

**KAKINADA INSTITUTE OF ENGINEERING AND TECHNOLOGY-II**  
**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

A PROJECT REPORT ON

**“A METHODOLOGY FOR SECURE SHARING OF PERSONAL  
HEALTH RECORDS IN THE CLOUD”**

SUBMITTED TO

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, KAKINADA**

A Project Report Submitted in partial fulfilment of the requirements For the Award of  
degree of

**BACHELOR OF TECHNOLOGY  
IN  
COMPUTER SCIENCE AND ENGINEERING**

Submitted By

**VADDI SRAVANI**

**( 196Q1A0509 )**

Under the Esteemed Guidance of

**Mr. CH.NARESH, M.Tech**

Assistant Professor

Department of Computer Science and Engineering



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

**KAKINADA INSTITUTE OF ENGINEERING & TECHNOLOGY-II**

(Approved by AICTE, Govt. of AP. & Affiliated to JNT University Kakinada)

Yanam Road, Korangi- 533461 E.G Dist. (A.P)

2019-2023

**KAKINADA INSTITUTE OF ENGINEERING AND TECHNOLOGY-II**  
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

A PROJECT REPORT ON

**“A METHODOLOGY FOR SECURE SHARING OF PERSONAL  
HEALTH RECORDS IN THE CLOUD”**

SUBMITTED TO

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, KAKINADA**

A Project Report Submitted in partial fulfilment of the requirements for the

Award of degree of

**BACHELOR OF TECHNOLOGY  
IN  
COMPUTER SCIENCE AND ENGINEERING**

Submitted By

**VADDI SRAVANI**

**(196Q1A0509)**

Under the Esteemed Guidance of

**Mr. CH.NARESH, M.Tech.**

Assistant Professor

Department of Computer Science and Engineering



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**KAKINADA INSTITUTE OF ENGINEERING AND TECHNOLOGY-II**

(Approved by AICTE, Govt. of AP. & Affiliated to JNT University Kakinada)

Yanam Road, Korangi- 533461 E.G Dist. (A.P).

**2019-2023**

# **KAKINADA INSTITUTE OF ENGINEERING AND TECHNOLOGY-II**

(Approved by AICTE, Govt. of AP. & Affiliated to JNT University Kakinada) Yanam Road,  
Korangi- 533461 E.G Dist. (A. P)

## **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



### **CERTIFICATE**

This is to certify that the project entitled **“A METHODOLOGY FOR SECURE SHARING OF PERSONAL HEALTH RECORDS IN THE CLOUD”** is bonafide work of **VADDI SRAVANI (196Q1A0509)**, in Partial Fulfillment of the requirements for the award of the Degree Of Bachelor of Technology in **Computer Science & Engineering** in the academic year 2022-2023 from **Kakinada Institute of Engineering And Technology-II** affiliated to Jawaharlal Nehru Technological University Kakinada, is a record of Bonafide work carried out by them under my guidance and supervision.

The results embodied in this project have not been submitted to any other University or Institute for the award of any degree or diploma.

**INTERNAL SUPERVISOR**

**HEAD OF THE DEPARTMENT**

**EXTERNAL EXAMINER**

## **DECLARATION**

I hereby declare that the project entitled **A METHODOLOGY FOR SECURE SHARING OF PERSONAL HEALTH RECORDS IN THE CLOUD** is the work done by us during the academic year 2022-2023, and is submitted in partial fulfilment of the requirements for the award of Bachelor of Technology in **COMPUTER SCIENCE AND ENGINEERING** from **KAKINADA INSTITUTE OF ENGINEERING AND TECCHNOLOGY-II** affiliated to **Jawaharlal Nehru Technological University Kakinada,**

I also declare that this project is the outcome of my own effort, that it has not been submitted to any other university for the award of any degree

**BY**

**VADDI SRAVANI**

**(196Q1A0509)**

## ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to our Honorable Chairman **SRI. P. VENKATA VISWAM** for the guidance and advice which is given and for providing sufficient resources.

I wish to avail this opportunity to thank **Dr. CH BHAVANARAYANA, (Ph.D.)**, Principal, Kakinada Institute of Engineering and Technology for his continuous support and giving valuable suggestions during the entire period of the project work.

