**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

**PANIMALAR ENGINEERING COLLEGE**

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

## 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., HEAD OF THE DEPARTMENT** | **Dr.KAVITHA SUBRAMANI M.E,PhD., 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.......................................

**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 **Dr.KAVITHA SUBRAMANI (M.E., PhD.)** 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** and **Guide Dr.KAVITHA SUBRAMANI M.E. PhD.,** and 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.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER**  **NO.** | **TITLE** | **PAGE**  **NO.** |
|  | **ABSTRACT**  **LIST OF FIGURES**  **LIST OF ABBREVIATIONS** | IV VII IX |
| **1.** | **INTRODUCTION** |  |
|  | 1.1 PROBLEM STATEMENT | 2 |
| 1.2 EXISTING SYSTEM | 3 |
| **2** | **LITERURE SURVEY** | 4 |
| **3.** | **THEORETICAL BACKGROUND**  3.1 IMPLEMENTATION ENVIRONMENT | 12  13 |
|  | 3.2 SYSTEM ARCHITECTURE | 15 |
|  | 3.3 PROPOSED METHODOLOGY | 16 |
|  | 3.3.1 INPUT DESIGN | 18 |
|  | 3.3.2 UML DIAGRAM | 19 |
|  | 3.3.3 USE CASE DIAGRAM | 19 |
|  | 3.3.4 ACTIVITY DIAGRAM | 20 |
|  | 3.3.5 SEQUENCE DIAGRAM | 21 |
|  | 3.3.6 ER DIAGRAM | 22 |

|  |  |  |
| --- | --- | --- |
| **4.** | **SYSTEM IMPLEMENTATION**   * 1. **MODULES**      1. USER MODULE      2. ADMIN MODULE      3. DOCTOR MODULE      4. SURGEON MODULE      5. RADIOLOGIST MODULE      6. PHARMACIST MODULE      7. CHAT MODULE   2. **ALGORITHMS**      1. SHA-256 ALGORITHM      2. ForodoKEM ALGORITHM      3. BLOCKCHAIN | 25  26  26  26  27  27  28  28  29  30  30  30  31 |
| **5.** | **RESULTS & DISCUSSION**   * 1. SYSTEM TESTING   2. RESULT & DISCUSSION | 33  34  36 |
| **6.** | **CONCLUSION AND FUTURE WORK**   * 1. CONCLUSION   2. FUTURE WORK | 38  39  39 |
|  | **APPENDICES**   * 1. SDG Goals   2. Source Code   3. Screen Shots | 40  40  40  58 |

|  |  |
| --- | --- |
| * 1. Plagiarism Report   2. Paper Publication | 63  75 |
| **REFERENCES** | 78 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **FIGURE NAME** | **PAGE NO** |
| 3.1 | SYSTEM ARCHITECTURE | 15 |
| 3.2 | INPUT DESIGN | 18 |
| 3.3 | USE CASE DIAGRAM | 19 |
| 3.4 | DFD DIAGRAM | 20 |
| 3.5 | ACTIVITY DIAGRAM | 21 |
| 3.6 | SEQUENCE DIAGRAM | 22 |
| 3.7 | ER DIAGRAM | 23 |
| A.3.1 | HOME PAGE | 58 |
| A.3.2 | USER REGISTRATION AND LOGIN | 59 |
| A.3.3 | USER MODULE | 60 |
| A.3.4 | DOCTOR REGISTRSTION AND SIGN- IN PAGE | 61 |
| A.3.5 | SURGEON PAGE | 62 |
| A.3.6 | PHARMACY SIGN-IN AND LOGIN PAGE | 63 |

|  |  |  |
| --- | --- | --- |
| A.3.7 | CHAT MODULE | 64 |
| A.3.8 | ADMIN LOGIN | 65 |
| A.3.9 | RADIOLOGIST REGISTER AND LOGIN PAGE | 66 |
| A.3.10 | RADIOLOGIST MODULE | 67 |

**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

## 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.

## 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.

## 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 REVIEW

## CHAPTER 2 LITERATURE SURVEY

**"Achieving Secure, Verifiable, and Efficient Boolean Keyword Searchable Encryption for Cloud Data Warehouse" – Thanaruk Theeramunkong, Somchart Fugkeaw [1] .**This study proposes a secure and verifiable searchable encryption scheme tailored for Cloud Data Warehouses (CDWs), addressing the limitations of traditional Searchable Encryption (SE) techniques. It introduces support for Boolean expression queries over encrypted data, combining Partial Homomorphic Encryption (PHE), B+Tree, Inverted Index, and bitmapping functions for privacy-preserving and efficient search performance. By integrating blockchain and smart contracts, the scheme automates authentication, index retention, and trapdoor generation without relying on third-party verifications. Comparative evaluations reveal its superior efficiency and effectiveness over existing approaches. **Advantage:** Supports Boolean expression queries, enables scalable and privacy-preserving search operations through blockchain, and ensures trustworthiness without third-party dependency. **Disadvantage:** Complex integration and higher costs due to the employment of advanced technologies like blockchain and PHE.

### "MaxD K-means: A clustering algorithm for Auto-generation of centroids and distance of data points in clusters" – Tutut Herawan, Abul Beg [2].

This study introduces the MaxD K-Means clustering algorithm, a novel enhancement to the traditional K-Means method. Unlike standard K-Means, where users must specify the number of clusters (k) beforehand, MaxD K-Means automatically determines the initial value of k and employs a unique strategy for setting initial centroids. Experiments conducted with synthetic data from Lloyd's

K-Means tests demonstrate significant improvements, including a reduction in the number of iterations by up to 78%. **Advantage:** Eliminates the need for user- defined cluster count, enhances initialization of centroids, and significantly improves iteration efficiency. **Disadvantage:** Limited testing on real-world data may require further validation for broader applicability.

### "Privacy-Preserving Patient-Centric Clinical Decision Support System on Naive Bayesian Classification" – Rongxing Lu, Ximeng Liu, Jianfeng Ma[3].

This study presents PPCD, a privacy-preserving clinical decision support system designed to assist clinicians in diagnosing patient disease risks while safeguarding sensitive medical data. Leveraging naive Bayesian classification, the system analyzes large amounts of clinical data without compromising individual privacy. Historical patient data stored in the cloud trains the classifier securely, using an innovative cryptographic tool—additive homomorphic proxy aggregation scheme—and a privacy-preserving protocol for retrieving top-k disease names based on patient preferences. Extensive simulations show PPCD efficiently computes disease risks with high accuracy while maintaining robust privacy measures. **Advantage:** Improves diagnosis accuracy and efficiency, ensures patient data privacy using advanced cryptographic techniques, and enables secure top-k disease retrieval. **Disadvantage:** Practical implementation may face challenges due to the complexity of cryptographic processes and reliance on cloud- based infrastructures.

**"Handling Privacy-Sensitive Medical Data With Federated Learning: Challenges and Future Directions" – Ons Aouedi, Alessio Sacco, Kandaraj Piamrat, Guido Marchetto [4] .**This study investigates the use of Federated Learning (FL) in healthcare to manage privacy-sensitive medical data securely. FL

enables multiple institutions to collaboratively train machine learning models without sharing raw patient data, ensuring privacy and security. The research outlines key challenges such as data heterogeneity, communication overhead, and security vulnerabilities like inference and poisoning attacks. It also explores potential solutions, including blockchain integration, differential privacy, and homomorphic encryption, to enhance security and efficiency within federated healthcare systems.**Advantage:** Ensures data privacy during collaborative model training, addresses healthcare-specific challenges, and offers innovative solutions for secure systems integration. **Disadvantage:** Issues like communication overhead and vulnerabilities to attacks require further refinement for widespread practical implementation.

### "An Investigation Into Patient Privacy Disclosure in Online Medical Platforms" – Chun-Lin Feng, Zhi-Chao Cheng, Li-Juan Huang [5] .

This study investigates the factors that influence patients’ willingness to disclose personal health information on online medical platforms. It emphasizes the importance of fairness perceptions—both outcome fairness and procedural fairness—and examines the role of perceived platform interactivity in shaping privacy disclosure behaviors. Based on a survey of 1,546 users, the research highlights that fairness and interactivity significantly affect patients' decisions, offering valuable insights for enhancing privacy management in online healthcare communities. **Advantage:** Provides insights into improving patient privacy management and highlights key factors—fairness and interactivity—that influence privacy disclosure. **Disadvantage:** Relies on survey data, which may not fully account for variations in behavior across diverse patient demographics or platform designs.

### "Secure ID Privacy and Inference Threat Prevention Mechanisms for Distributed Systems" – Tahani Hamad Aljohani, Ning Zhang [6]

This study proposes the SPID framework, a security solution for distributed healthcare systems designed to prevent inference attacks and protect patient identity privacy. By leveraging a distributed set of servers owned by different providers, SPID ensures secure upload and encryption of health data to multiple foreign servers, blocking unauthorized access and inference-based tracking. Key mechanisms include elliptic curve cryptography (ECC), anonymous authentication, pseudonym-based access control, and double encryption techniques. Performance evaluations using benchmarking tools and queuing theory demonstrate the framework's effectiveness in enhancing security and mitigating threats. **Advantage:** Provides strong protection against inference attacks and identity breaches, utilizes advanced cryptographic techniques for secure data handling, and offers improved performance through distributed architecture. **Disadvantage:** Implementation may be resource-intensive and complex due to reliance on cryptography and distributed server systems.

**"A Conceptual Framework to Ensure Privacy in Patient Record Management System" – K. G. Srinivasa, K. R. Venugopal, S. S. Manvi, L. M. Patnaik [7]** This study proposes a conceptual framework aimed at preserving privacy in cloud- based patient record management systems. It classifies health data into personal and medical information, applies encryption techniques such as SHA-256 and public key cryptography, and ensures controlled access based on defined privacy policies. The framework emphasizes secure storage and retrieval by authorized personnel only, and highlights patient-centric control over health data. **Advantage:** Offers strong privacy control and data confidentiality through

advanced encryption, supports role-based access, and enhances patient trust in cloud environments. **Disadvantage:** Remains conceptual without real-world implementation or performance validation, and may face practical integration and key management issues.

**"A Fog-Based Middleware for Automated Compliance With OECD Privacy Principles in Internet of Healthcare Things" – Ahmed A. Alzahrani, Rajkumar Buyya[8]** .This study proposes a fog-based middleware architecture to ensure automated compliance with OECD privacy principles in Internet of Healthcare Things (IoHT) applications. The middleware acts as a bridge between healthcare applications and IoHT infrastructure, enforcing privacy policies through components like the Policy Mapper, Policy Decision Point (PDP), Policy Enforcement Point (PEP), and Context Handler. By translating human-readable privacy rules into enforceable system policies, the framework restricts unauthorized access and enforces purpose-based data sharing. **Advantage**: Enables real-time, automated privacy enforcement without manual intervention; ensures compliance with international privacy guidelines; reduces data exposure risks through fog-based local processing. **Disadvantage**: May introduce computational overhead at the fog layer; complexity in handling diverse privacy rules across jurisdictions; lacks empirical performance evaluation in real-world scenarios

**"Digital Privacy in Healthcare: State-of-the-Art and Future Vision" – Khaled Salah, Raja Jayaraman, and Davor Svetinovic [9]** This study presents a comprehensive analysis of digital privacy challenges in healthcare and explores current and future technologies to address these concerns.

The paper reviews existing privacy-preserving frameworks and emphasizes the importance of patient-centric privacy models, particularly in the era of digital health transformation. It discusses technologies such as blockchain, differential privacy, and secure multi-party computation, evaluating their roles in safeguarding sensitive patient information. Moreover, it highlights regulatory frameworks and the need for better interoperability and user control in managing health data. The future vision includes AI-driven privacy risk assessment tools and enhanced cryptographic protocols for real-time privacy enforcement. **Advantages:**Provides a broad overview of privacy technologies, integrates legal and technical perspectives, proposes forward-looking privacy enhancements using AI and cryptography, and supports patient empowerment in data control.**Disadvantages:** Lacks a specific implementation model, and some future visions are theoretical and may require significant development before practical adoption.

.

**"Advancing Federated Learning Through Novel Mechanism for Privacy Preservation in Healthcare Applications" – Mehdi Sookhak, Abbas Kiani, Alireza Jolfaei, etal.[10]**This study presents a privacy-preserving approach for healthcare systems using a federated learning framework that eliminates the need for centralized data collection. The proposed mechanism ensures data confidentiality by training models locally on user devices and only sharing the model updates with the server. To strengthen privacy, the framework introduces a novel Hybrid Aggregation Technique (HAT) and employs optimized differential privacy techniques to prevent information leakage. This allows hospitals and medical institutions to collaboratively train accurate machine learning models without compromising sensitive patient data. The work also integrates blockchain for auditability and secure model update tracking.

**Advantage:**Preserves data privacy by avoiding raw data sharing, enhances accuracy through hybrid aggregation, provides auditability with blockchain integration, and ensures privacy with optimized differential privacy.**Disadvantage:** Increased complexity in implementation due to integration of multiple technologies like federated learning, differential privacy, and blockchain; performance may be affected by heterogeneous local data and device limitations.

**"Hilbert Convex Similarity for Highly Secure Random Distribution of Patient Privacy Steganography" – V. Rajesh, S. Vijayakumar, T. Devi, K. Balasubramanian[11].** This study proposes a novel steganography method using Hilbert Convex Similarity (HCS) to ensure high-security data hiding in patient privacy protection. The framework integrates secure data embedding with random distribution and advanced similarity-based encoding. It effectively conceals sensitive medical information in medical images using an optimized secret key generation and distribution method. The HCS-based approach increases imperceptibility and security by maintaining similarity between the original and stego image, making unauthorized extraction difficult. Experimental analysis confirms the method's robustness, image quality retention, and resilience against steganalysis attacks. **Advantage:** Ensures highly secure data hiding using random distribution and similarity-based encoding, maintains image quality and resists steganalysis. **Disadvantage:** Computational complexity may increase due to multiple transformation steps and secret key generation; limited to image-based data hiding scenarios.

# CHAPTER 3 THEORETICAL BACKGROUND

## CHAPTER 3 THEORETICAL BACKGROUND

* 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.

