DEVELOPMENT OF A SECURE HEALTHCARE MANAGEMENT SYSTEM UTILIZING BLOCKCHAIN TECHNOLOGY FOR ENCRYPTED PATIENT DATA TRANSMISSION

A PROJECT REPORT

Submitted by

DHARSHINI B

ELAKKIYA S

GAJALAKSHMI S

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2025

DEVELOPMENT OF A SECURE HEALTHCARE MANAGEMENT SYSTEM UTILIZING BLOCKCHAIN TECHNOLOGY FOR ENCRYPTED PATIENT DATA TRANSMISSION

A PROJECT REPORT

Submitted by

DHARSHINI B [211421104058]

ELAKKIYA S[211421104069]

GAJALAKSHMI S[211421104070]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2025

BONAFIDE CERTIFICATE

Certified that this project report **"DEVELOPMENT OF A SECURE HEALTHCARE MANAGEMENT SYSTEM UTILIZING BLOCKCHAIN TECHNOLOGY FOR ENCRYPTED PATIENT DATA TRANSMISSION"** is the bonafide work of "**DHARSHINI B (211421104058) ELAKKIYA S (211421104069) GAJALAKSHMI S(211421104070).**"who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.L.JABASHEELA M.E., Ph.D.,	**Dr.KAVITHA SUBRAMANI M.E…,PhD.,**
HEAD OF THE DEPARTMENT	**SUPERVISOR, PROFESSOR**
Department of CSE, **Panimalar engineering college,** **Nasarathpettai,** **Poonamallee,** **Chennai – 123**	**Department of CSE,** **Panimalar Engineering College,** **Nasarathpettai,** **Poonamallee,** **Chennai – 123**

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on......................................

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We **DHARSHINI B(211421104058) , ELAKKIYA S(211421104069)** , **GAJALAKSHMI S(211421104070)** hereby declare that this project report titled "**DEVELOPMENT OF A SECURE HEALTHCARE MANAGEMENT SYSTEM UTILIZING BLOCKCHAIN TECHNOLOGY FOR ENCRYPTED PATIENT DATA TRANSMISSION**" under the guidance of **Mrs.S.T.SANTHANALAKSHMI M.Tech(Ph.D)** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

DHARSHINI B

ELAKKIYA S

GAJALAKSHMI S

ACKNOWLEDGEMENT

Our profound gratitude is directed towards our esteemed Secretary and Correspondent, **Dr. P. CHINNADURAI, M.A., Ph.D**., for his fervent encouragement. His inspirational support proved instrumental in galvanizing our efforts, ultimately contributing significantly to the successful completion of this project.

We want to express our deep gratitude to our Directors,**Tmt. C. VIJAYARAJESWARI, Dr. C. SAKTHI KUMAR, M.E., Ph.D., and Dr. SARANYASREE SAKTHI KUMAR, B.E., M.B.A., Ph.D.,** for graciously affording us the essential resources and facilities for undertaking of this project.

Our gratitude is also extended to our Principal, **Dr. K. MANI, M.E., Ph.D.,** whose facilitation proved pivotal in the successful completion of this project.

We express our heartfelt thanks to **Dr. L. JABASHEELA, M.E., Ph.D.,** Head of the Department of Computer Science and Engineering, for granting the necessary facilities that contributed to the timely and successful completion of project.

We would like to express our sincere thanks to **Project Coordinator Dr.KAVITHA SUBRAMANI M.E…,PhD.,** and **Project Guide Mrs.S.T.SANTHANALAKSHMI M.Tech(Ph.D)** and all the faculty members of the Department of CSE for their unwavering support for the successful completion of the project.

DHARSHINI B

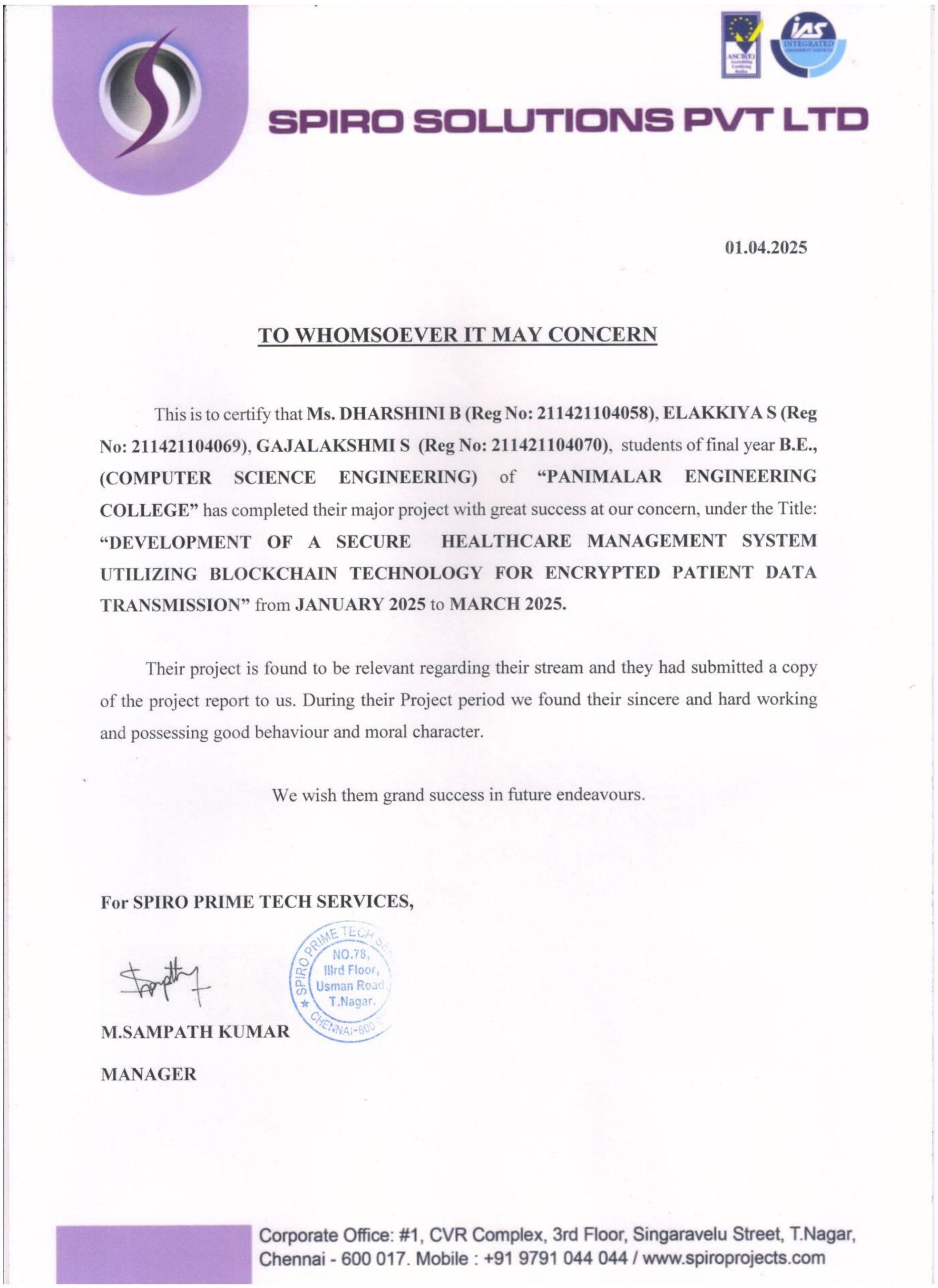
ELAKKIYA S

GAJALAKSHMI S

ABSTRACT

Effective management of healthcare data is essential for ensuring the safety of sensitive patient information in today's digital landscape. An innovative system utilizes the cutting-edge FrodoKEM encryption algorithm along with real-time secure chat features to protect information shared among various hospital departments. FrodoKEM uses adaptive encryption methods and dynamic key management to provide strong data security during transmission. Simultaneously, blockchain-based logging and SHA-256 hashing are employed to preserve data integrity and create an unchangeable audit trail for all accesses and alterations. The secure chat component allows for private, real-time communication between physicians and patients, enhancing telehealth capabilities and overall clinical productivity. Experimental tests indicate that the system delivers low latency and high security performance in fluctuating healthcare environments. By integrating state-of-the-art encryption techniques with secure communication solutions, this approach not only strengthens data confidentiality and integrity but also improves clinical workflows, presenting a promising avenue for future digital healthcare management.

Project completion certificate



SPIRO SOLUTIONS PVT LTD

01.04.2025

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Ms. DHARSHINI B (Reg No: 211421104058), ELAKKIYA S (Reg No: 211421104069), GAJALAKSHMI S (Reg No: 211421104070)**, students of final year **B.E., (COMPUTER SCIENCE ENGINEERING)** of **"PANIMALAR ENGINEERING COLLEGE"** has completed their major project with great success at our concern, under the Title: **"DEVELOPMENT OF A SECURE HEALTHCARE MANAGEMENT SYSTEM UTILIZING BLOCKCHAIN TECHNOLOGY FOR ENCRYPTED PATIENT DATA TRANSMISSION"** from **JANUARY 2025** to **MARCH 2025.**

Their project is found to be relevant regarding their stream and they had submitted a copy of the project report to us. During their Project period we found their sincere and hard working and possessing good behaviour and moral character.

We wish them grand success in future endeavours.

For SPIRO PRIME TECH SERVICES,

[illegible]

SPIRO PRIME TECH SER NO.78, IIIrd Floor, Usman Road, T.Nagar, CHENNAI-600

M.SAMPATH KUMAR

MANAGER

Corporate Office: #1, CVR Complex, 3rd Floor, Singaravelu Street, T.Nagar, Chennai - 600 017. Mobile : +91 9791 044 044 / www.spiroprojects.com

LIST OF FIGURES

LIST OF ABBRIVATIONS

API	Application Programming Interface
CDSS	Clinical Decision Support System
CDW	Clinical Data Warehouse
DW	Data Warehouse
ECC	Elliptic Curve Cryptography
IDE	Integrated Development Environment
LAN	Local Area Network
LWE	Learning With Errors
SDG	Sustainable Development Goals
SDD	Solid State Drive
UFT	Unicode Transformation Format
TLS	Transport Layer Security

CHAPTER-1

INTRODUCTION :

Patients who come to a hospital are first referred to a general doctor. A doctor updates the prescription with the patient's ID and medical notes. Such a prescription is safely shared with other departments: surgery, radiology, and pharmacy, among others. Only by authorized staff, a department-specific secret key can enable access to the prescription. For the purpose of security, a prescription is encrypted using the AES algorithm before sending it to each department. Such information, accessed only by staff in the department, can be viewed to continue the required treatment. Then, patient details are updated in the database. All patient data in the database is encrypted with the SHA-256 algorithm for extra security. Thus, sensitive medical information is shared and stored across departments securely, accessible only to authorized personnel.

1.1 PROBLEM STATEMENT:

In modern healthcare systems, ensuring the security and privacy of patient data is a critical challenge. Traditional hospital management systems rely on centralized cloud storage, which is vulnerable to data breaches, unauthorized access, and privacy violations. Additionally, the lack of secure and efficient data-sharing mechanisms across hospital departments leads to inefficiencies in patient care. As cyber threats evolve, conventional encryption techniques may become insufficient, especially with the potential risks posed by quantum computing. There is a need for a robust and decentralized solution that ensures secure patient data storage, access control, and protection against emerging security threats.

1.2 EXISTING SYSTEM:

The existing healthcare system primarily relies on centralized databases for storing and managing patient records. Hospitals and third-party service providers maintain these databases, making them responsible for data security and accessibility. Patient records, including medical history, prescriptions, and test reports, are stored in digital or physical formats. Access to these records is often restricted to hospital staff, and any modifications or updates are managed within the centralized system. While this system has been widely used for years, it suffers from several limitations that affect data security, integrity, and accessibility.

DISADVANTAGES:

- Centralized databases are highly susceptible to cyberattacks, making patient records prone to hacking, data breaches, and unauthorized access.
- Medical records can be altered, lost, or manipulated due to system failures, cyber threats, or insider attacks, leading to inaccurate or incomplete patient information.
- Patients have minimal control over their data, while unauthorized users may find loopholes to access sensitive medical information.
- Sharing patient records between hospitals and healthcare providers is slow and insecure, causing delays in patient treatment and coordination.
- There is no proper audit mechanism to track who accesses or modifies patient data, increasing the risk of data misuse.
- Since all patient data is stored in a central location, any technical failure, cyberattack, or database corruption can lead to complete data loss or system downtime

CHAPTER-2

LITERATURE SURVEY

TITLE: Achieving Secure, Verifiable, and Efficient Boolean Keyword Searchable Encryption for Cloud Data Warehouse

AUTHOR: Thanaruk Theeramunkong, Somchart Fugkeaw

YEAR: · January 2024

DESCRIPTION:

Cloud data warehouse (CDW) platforms have been offered by many cloud service providers to provide abundant storage and unlimited accessibility service to business users. Sensitive data warehouse (DW) data consisting of dimension and fact data is typically encrypted before it is outsourced to the cloud. However, the query over encrypted DW is not practically supported by any analytical query tools. The Searchable Encryption (SE) technique is palpable for supporting the keyword searches over the encrypted data. Although many SE schemes have introduced their own unique searching methods based on indexing structure on top of searchable encryption techniques, there are no schemes that support Boolean expression queries essential for the search conditions over the DW schema. In this paper, we propose a secure and verifiable searchable encryption scheme with the support of Boolean expressions for CDW. The technical construct of the proposed scheme is based on the combination of Partial Homomorphic Encryption (PHE), B+Tree and Inverted Index, and bitmapping functions to enable privacy-preserving SE with efficient search performance suitable for encrypted DW. To enhance the scalability without requiring a third party to support the verification of search results, we employed blockchain and smart contracts to automate authentication, search index retention, and trapdoor generation. For the evaluation, we conducted comparative experiments to show that our scheme is more proficient and effective than related works.

TITLE: MaxD K-means: A clustering algorithm for Auto-generation of centroids and distance of data points in clusters

AUTHOR: Tutut Herawan, Abul Beg

YEAR: · January 2012

DESCRIPTION:

K-Means is one of the unsupervised learning and partitioning clustering algorithms. It is very popular and widely used for its simplicity and fastness. The main drawback of this algorithm is that user should specify the number of cluster in advance. As an iterative clustering strategy, K-Means algorithm is very sensitive to the initial starting conditions. In this paper, we propose a clustering technique called MaxD K-Means clustering algorithm. MaxD K-Means algorithm auto generates initial k (the desired number of cluster) without asking for input from the user. MaxD K-means also used a novel strategy of setting the initial centroids. The experiment of the Max-D means has been conducted using synthetic data, which is taken from the Llyod's K-Means experiments. The results from the new algorithm show that the number of iteration improves tremendously, and the number of iterations is reduced by confirming an improvement rate is up to 78%

TITLE: Privacy-Preserving Patient-Centric Clinical Decision Support System on Naive Bayesian Classification

AUTHOR: Rongxing Lu, Ximeng Liu , Jianfeng Ma

YEAR: January 2015

DESCRIPTION: Clinical decision support system (CDSS), which uses advanced data mining techniques to help clinician make proper decisions, has received considerable attention recently. The advantages of CDSS include not only improving diagnosis accuracy but also reducing diagnosis time. Specifically, with large amounts of clinical data generated everyday, na¨ ıve Bayesian classification can be utilized to excavate valuable information to improve CDSS. Although CDSS is quite promising, the flourish of CDSS still faces many challenges including information security and privacy concerns. In this paper, we propose a new privacy preserving patient-centric clinical decision support system, called PPCD, which helps clinician complementary to diagnose the risk of patients' disease in a privacy-preserving way. In PPCD, the past patients' historical data are stored in cloud and can be used to train the na¨ ıve Bayesian classifier without leaking any individual patient medical data, and then the trained classifier can be applied to compute the disease risk for new coming patients and also allow these patients to retrieve the top-k disease names according to their own preferences. Specifically, to protect the privacy of past patients' historical data, a new cryptographic tool called additive homomorphic proxy aggregation scheme is designed. Moreover, to leverage the leakage of na¨ ıve Bayesian classifier, we introduce a privacy-preserving top-k disease names retrieval protocol in PPCD. Detailed privacy analysis ensures that patient's information is private and will not be leaked out during the disease diagnosis phase. In addition, performance evaluation via extensive simulation also demonstrates that our PPCD can efficiently calculate patient's disease risk with high accuracy in a privacy-preserving way

TITLE: Handling Privacy-Sensitive Medical Data With Federated Learning: Challenges and Future Directions
AUTHORS: Ons Aouedi, Alessio Sacco, Kandaraj Piamrat, Guido Marchetto
YEAR:2023
DESCRIPTION: This paper explores the application of Federated Learning (FL) in healthcare for handling privacy-sensitive medical data. It discusses how FL allows multiple institutions to collaboratively train machine learning models without sharing raw patient data, ensuring privacy and security. The study highlights challenges such as data heterogeneity, communication overhead, and security vulnerabilities, including inference and poisoning attacks. It also

examines potential solutions, including blockchain integration, differential privacy, and homomorphic encryption, to enhance security and efficiency in federated healthcare systems.