I am sincere thanks and deep sense of gratitude to **Mr. B. MAHALAKSHMI RAO, M.Tech Ph.D. Associate professor**, Head of the Department CSE, for his valuable guidance, in completion of this project successfully.

I express a great pleasure to acknowledge my profound sense of gratitude to our project guide **Mr. CH.NARESH, M.Tech**, Assistant Professor in CSE Department for his valuable guidance, comments, suggestions, and encouragement throughout the course of this project.

I am also place our floral gratitude to all other teaching staff and lab technicians for their constant support and advice throughout the project.

## PROJECT ASSOCIATES

VADDI SRAVANI

(196Q1A0509)

## **ABSTRACT**

The widespread acceptance of cloud based services in the healthcare sector has resulted in cost effective and convenient exchange of Personal Health Records (PHRs) among several participating entities of the e-Health systems. Nevertheless, storing the confidential health information to cloud servers is susceptible to revelation or theft and calls for the development of methodologies that ensure the privacy of the PHRs. Therefore, we propose a methodology called SeSPHR for secure sharing of the PHRs in the cloud. The SeSPHR scheme ensures patient-centric control on the PHRs and preserves the confidentiality of the PHRs. The patients store the encrypted PHRs on the un-trusted cloud servers and selectively grant access to different types of users on different portions of the PHRs. A semi-trusted proxy called Setup and Re-encryption Server (SRS) is introduced to set up the public/private key pairs and to produce the re-encryption keys. Moreover, the methodology is secure against insider threats and also enforces a forward and backward access control. Furthermore, we formally analyze and verify the working of SeSPHR methodology through the High Level Petri Nets (HLPN). Performance evaluation regarding time consumption indicates that the SeSPHR methodology has potential to be employed for securely sharing the PHRs in the cloud.

## **INDEX**

<b>S.NO</b>	<b>TITLE</b>	<b>PAGE.NO</b>
	<b>CERTIFICATE</b>	
	<b>ACKNOWLEDGEMENT</b>	
	<b>DECLARATION</b>	
	<b>TABLE OF CONTENTS</b>	
	<b>LIST OF FIGURES</b>	
	<b>ABSTRACT</b>	
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Brief information	
	1.2 Literature Survey	
	1.3 Motivation	
	1.4 Objective	
	1.5 Problem statement	
<b>2</b>	<b>SYSTEM ANALYSIS</b>	<b>9</b>
	2.1 Existing system	
	2.2 Proposed system	
	2.3 Feasibility study	
	2.3.1 Economic feasibility	
	2.3.2 Technical feasibility	
	2.3.3 Social Feasibility	
	2.4 System requirement specification	
	2.4.1 Functional requirements	
	2.4.2 Non-functional requirements	
<b>3</b>	<b>SOFTWARE REQUIREMENTS</b>	<b>18</b>
	3.1 Hardware requirements	
	3.2 Software requirements	
<b>4</b>	<b>SYSTEM DESIGN</b>	<b>19</b>
	4.1 System architecture	
	4.2 System Modules	

	4.3Introduction to UML Diagrams	
	4.3.1 Use case diagram	
	4.3.2 Class diagram	
	4.3.3 Sequence diagram	
	4.3.4 Activity diagram	
	4.3.5 Component diagram	
	4.3.6 Deployment diagram	
	4.3.7 Database design	
	4.3.8 Normalization	
5	<b>SYSTEM IMPLEMENTATION</b>	<b>32</b>
	5.1 Frontend implementation	
	5.2 Backend implementation	
6	<b>SYSTEM TESTING</b>	<b>41</b>
	6.1 Testing concepts	
	6.2 Testing strategies	
	6.3 Test cases	
7	<b>SOURCE CODE</b>	<b>46</b>
	7.1 ABE.JAVA	
	7.2 ContentExtractor.JAVA	
	7.3Dbconn.JAVA	
	7.4 Ftpcon.JAVA	
	7.5 Upload.JAVA	
8	<b>OUTPUT SCREEN</b>	<b>53</b>
9	<b>CONCLUSION</b>	<b>59</b>
10	<b>REFERENCE</b>	<b>60</b>