## 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

## 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.

## TECHNOLOGIES UTILIZED

### Blockchain

Blockchain is a **decentralized and secure digital ledger** technology that stores data in blocks linked together in a chain. Each block contains data, a timestamp, and a cryptographic hash of the previous block, making it tamper-proof. In healthcare, blockchain is used to **securely store patient data**, prevent unauthorized access, and maintain data integrity across distributed systems.

### FrodoKEM

FrodoKEM is a **post-quantum cryptographic algorithm** designed to stay secure even against future quantum computers. It belongs to a class of algorithms called **Key Encapsulation Mechanisms (KEM)**, which are used for safely sharing encryption keys. FrodoKEM is based on hard mathematical problems (Learning with Errors) and is used in systems that need **high security for key exchange**, such as healthcare platforms protecting patient data.

## .3.2 SYSTEM ARCHITECTURE

The systems architect establishes the basic structure of the system, we propose a Hash code Solomon algorithm and we can put a small part of data in local machine and fog server in order to protect the privacy. Moreover, based on computational intelligence, this algorithm can compute the distribution proportion stored in cloud, fog, and local machine, respectively..

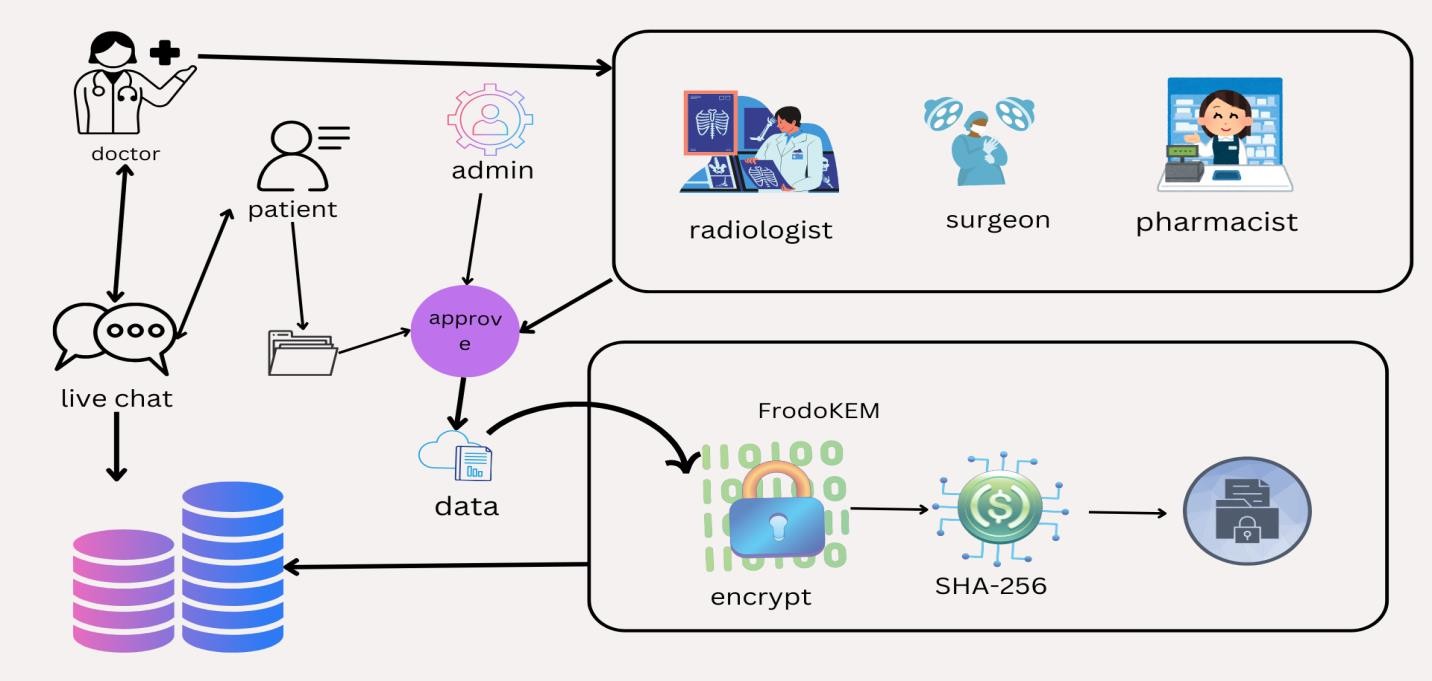


Fig 3.1 System Architecture Diagram

## PROPOSED METHODOLOGY

The proposed system addresses privacy concerns in healthcare by integrating edge computing and blockchain technology to create a secure and decentralized data management environment. Patient data is processed locally on edge devices within healthcare institutions, minimizing the risk of data breaches associated with cloud storage and ensuring faster response times. A blockchain ledger is used to record all access and transactions transparently, preventing tampering and unauthorized changes. Encryption techniques like **FrodoKEM** and **SHA-256** protect sensitive health information, while the open-source nature of the system ensures adaptability and transparency. Patients maintain full control over their medical records through data sovereignty, allowing them to selectively grant access to authorized healthcare providers. This architecture not only strengthens privacy and security but also builds patient trust by offering transparency, accountability, and compliance with privacy regulations.

## OBJECTIVES

The primary objective of this project is to design and implement a secure and efficient healthcare data management system by leveraging blockchain technology and encryption mechanisms. The system aims to enhance data protection, streamline communication between healthcare departments, and ensure patient confidentiality and regulatory compliance.

### Enhance Data Security and Integrity:

Implement blockchain technology to create a secure and tamper-proof ledger for recording patient information. This ensures that all data remains accurate, immutable, and protected from unauthorized alterations or tampering.

### Streamline Specialist and Department Integration:

Develop a structured registration and login system for various healthcare professionals—such as client specialists, radiologists, surgeons, and pharmacists—allowing them to access patient records through a role-based endorsement process managed by administrators.

### Maintain Patient Confidentiality:

Introduce a unique encrypted identifier system for each patient to prevent data misuse. This ensures that only authorized personnel from relevant departments can access sensitive health information.

* **Facilitate Efficient Data Handling Across Departments:** Enable seamless communication and data flow between different healthcare departments by forwarding encrypted patient identifiers. This helps maintain accuracy and security while transitioning patient data across various stages of medical care.

### Ensure Compliance and System Reliability:

Build a robust and trustworthy system that complies with existing healthcare data protection standards. The system should foster patient trust by ensuring the integrity, confidentiality, and availability of personal medical information at all times.

## INPUT DESIGN

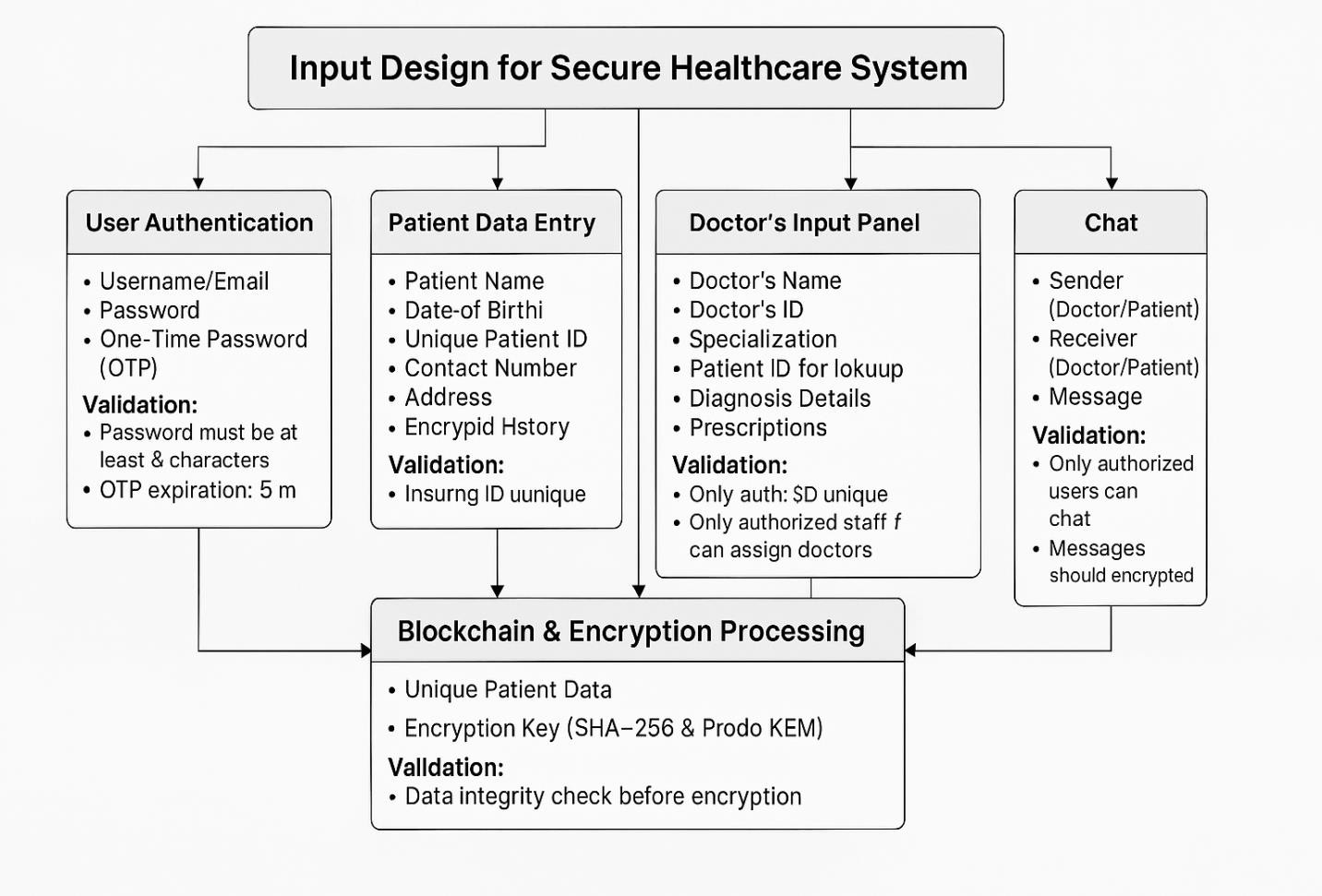
****

Fig 3.2 Input Design

The secure healthcare system ensures authorized access through user authentication using a **username,** data entry includes **personal details, contact information, and a unique ID**, ensuring no duplicates. Doctors can access a panel to **enter diagnoses, prescriptions, and lookup patient details**, with only authorized staff allowed to assign doctors. A **secure chat system** enables encrypted communication between doctors and patients, restricting access to authorized users. All data is processed using **SHA-256 and FrodoKEM encryption**, ensuring integrity and protection against unauthorized access.

## UML DIAGRAM

### Usecase Diagram

The use case diagram is the main building block of [object oriented](http://en.wikipedia.org/wiki/Object_oriented) modeling. It is used both for general [conceptual modeling](http://en.wikipedia.org/wiki/Conceptual_model) of the systematic of the application, and for detailed modeling translating the models into [programming code](http://en.wikipedia.org/wiki/Programming_code). For this in our component diagram first propose a data In this proposed method we are using Hash-Solomon Code Algorithm to encrypt the data.

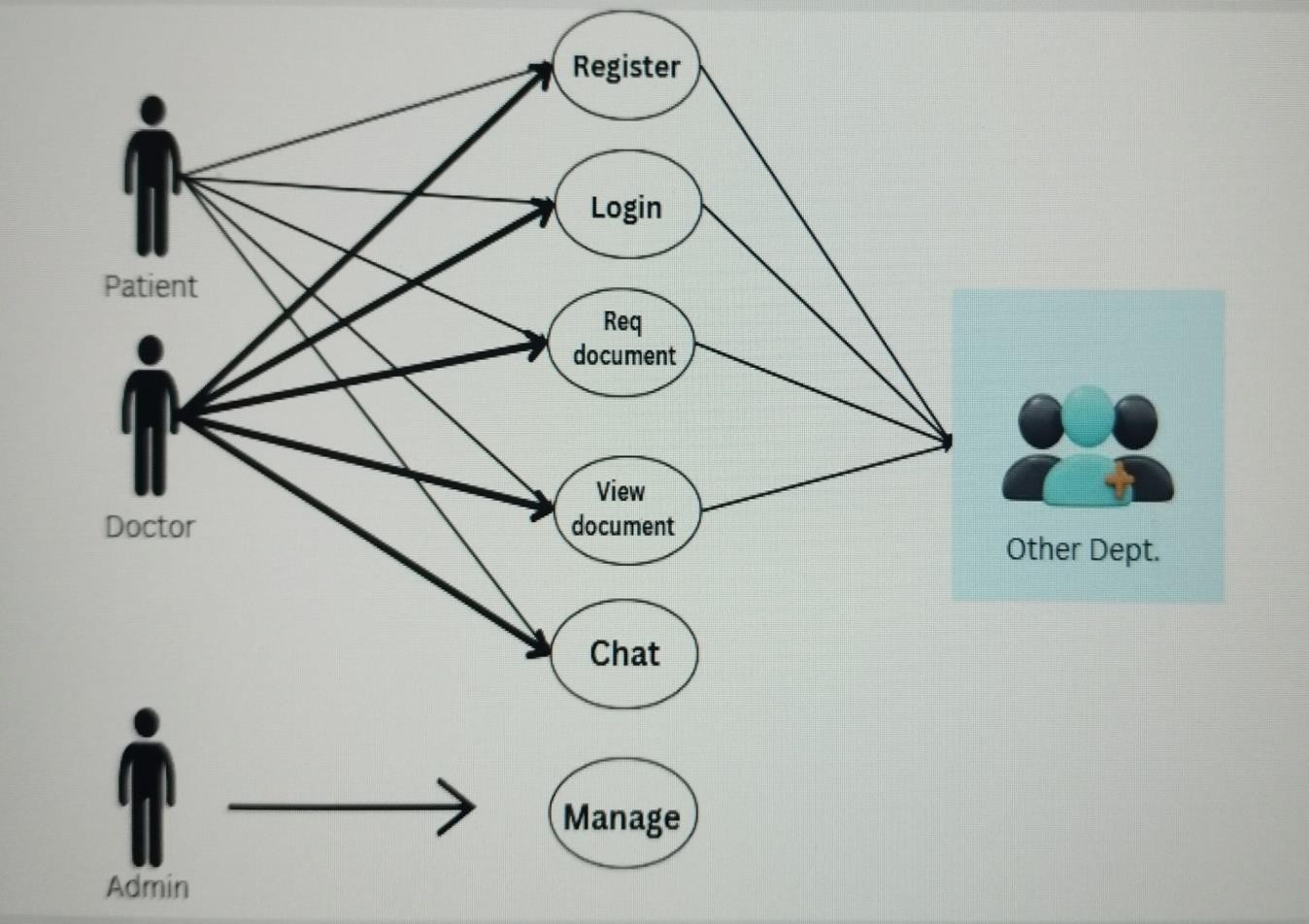


Fig 3.3 Usecase Diagram

### Activity Diagram

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

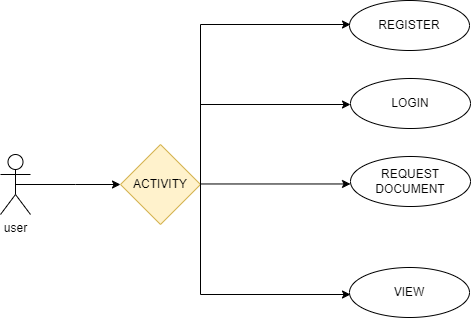


Fig 3.4 Activity Diagram

### Sequence Diagram

In our sequence diagram specifying processes operate with one another and in order. In our sequence diagram first propose a f or this in our component diagram first propose a data in this proposed method we are using Hash-Solomon Code Algorithm to encrypt the data.