TITLE: An Investigation Into Patient Privacy Disclosure in Online Medical Platforms
AUTHORS: Chun-Lin Feng, Zhi-Chao Cheng, Li-Juan Huang
YEAR:2019
DESCRIPTION: This paper examines the factors influencing patient privacy disclosure in online medical platforms. It explores the role of fairness perceptions—outcome fairness and procedural fairness—on patients' willingness to share personal health information. The study also highlights the impact of perceived platform interactivity on privacy disclosure behaviors. Through a survey of 1,546 users, the research reveals that fairness and interactivity significantly affect patients' decisions to disclose personal data, providing insights for improving privacy management in online healthcare communities.

TITLE: Secure ID Privacy and Inference Threat Prevention Mechanisms for DistributedSystems.
AUTHORS: Tahani Hamad Aljohani, Ning Zhang
YEAR:2023
DESCRIPTION: This paper investigates security and privacy challenges in distributed healthcare systems, focusing on preventing inference attacks and preserving patient identity privacy. It introduces the SPID framework (Secure, ID Privacy, and Inference Threat Prevention Mechanisms), which enhances data security by utilizing a distributed set of servers owned by different service providers. The framework allows patients to securely upload encrypted health data to multiple foreign servers, preventing unauthorized access and inference-based tracking. Key security mechanisms include elliptic curve cryptography

(ECC), anonymous authentication, pseudonym-based access control, and double encryption techniques. The study evaluates SPID using benchmarking tools and queuing theory, demonstrating improved performance and strong protection against security threats.

Title: An Investigation Into Patient Privacy Disclosure in Online Medical Platforms

Author: [Author information not explicitly found—please confirm from the document]

Description: This paper explores the factors influencing patient privacy disclosure on online medical platforms. It examines privacy concerns, user behaviors, and potential risks associated with sharing medical information online. The study discusses regulatory frameworks, data security measures, and user awareness in the context of digital health platforms. Additionally, it investigates the role of platform policies, third-party data access, and the ethical implications of data sharing in telemedicine and digital health services. The research highlights the balance between patient convenience and privacy risks, offering insights into how platforms can enhance security while maintaining user trust.

CHAPTER -3

3.THEORETICAL BACKGROUND:

3.1 IMPLEMENTATION ENVIRONMENT

The implementation of the **Secure Healthcare Management System** is carried out in a robust and scalable environment to ensure efficient execution, security, and ease of integration. The system is developed using modern technologies that provide secure data handling, user authentication, and encrypted communication.

3.1.1 HARDWARE ENVIRONMENT

The following hardware configurations are required to ensure smooth execution of the system:

- **Processor**: Intel Core i5 or higher
- **RAM**: Minimum 8GB DDR4
- **Hard Disk**: At least 250GB SSD for fast data access
- **Network**: Secure LAN/WiFi connectivity for data exchange
- **GPU (Optional)**: For advanced computations if needed

3.1.2 SOFTWARE ENVIRONMENT

To build and deploy the system, the following software components are used:

- **Operating System**: Windows 10 / Linux Ubuntu 20.04
- **Frontend Technologies**:
  - HTML, CSS, JavaScript, Bootstrap (for responsive UI)
- **Backend Technologies**:

  - Java, Spring Boot (for API and business logic)
- **Database Management**:
  - MySQL (for structured data storage)
- **Blockchain Implementation**:
  - Hyperledger Fabric / Ethereum for decentralized storage
- **Security Algorithms**:
  - ForodoKEM for data encryption
  - SHA-256 for hashing sensitive information
- **Development Environment**:
  - **IDE**: Spring Tool Suite (STS) / IntelliJ IDEA
  - **Version Control**: GitHub / GitLab for source code management
  - **Testing Tools**: JUnit for unit testing, Postman for API testing

