitals.
Email Service: Gmail API for sending emails and OTPs.
Security: ECDSA for integrity, homomorphic encryption for secure computations, and identity-based encryption for data protection.
Monitoring and Alerts: Real-time data processing and alert system using Gmail API and OpenText Core Share API.

3. Detailed Functionality

3.1. Doctor Interface

Prescription Management: Doctors can add, update, and view patient prescriptions.
Vital Sign Monitoring: Doctors can view patient vitals in real-time.
Alerts: Receive email alerts for abnormal patient vitals.
Data Sharing: Securely upload and share educational videos and medical documents with patients.

3.2. Patient Interface

Prescription Access: Patients can view their prescriptions securely.
Vital Sign Monitoring: Patients can view their vitals in real-time.
Alerts: Receive email alerts for abnormal vitals or device malfunctions.
Data Access: Access shared educational videos and medical documents.

3.3. Security Measures

ECDSA: Ensures data integrity and authenticity.
Homomorphic Encryption: Allows secure computations on encrypted data.
Identity-Based Encryption: Protects patient data by encrypting it with the patient's identity.

3.4. Email and OTP Service

Gmail API: Used for sending emails and OTPs for login authentication.

4. Integration with OpenText API

4.1. Selected OpenText API: OpenText Core Share API

Purpose: Enhance communication and data sharing between doctors and patients by allowing secure sharing of educational videos, Additional medical documents, and reports.
Integration Plan:
- API Setup: Configure the OpenText Core Share API in the Django application.
- Usage: Enable doctors to securely upload and share videos and medical documents with patients.
- Security: Ensure the shared documents and videos are encrypted and secure.

5. Implementation Details

5.1. Django Backend

Models: Define models for users (doctors and patients), prescriptions, and vitals.
Views: Create views for handling user interactions and API requests.
Forms: Implement forms for inputting prescriptions, user details, and uploading documents.

5.2. Frontend

Templates: Design Django templates for doctor and patient interfaces.
JavaScript: Implement real-time updates for vital signs monitoring.

5.3. Security

ECDSA: Integrate ECDSA for signing and verifying data integrity.
Homomorphic Encryption: Implement libraries for performing secure computations.
Identity-Based Encryption: Use identity-based encryption for protecting patient data.

5.4. Email Service

Gmail API Integration: Set up Gmail API for sending emails and OTPs.
Email Alerts: Configure email alerts for abnormal vitals and device malfunctions.

5.5. OpenText Core Share API Integration

File Upload: Implement functionality for doctors to upload educational videos and medical documents.

File Sharing: Enable sharing of uploaded files with patients securely.
File Access: Allow patients to access shared files securely through the application.

6. Addressing Problem Statement Challenges

6.1. Data Availability and Access

OpenText Core Share API: Ensures secure and efficient sharing of educational videos and additional medical documents .

6.2. Data Quality and Integrity

ECDSA: Ensures the integrity and authenticity of data.
Identity-Based Encryption: Protects patient data by encrypting it with the patient's identity.

6.3. Data Security and Privacy

Homomorphic Encryption: Allows secure computations on encrypted data without exposing it.
OpenText Core Share API: Ensures secure file sharing and storage.

6.4. Real-Time Data Processing and Analysis

Real-Time Monitoring: Continuous monitoring and processing of patient vitals.
Alert System: Immediate alerts for abnormal vital signs or device malfunctions.

7. Project Challenges and Solutions

7.1. Challenges

Data Security: Ensuring the highest level of security for sensitive patient data.
Real-Time Processing: Handling real-time data processing for vitals monitoring.
User Authentication: Implementing secure and efficient user authentication mechanisms.
Data Sharing: Securely sharing large files like videos and documents between doctors and patients.

7.2. Solutions

Advanced Cryptography: Use of ECDSA, homomorphic encryption, and identity-based encryption.
Efficient Algorithms: Optimizing algorithms for real-time data processing.
Gmail API: Utilizing Gmail API for secure and reliable email services.
OpenText Core Share API: Securely sharing large files between doctors and patients.

8. Future Enhancements

Mobile Application: Develop a mobile version of the application for greater accessibility.
AI Integration: Incorporate AI for predictive analysis of patient vitals.
Expanded API Use: Explore additional OpenText APIs for enhanced communication and data sharing.

9. Conclusion

This healthcare application provides a secure, efficient, and user-friendly platform for managing patient prescriptions and monitoring vitals. By leveraging advanced cryptographic techniques, reliable email services, and the OpenText Core Share API, it ensures data privacy, integrity, and real-time alerting, addressing critical needs in modern healthcare.