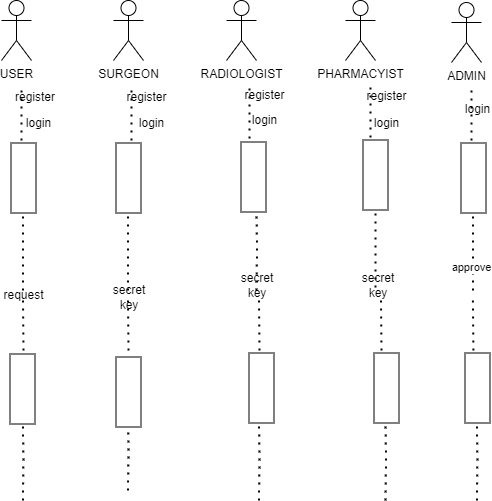


Fig 3.5 Sequence Diagram

### ER Diagram

The diagram represents a structured request approval system in a secure healthcare environment. Users initiate requests such as applying for access or approvals, which are stored in a central database. Administrators manage the approval process by logging in, verifying users, and forwarding requests to relevant departments. Departments cross-verify the requests to ensure compliance and security before granting final approval. If validated, the request is either approved or accepted, allowing users to proceed. This workflow ensures that only authorized users gain access, enhancing data integrity, privacy, and security within the system.

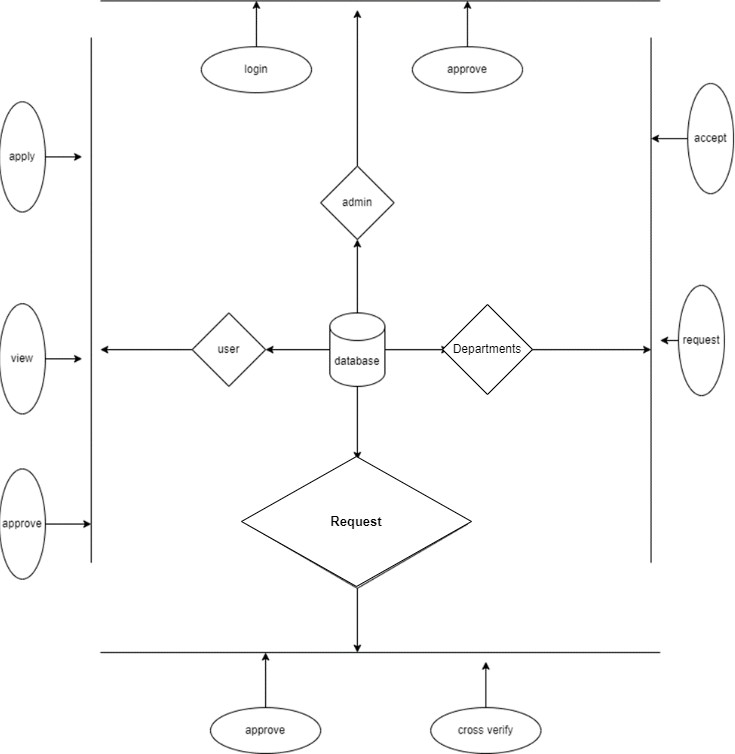


Fig 3.6 ER Diagram

### DF Diagram

The diagram illustrates a secure healthcare data management system where different users, including doctors, surgeons, radiologists, and pharmacists, interact with a centralized database. Users can send requests, while doctors can upload data and engage in live chat for communication. Each department accesses data by entering a secret key, ensuring security before viewing information. Admins oversee the approval process, maintaining control over access permissions. This structure enhances security, ensures authorized data access, and facilitates seamless collaboration among healthcare professionals while protecting patient data.

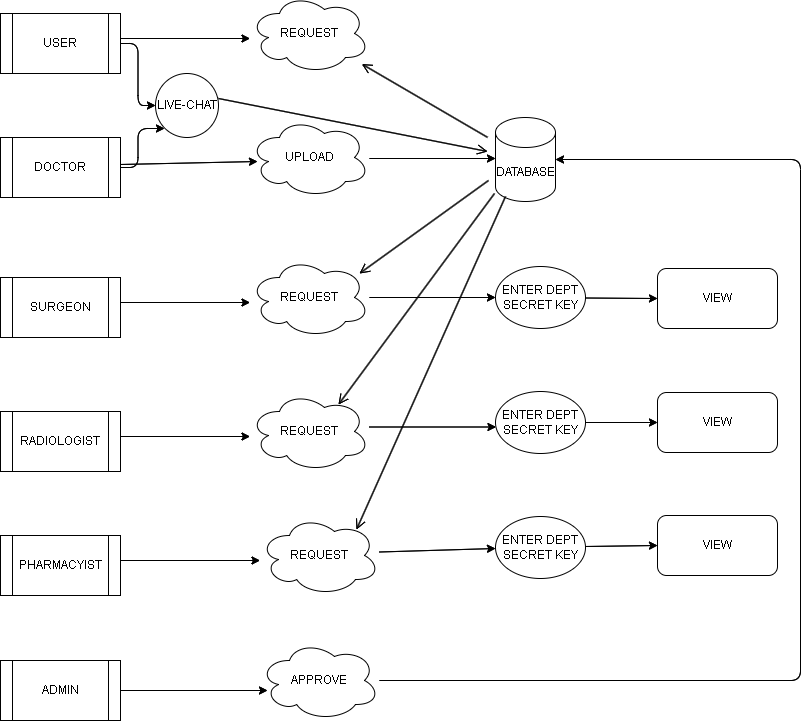


Fig 3.7 DF Diagram

### State Diagram

A record is maintained containing medical details with an assigned ID for prescription; notes of the treatment are mentioned and prescription along with an ForodoKEM encrypted key generated and transmitted, encrypted, with unique IDs by various departments-surgery, radiology, or the pharmacy department-departmental authority decryption with individual IDs using keys access for prescriptions. Any changes or updates to the treatment plan are documented and stored in the hospital's database. Data at rest is encrypted using the SHA-256 hashing algorithm. This combination of AES encryption for transmission and SHA-256 hashing for storage ensures data confidentiality and integrity at all stages of the process.

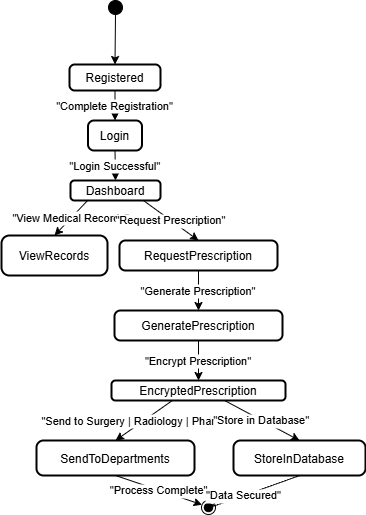


Fig 3.8 State Diagram

# CHAPTER 4 SYSTEM IMPLEMENTATION

**CHAPTER 4 SYSTEM IMPLEMENTATION**

* 1. **MODULES**
     1. **USER MODULE**

The **User Module** is a fundamental part of our secure healthcare system, enabling users to register, log in, request documents, and view authorized medical records. Users begin by registering their details to create a secure account, ensuring only authorized individuals access the system. After successful registration, they can log in using their credentials, undergoing authentication for security. Through the document request feature, users can formally request access to specific healthcare records, which are securely processed to maintain data privacy. Once approved, users can view the requested documents within their access permissions. This module ensures a secure, efficient, and privacy-focused system for managing sensitive patient data while preventing unauthorized access.

## ADMIN MODULE

The **Admin Module** plays a crucial role in managing doctor registrations and maintaining system integrity within the secure healthcare system. The process begins with the admin logging into the system, ensuring only authorized personnel can oversee doctor approvals. Upon logging in, the admin can view doctor registration requests submitted by healthcare professionals. After reviewing these requests, the admin has the authority to approve or reject them based on verification criteria. Once approved, the system maintains the records, ensuring that only authenticated doctors can access patient data securely. This module

ensures a streamlined, secure, and efficient approval process for doctors while maintaining system integrity.

## DOCTORS MODULE

The **Doctor Consultation Module** is a structured system that allows doctors to manage patient interactions efficiently. The process begins with **registration**, where doctors create an account to access the platform, or they can directly **log in** if they are already registered. After logging in, doctors are directed to the **dashboard**, which serves as the central hub for managing their activities. From the dashboard, doctors can **view free consultations**, allowing them to monitor and respond to general patient inquiries. They can also **provide suggestions**, offering medical advice based on patient concerns. Additionally, doctors can **view patient requests**, where they can see appointment or consultation requests submitted by patients. Another key feature is the ability to **accept or reject appointments**, enabling doctors to manage their schedules effectively. If an appointment is accepted, doctors can engage in **manual chat with patients**, facilitating direct communication for more personalized medical guidance. This module enhances doctor-patient interaction by streamlining consultation processes, improving accessibility, and ensuring efficient communication, all while maintaining data security and confidentiality.

## SURGEON MODULE

The **Surgeon Access Module** is a secure system that allows surgeons to access patient reports through a multi-step authentication process. First, the surgeon must

register in the system and log in using their credentials. After logging in, they can navigate to the patient reports section. To enhance security, the system requires the surgeon to enter a **department secret key** for authorization, followed by their **staff ID** for additional verification. Once all credentials are validated, the surgeon can submit the request and gain access to the reports. This module ensures strict authentication, restricted access, and data confidentiality, preventing unauthorized access to sensitive patient information.

## RADIOLOGIST MODULE

The **Radiologist Access Module** is a secure system that enables radiologists to access patient reports while ensuring strict authentication and authorization. The process begins with the **registration**, where the radiologist creates an account to gain system access. After registration, they must **log in** using their credentials. Once authenticated, they can proceed to **view patient reports**, but access is further restricted through additional security measures. To ensure department-level authorization, the radiologist must enter a **department secret key**, which verifies their association with the medical division handling the reports. Further verification is required by entering their **staff ID**, ensuring that only authorized personnel can proceed. Once all credentials are validated, the **submit** action grants final access to the patient reports. This multi-layered security approach ensures that only registered and authorized radiologists can access sensitive medical data, maintaining confidentiality, preventing unauthorized access, and ensuring compliance with data protection regulations.

## PHARMACIST MODULE

The **Pharmacist Access Module** is a secure system that allows pharmacists to access patient reports while ensuring authentication and data protection. The process starts with the **registration**, where the pharmacist creates an account to gain system access. Once registered, they must **log in** using their credentials before proceeding further. After successful login, they can navigate to the **patient reports section**, where they can view necessary medical information related to prescriptions and medications. However, access is restricted with multiple security layers. The pharmacist must enter a **department secret key** to verify their authorization within the system. Following this, they must provide their **staff ID** as an additional authentication step to confirm their identity. Once all security checks are passed, they can **submit** the request and gain access to the patient reports. This module ensures that only registered and authorized pharmacists can access sensitive medical data, maintaining confidentiality, preventing unauthorized access, and ensuring compliance with healthcare data security regulations.

## 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.

## ALGORITHMS

* + 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.

### 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.**

### 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, preventing unauthorized access, and enhancing data privacy. By integrating blockchain, healthcare systems can overcome challenges related to data breaches, unauthorized modifications, and trust issues in electronic health records (EHRs).

### Key Features of Blockchain in This System:

* + - * **Decentralization:** Unlike traditional centralized databases, blockchain distributes data across multiple nodes, eliminating the risk of single-point failures and reducing dependency on a single authority. This ensures that patient records are always available, even in the event of system failures.
      * **Immutability:** Once recorded, patient data cannot be altered or deleted, ensuring data integrity and preventing malicious modifications. Every transaction is permanently stored, creating an audit trail that can be used for verification and compliance purposes.
      * **Security:** Using advanced cryptographic techniques such as SHA-256 hashing, FrodoKEM encryption, and digital signatures, blockchain ensures that patient records remain protected from cyber threats, unauthorized access, and data manipulation. Encryption safeguards patient confidentiality, while cryptographic hashing secures data integrity.
      * **Transparency and Trust:** Blockchain’s distributed ledger provides a transparent system where all authorized stakeholders can verify records without compromising patient privacy. Smart contracts can automate access control, ensuring that only authorized personnel can access or update records based on predefined rules.
      * **Interoperability:** Blockchain facilitates seamless data sharing between healthcare providers, insurance companies, and patients while maintaining security and compliance with regulations like HIPAA and GDPR. This eliminates data silos and enhances coordination among healthcare entities.