3.2 SYSTEM ARCHITECTURE

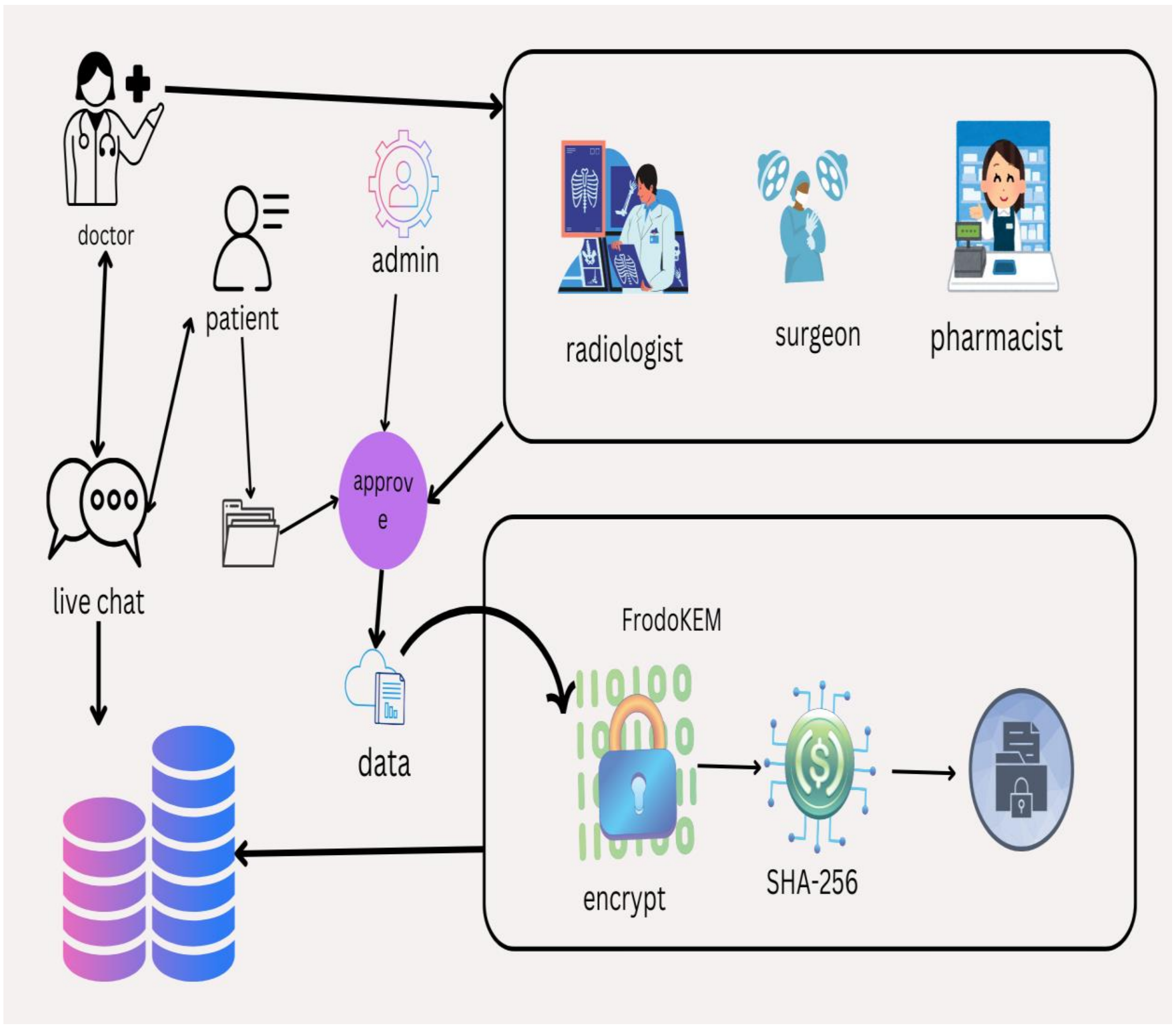


FIG 3.1 System Architecture Diagram

3.3 PROPOSED METHODOLOGY

INPUT DESIGN

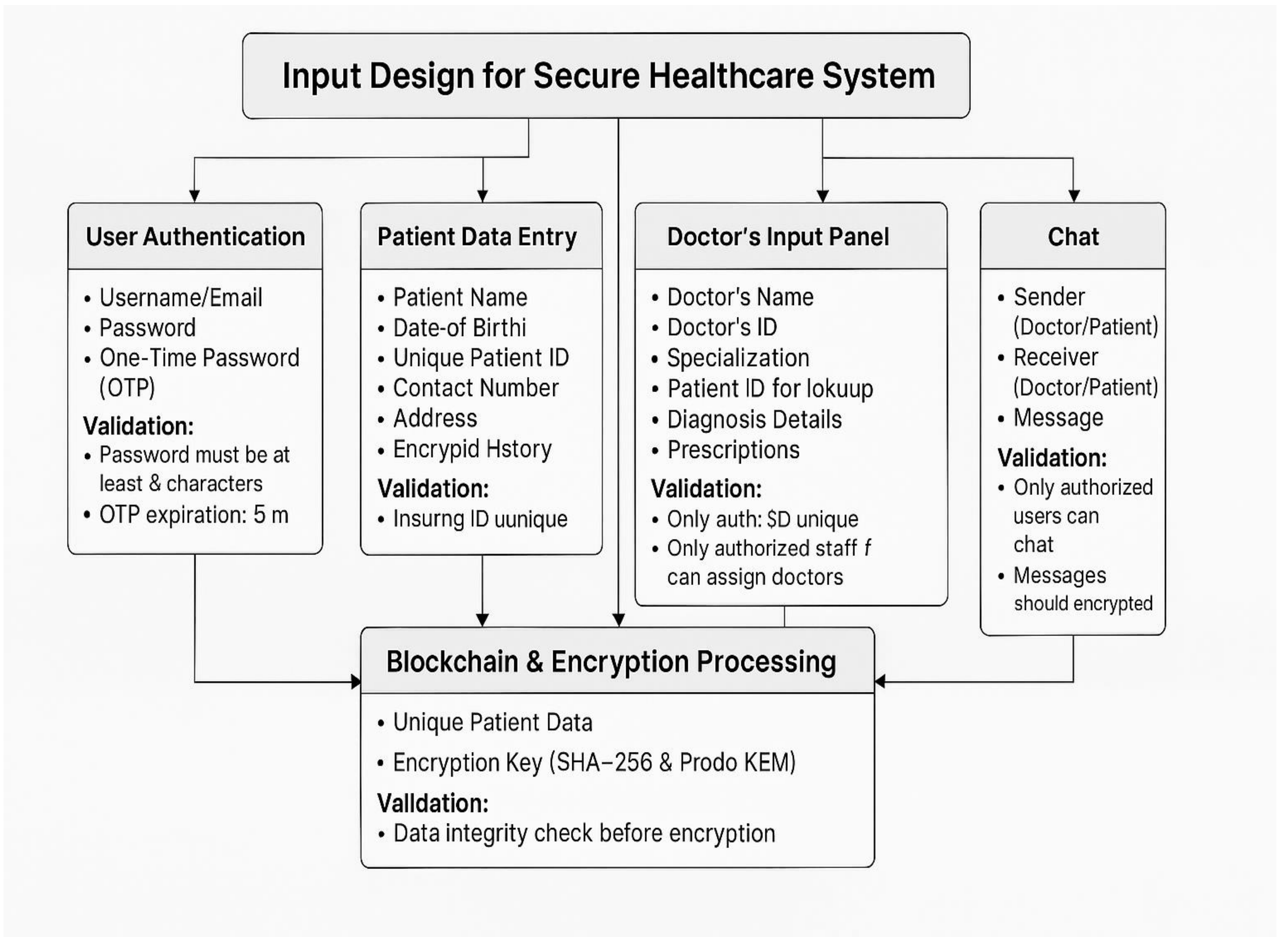


Fig 3.2 Input Design for Secure Healthcare System

3.4 MODULE DESIGN

USECASE

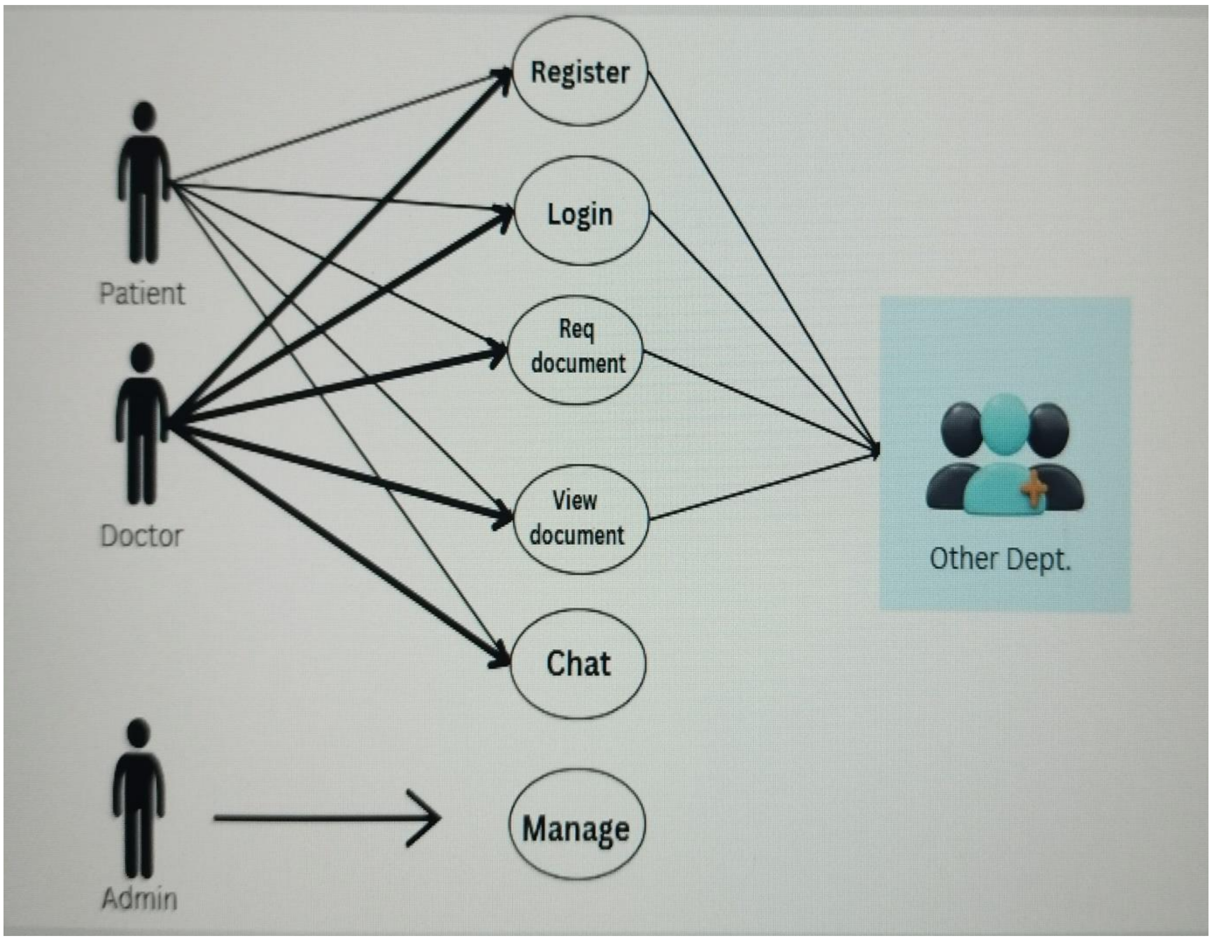


Fig 3.3 Usecase Diagram

ACTIVITY DIAGRAM

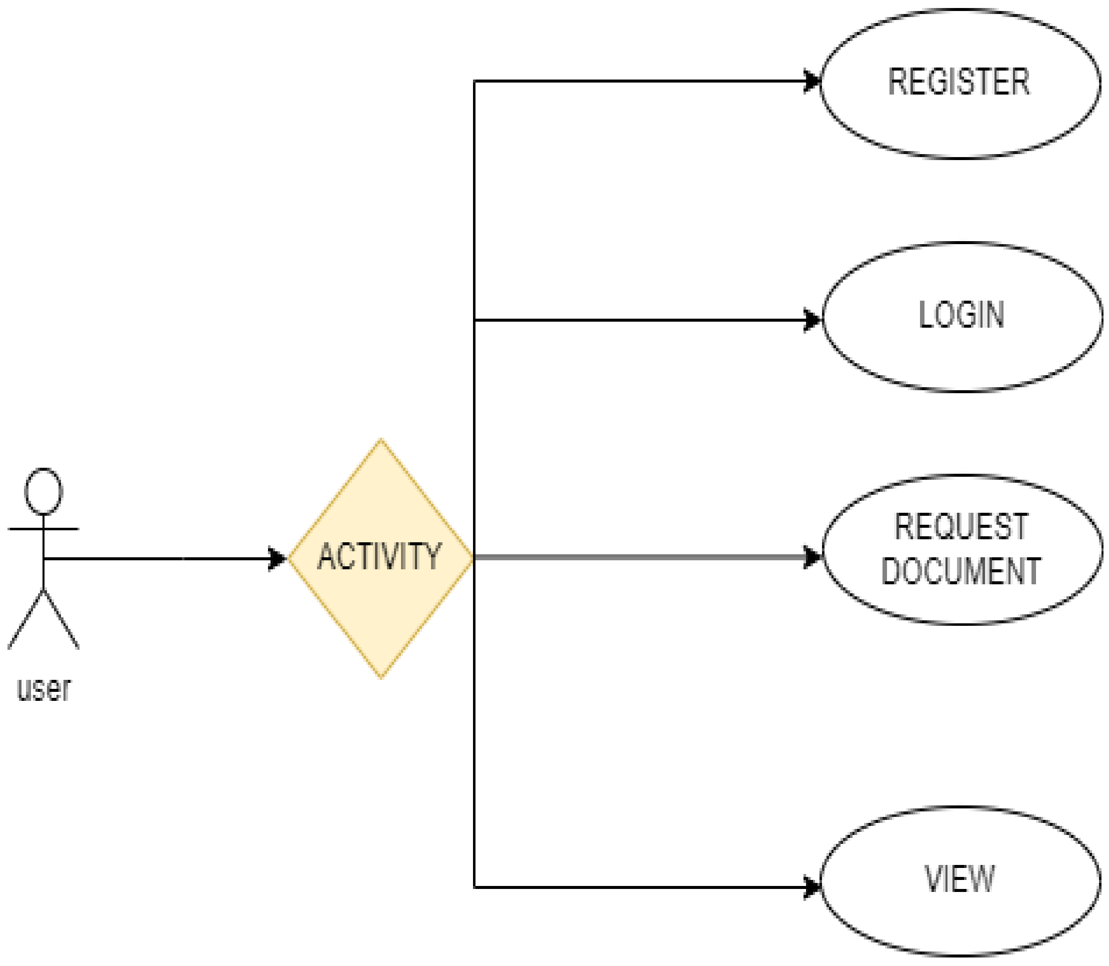


Fig 3.4 Activity Diagram

SEQUENCE DIAGRAM

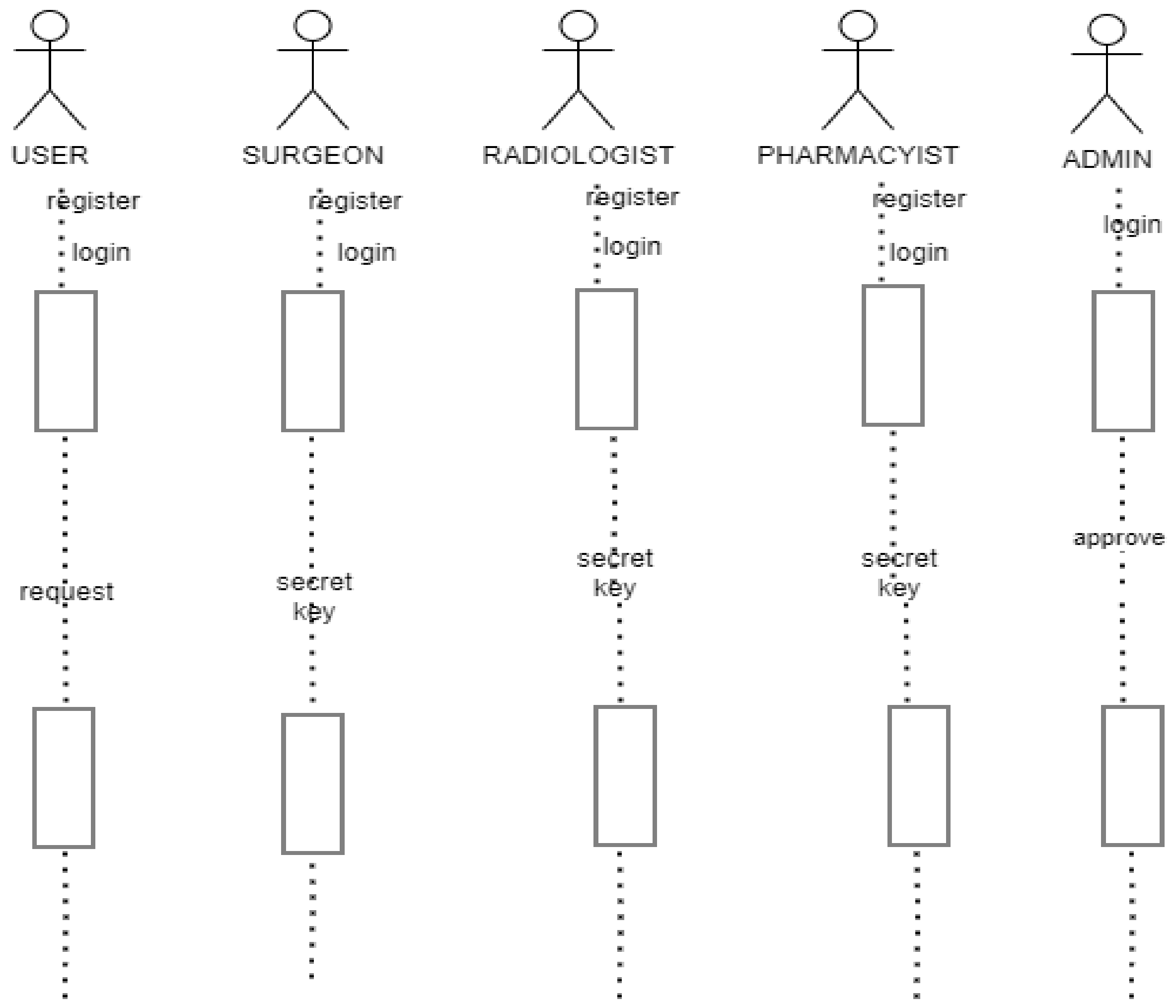


Fig 3.5 Sequence Diagram

ER DIAGRAM

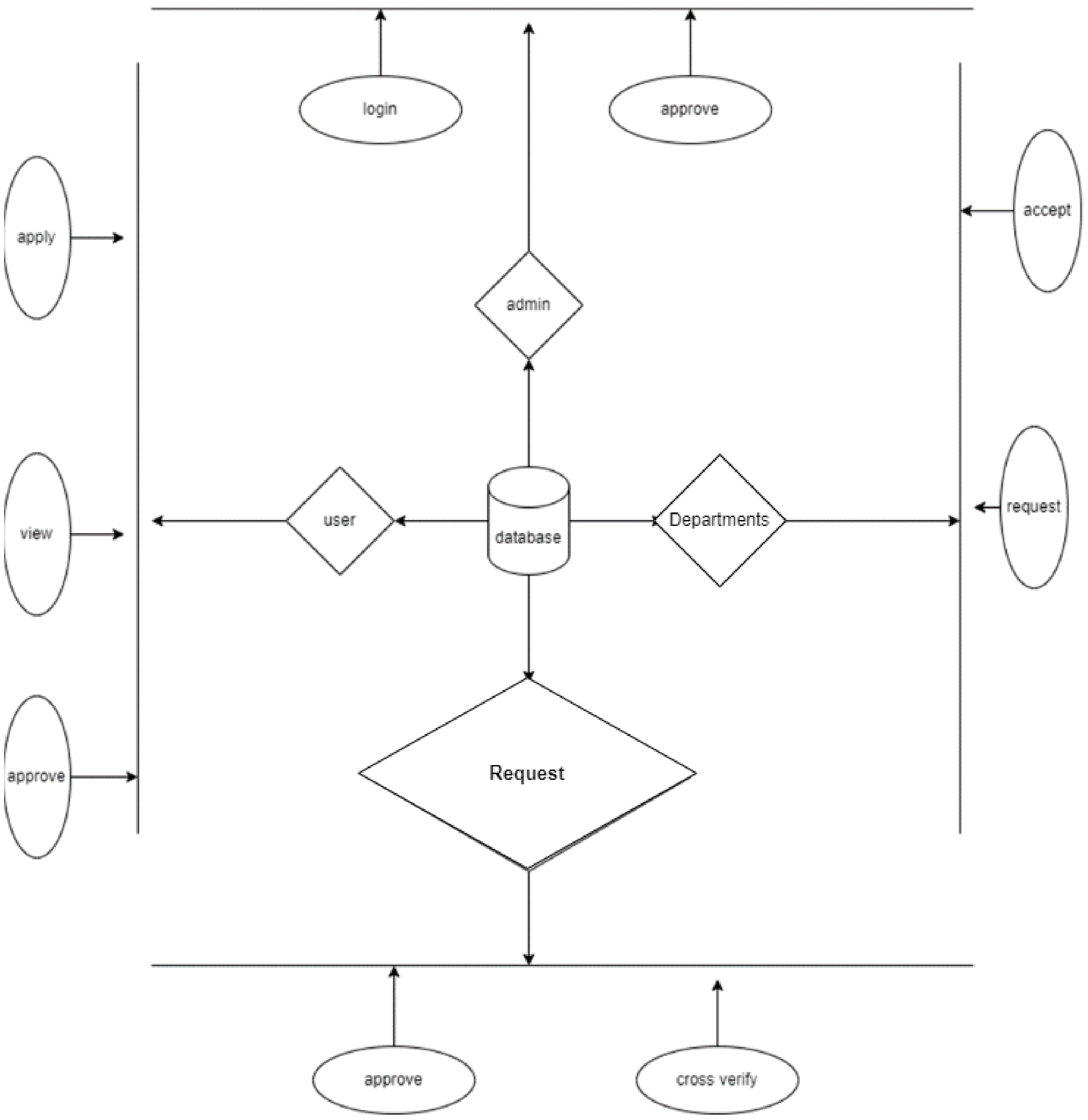


Fig 3.6 ER Diagram

DF DIAGRAM

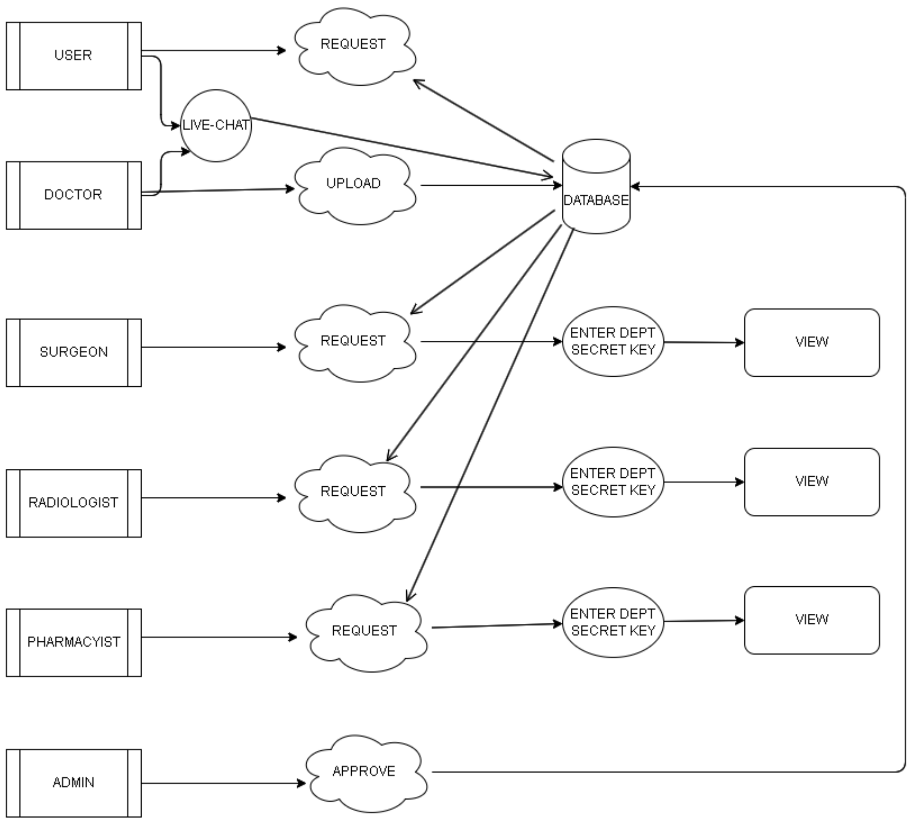


Fig 3.7 DataFlow Diagram

CHAPTER-4

SYSTEM IMPLEMENTATION

4.1.1 USER MODULE:

This is the first module in our project, here symbolizes a unit of work performed within a database management system (or similar system) against a database, and treated in a coherent and reliable way independent of other transactions. A transaction generally represents any change in database user will transfer the amount to provider. They registered with datas and then login

4.1.2 ADMIN MODULE:

This is the first module in our project, here symbolizes a unit of work performed within a database management system (or similar system) against a database, and treated in a coherent and reliable way independent of other transactions. The admin login in to the application, they verify the user application and approve to check it manually in these application.

4.1.3 DOCTORS MODULE:

This is the first module in our project, here symbolizes a unit of work performed within a database management system (or similar system) against a database, and treated in a coherent and reliable way independent of other transactions. A transaction generally represents any change in database user will transfer the amount to provider. They registered with datas and then login

4.1.4 SURGEON MODULE:

This is the first module in our project, here symbolizes a unit of work performed within a database management system (or similar system) against a database, and treated in a coherent and reliable way independent of other transactions. A transaction generally represents any change in database user will transfer the amount to provider. They registered with datas and then login

4.1.5 RADIOLOGIST MODULE:

This is the first module in our project, here symbolizes a unit of work performed within a database management system (or similar system) against a database, and treated in a coherent and reliable way independent of other transactions. A transaction generally represents any change in database user will transfer the amount to provider. They registered with datas and then login

4.1.6 PHARMACIST MODULE:

This is the first module in our project, here symbolizes a unit of work performed within a database management system (or similar system) against a database, and treated in a coherent and reliable way independent of other transactions. A transaction generally represents any change in database user will transfer the amount to provider. They registered with data and then login.

4.1.7 CHAT MODULE:

The Chat Module facilitates secure, real-time communication between patients and healthcare providers. It allows for text, audio, and video interactions, enabling telemedicine capabilities. All communications are protected by end-to-end encryption in accordance with the system's security measures, ensuring that sensitive conversations remain private.

4.2 ALGORITHM DESCRIPTION

4.2.1 SHA 256:

SHA-256 (Secure Hash Algorithm 256-bit) is a cryptographic hash function that generates a unique 256-bit (32-byte) fixed-size hash value for any given input, regardless of its size. Developed by the National Security Agency (NSA) and published by the National Institute of Standards and Technology (NIST), SHA-256 is part of the SHA-2 family of algorithms and is widely used in various security applications and protocols, including SSL/TLS, digital signatures, and blockchain technology. Unlike encryption, SHA-256 is a one-way function; it is designed to be irreversible, meaning that the original input cannot be reconstructed from the hash value. The algorithm works by processing the input data in 512-bit blocks and applying a series of bitwise operations, logical functions, and modular arithmetic to generate the final hash. This ensures that even a small change in the input (e.g., flipping a single bit) results in a completely different hash output, demonstrating the algorithm's high sensitivity to input variations. SHA-256 is known for its robustness and security, making it highly resistant to collision attacks (where two different inputs produce the same hash) and preimage attacks (where an input is derived from a given hash). Due to its efficiency and strong security properties, SHA-256 remains a popular and trusted choice for ensuring data integrity and authenticity in modern cryptographic systems.