## LIST OF FIGURES

<b>S No</b>	<b>Figure No</b>	<b>Name Of the Figure</b>	<b>Page No</b>
1	Fig 4.1	System Architecture	21
2	Fig 4.2	Use Case Diagram	24
3	Fig 4.3	Class Diagram	25
4	Fig 4.4	Sequence Diagram	26
5	Fig 4.5	Activity diagram	27
6	Fig 4.6	Component diagram	28
7	Fig 4.7	Deployment diagram	29
8	Fig 8.1	Home page	53
9	Fig 8.2	Login page	54
10	Fig 8.3	Owner File Upload	54
11	Fig 8.4	User List	55
12	Fig 8.5	Data Owners	56
13	Fig 8.6	User Send Request	57
14	Fig 8.7	Key Accept	57
15	Fig 8.8	Encrypted Key	58

# **CHAPTER -1**

## **INTRODUCTION**

### **1. INTRODUCTION**

Distributed computing is an ongoing innovation that targets giving access to assets in a flash according to the requirements of the end clients. Cloud empowers its clients to utilize the assets that are broadly circulated in the web to perform calculations without introducing in their very own PC's and needs to pay just for the administration they devoured. All the computational necessities will be dealt with by the cloud specialist organizations and subsequently every one of the complexities included will be escaped the client. NIST recognizes the five key qualities of distributed computing as on-request self administration, asset pooling, and wide system get to, fast versatility and estimated administration. Cloud offers benefits in three fundamental structures specifically foundation (IaaS), stages (PaaS) and Software (SaaS) and is on the phase of development to give everything as an administration (XaaS). As enormous extents of information are moving onto the cloud, the assailants are quicker to misuse the vulnerabilities related with cloud and in this way to take the touchy information. Among the different dangers to distributed computing, Personal Health Records (PHR) assaults can demonstrate to be the deadliest assault and even the Cloud Security Alliance has distinguished PHR assault as one of the nine noteworthy dangers.

Cloud computing primarily provides three service models:

‡ **Software as a service (SaaS):**

- Software as a service is a business strategy that allows you to quickly access cloud-based web applications.
- SaaS is the highest level of cloud computing in which all services are provided by cloud providers
- The Cloud provider is in charge of overseeing the computing stack, which can be activated via a web browser.
- These cloud-based applications require a paid subscription to use. Some are available for free but some have limited access.
- Some examples of software as service are G-suite, Drop box

### ‡ **Platform as a service (PaaS):**

- Platform as a service is essentially a cloud model base that enables users to create, test, and arrange diverse business applications.
- By use of this model clarifies the process of creating enterprise software.
- PaaS offers an interactive runtime environment in which you can create and test different applications.
- All resources are presented in the form of cloud data storage, servers and network infrastructure that can be easily managed by the company or platform provider.
- Examples of Platform as service are AWS Elastic Beanstalk, Google App Engine.

### ‡ **Infrastructure as a service (IaaS):**

- Infrastructure as a service is essentially a virtual configuration of computing resources that can be accessed via the cloud.
- An IaaS cloud provider can provide your company with a complete computing infrastructure, including a storage device and network interface
- This infrastructure is also maintained and supported by the vendor.
- Businesses can use computing resources to meet their needs without installing equipment on their premises
- Some Examples of Infrastructure as a service

## **1.1 BRIEF INFORMATION:**

Cloud computing in healthcare makes medical record-sharing easier and safer, automates backend operations and even facilitates the creation and maintenance of telehealth apps. Utilizing the cloud increases the efficiency of the healthcare industry, while also decreasing costs. The use of cloud technology in healthcare is growing so fast that total global spending is estimated to reach a global market value at over \$89 billion by 2027. In addition, IaaS, a cloud computing model popular for migrating healthcare infrastructures to the cloud, is currently the fastest growing cloud service with a projected CAGR of 32 percent by 2027.

Benefits of cloud computing in health care:

- Accelerates clinical analyses and care processes.
- Automates data processing and scalability.
- Increases patient data accessibility.
- Reduces network equipment and staff costs.

Health Cloud provides objects you can use to review and evaluate medical care services, communicate about clinical policies, and help health plan members ensure they receive the right care in the right setting at the right time.