# CHAPTER 5 RESULTS AND DISCUSSION

## CHAPTER 5 RESULTS AND DISCUSSION

* 1. **SYSTEM TESTING**

### White Box Testing

White box testing is crucial for verifying the internal logic, security mechanisms, and blockchain transactions in the system. It ensures that patient data remains immutable by validating encryption mechanisms like SHA-256 hashing and FrodoKEM encryption**.** This testing method also examines role-based access control (RBAC) to confirm that only authorized users, such as doctors, pharmacists, and radiologists, can access patient data. Additionally, it helps in error handling and exception testing by ensuring that invalid inputs, authentication failures, and unauthorized access attempts are properly managed. Security testing within white box testing verifies encryption algorithms, digital signatures, and smart contract execution**,** preventing cyber threats. Furthermore, it optimizes code by detecting redundant logic, inefficient loops, and security vulnerabilities**,** ensuring a robust and efficient system.

### Black Box Testing

Black box testing is essential for evaluating the functional behavior of the healthcare system without inspecting its internal code structure. This method focuses on verifying whether patient data access, appointment scheduling, and medical report retrieval work as intended. Testers provide inputs and examine the outputs without considering how the system processes data internally. It ensures that the dashboard functionalities for doctors, pharmacists, and radiologists operate

correctly, preventing incorrect access to confidential patient records. Since blockchain integration adds complexity, black box testing validates whether transactions are successfully recorded and retrieved while ensuring system usability and responsiveness for healthcare professionals and patients.

### Unit Testing

Unit testing is critical for validating individual **modules and components** of the healthcare system. Each feature, such as user registration, login authentication, patient report access, and appointment management, is tested independently to ensure it functions correctly. In blockchain-based systems, unit testing is particularly important for verifying smart contract execution, cryptographic key management, and transaction processing to ensure data integrity. It also helps in debugging errors at an early stage, reducing risks before integrating different system components. By isolating and testing each module, unit testing ensures better performance, security, and reliability in handling sensitive healthcare data.

### Functional Testing

Functional testing ensures that the healthcare system operates as per the defined requirements by testing each feature for expected behavior. It validates key functionalities such as secure login, encrypted patient data storage, appointment booking, and report sharing among doctors, radiologists, and pharmacists. The process involves providing inputs, executing test cases, and comparing actual outputs with expected results to ensure accuracy. Functional testing also checks whether blockchain transactions are correctly recorded, verified, and retrieved, preventing unauthorized modifications.

## 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**

## 6.1. 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.

## 6.2 FUTURE ENHANCEMENTS

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.

## APPENDICES

* 1. **SDG Goals**

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

* 1. **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 + "]";