4.2.2 ForodoKEM: A Post-Quantum Key Encapsulation Mechanism

FrodoKEM is a **post-quantum cryptographic algorithm** designed for secure key exchange, resistant to attacks from quantum computers. It is based on the **Learning With Errors (LWE)** problem, which involves adding controlled noise to matrix multiplications, making it computationally infeasible to reverse. Unlike structured lattice-based schemes, FrodoKEM uses **unstructured lattices**, providing stronger security but requiring more computational resources.

The algorithm follows three main steps:

- **Key Generation:** The recipient creates a public-private key pair using a randomly chosen matrix and error values.

- **Encapsulation:** The sender encrypts a randomly generated **shared secret key** using the recipient's public key, producing a ciphertext.
- **Decapsulation:** The recipient decrypts the ciphertext with their private key to recover the shared secret key, which is then used for secure communication.

FrodoKEM is highly secure and resistant to both classical and quantum attacks. However, it requires **larger key sizes and higher computational power** compared to traditional methods. Despite this, it is a strong candidate for securing future communications in sensitive fields like **healthcare, finance, and cloud security.**

4.2.3 BLOCKCHAIN TECHNOLOGY IN SECURE HEALTHCARE SYSTEMS:

Blockchain technology is a decentralized and distributed ledger system that ensures secure, transparent, and tamper-proof data management. In this project, blockchain plays a crucial role in securing patient data by providing immutable records and preventing unauthorized access.

Key Features of Blockchain in This System:

- **Decentralization:** Unlike traditional centralized databases, blockchain distributes data across multiple nodes, reducing the risk of single-point failures.
- **Immutability:** Once recorded, patient data cannot be altered or deleted, ensuring data integrity and preventing malicious modifications.
- **Security:** Using cryptographic techniques like **SHA-256 hashing** and **Prodo KEM encryption**, patient records remain protected from cyber threats and unauthorized access.

CHAPTER-5

RESULT AND DISCUSSION

5.1 Performance Parameters / Testing

The performance of the **Blockchain-Based Secure Healthcare System** was evaluated based on key security, efficiency, and scalability metrics to ensure reliability and effectiveness. The primary performance parameters considered include **data integrity, encryption strength, transaction speed, scalability, and resource utilization.**

Data Integrity

Data integrity ensures that patient records remain immutable and unaltered after being stored on the blockchain. This was tested by verifying cryptographic hash values using **SHA-256**, ensuring that any unauthorized modifications would be detected.

Encryption Strength

The security of patient data was assessed using **Prodo KEM and SHA-256** encryption algorithms. Testing involved measuring encryption and decryption times to ensure minimal performance overhead while maintaining strong security.

Transaction Speed

Transaction speed was evaluated by measuring the time taken to store and retrieve patient records on the blockchain. Optimized smart contracts were used to reduce latency and improve system responsiveness.

Scalability and Storage Efficiency

The system's ability to handle large volumes of patient records was assessed by simulating increasing data loads. The impact of blockchain's immutable ledger on storage growth was analyzed, ensuring the system remains efficient without excessive resource consumption.

Resource Utilization

The CPU and memory usage of the system were monitored during encryption, decryption, and transaction processes to prevent excessive computational overhead. Blockchain nodes were optimized to maintain a balance between security and performance.

Testing Methodology

To ensure comprehensive evaluation, multiple testing strategies were employed:

- **Load Testing:** Simulated high volumes of patient data transactions to test system performance under stress.
- **Security Testing:** Assessed vulnerabilities using penetration testing and cryptographic analysis.
- **Real-Time Testing:** Evaluated system performance in a simulated hospital environment to ensure practical applicability.

By focusing on these performance parameters, the **Blockchain-Based Secure Healthcare System** ensures high security, efficiency, and scalability, making it a robust solution for protecting patient data in hospital management systems.

5.2 RESULTS & DISCUSSION:

The implementation of the **Blockchain-Based Secure Healthcare System** successfully enhances data security, integrity, and accessibility for hospitals. The system encrypts patient records using **SHA-256 hashing** and **Prodo KEM encryption**, ensuring that sensitive medical data remains protected from unauthorized access and cyber threats. The key outcomes observed during the implementation and testing phases include:

- **Data Security & Integrity:** Patient records stored on the blockchain remain immutable, ensuring protection against tampering or unauthorized modifications.
- **Efficient Data Management:** The system enables hospitals to securely store, retrieve, and manage patient data, reducing administrative overhead.
- **Enhanced Privacy:** Cryptographic techniques ensure that only authorized personnel can access patient information, preventing data breaches.
- **Decentralized Access:** The blockchain network ensures that data is distributed across multiple nodes, eliminating single points of failure and improving system reliability.
- **Improved Traceability:** Every transaction in the system is logged, enabling hospitals to track data modifications and maintain transparency.

The results demonstrate that blockchain technology provides a **secure and reliable** solution for managing healthcare data. Traditional centralized databases are vulnerable to hacking, data breaches, and single-point failures, whereas blockchain offers a **decentralized and tamper-proof alternative**.

However, the study also highlights certain challenges:

- **Computational Overhead:** Implementing blockchain increases processing time due to encryption and consensus mechanisms.
- **Storage Requirements:** Since blockchain maintains a complete transaction history, data storage can grow significantly over time.
- **Scalability Issues:** As the number of records increases, the network requires more computational power and efficient optimization strategies.

Despite these challenges, the system proves to be an **effective solution for securing healthcare records**, ensuring regulatory compliance, and reducing risks associated with data breaches. Future improvements may focus on optimizing **storage efficiency**, **reducing computational overhead**, and integrating **smart contracts** for automated verification and secure access control.

CHAPTER-6

CONCLUSION AND FUTURE WORK

CONCLUSION:

In summary, your project integrates blockchain technology to enhance the integrity and confidentiality of patient data within a healthcare system. The system involves multiple roles, including client specialists, radiologists, surgeons, pharmacists, and administrators, each with specific responsibilities and access rights. After a patient registers and logs in, they receive a unique encrypted number from their specialist, which is then forwarded to various departments such as surgery, radiology, and pharmacy. This unique number ensures that sensitive patient information is protected and cannot be misused. The blockchain framework guarantees data integrity by providing an immutable ledger, while encryption and a structured endorsement process safeguard patient confidentiality. This approach not only streamlines data handling across different departments but also builds a secure and reliable system that fosters trust and efficiency in patient care.

FUTURE WORK:

Future improvements will focus on optimizing storage and scalability by integrating off-chain storage solutions and hybrid blockchain models. Implementing smart contracts can automate access control, ensuring secure and transparent data retrieval. Enhancing interoperability with Electronic Health Records (EHR) and Hospital Information Systems (HIS) will enable seamless data exchange. Performance optimization through lightweight consensus algorithms like Proof-of-Authority (PoA) can reduce energy consumption.

A.1 SDG Goals

SDG 3 – Ensure healthy lives and promote well-being for all at all ages

A.2 Source Code

DOCTOR MODULE:

```
package com.spring.graph.api.services;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.spring.graph.api.entity.Docterreg;
import com.spring.graph.api.repository.Docregrepository;

@Service
public class Docservice {

@Autowired
private Docregrepository  docrepo;
public boolean checkEmail(String email) {
// TODO Auto-generated method stub
return docrepo.existsByEmail(email);
}

public Docterreg updatedoctorstatus(Docterreg seller) {

return docrepo.save(seller);
}

public Docterreg getdoctorid(Long id) {

Optional<Docterreg> seller = docrepo.findById(id);
return seller.orElse(null);


}




public Docterreg getdocByEmailAndPassword(String sellemail, String
sellpassword) {
// TODO Auto-generated method stub
System.out.println("qqqqqqqq");
Optional<Docterreg> userOptional =
docrepo.findByEmailAndPassword(sellemail, sellpassword);
if (userOptional.isPresent()) {
Docterreg user = userOptional.get();

if (user.getStatus().equalsIgnoreCase("Approved")) {

System.out.println("Seller status is  approved for email: " + sellemail);

return user;
} else {
// If status is not "Approved", print a message and return null
System.out.println("Seller status is not approved for email: " + sellemail);
return null;
}
} else {
System.out.println("User not found for email: " + sellemail);
return null;
}
}

}

package com.spring.graph.api.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Docterreg {

   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private long id;

   private String name;
   private String email;
   private String password;
   private String doctors;
   private String contact;
```

```
    private String image;
    // Default constructor
    private String status;
    public Docterreg(Object ob) {
                // TODO Auto-generated constructor stub
        }

    public Docterreg() {
        super();
    }

    // Constructor with parameters


    // Getters and Setters
    public long getId() {
        return id;
    }
public Docterreg(long id, String name, String email, String password, String
doctors, String contact, String image,
                        String status) {
                super();
                this.id = id;
                this.name = name;
                this.email = email;
                this.password = password;
                this.doctors = doctors;
                this.contact = contact;
                this.image = image;
                this.status = status;

        }
        public void setId(long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
```

```
public String getEmail() {
   return email;
}

public void setEmail(String email) {
   this.email = email;
}

public String getPassword() {
   return password;
}

public void setPassword(String password) {
   this.password = password;
}

public String getDoctors() {
   return doctors;
}

public void setDoctors(String doctors) {
   this.doctors = doctors;
}

public String getContact() {
   return contact;
}

public void setContact(String contact) {
   this.contact = contact;
}



public String getStatus() {
		return status;
	}
	public void setStatus(String status) {
		this.status = status;
	}
	public String getImage() {
		return image;
	}

	public void setImage(String image) {
		this.image = image;
	}


}
```

PHARMACY MODULE

```
package com.spring.graph.api.services;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.spring.graph.api.entity.pharmacyentity;
import com.spring.graph.api.repository.pharmacyrepo;

@Service
public class pharmacyservice {

@Autowired
private pharmacyrepo pharmacyrepo;
public boolean checkEmail(String email) {
// TODO Auto-generated method stub
return pharmacyrepo.existsByEmail(email);
}
public pharmacyentity updatedpharmacystatus(pharmacyentity seller) {

return pharmacyrepo.save(seller);
}

public pharmacyentity getpharmacyid(Long id) {

Optional<pharmacyentity> seller = pharmacyrepo.findById(id);
return seller.orElse(null);
}

public pharmacyentity getdocByEmailAndPassword(String sellemail, String
sellpassword) {
// TODO Auto-generated method stub
System.out.println("qqqqqqqq");
```

```
Optional<pharmacyentity> userOptional =
pharmacyrepo.findByEmailAndPassword(sellemail, sellpassword);
if (userOptional.isPresent()) {
pharmacyentity user = userOptional.get();

if (user.getStatus().equalsIgnoreCase("Approved")) {

System.out.println("pharmacy status is  approved for email: " + sellemail);

return user;
} else {
// If status is not "Approved", print a message and return null
System.out.println("pharmacy status is not approved for email: " + sellemail);
return null;
}
} else {
System.out.println("pharmacy not found for email: " + sellemail);
return null;
}
}
}


package com.spring.graph.api.entity;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class pharmacyentity {
 @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    private String name;
    private String email;
    private String password;
    private String contact;
    private String image;
    // Default constructor
    private String catagory;
    private String status;
    public pharmacyentity() {
	}

   public pharmacyentity(long id, String name, String email, String password, String contact, String image,
			String catagory, String status) {
		super();
		this.id = id;
		this.name = name;
		this.email = email;
		this.password = password;
		this.contact = contact;
		this.image = image;
		this.catagory = catagory;
		this.status = status;
	}
	public long getId() {
		return id;
	}
	public void setId(long id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}
	public String getContact() {
		return contact;
	}
	public void setContact(String contact) {
		this.contact = contact;
	}
	public String getImage() {
		return image;
	}
	public void setImage(String image) {
		this.image = image;
	}
	public String getCatagory() {
		return catagory;
	}
	public void setCatagory(String catagory) {
		this.catagory = catagory;
	}
	public String getStatus() {
		return status;
	}
	public void setStatus(String status) {
		this.status = status;
	}



	@Override
    public String toString() {
        return "pharmacyentity [id=" + id + ", name=" + name + ", email=" + email
+ ", password=" + password + ", catagory=" + catagory + ", contact=" + contact
+   ", image=" + image +
                ", status=" + status + "]";
    }
```

RADIOLOGIS MODULE:

```
package com.spring.graph.api.services;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.spring.graph.api.entity.radiologistentity;
import com.spring.graph.api.repository.radiologistrepo;
@Service
public class radiologistservices {

@Autowired
private radiologistrepo radiologistrepo;
public boolean checkEmail(String email) {
// TODO Auto-generated method stub
return radiologistrepo.existsByEmail(email);
}
public radiologistentity updatedradiologiststatus(radiologistentity seller) {

return radiologistrepo.save(seller);
}

public radiologistentity getradiologyid(Long id) {

Optional<radiologistentity> seller = radiologistrepo.findById(id);
return seller.orElse(null);



}

public radiologistentity getdocByEmailAndPassword(String sellemail, String
sellpassword) {
// TODO Auto-generated method stub
System.out.println("qqqqqqqq");
Optional<radiologistentity> userOptional =
radiologistrepo.findByEmailAndPassword(sellemail, sellpassword);
if (userOptional.isPresent()) {
radiologistentity user = userOptional.get();

if (user.getStatus().equalsIgnoreCase("Approved")) {
System.out.println("Radiologist status is  approved for email: " + sellemail);

return user;
} else {
// If status is not "Approved", print a message and return null
System.out.println("Radiologist status is not approved for email: " + sellemail);
return null;
}
} else {
System.out.println("User not found for email: " + sellemail);
return null;
}

}
package com.spring.graph.api.entity;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class radiologistentity {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private long id;

private String name;
private String email;
private String password;
private String contact;
private String image;
// Default constructor
private String catagory;
private String status;
public radiologistentity() {

}

public radiologistentity(long id, String name, String email, String password,
String contact, String image,
String catagory, String status) {
super();
this.id = id;
this.name = name;
this.email = email;
this.password = password;
this.contact = contact;
this.image = image;
		this.catagory = catagory;
		this.status = status;
	}
	public long getId() {
		return id;
	}
	public void setId(long id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}
	public String getContact() {
		return contact;
	}
	public void setContact(String contact) {
		this.contact = contact;
	}
	public String getImage() {
		return image;
	}
	public void setImage(String image) {
		this.image = image;
	}
	public String getCatagory() {
		return catagory;
```

```
        }
        public void setCatagory(String catagory) {
this.catagory = catagory;
}
public String getStatus() {
return status;
}
public void setStatus(String status) {
this.status = status;
}

@Override
public String toString() {
return "surgeonentity [id=" + id + ", name=" + name + ", email=" + email + ",
password=" + password + ", catagory=" + catagory + ", contact=" + contact +
", image=" + image +
", status=" + status + "]";
}
```