}

}

### 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));

}

}

## SCREEN SHOTS HOME PAGE

****

Fig A.3.1 Home Page

## USER REGISTRATION AND LOGIN

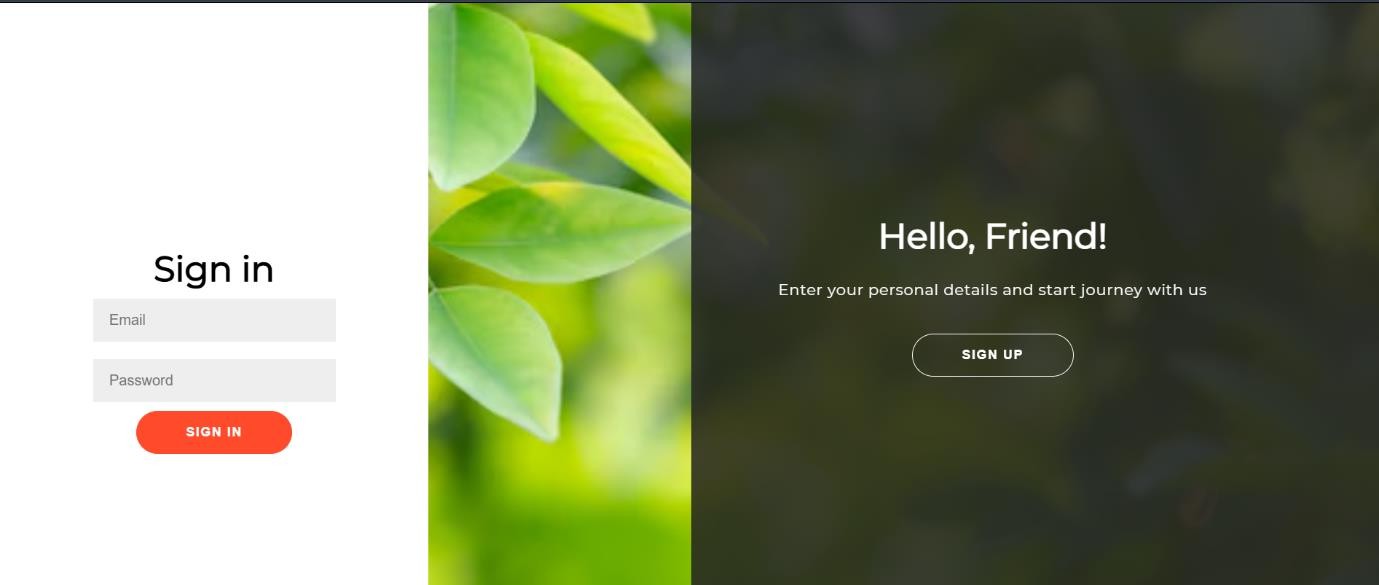
****

Fig A.3.2 User Registration and Login Page

## USER MODULE

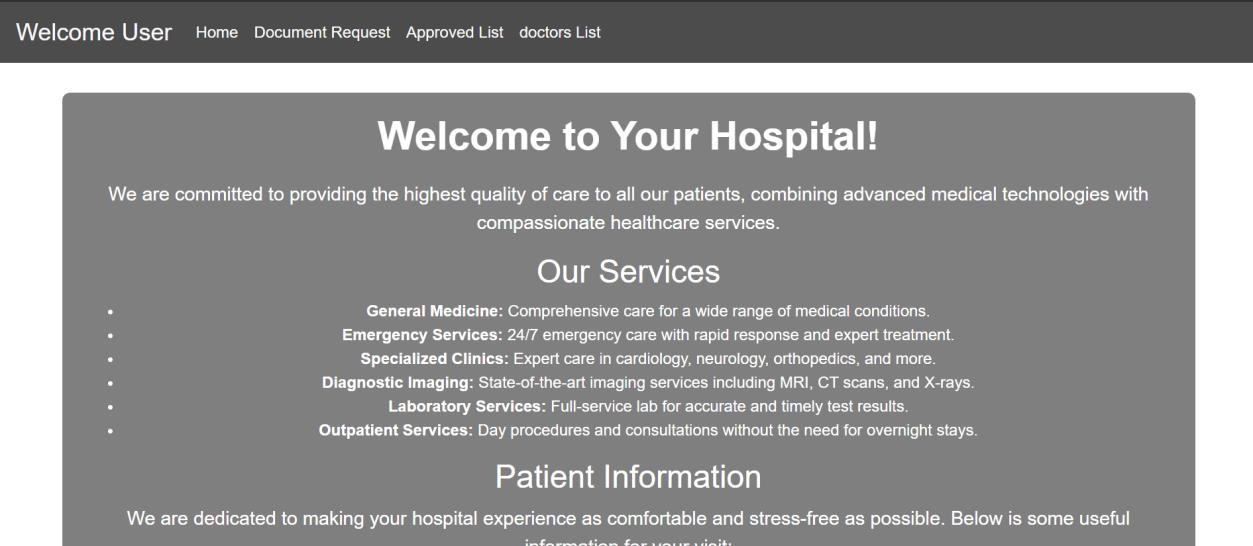
****

Fig A.3.3 User Module

## DOCTOR REGISTRATION AND SIGN-IN PAGE

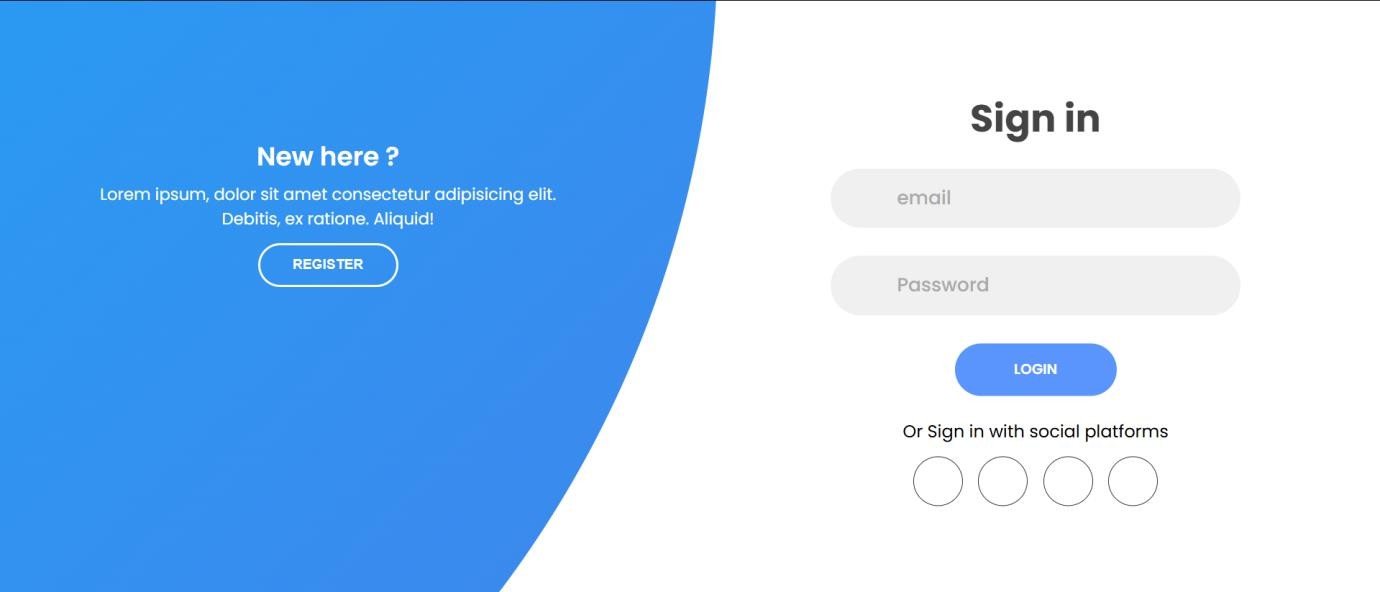
****

Fig A.3.4 Doctor Registration and Sign-in Page

## SURGEON PAGE

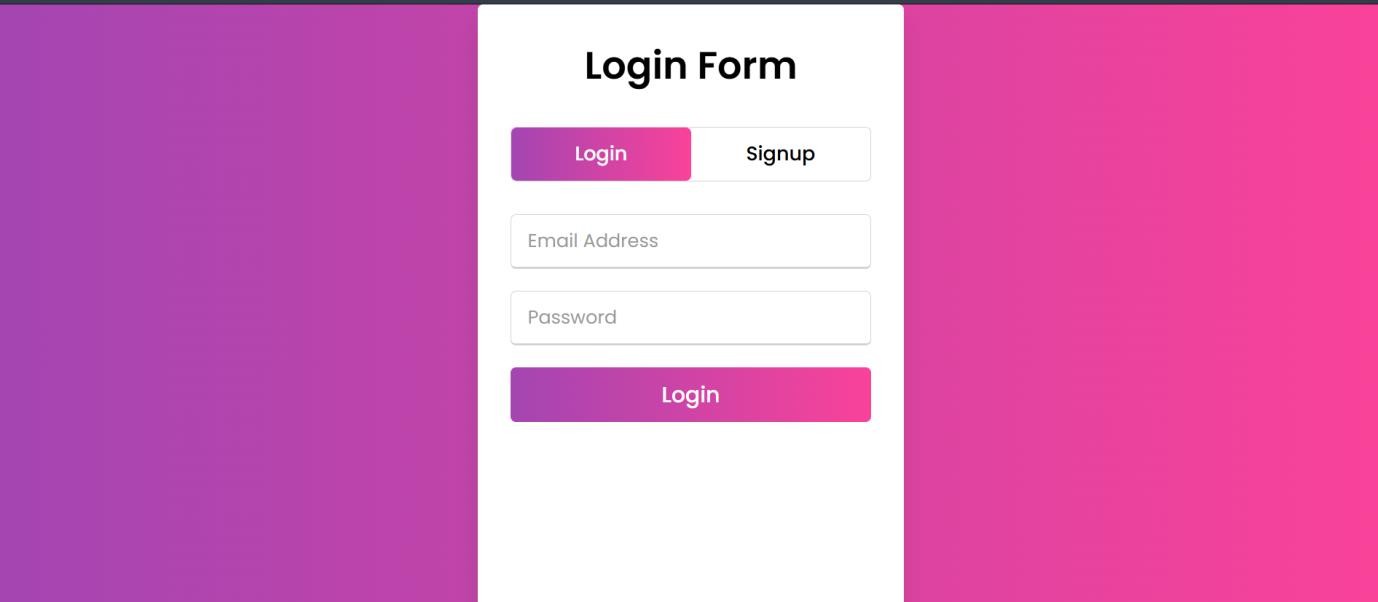
****

Fig A.3.5 Surgeon Page