SURGEON MODULE:

```
package com.spring.graph.api.services;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.spring.graph.api.entity.sdocrequet;
import com.spring.graph.api.entity.surgeonentity;
import com.spring.graph.api.repository.surgeonrepo;

@Service
public class surgeonservices {

@Autowired
private surgeonrepo surgeonrepo;


public boolean checkEmail(String email) {
// TODO Auto-generated method stub
return surgeonrepo.existsByEmail(email);
}
public surgeonentity updatedsurgeonstatus(surgeonentity seller) {
```

```
return surgeonrepo.save(seller);
}

public surgeonentity getsurgeonid(Long id) {

Optional<surgeonentity> seller = surgeonrepo.findById(id);
return seller.orElse(null);



}

public surgeonentity getdocByEmailAndPassword(String sellemail, String
sellpassword) {
// TODO Auto-generated method stub
System.out.println("qqqqqqqq");
Optional<surgeonentity> userOptional =
surgeonrepo.findByEmailAndPassword(sellemail, sellpassword);
if (userOptional.isPresent()) {
surgeonentity user = userOptional.get();
if (user.getStatus().equalsIgnoreCase("Approved")) {
System.out.println("Surgeon status is  approved for email: " + sellemail);

return user;
} else {
// If status is not "Approved", print a message and return null
System.out.println("Surgeon status is not approved for email: " + sellemail);
return null;
}
} else {
System.out.println("Surgeon not found for email: " + sellemail);
return null;
}
}
}

package com.spring.graph.api.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
```

```
public class surgeonentity {

   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private long id;
   private String name;
   private String email;
   private String password;

   private String contact;
   private String image;
   // Default constructor

   private String catagory;
   private String status;




   public surgeonentity() {

        }
        public surgeonentity(long id, String name, String email, String password,
String contact, String image,
                        String catagory, String status) {
                super();
                this.id = id;
                this.name = name;
                this.email = email;
                this.password = password;
                this.contact = contact;
                this.image = image;
                this.catagory = catagory;
                this.status = status;
        }
        public long getId() {
                return id;
        }
        public void setId(long id) {
                this.id = id;
        }
        public String getName() {
                return name;
```

```
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}
	public String getContact() {
		return contact;
	}
	public void setContact(String contact) {
		this.contact = contact;
	}
	public String getImage() {
		return image;
	}
	public void setImage(String image) {
		this.image = image;
	}
	public String getCatagory() {
		return catagory;
	}
	public void setCatagory(String catagory) {
		this.catagory = catagory;
	}
	public String getStatus() {
		return status;
	}
	public void setStatus(String status) {
		this.status = status;
	}

@Override
  public String toString() {
       return "surgeonentity [id=" + id + ", name=" + name + ", email=" + email
+ ", password=" + password + ", catagory=" + catagory + ", contact=" + contact
+   ", image=" + image +
                ", status=" + status + "]";
   }

 }
```

USER REGISTRATION MODULE:

```
package com.spring.graph.api.services;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.spring.graph.api.entity.User;
import com.spring.graph.api.repository.Userrepo;
@Service
public class Usereg {
@Autowired
private Userrepo userrepo;
public User saveUsers(User us) {
// TODO Auto-generated method stub
        return userrepo.save(us);
        }
public User updateduserstatus(User ok) {

                return userrepo.save(ok);
        }

                public User getuserid(Long id) {

                        Optional<User> seller = userrepo.findById(id);
                     return seller.orElse(null);



                }



        public User getUserByEmailAndPassword(String email, String password)
{



                 System.out.println("qqqqqqqq");
Optional<User>userOptional= userrepo.findByEmailAndPassword(email,
password);
                 if (userOptional.isPresent()) {
                  User user = userOptional.get();


                  if (user.getStatus().equalsIgnoreCase("Approved")) {


                 System.out.println("Seller status is  approved for email: " + email);

                        return user;
                  } else {
                     // If status is not "Approved", print a message and return null
             System.out.println("Seller status is not approved for email: " + email);
                     return null;
                  }
               } else {
                  System.out.println("User not found for email: " + email);
                  return null;
               }


                }



        }

package com.spring.graph.api.entity;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
@Entity
public class User {


@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
   private long id;
   private String name;
   private String email;
   private String password;
```

```
    private String age;
    private String contact;
    private String userid;
    private String status;//Default constructor

public User(Object ob) {
                // TODO Auto-generated constructor stub
        }

public User() {

        }

public long getId() {
                return id;
        }
public void setId(long id) {
                this.id = id;
        }

public String getName() {
                return name;
        }
public void setName(String name) {
                this.name = name;
        }

public String getEmail() {
                return email;
        }

public void setEmail(String email) {
                this.email = email;
        }

public String getPassword() {
                return password;
        }

public void setPassword(String password) {
                this.password = password;
        }
```

```
          public String getAge() {
                    return age;
}

public void setAge(String age) {
                    this.age = age;
          }

public String getContact() {
                    return contact;
          }


public void setContact(String contact) {
                    this.contact = contact;
          }

public String getUserid() {
                    return userid;
          }

public void setUserid(String userid) {
                    this.userid = userid;
          }
  public String getStatus() {
                    return status;
          }
          public void setStatus(String status) {
                    this.status = status;
          }

public User(long id, String name, String email, String password, String age,
String contact, String userid) {
                    super();
                    this.id = id;
                    this.name = name;
                    this.email = email;
                    this.password = password;
                    this.age = age;
                    this.contact = contact;
                    this.userid = userid;

          }
```

```
@Override
   public String toString() {
      return "userreg [id=" + id + ", name=" + name + ", email=" + email + ",
password=" + password + ", age=" + age + ", contact=" + contact +   ", userid="
+ userid", status=" + status + "]";
   }

}
```

ENCRYPTION MODULE:

```
package com.spring.graph.api.encryptionservice;

import org.springframework.stereotype.Service;

import com.spring.graph.api.algorithms.AESUtil;

import javax.crypto.SecretKey;

@Service
public class EncryptionService {

   private final SecretKey secretKey;

   public EncryptionService() throws Exception {
      // Generate a new AES key. You can save this key securely and reuse it.
      this.secretKey = AESUtil.generateKey(128);
   }

   public String encryptData(byte[] bs) throws Exception {
      return AESUtil.encrypt(bs, secretKey);
   }

   public String decryptData(String encryptedData) throws Exception {
      return AESUtil.decrypt(encryptedData, secretKey);
   }

   public String getSecretKey() {
      return AESUtil.keyToString(secretKey);
   }
```

```
}

package com.spring.graph.api.algorithms;
import java.util.Date;




public class Block {

	public String hash;
	public String previousHash;
	private String data; //our data will be a simple message.
	private long timeStamp; //as number of milliseconds since 1/1/1970.

	public Block(String data,String previousHash ) {
		this.data = data;
		this.previousHash = previousHash;
		this.timeStamp = new Date().getTime();
		this.hash = calculateHash(); //Making sure we do this after we set
the other values.
	}

	public String calculateHash() {
		String calculatedhash = StringUtil.applySha256(
				previousHash +
				Long.toString(timeStamp) +
				data
				);
		return calculatedhash;
	}
}


package com.spring.graph.api.algorithms;


import java.security.MessageDigest;

public class StringUtil{
	//Applies Sha256 to a string and returns the result.
	public static String applySha256(String input){
```

```
                try {
                        MessageDigest digest = MessageDigest.getInstance("SHA-
256");
                        //Applies sha256 to our input,
                        byte[] hash = digest.digest(input.getBytes("UTF-8"));
                        StringBuffer hexString = new StringBuffer(); // This will
contain hash as hexidecimal
                        for (int i = 0; i < hash.length; i++) {
                                String hex = Integer.toHexString(0xff & hash[i]);
                                if(hex.length() == 1) hexString.append('0');
                                hexString.append(hex);
                        }
                        return hexString.toString();
                }
                catch(Exception e) {
                        throw new RuntimeException(e);
                }
        }
}



package com.spring.graph.api.algorithms;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
public class mining {

private class Block {
      private int index;
      private String previousHash;
      private String data;
      private long timestamp;
      private String hash;
      private int nonce;

      public Block(int index, String previousHash, String data) {
         this.index = index;
         this.previousHash = previousHash;
         this.data = data;
         this.timestamp = new Date().getTime();
         this.hash = calculateHash();
         this.nonce = 0;
      }

      public String calculateHash() {
         return applySha256(
             index + previousHash + timestamp + data + nonce
         );
      }

      public void mineBlock(int difficulty) {
         String target = new String(new char[difficulty]).replace('\0', '0');

         while (!hash.substring(0, difficulty).equals(target)) {
             nonce++;
             hash = calculateHash();
         }

   System.out.println("Block mined: " + hash);
      }
  }

  private List<Block> blockchain;
  private int difficulty;

  public mining(int difficulty) {
      this.blockchain = new ArrayList<>();
      this.difficulty = difficulty;
      // Genesis block
      addBlock("Genesis Block");
  }

  public void addBlock(String data) {
      int index = blockchain.size();
      Block previousBlock = (index > 0) ? blockchain.get(index - 1) : null;
      String previousHash = (previousBlock != null) ? previousBlock.hash : "0";

      Block newBlock = new Block(index, previousHash, data);
      newBlock.mineBlock(difficulty);

      blockchain.add(newBlock);
  }
```

```
public boolean isChainValid() {
    for (int i = 1; i < blockchain.size(); i++) {
        Block currentBlock = blockchain.get(i);
        Block previousBlock = blockchain.get(i - 1);

        if (!currentBlock.hash.equals(currentBlock.calculateHash())) {
            System.out.println("Block hash mismatch at index " + i);
            return false;
        }

        if (!currentBlock.previousHash.equals(previousBlock.hash)) {
            System.out.println("Previous hash mismatch at index " + i);
            return false;
        }
    }

    System.out.println("Blockchain is valid.");
    return true;
}

private String applySha256(String input) {
    try {
        MessageDigest digest = MessageDigest.getInstance("SHA-256");
        byte[] hash = digest.digest(input.getBytes());

        StringBuilder hexString = new StringBuilder();
        for (byte b : hash) {
            String hex = Integer.toHexString(0xff & b);
            if (hex.length() == 1) {
                hexString.append('0');
            }
            hexString.append(hex);
        }

        return hexString.toString();
    } catch (NoSuchAlgorithmException e) {
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    int initialDifficulty = 1;
    mining blockchain = new mining(initialDifficulty);
```

```
        // Adding blocks to the blockchain
        blockchain.addBlock("Transaction 1");


        // Validating the blockchain
        blockchain.isChainValid();
    }
}
```

ForodoKEM:

```
package com.spring.graph.api.algorithms;

import java.security.SecureRandom;
import java.util.Arrays;

public class FrodoKem {

    private static final int N = 512;  // Dimension of the matrix
    private static final int Q = 12289; // Modulo value for operations

    // Simulate a secret key generation
    public static int[] generateSecretKey() {
        SecureRandom random = new SecureRandom();
        int[] secretKey = new int[N];
        for (int i = 0; i < N; i++) {
            secretKey[i] = random.nextInt(Q);  // Random values within the range of
Q
        }
        return secretKey;
    }

    // Simulate public key generation
    public static int[] generatePublicKey(int[] secretKey) {
        // In FrodoKEM, the public key generation involves a matrix-based
operation
        int[] publicKey = new int[N];
        for (int i = 0; i < N; i++) {
            publicKey[i] = (secretKey[i] + new SecureRandom().nextInt(Q)) % Q;
        }
        return publicKey;
    }

    // Encapsulation (Encryption step)
    public static int[] encapsulate(int[] publicKey) {
        int[] ciphertext = new int[N];
        SecureRandom random = new SecureRandom();
        for (int i = 0; i < N; i++) {
            // Simulate the encapsulation process by adding randomness to the
public key
            ciphertext[i] = (publicKey[i] + random.nextInt(Q)) % Q;
        }
        return ciphertext;
    }

    // Decapsulation (Decryption step)
    public static int[] decapsulate(int[] ciphertext, int[] secretKey) {
        int[] decryptedMessage = new int[N];
        for (int i = 0; i < N; i++) {
            decryptedMessage[i] = (ciphertext[i] - secretKey[i]) % Q;
        }
        return decryptedMessage;
    }

    // Testing the FrodoKEM Algorithm
    public static void main(String[] args) {
        // Generate a secret key and public key
        int[] secretKey = generateSecretKey();
        int[] publicKey = generatePublicKey(secretKey);

        // Encrypt (Encapsulate)
        int[] ciphertext = encapsulate(publicKey);
        System.out.println("Ciphertext: " + Arrays.toString(ciphertext));

        // Decrypt (Decapsulate)
        int[] decryptedMessage = decapsulate(ciphertext, secretKey);
        System.out.println("Decrypted          Message:          "          +
Arrays.toString(decryptedMessage));
    }
}
```

ADMIN MODULE:

```
package com.spring.graph.api.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Adminentity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    private String email;

    private String password;

    // Getters and setters for id, email, and password
    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    // toString method for debugging or logging purposes
    @Override
    public String toString() {
        return "Adminentity [id=" + id + ", email=" + email + ", password=" +
password + "]";
    }
}
```

PATIENT MODULE:

```
package com.spring.graph.api.entity;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Lob;

@Entity
public class Patiententity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String patientid;
    private String email;
    private String msg;
    private String docName;
    private String docType;
    private String docId;
    private String encryptData;

    @Lob
    @Column(name = "phash", columnDefinition = "LONGTEXT")
    private String phash;

    @Lob
    @Column(name = "ahash", columnDefinition = "LONGTEXT")
    private String ahash;

    @Lob
    @Column(name = "data", columnDefinition = "LONGBLOB")
    private byte[] data;

    private String status;

    public Patiententity() {}

    public Patiententity(Long id, String patientid, String email, String msg, String
docName, String docType,
                         String docId, String encryptData, String phash, String ahash,
byte[] data, String status) {
        this.id = id;
        this.patientid = patientid;
        this.email = email;
        this.msg = msg;
        this.docName = docName;
        this.docType = docType;
        this.docId = docId;
        this.encryptData = encryptData;
        this.phash = phash;
        this.ahash = ahash;
        this.data = data;
        this.status = status;
    }

    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }

    public String getPatientid() { return patientid; }
    public void setPatientid(String patientid) { this.patientid = patientid; }

    public String getEmail() { return email; }
    public void setEmail(String)
```

A.3 SCREEN SHOTS:

HOME PAGE:

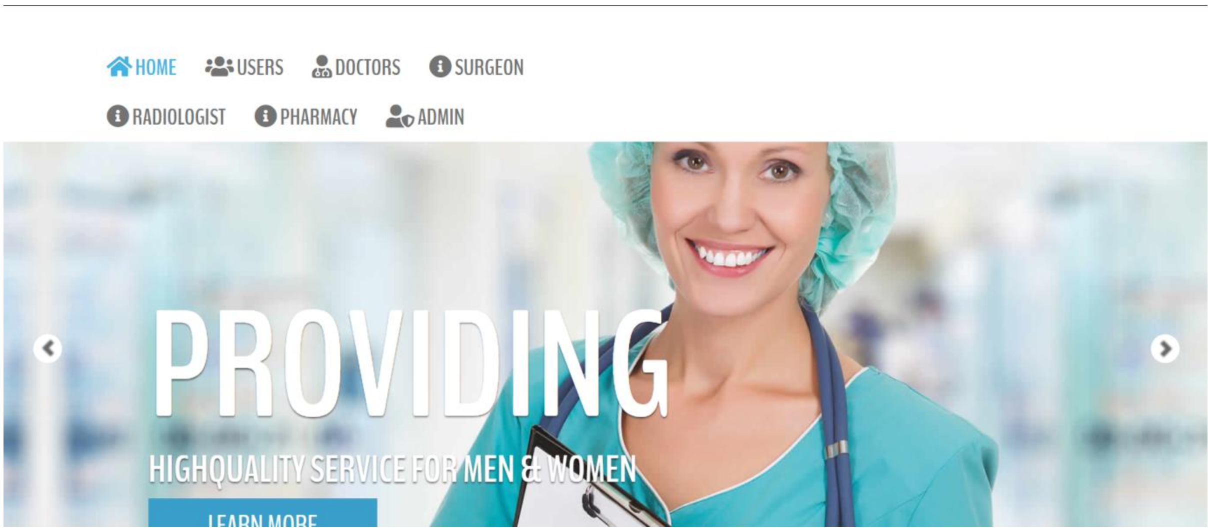


Fig A.3.1 Home Page

USER REGISTRATION AND LOGIN:

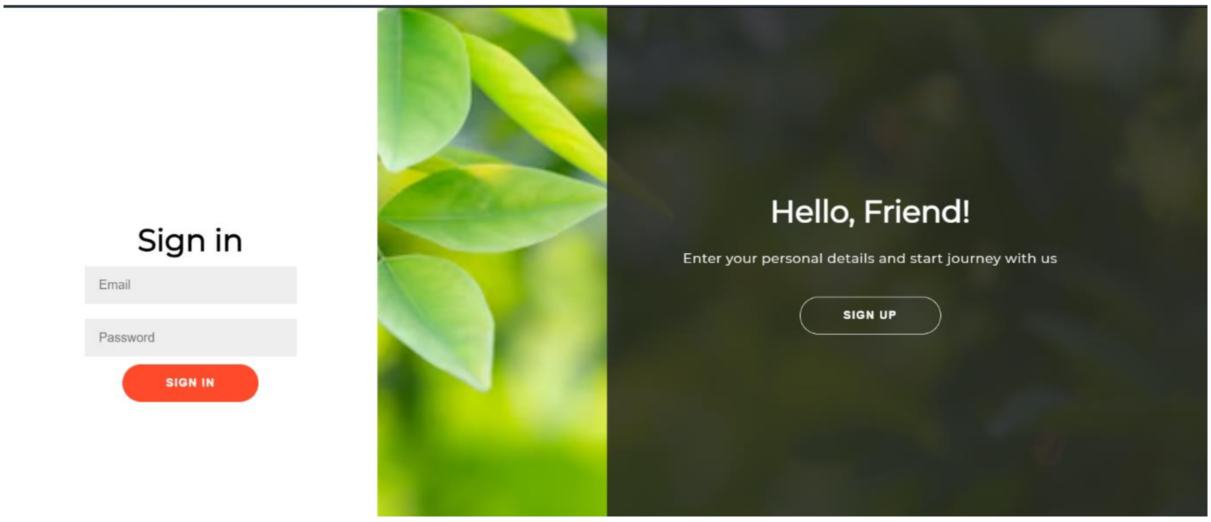
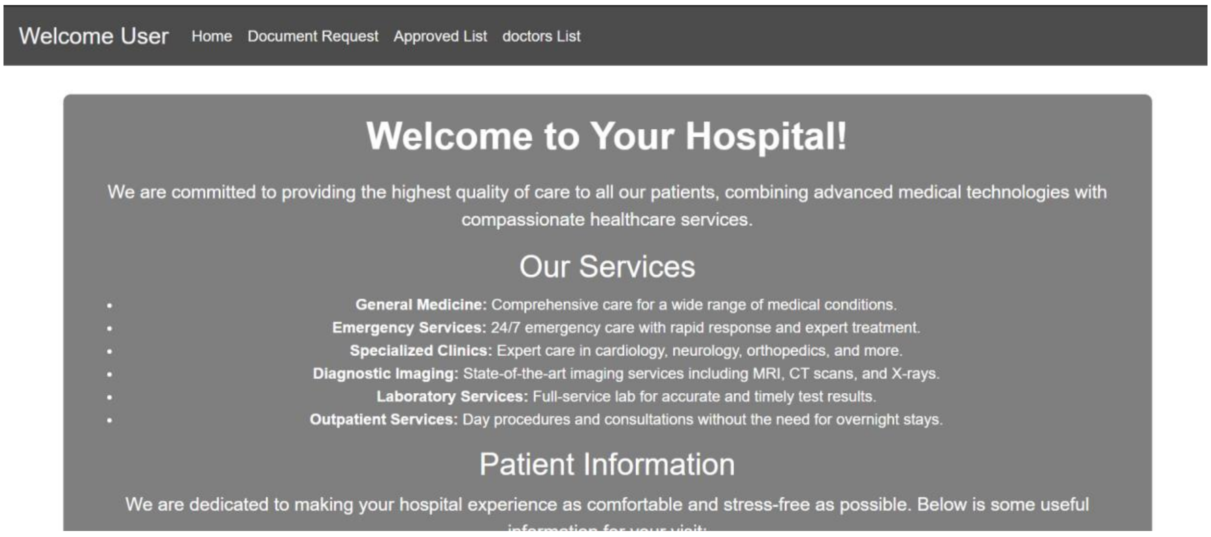


Fig A.3.2 User Registration and Login Page

USER MODULE:



Welcome to Your Hospital!

We are committed to providing the highest quality of care to all our patients, combining advanced medical technologies with compassionate healthcare services.

Our Services

- **General Medicine:** Comprehensive care for a wide range of medical conditions.
- **Emergency Services:** 24/7 emergency care with rapid response and expert treatment.
- **Specialized Clinics:** Expert care in cardiology, neurology, orthopedics, and more.
- **Diagnostic Imaging:** State-of-the-art imaging services including MRI, CT scans, and X-rays.
- **Laboratory Services:** Full-service lab for accurate and timely test results.
- **Outpatient Services:** Day procedures and consultations without the need for overnight stays.

Patient Information

We are dedicated to making your hospital experience as comfortable and stress-free as possible. Below is some useful

Fig A.3.3 User Module

DOCTOR REGISTRATION AND SIGN-IN PAGE:

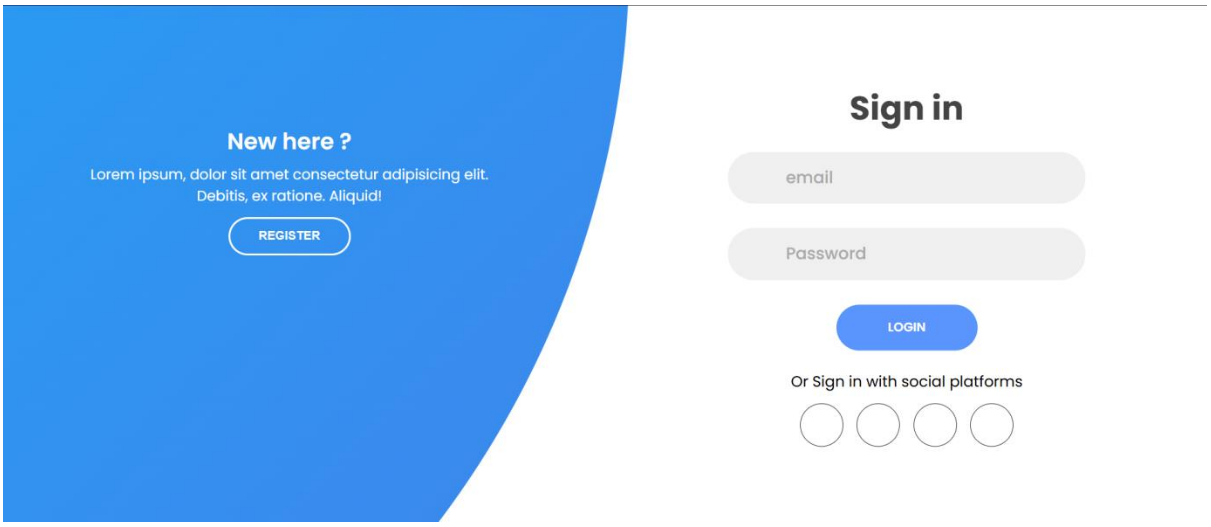

New here ?
Lorem ipsum, dolor sit amet consectetur adipisicing elit. Debitis, ex ratione. Aliquid!
REGISTER
Sign in
email
Password
LOGIN
Or Sign in with social platforms

Fig A.3.4 Doctor Registration and Sign-in Page

SURGEON PAGE:

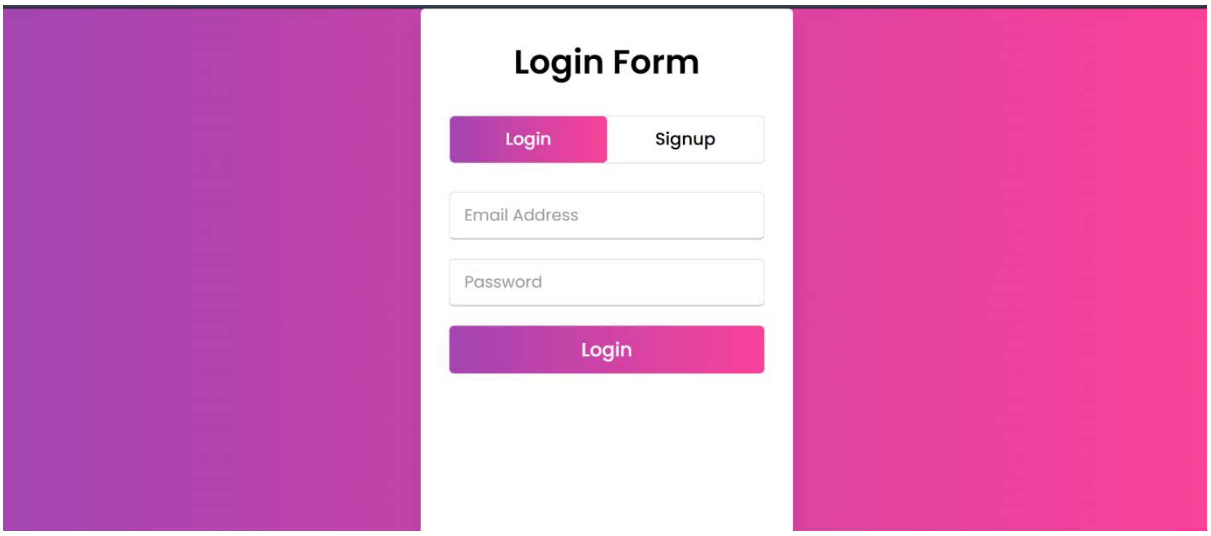


Fig A.3.5 Surgeon Page

PHARMACY SIGN UP AND LOGIN PAGE:

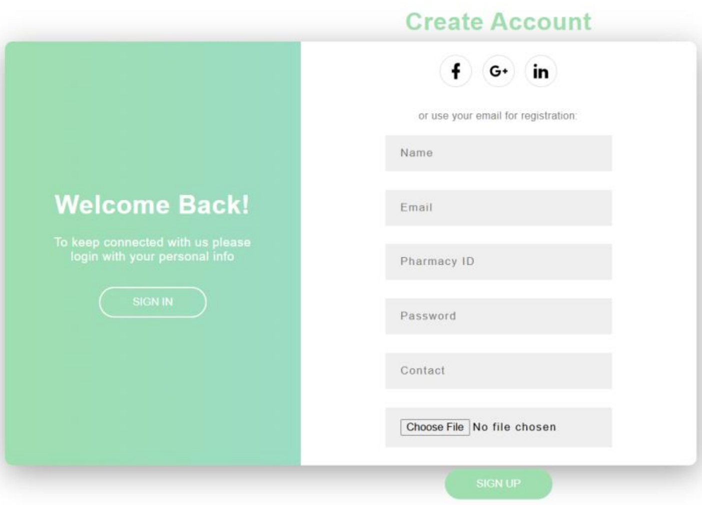
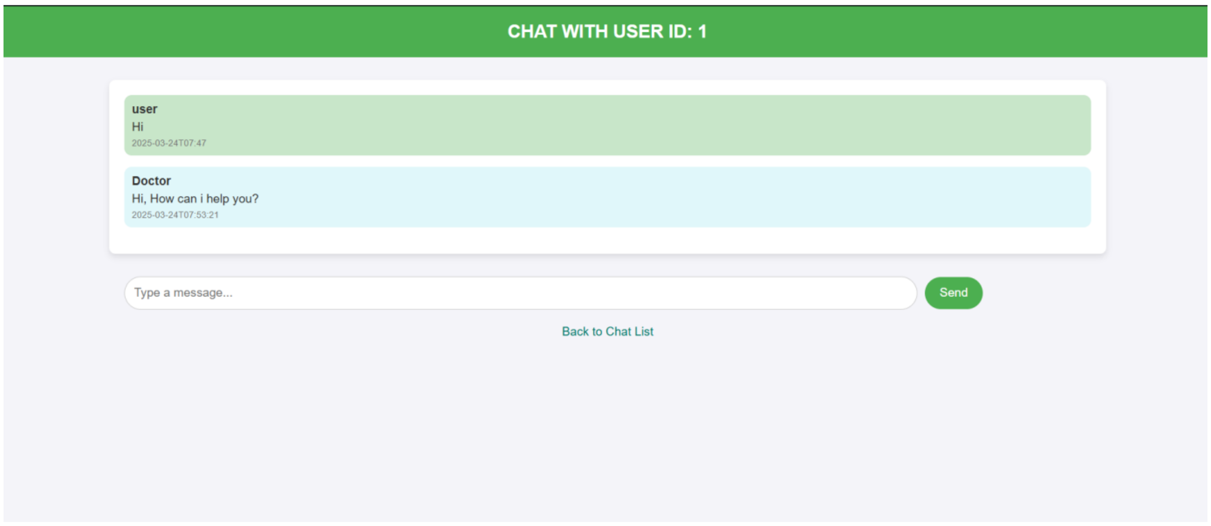

Create Account
or use your email for registration:
Name
Email
Pharmacy ID
Password
Contact
Choose File No file chosen
SIGN UP
Welcome Back!
To keep connected with us please login with your personal info
SIGN IN

Fig A.3.6 Pharmacy Sign-up and Login Page

CHATTING MODULE:


CHAT WITH USER ID: 1

user
Hi
2025-03-24T07:47

Doctor
Hi, How can i help you?
2025-03-24T07:53:21

Fig A.3.7 Chatting Module

ADMIN LOGIN:

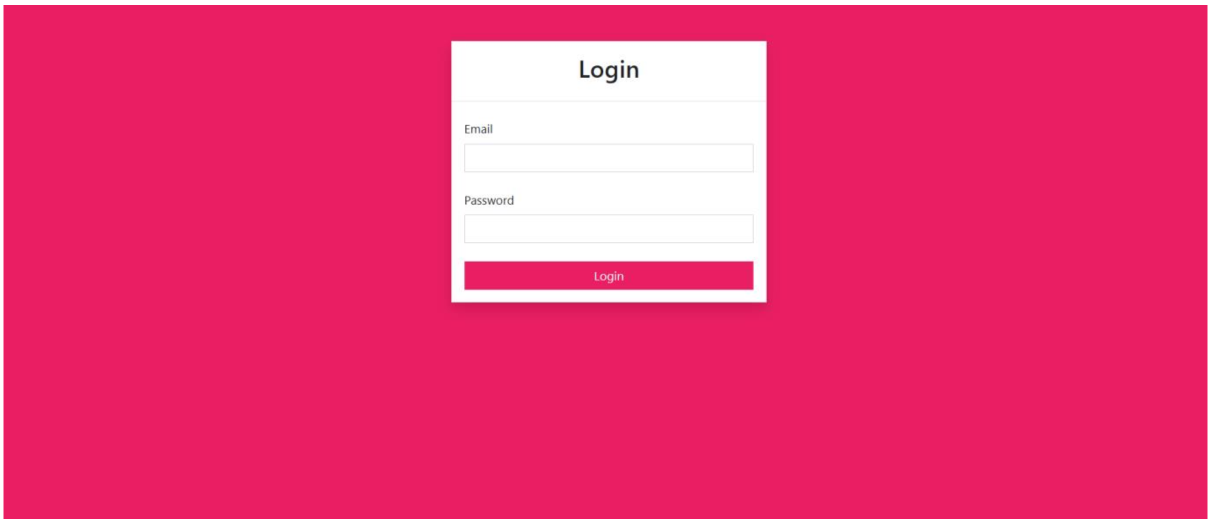


Fig A.3.8 Admin Login Page

RADIOLOGIST REGISTER AND LOGIN PAGE:

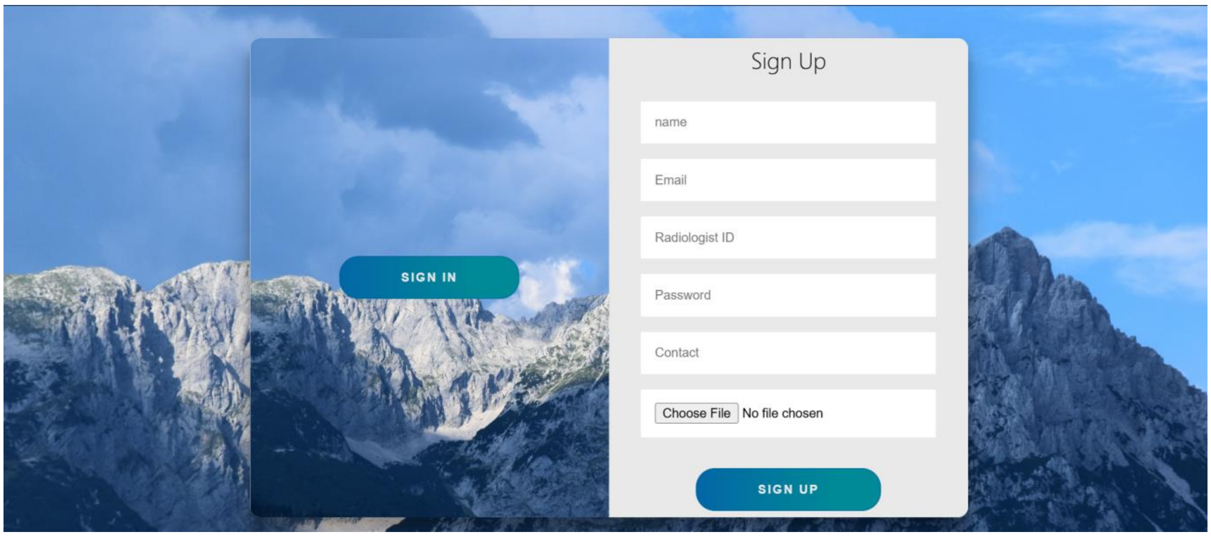
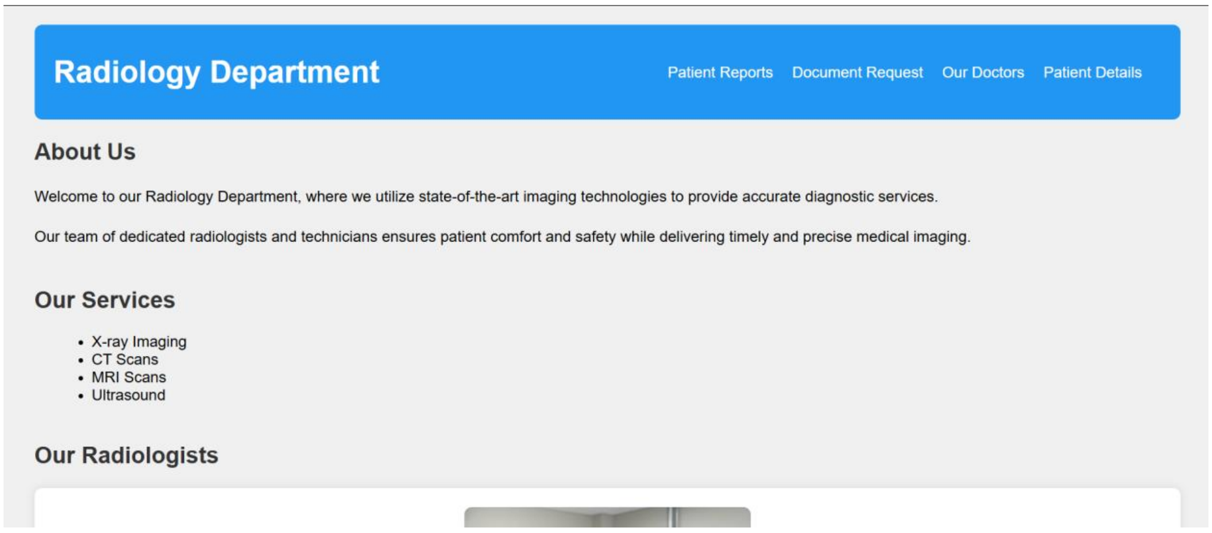


Fig A.3.9 Radiologist Register and Login Page

RADIOLOGIST MODULE:



About Us

Welcome to our Radiology Department, where we utilize state-of-the-art imaging technologies to provide accurate diagnostic services.

Our team of dedicated radiologists and technicians ensures patient comfort and safety while delivering timely and precise medical imaging.

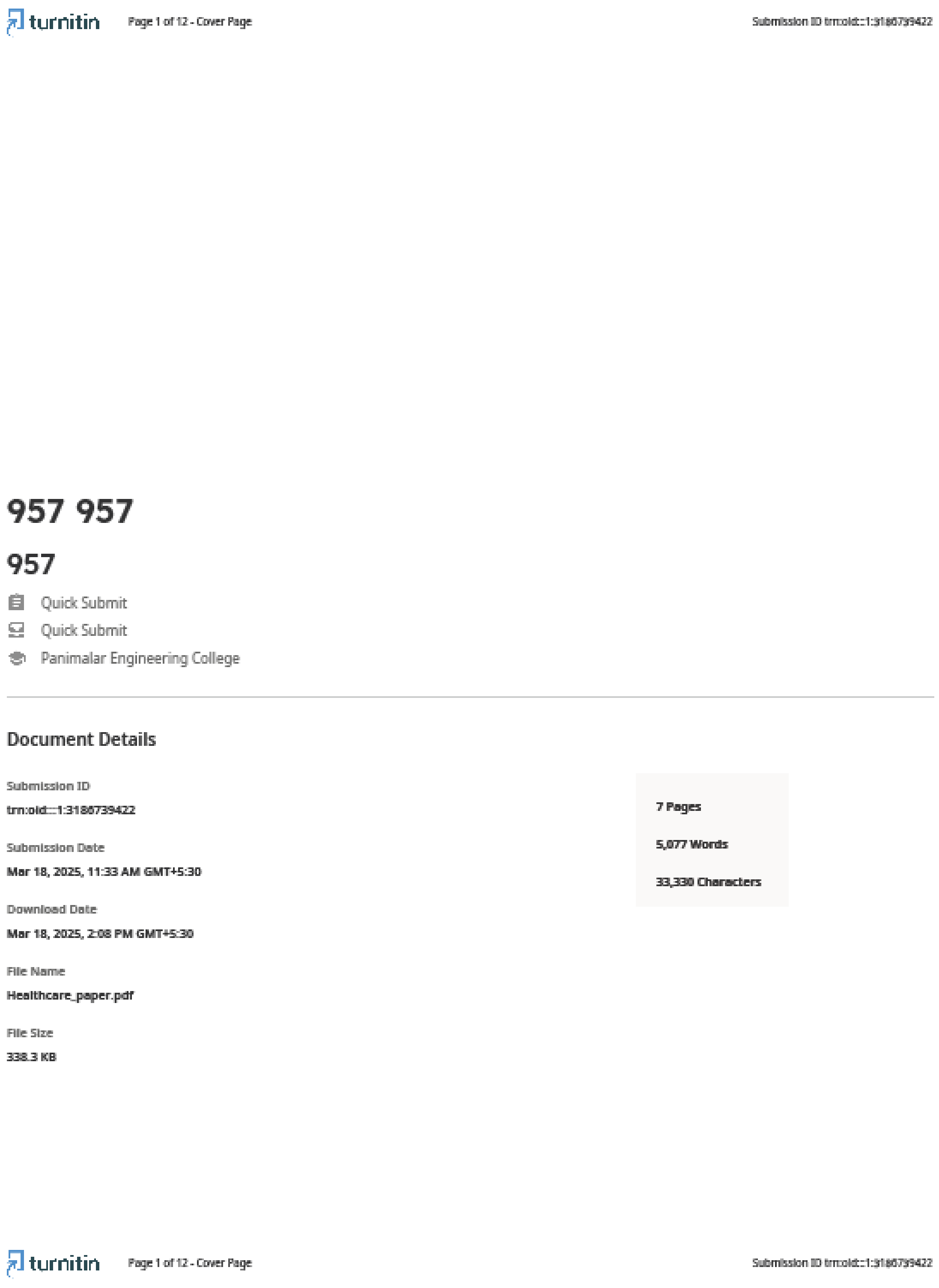
Our Services

- X-ray Imaging
- CT Scans
- MRI Scans
- Ultrasound

Our Radiologists

Fig A.3.10 Radiologist Module

A.4 PLAGIARISM REPORT:



turnitin Page 1 of 12 - Cover Page Submission ID trn:oid:::1:3186739422

957 957

957

Quick Submit

Quick Submit

Panimalar Engineering College

Document Details

Submission ID trn:oid:::1:3186739422	7 Pages
Submission Date Mar 18, 2025, 11:33 AM GMT+5:30	5,077 Words
Download Date Mar 18, 2025, 2:08 PM GMT+5:30	33,330 Characters
File Name Healthcare_paper.pdf	
File Size 338.3 KB	

turnitin Page 1 of 12 - Cover Page Submission ID trn:oid:::1:3186739422

Match Groups

- **48** Not Cited or Quoted 10%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

6%	Internet sources
8%	Publications
2%	Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Student papers	Loyola University, Chicago	<1%
2	Internet	ebin.pub	<1%
3	Publication	Alex Khang, Kali Charan Rath. "The Quantum Evolution - Application of AI and Ro...	<1%
4	Publication	Dan Zhu, Hui Zhu, Cheng Huang, Rongxing Lu, Dengguo Feng, Xuemin Shen. "Effi...	<1%
5	Internet	ai.jmir.org	<1%
6	Publication	Xiaofeng Wang, Xiaoguang Yue, Ahthasham Sajid, Noshina Tariq. "AllianceBlockc...	<1%
7	Internet	www.gwcet.ac.in	<1%
8	Publication	Anurag Tiwari, Manuj Darbari. "Emerging Trends in Computer Science and Its Ap...	<1%
9	Internet	www.talk-business.co.uk	<1%
10	Publication	Fatma Khallaf, Walid El-Shafai, El-Sayed M. El-Rabaie, Fathi E. Abd El-Samie. "Block...	<1%

DEVELOPMENT OF A SECURE HEALTHCARE MANAGEMENT SYSTEM UTILIZING BLOCKCHAIN TECHNOLOGY FOR ENCRYPTED PATIENT DATA TRANSMISSION

17

S.T. Santhanalakshmi[1]
Associate Professor
Department of Computer Science and Engineering
Panimalar Engineering College
santhanalakshmi.peccse2024@gmail.com

Dr. Kavitha Subramani[2]
Professor
Department of Computer Science and Engineering
Panimalar Engineering College
kavitha.pec2022@gmail.com

Dharshini B[3]
Department of Computer Science and Engineering
Panimalar Engineering College
bdharshini16@gmail.com

1

Elakkiya S[4]
Department of Computer Science and Engineering
Panimalar Engineering College
elakkimuthu@gmail.com

Gajalakshmi S[5]
Department of Computer Science and Engineering
Panimalar Engineering College
gajalakshmisaravanan2804@gmail.com

5

***ABSTRACT*—Effective management of healthcare data is essential for ensuring the safety of sensitive patient information in today's digital landscape. An innovative system utilizes the cutting-edge FrodoKEM encryption algorithm along with real-time secure chat features to protect information shared among various hospital departments. FrodoKEM uses adaptive encryption methods and dynamic key management to provide strong data security during transmission. Simultaneously, blockchain-based logging and SHA-256 hashing are employed to preserve data integrity and create an unchangeable audit trail for all accesses and alterations. The secure chat component allows for private, real-time communication between physicians and patients, enhancing telehealth capabilities and overall clinical productivity. Experimental tests indicate that the system delivers low latency and high security performance in fluctuating healthcare environments. By integrating state-of-the-art encryption techniques with secure communication solutions, this approach not only strengthens data confidentiality and integrity but also improves clinical workflows, presenting a promising avenue for future digital healthcare management.**

4

Keywords*—Healthcare security, FrodoKEM, secure real-time communication, blockchain logging, SHA-256 hashing, dynamic key management, data integrity, digital healthcare management.

I. INTRODUCTION

5

The contemporary digital healthcare environment requires strong safeguards for sensitive patient information. Traditional encryption techniques often struggle with static key management and limited flexibility in dynamic, high-throughput settings. To tackle these issues, the innovative FrodoKEM encryption algorithm has been introduced. FrodoKEM utilizes adaptive encryption methods and dynamic key management to protect sensitive medical information as it is shared across different hospital departments. This strategy significantly reduces risks linked to emerging cyber threats and provides improved security for medical data. At the same time, the growth of telemedicine highlights the need for secure, real-time communication pathways between healthcare providers and patients. A specialized secure chat module has been incorporated into the system, allowing private, immediate exchanges that comply with strict regulatory requirements. Additionally, to enhance data integrity and accountability, blockchain-based logging and SHA-256 hashing are utilized. These technologies work together to establish an unchanging audit trail, protecting against unauthorized alterations to data and ensuring transparency in data access. By merging sophisticated encryption with secure communication protocols and stringent data integrity practices, the proposed system presents a holistic solution for next-generation digital healthcare management. This integrated approach not only secures sensitive information but also improves clinical efficiency, laying the groundwork for more resilient and trustworthy healthcare systems.

II. LITERATURE SURVEY

Privacy preservation and security in healthcare systems have become critical areas of research due to the increasing reliance on digital healthcare platforms, mobile health (mHealth) applications, and Internet of Medical Things (IoMT). Numerous techniques have been proposed, including federated learning, blockchain, encryption algorithms, and hybrid privacy models. Zhu et al. [1] proposed an efficient and privacy-preserving cloud-assisted medical pre-diagnosis system, which enhances data accuracy and ensures patient data confidentiality through encrypted data sharing mechanisms. Zhang and Liu [11] introduced security models for healthcare applications deployed in cloud environments, emphasizing access control, secure data transmission, and data integrity measures. Federated learning (FL) has emerged as a promising technique for collaborative model training without exposing sensitive data. Narmadha and Varalakshmi [7] presented a privacy-preserving FL framework for healthcare, enabling hospitals to collaboratively train models without sharing patient records. Pati et al. [8] elaborated on privacy techniques integrated into FL, incorporating differential privacy and secure aggregation. Abaoud et al. [3] introduced novel privacy mechanisms in FL, enhancing data obfuscation and user-level privacy guarantees in healthcare settings. Javed et al. [16] proposed ShareChain, a blockchain-enabled FL model with differential privacy, ensuring traceability, transparency, and privacy-preservation during collaborative training. Blockchain has been extensively a blockchain ledger, creating an unalterable record of data interactions. Furthermore, the system applies SHA-256 hashing to create unique identifiers for data entries, enabling swift detection of any unauthorized changes. This integration of blockchain and hashing technologies strengthens the system against data tampering and builds trust among users.

F. Experimental Design and Evaluation Criteria

19 To evaluate the effectiveness and robustness of the proposed system, a series of meticulously organized experiments will be carried out within a controlled setting that aims to closely replicate real-world healthcare situations. These experiments will seek to emulate the standard communication and data transfer processes between healthcare providers and patients, ensuring that the performance and security assessments are reflective of genuine healthcare workflows.

The evaluation process will concentrate on several key performance indicators, which are vital for confirming that the system satisfies both security standards and operational efficiency. The primary evaluation criteria consist of:

- **Latency:** This measure will assess the time needed for various data transactions and communications to be completed. The goal is to ensure that the integration of robust encryption and secure communication protocols does not result in significant delays that could impede system responsiveness. Keeping latency low is particularly critical in healthcare environments, where prompt access to patient information is essential for accurate diagnosis and treatment. 24 8
- **Throughput:** The throughput evaluation will measure the system's ability to manage a high volume of simultaneous data exchanges and communications. 6 This is crucial to illustrate the system's scalability and efficiency, particularly in scenarios where multiple healthcare providers and patients are interacting with the platform at the same time. The capability to handle large volumes of secure transactions without a decline in performance is a vital success factor for the system's practical implementation. 32
- **Security Robustness:** A thorough analysis will be performed to assess the system's strength against various security threats and attack vectors. This includes simulated attempts to compromise the encryption mechanisms, gain unauthorized access to sensitive patient information, and manipulate or alter data records. By subjecting the system to different attack scenarios, its ability to maintain data confidentiality, integrity, and authenticity will be rigorously evaluated.

The findings from these experiments will furnish empirical evidence demonstrating the system's potential to enhance the security of healthcare data while upholding acceptable levels of performance and user

By incorporating advanced cryptographic techniques, secure communication protocols, and blockchain technology into a unified framework, this research aspires to provide a comprehensive and practical solution to the urgent issues of data security, privacy, and integrity within contemporary digital healthcare settings.

IV. System Architecture

The proposed healthcare management system is structured with a modular design that incorporates sophisticated security features to maintain data confidentiality, integrity, and availability. It consists of five main modules: User Module, Doctor Module, Department Module, Security Management Module, and Chat Module. Each of these modules is connected via secure communication pathways and together they form a strong and effective ecosystem for healthcare data management.

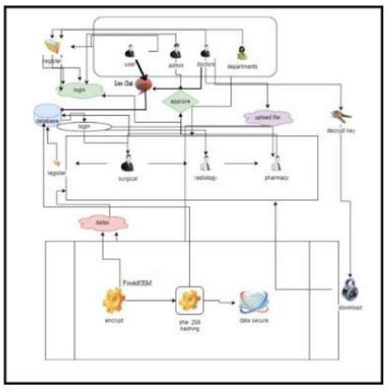


Fig. 1. System Architecture Diagram

A. User Module

This module oversees functionalities related to patients, such as registration, authentication, and profile administration. Patients are able to securely view their medical records, book appointments, and interact with healthcare professionals. The module utilizes robust authentication methods to thwart unauthorized access and ensures patient data is encrypted both during storage and transmission. management.

B. Doctor Module

The Doctor Module is tailored for healthcare providers to oversee patient consultations, access medical histories, and create prescriptions. It offers an intuitive interface for doctors to efficiently enter and retrieve patient data. Access is limited to verified medical staff, and all activities are documented for accountability.