## PHARMACY SIGN UP AND LOGIN PAGE

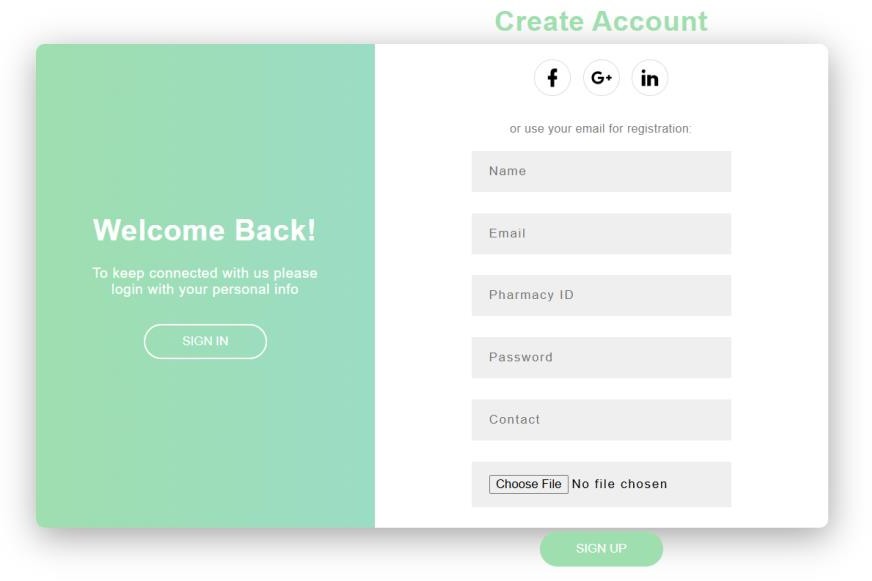
****

Fig A.3.6 Pharmacy Sign-up and Login Page

## CHATTING MODULE

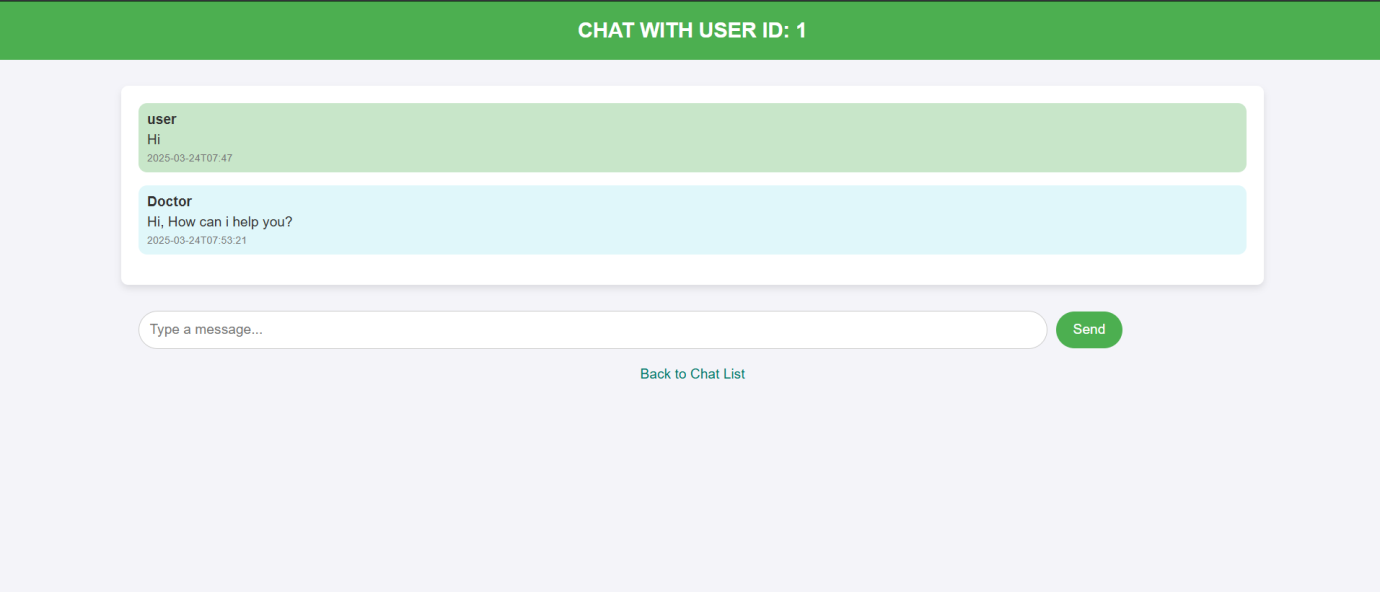
****

Fig A.3.7 Chatting Module

## ADMIN LOGIN

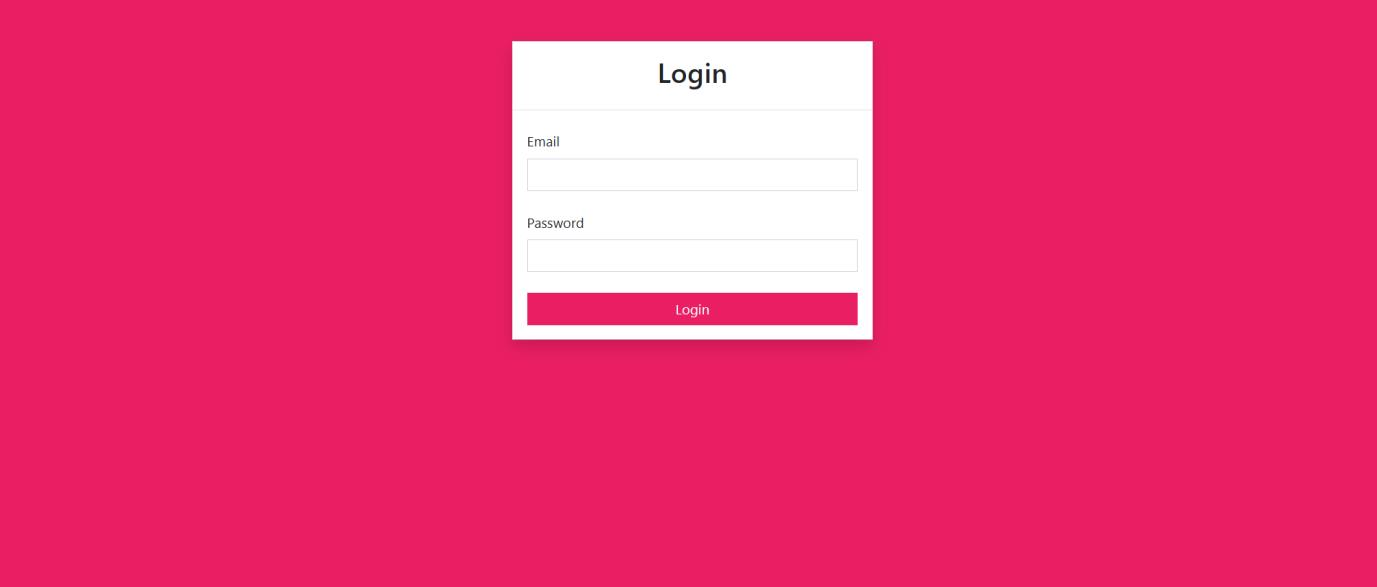
****

Fig A.3.8 Admin Login Page

## RADIOLOGIST REGISTER AND LOGIN PAGE

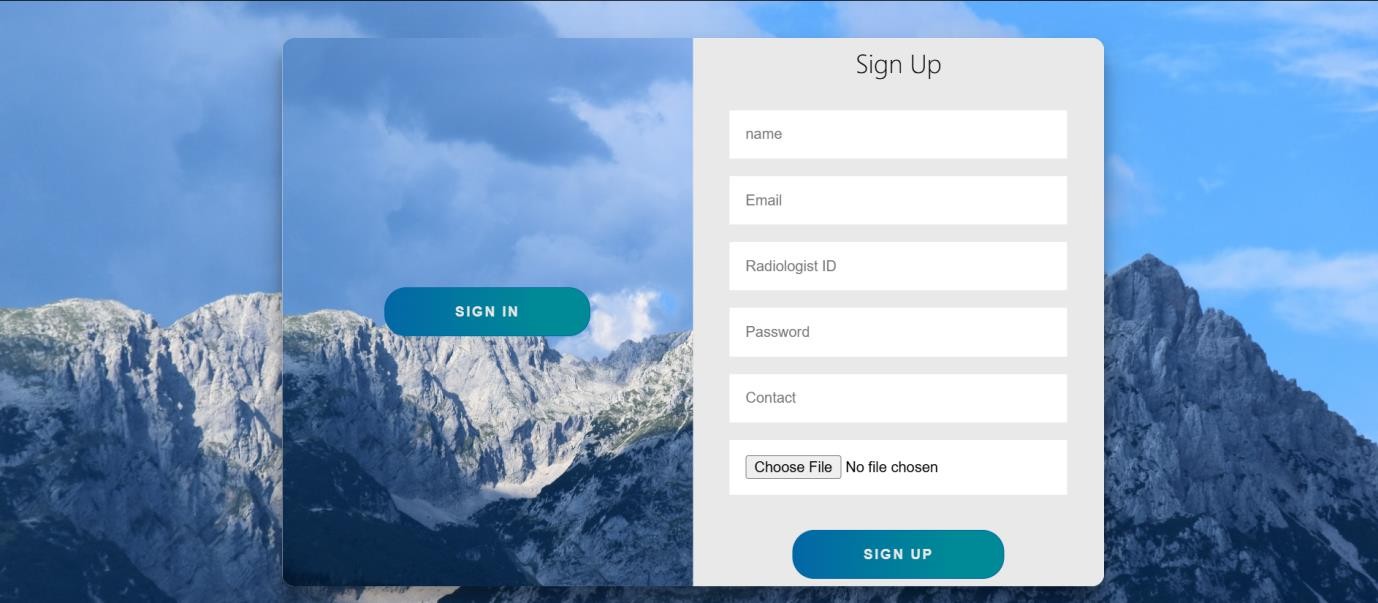
****

Fig A.3.9 Radiologist Register and Login Page

## RADIOLOGIST MODULE

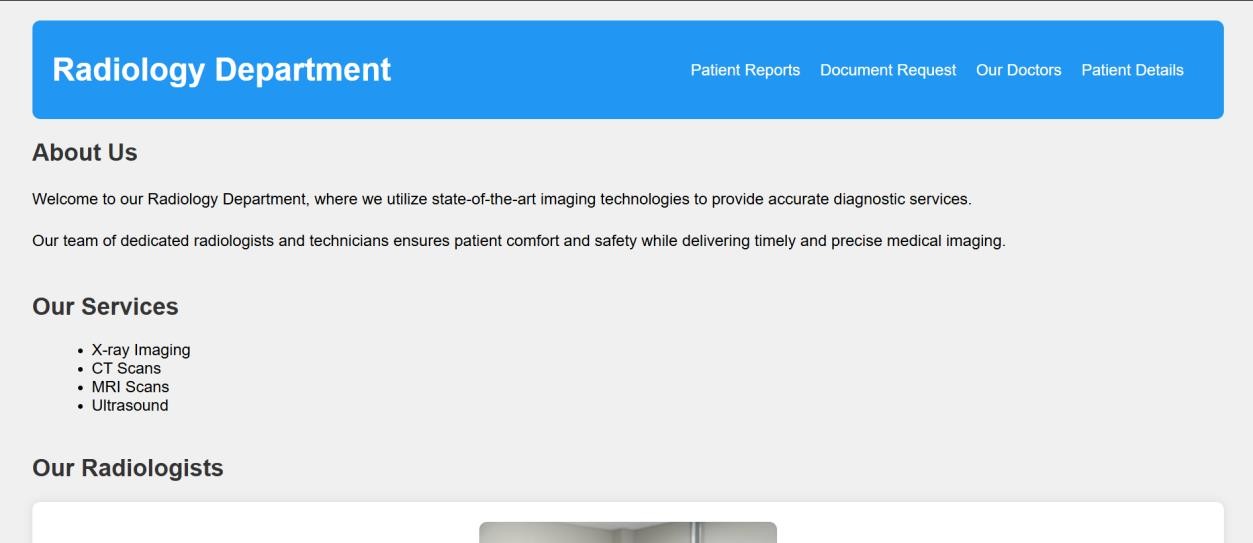
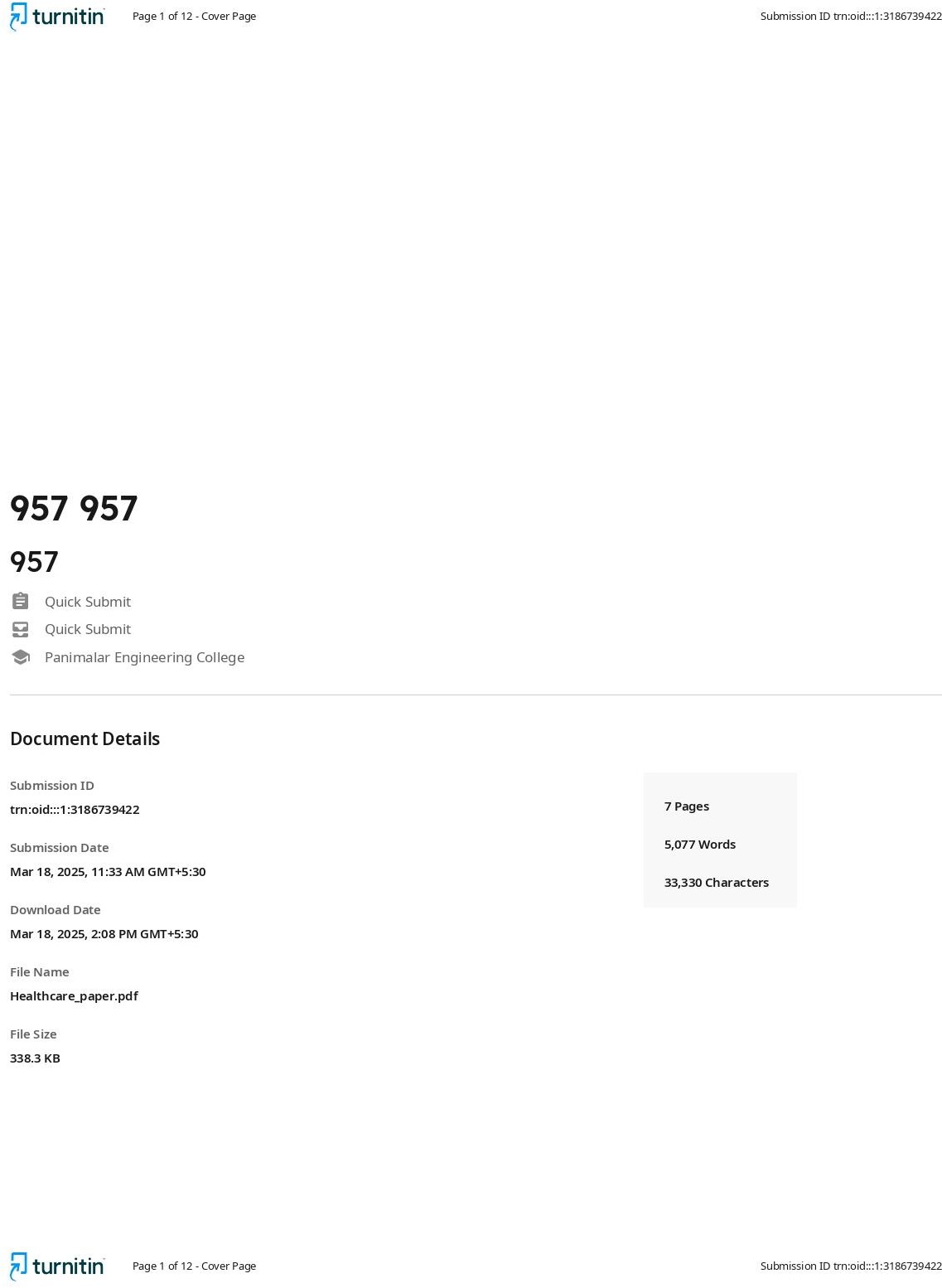
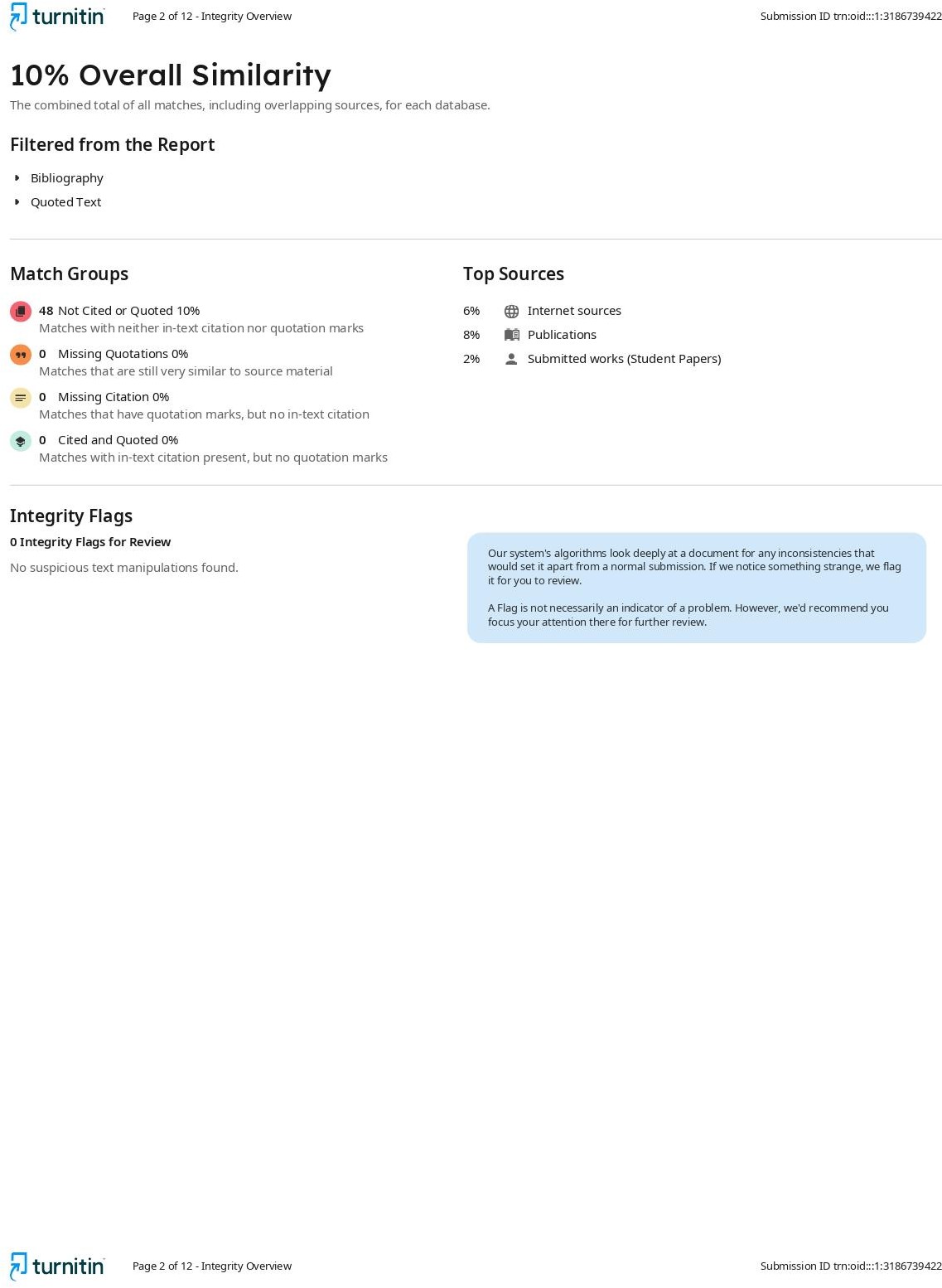
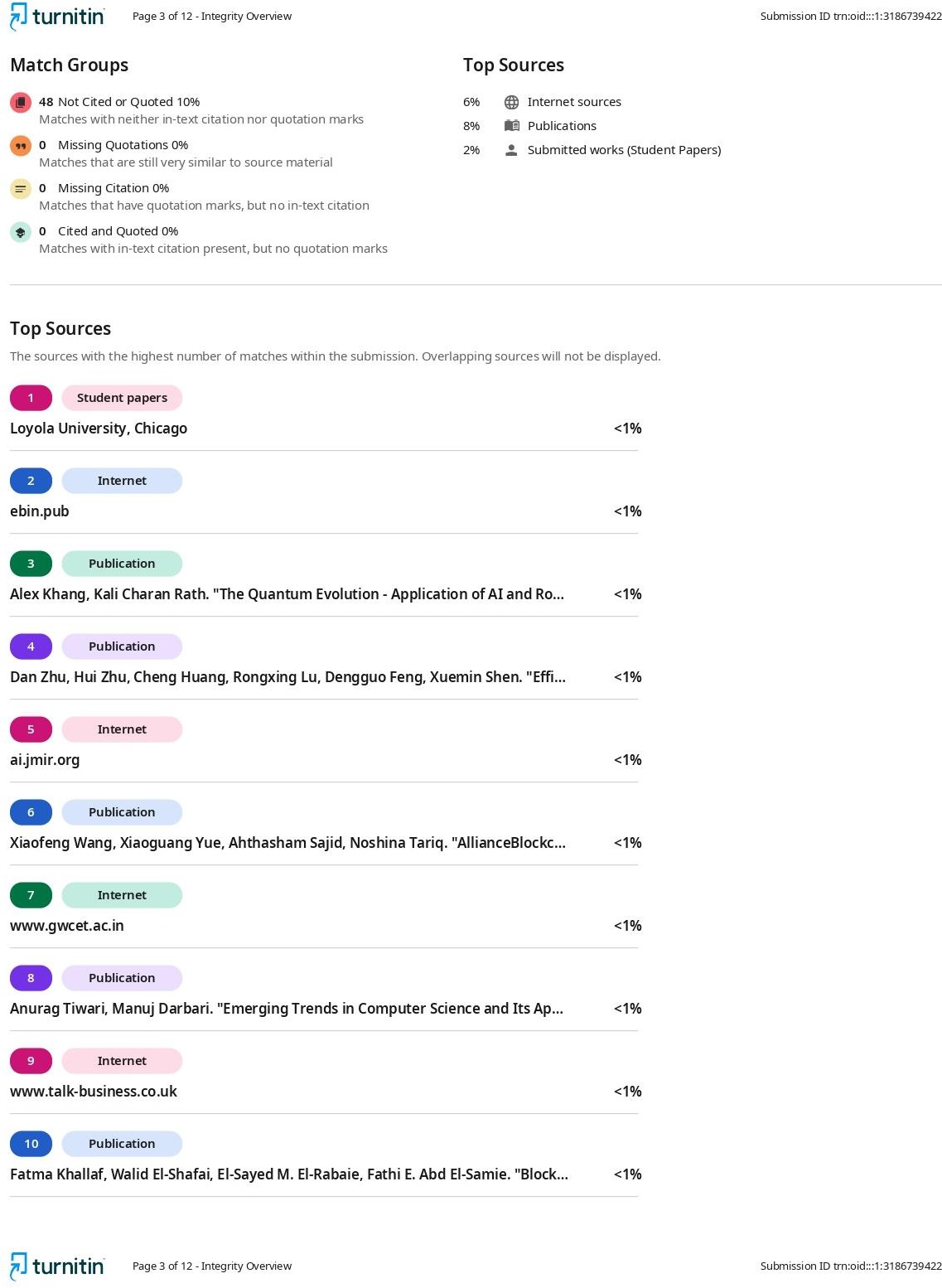
****