C. Department Module

This module promotes collaboration among various hospital departments like radiology, laboratory, and pharmacy. It allows departments to retrieve relevant patient data, update test results, and facilitate communications between departments. Role-based access control guarantees that each department has access only to the information relevant to its operations, upholding data privacy and adherence to healthcare regulations.

D. Security Management Module

The Security Management Module serves as the foundation of the system's strategy for data protection. It incorporates several essential security elements:

- **FrodoKEM Encryption:** Implements the FrodoKEM algorithm, a lattice-based post-quantum cryptography method, to secure sensitive information. This provides resilience against both classical and quantum threats, protecting patient data from potential future risks.
- **Dynamic Key Management:** Establishes a framework for frequent key updates and rotations, reducing the risk related to key breaches. This proactive management ensures that even if a key is compromised, the duration of vulnerability remains short.
- **Blockchain-Based Logging:** Utilizes blockchain technology to generate an unchangeable record of all system transactions and data accesses. Each log entry is hashed using SHA-256 and incorporated into the blockchain, offering a tamper-proof record that enhances both transparency and trust.

Fig. 2. Encrypted Patient Data

E. Chat Module

The Chat Module enables secure and immediate communication between patients and healthcare professionals, facilitating smooth interactions no matter the physical distance. It accommodates various forms of communication, such as text messaging, audio calls, and video consultations, thereby enhancing the system's telemedicine functionalities. This adaptability allows healthcare professionals to provide remote consultations, follow-ups, and quick clarifications, which boosts patient engagement and access to healthcare. All communications within this module are secured with end-to-end encryption and are fully compliant with the system's overall security measures. This guarantees that all sensitive discussions, including personal health information and medical guidance, are kept private and protected from unauthorized access.

F. Data Flow and Interaction

The architecture of the system guarantees smooth data flow across modules while upholding stringent security protocols:

- **Data Access:** When a user (whether a patient or a doctor) seeks access to data, the request is authenticated and authorized based on established roles and permissions. This process is managed by the Security Management Module to ensure that only valid requests are processed.
- **Data Transmission:** All data exchanged between modules is protected by encryption through FrodoKEM, safeguarding it from interception and unauthorized access during transmission.
- **Data Logging:** Each access and modification of data is recorded by the Security Management Module. These logs are saved on the blockchain, creating an unalterable record that can be reviewed to identify and prevent malicious acts.

This modular and security-focused architecture guarantees that the healthcare management system remains strong, adaptable, and capable of defending sensitive patient information against emerging cyber threats.

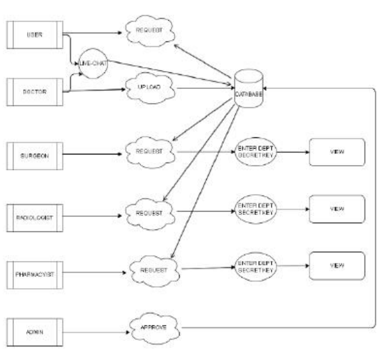


Fig. 3. Data Flow Diagram

V. Features and Functionalities

A. Secure Chat Interface

The incorporation of a secure messaging application into healthcare systems provides a specialized communication platform that thoroughly complies with the Health Insurance Portability and Accountability Act (HIPAA) regulations. This adherence guarantees that all communications fulfill rigorous data protection benchmarks, thereby improving both clinical cooperation and operational workflow efficiency. The secure chat interface functions as a smooth channel for real-time communication between healthcare providers and patients, enabling quick exchanges of vital health information. By encrypting all data transmitted, the system guarantees that sensitive patient discussions remain private and safeguarded from unauthorized access. Additionally, the user-friendly interface caters to individuals with diverse levels of technical expertise, allowing for simple and intuitive communication. This user-friendly design encourages smoother interactions between healthcare professionals and patients, leading to increased patient satisfaction, enhanced

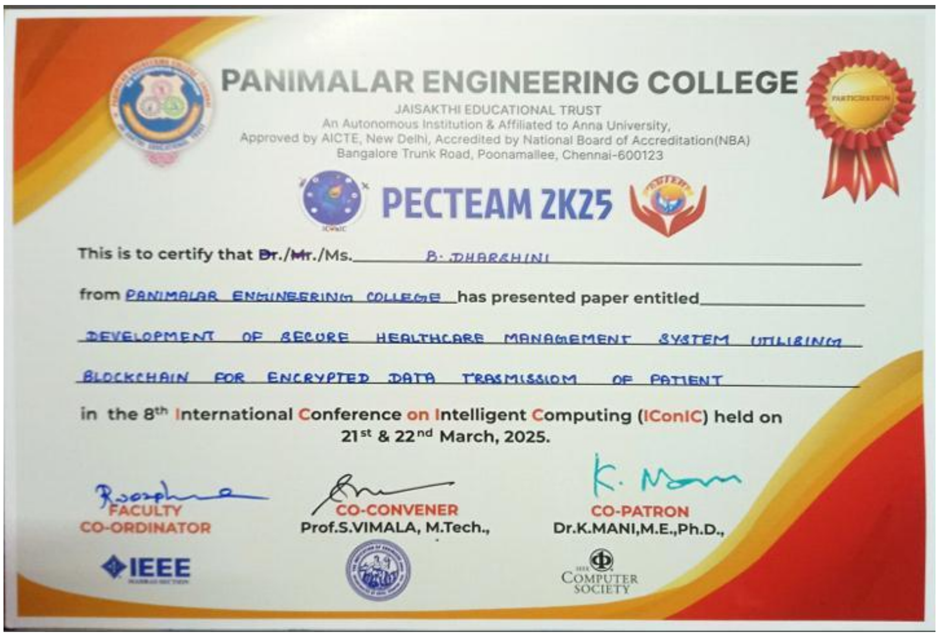
A.5 PAPER PUBLICATION:

PANIMALAR ENGINEERING COLLEGE - CHENNAI
JAISAKTHI EDUCATIONAL TRUST
PROSPERITY ENDURANCE CHARACTER
IConIC
8th INTERNATIONAL CONFERENCE on INTELLIGENT COMPUTING
IConIC 2K25
21st & 22nd March 2025
Chennai, India

Conference: 8th INTERNATIONAL CONFERENCE on INTELLIGENT COMPUTING

Paper Id: 957

Title: Development Of A Secure Healthcare Management System Utilizing Blockchain Technology For Encrypted Patient Data Transmission



PANIMALAR ENGINEERING COLLEGE

JAISAKTHI EDUCATIONAL TRUST
An Autonomous Institution & Affiliated to Anna University,
Approved by AICTE, New Delhi, Accredited by National Board of Accreditation(NBA)
Bangalore Trunk Road, Poonamallee, Chennai-600123

PECTEAM 2K25

This is to certify that ~~Dr.~~/~~Mr.~~/Ms. B. DHARSHINI

from PANIMALAR ENGINEERING COLLEGE has presented paper entitled DEVELOPMENT OF SECURE HEALTHCARE MANAGEMENT SYSTEM UTILISING BLOCKCHAIN FOR ENCRYPTED DATA TRASMISSIOM OF PATIENT

in the 8th International Conference on Intelligent Computing (IConIC) held on 21st & 22nd March, 2025.

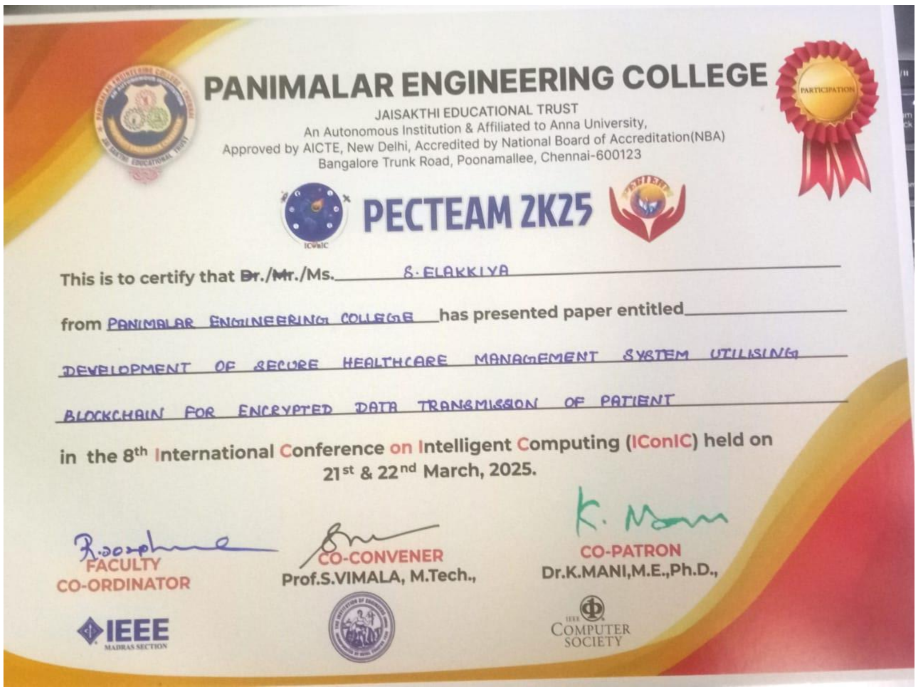
[illegible]
FACULTY CO-ORDINATOR

[illegible]
CO-CONVENER
Prof.S.VIMALA, M.Tech.,

[illegible]
CO-PATRON
Dr.K.MANI,M.E.,Ph.D.,

IEEE MADRAS SECTION

IEEE COMPUTER SOCIETY



PANIMALAR ENGINEERING COLLEGE

JAISAKTHI EDUCATIONAL TRUST
An Autonomous Institution & Affiliated to Anna University,
Approved by AICTE, New Delhi, Accredited by National Board of Accreditation(NBA)
Bangalore Trunk Road, Poonamallee, Chennai-600123

PECTEAM 2K25

This is to certify that ~~Dr.~~/~~Mr.~~/Ms. S. ELAKKIYA

from PANIMALAR ENGINEERING COLLEGE has presented paper entitled DEVELOPMENT OF SECURE HEALTHCARE MANAGEMENT SYSTEM UTILISING BLOCKCHAIN FOR ENCRYPTED DATA TRANSMISSION OF PATIENT

in the 8th International Conference on Intelligent Computing (IConIC) held on 21st & 22nd March, 2025.

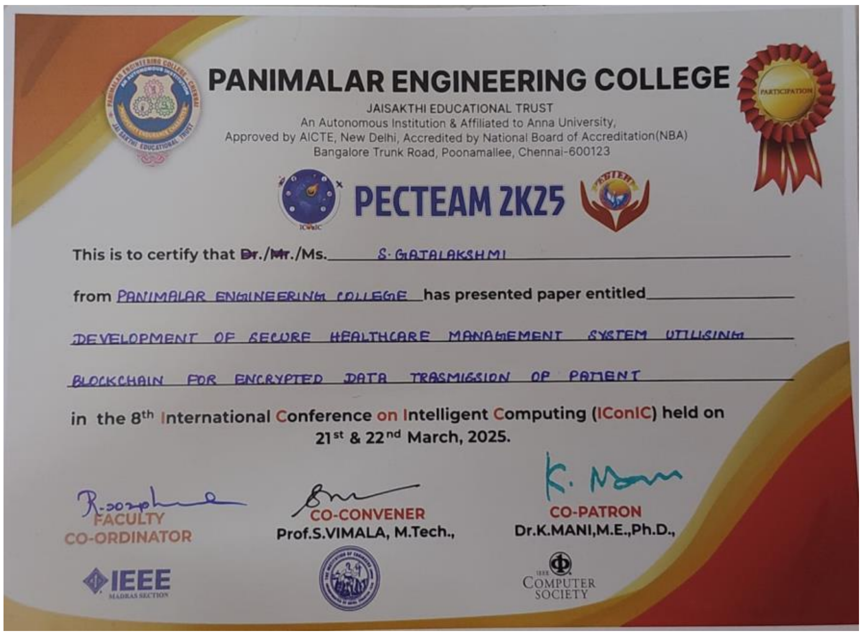
[illegible]
FACULTY CO-ORDINATOR

[illegible]
CO-CONVENER
Prof.S.VIMALA, M.Tech.,

[illegible]
CO-PATRON
Dr.K.MANI,M.E.,Ph.D.,

IEEE MADRAS SECTION

IEEE COMPUTER SOCIETY


PANIMALAR ENGINEERING COLLEGE

JAISAKTHI EDUCATIONAL TRUST
An Autonomous Institution & Affiliated to Anna University,
Approved by AICTE, New Delhi, Accredited by National Board of Accreditation(NBA)
Bangalore Trunk Road, Poonamallee, Chennai-600123

PARTICIPATION

PECTEAM 2K25

This is to certify that ~~Dr.~~/~~Mr.~~/Ms. S. GAJALAKSHMI

from PANIMALAR ENGINEERING COLLEGE has presented paper entitled

DEVELOPMENT OF SECURE HEALTHCARE MANAGEMENT SYSTEM UTILISING BLOCKCHAIN FOR ENCRYPTED DATA TRASMISSION OF PATIENT

in the 8th International Conference on Intelligent Computing (IConIC) held on 21st & 22nd March, 2025.

[illegible] FACULTY CO-ORDINATOR

[illegible] CO-CONVENER Prof.S.VIMALA, M.Tech.,

[illegible] CO-PATRON Dr.K.MANI,M.E.,Ph.D.,

IEEE MADRAS SECTION

IEEE COMPUTER SOCIETY

REFERENCES

[1] The Radicati Group Inc., "Cloud Email and Collaboration-Market Quadrant 2019," https://
www.radicati.com/wp/wp-content/uploads/2019/03/ Cloud-Email-and-Collaboration-Market-Quadrant-2019-Brochure. pdf, March 2019, accessed April 8, 2019.

[2] Tim Sadler, "The Year of Email Data Breaches," https://www.infosecurity magazine.com/opinions/
2017-email-data-breaches/, January 2018, accessed September 11,
2019.

[3] Wikileaks, "Hillary Clinton Email Archive," https://wikileaks.org/clinton-emails/, March 2016, accessed April 8, 2019.

[4] ——, "The Podesta Emails," https://wikileaks.org/ podesta-emails/, March 2016, accessed April 8, 2019.

[5] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer, "OpenPGP Message Format," https://tools.ietf.org/html/ rfc4880, November 2007, RFC 4880 (Proposed Standard).

[6] B. Ramsdell and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification," https: //tools.ietf.org/html/rfc5751, January 2010, RFC 5751 (Proposed Standard).

[7] R. Abu-Salma, M. A. Sasse, J. Bonneau, A. Danilova, A. Naiakshina, and M. Smith, "Obstacles to the adoption of secure communication tools," in 2017 IEEE Symposium on Security and Privacy. IEEE, 2017, pp. 137–153.

[8] S. Ruoti, J. Andersen, D. Zappala, and K. Seamons. (2015) Why johnny still, still can't encrypt: Evaluating the usability of a modern pgp client. [Online]. Available: https://arxiv.org/pdf/ 1510.08555.pdf

[9] S. Sheng, L. Broderick, C. A. Koranda, and J. J. Hyland, "Why johnny still can't encrypt: evaluating the usability of email encryption software," in Symposium On Usable Privacy and Security, 2006, pp. 3–4.

[10] A. Shamir, "Identity-based cryptosystems and signature schemes," in Advances in Cryptology–CRYPTO 1984. Springer, 1984, pp. 47– 53.

[11] Proofpoint, “Proofpoint Email Protection,” https://www.proofpoint.com/us/products/email-protection, 2005, accessed April 18, 2019.

[12] DataMotion, “DataMotion SecureMail,” https://www.proofpoint.com/us/products/email-protection, 2013, accessed April 18, 2019.

[13] N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith, “SoK:
secure messaging,” in 2015 IEEE Symposium on Security and Privacy. IEEE, 2015, pp. 232–249.

[14] H.-M. Sun, B.-T. Hsieh, and H.-J. Hwang, “Secure e-mail protocols providing perfect forward secrecy,” IEEE Communications Letters, vol. 9, no. 1, pp. 58–60, 2005.

[15] J. O. Kwon, I. R. Jeong, and D. H. Lee, “A forward-secure e-mail protocol without certificated public keys,” Information Sciences, vol. 179, no. 24, pp. 4227–4231, 2009.