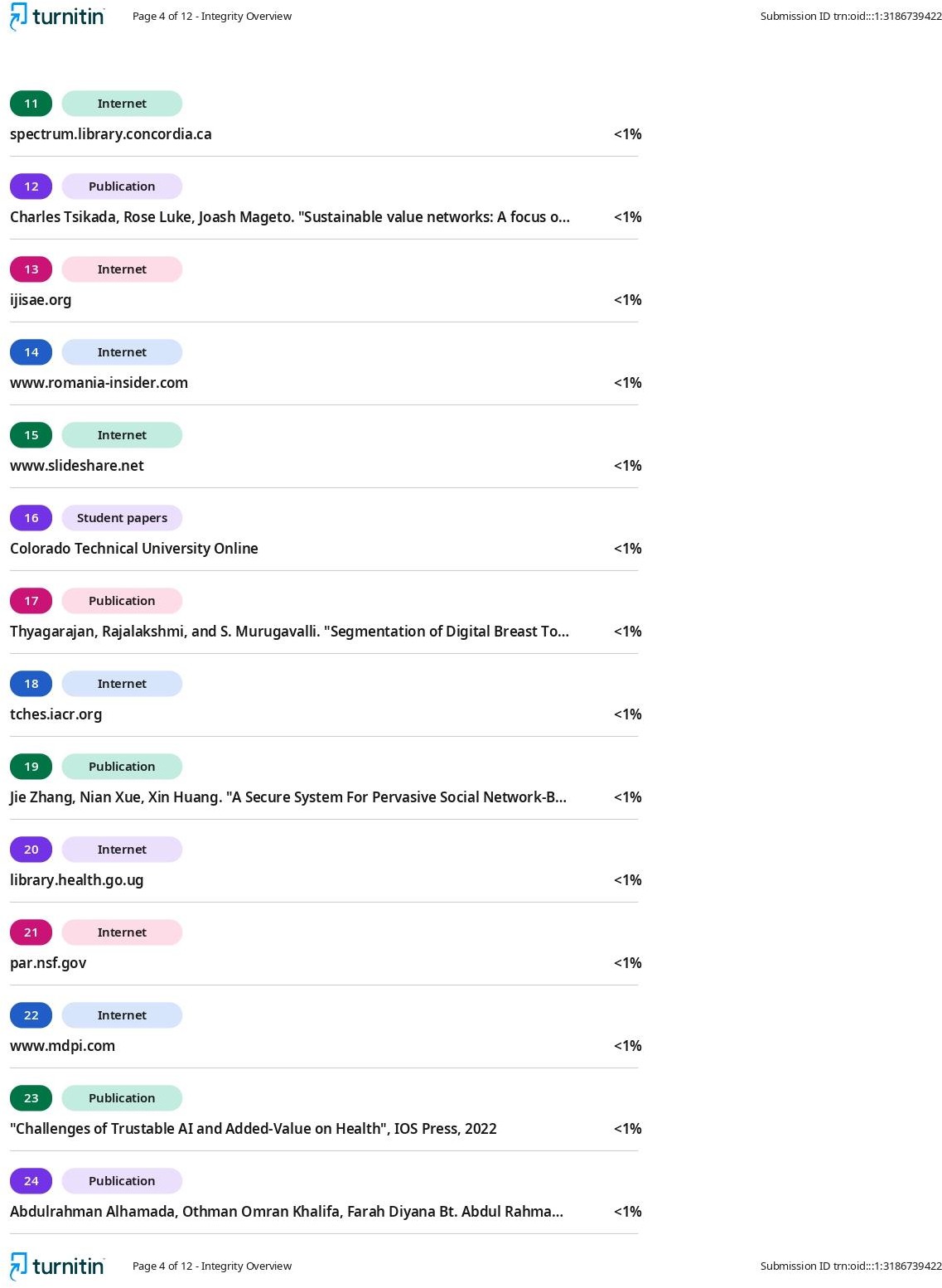
Fig A.3.10 Radiologist Module

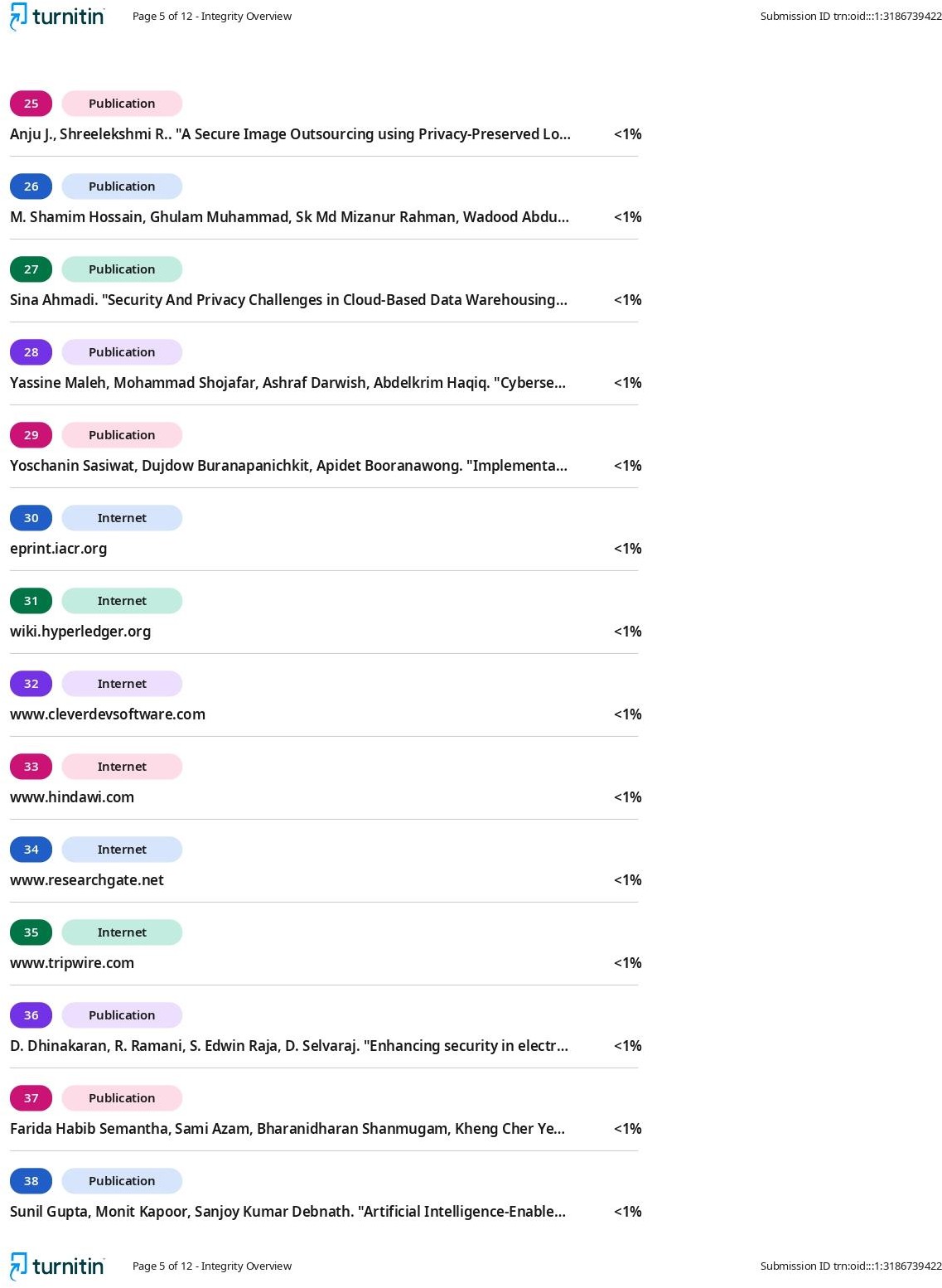
## PLAGIARISM REPORT

****

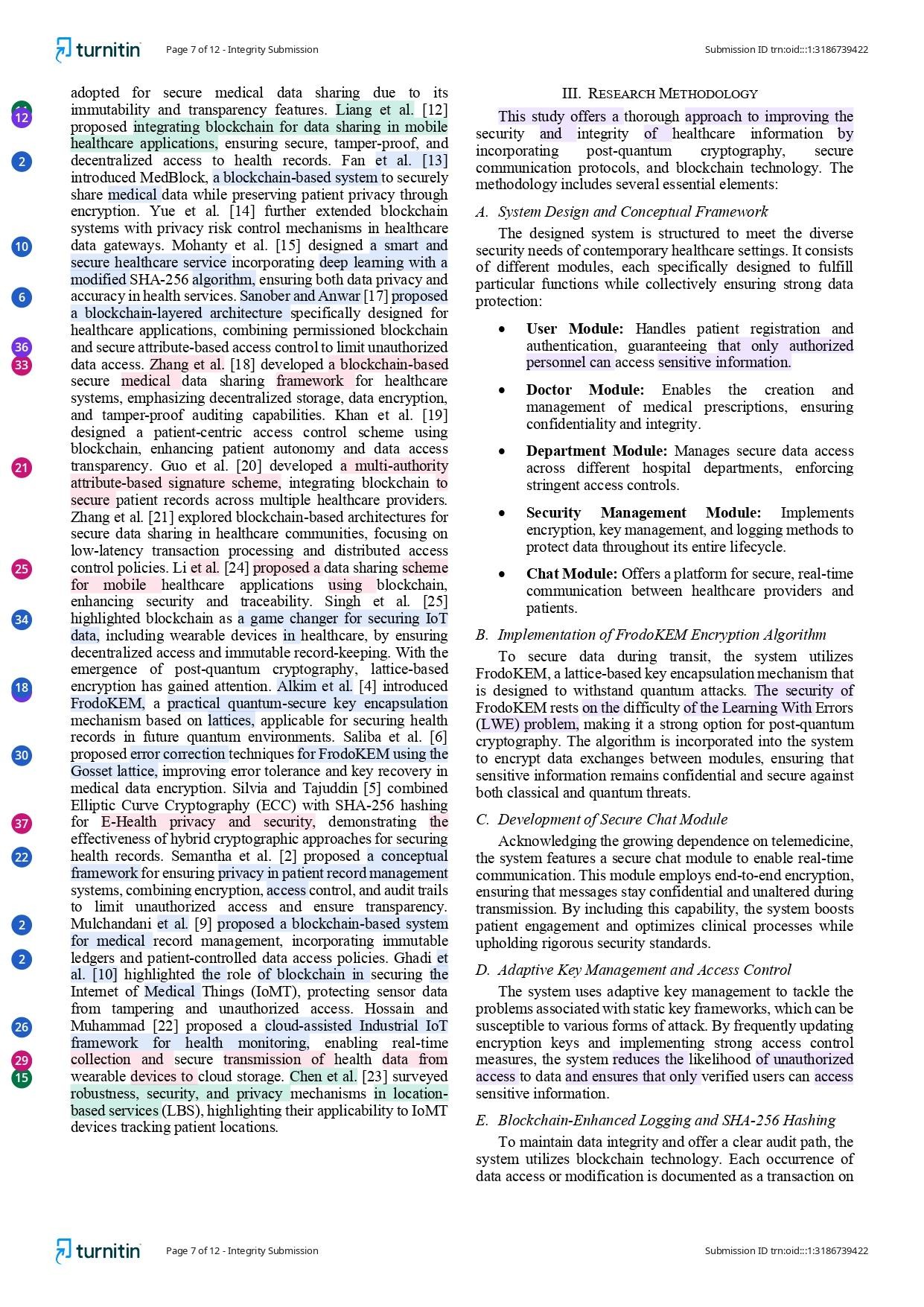


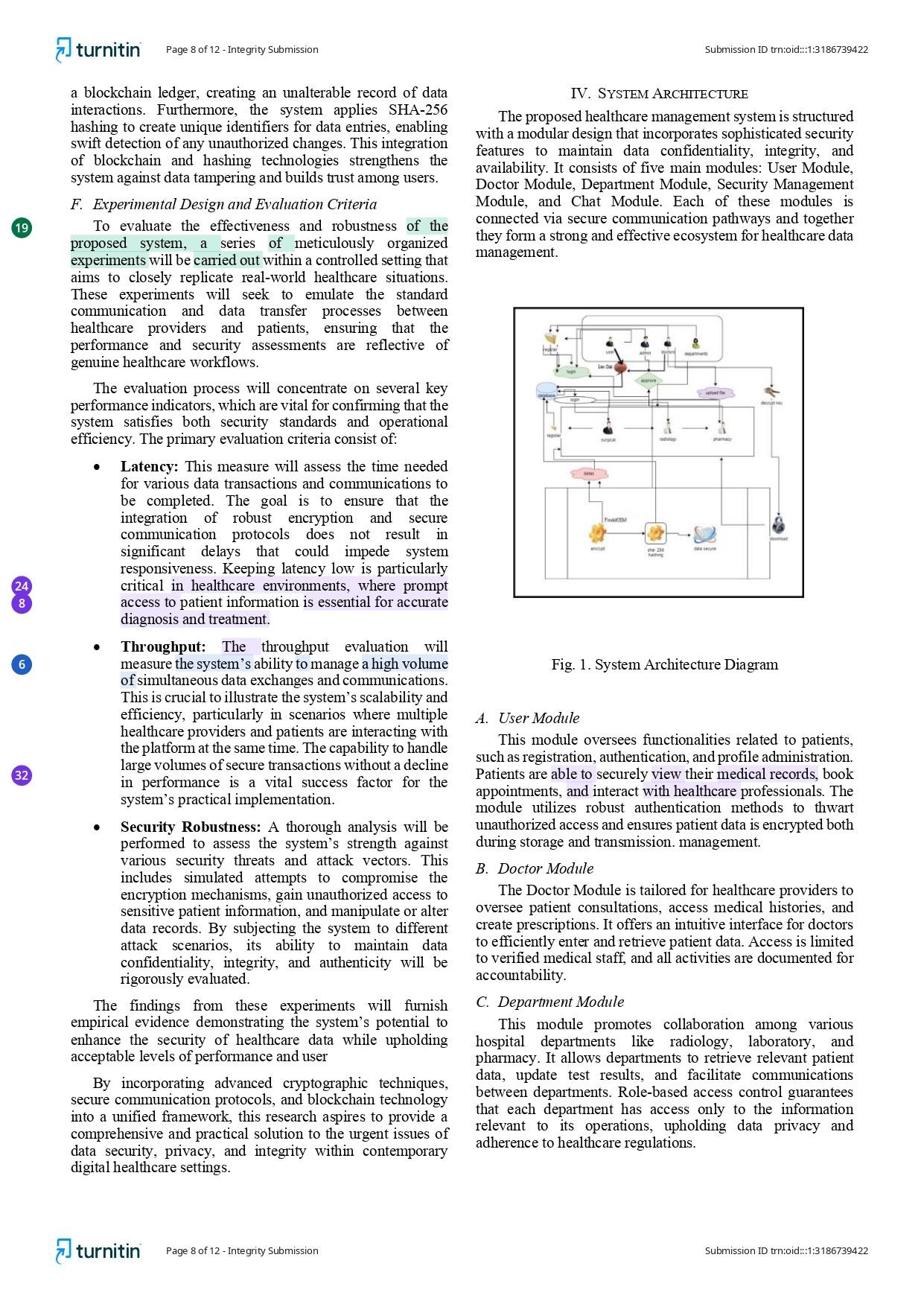






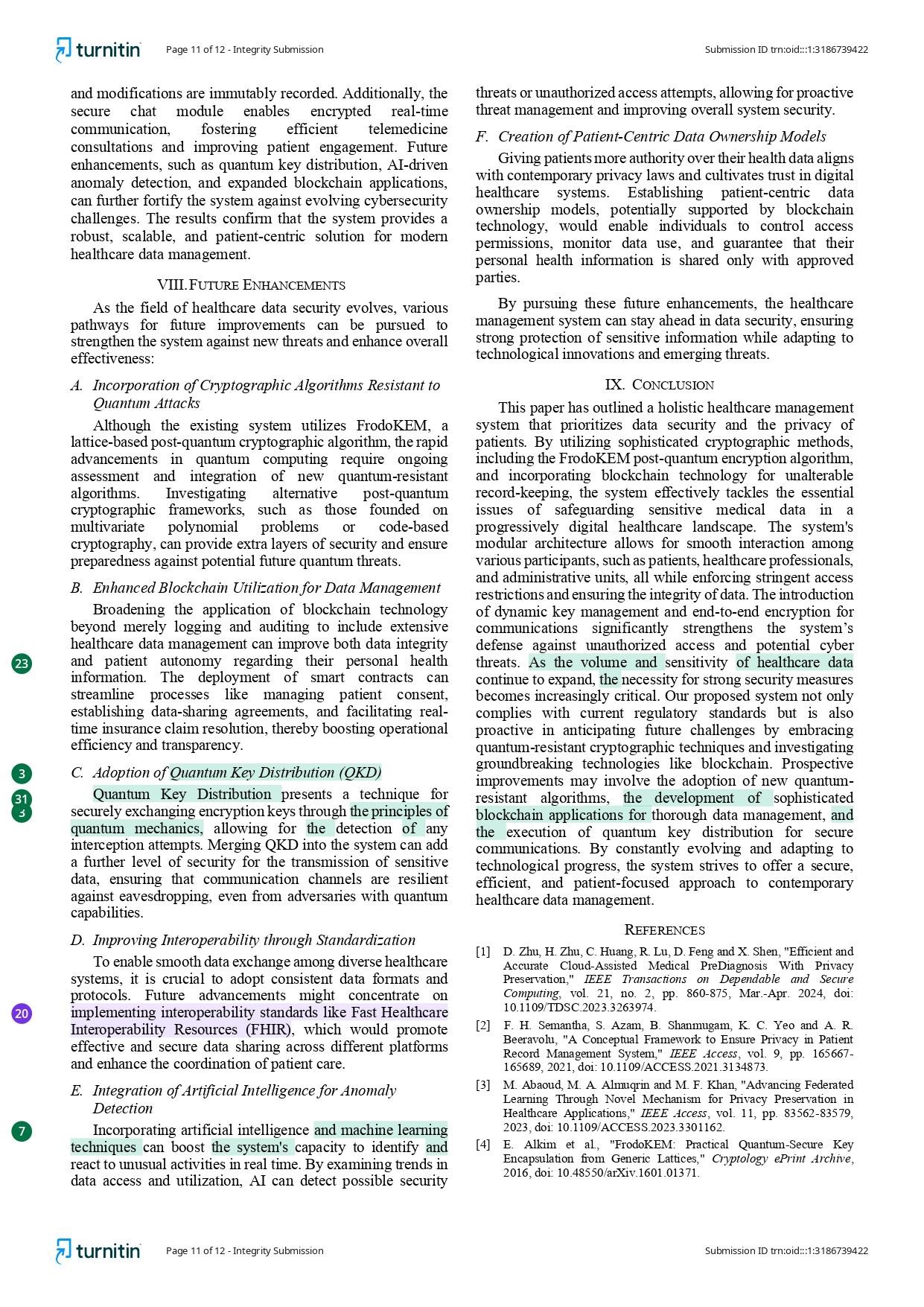
****

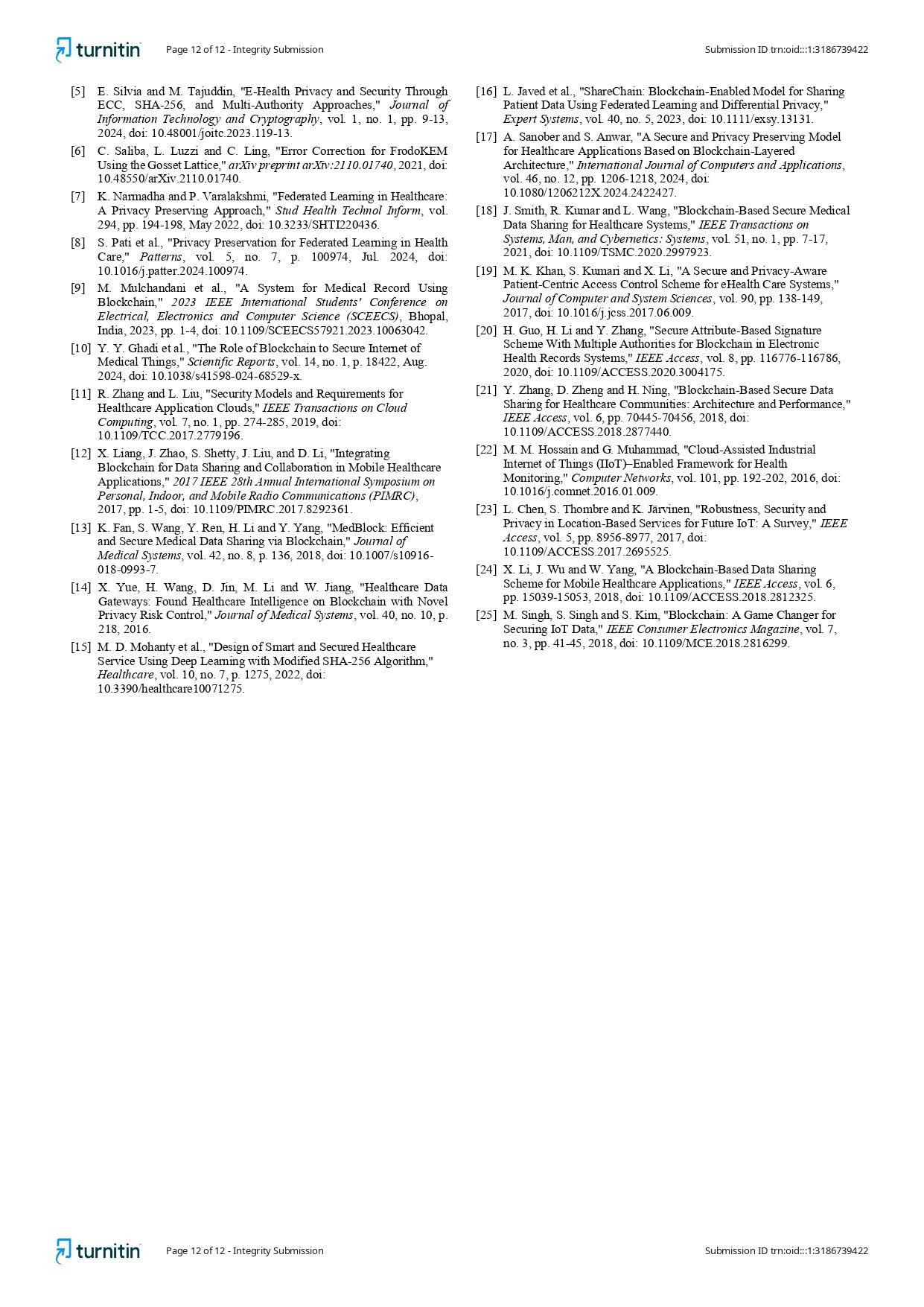
****

****

****

****

****

****

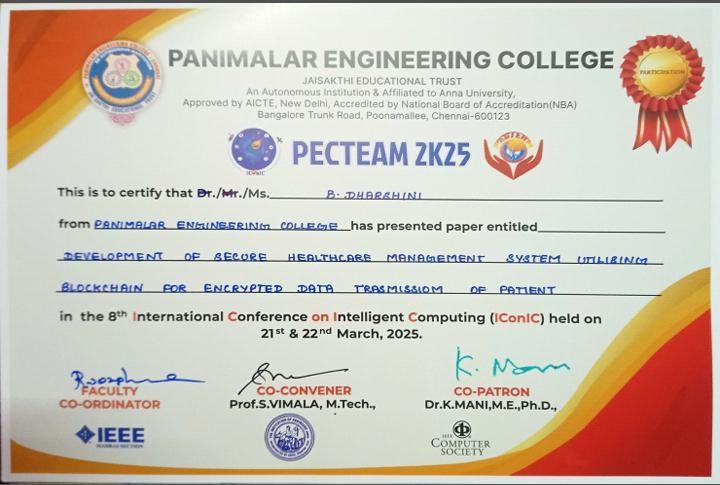
* 1. **PAPER PUBLICATION**

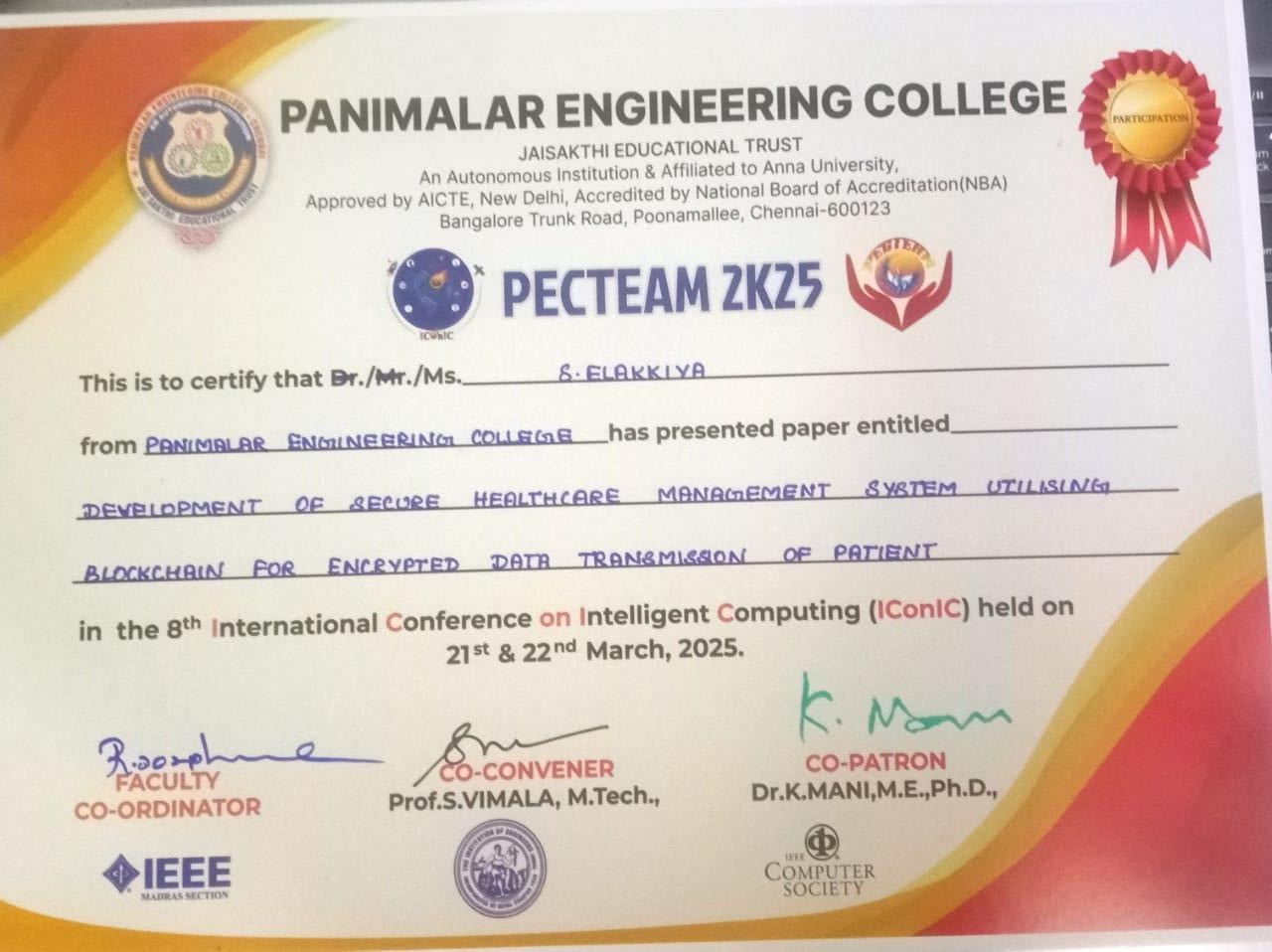
****

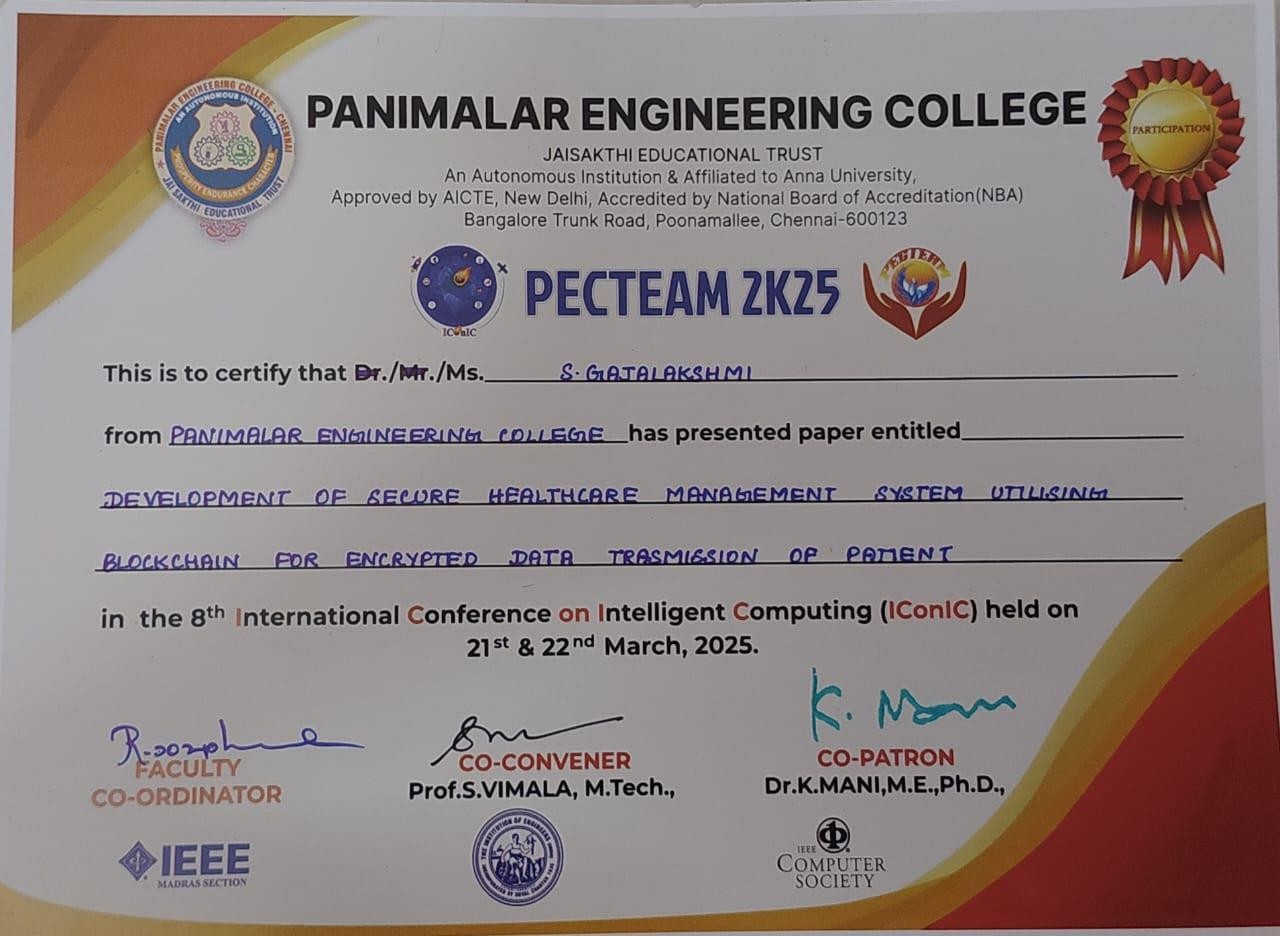
**Conference:** 8th INTERNATIONAL CONFERENCE on INTELLIGENT COMPUTING

**Paper Id:** 957

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







## REFERENCES

1. The Radicati Group Inc., ―Cloud Email and Collaboration-Market Quadrant 2019,‖ https://

[www.radicati.com/wp/wp-content/uploads/2019/03/](http://www.radicati.com/wp/wp-content/uploads/2019/03/) Cloud-Email-and- Collaboration-Market-Quadrant-2019-Brochure. pdf, March 2019, accessed April 8, 2019.

1. Tim Sadler, ―The Year of Email Data Breaches,‖ https://www.infosecurity magazine.com/opinions/

2017-email-data-breaches/, January 2018, accessed September 11,

2019.

1. Wikileaks, ―Hillary Clinton Email Archive,‖ https://wikileaks.org/clinton- emails/, March 2016, accessed April 8, 2019.
2. ——, ―The Podesta Emails,‖ <https://wikileaks.org/> podesta-emails/, March 2016, accessed April 8, 2019.
3. 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).
4. 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).
5. 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.
6. 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
7. 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.
8. A. Shamir, ―Identity-based cryptosystems and signature schemes,‖ in Advances in Cryptology–CRYPTO 1984. Springer, 1984, pp. 47– 53.
9. Proofpoint, ―Proofpoint Email Protection,‖ [https://www](https://www/). proofpoint.com/us/products/email-protection, 2005, accessed April 18, 2019.
10. DataMotion, ―DataMotion SecureMail,‖ https://www.proofpoint. com/us/products/email-protection, 2013, accessed April 18, 2019.
11. 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.

1. 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.
2. 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.