- Types of Registers Used in Health Records

1. Labor and Delivery Register.
2. Family Planning Register.
3. Daily Immunization Register.
4. Daily General Attendance Register.
5. Daily Ante Natal Clinic Attendance Register.
6. out-Patient Register.
7. In-Patient or Admission/Discharge Register.

Cloud computing in healthcare is all about applying remote servers used through the internet. It helps to store, handle, and process medical information. Cloud storage is convenient for healthcare experts and medical institutions to use online servers to store a massive amount of data securely. Generally, IT experts maintain such servers.

With the advent of the EMR (Electronic Medical Records) Mandate, medical companies have embraced cloud-based solutions to store and secure their patient records. Furthermore, healthcare organizations that did not plan to shift current data centers to the cloud also adopted cloud-based solutions.

## **1.2 Literature Survey**

Sharing personal health records (PHRs) in cloud computing is a complex and challenging task due to the sensitive nature of the data involved. In recent years, several researchers have proposed different solutions to ensure the security and privacy of PHRs in cloud computing. Secure and Privacy-Preserving Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption" by Y. Wang et al. (2016) proposes a novel approach to secure and privacy-preserving sharing of PHRs in cloud computing using attribute-based encryption

(ABE). The proposed approach allows patients to share their PHRs with authorized healthcare providers while keeping the data confidential and private.

Security and privacy concerns: One of the primary concerns when it comes to sharing PHRs in the cloud is the security and privacy of the data. Many studies have focused on different methods for securing PHRs, including encryption, access control, and authentication. Additionally, there is a need for clear policies and regulations around the use of cloud computing for PHRs.[1] Access control and authentication: Access control and authentication are crucial components of any secure PHR system. Many studies have explored different access control and authentication methods, including biometric authentication, two-factor authentication, and attribute-based access control.[2] Patient-centered approaches: Patient-centered approaches to sharing PHRs in the cloud have also been explored in the literature. These approaches aim to give patients more control over their own health data and allow them to share their data with healthcare providers as they see fit.[3] while there are many challenges to securely sharing PHRs in the cloud, there are also many potential benefits.[4] By implementing strong security measures, using a exploring patient-centered approaches, healthcare providers can take advantage of the efficiency and accessibility of cloud computing while also protecting patient privacy and security.

## **1.3 MOTIVATION**

The main motivation behind cloud computing is to enable businesses to get access to data centres and manage tasks from a remote location.

Cloud computing works on the pay-as-you-go pricing model, which helps businesses lower their operating cost and run infrastructure more efficiently.

Cloud computing helps start-ups manage shifting computing requirements by providing greater flexibility in the computing services they purchase. A cloud-based IT infrastructure is more versatile – notably in terms of scalability – than is local, intranet-based infrastructure.

### **Scalability**

Cloud computing helps startups manage shifting computing requirements by providing greater flexibility in the computing services they purchase. A cloud-based IT infrastructure is more versatile – notably in terms of scalability – than is local, intranet-based infrastructure.

### **Reliability**

Because cloud vendors can build more redundancy into a system than a company can build into its own intranet, the cloud vendor can spread its infrastructure investment costs across its entire customer base, allocating resources as necessary.

### **Virtualization**

Because cloud-based IT infrastructure can be virtualized and geographically dislocated, startups are freed from having to consider the physical location of its IT infrastructure and datacenters in business operations decisions.

### **Affordability**

Under traditional infrastructures, startups may not receive –or have Financial wherewithal to purchase – certain features that are often offered to cloud computing customers at substantial discounts. How do these benefits pass on to startups and other small companies? Because the marginal cost to the cloud computing provider of many features (such as enhanced security) may be very low (or even negligible), otherwise unaffordable services may be offered for free to startups using cloud computing options.

## 1.4 OBJECTIVE

We are focusing on how we can implement cloud computing in healthcare services.

- With the help of cloud computing lots of things can be implementing in traditional healthcare system
- The purpose of this study was to present a systematic literature review of cloud computing technologies in healthcare and to analyse previous research
- Providing motivation, limitations encountered by researchers, and recommendations made to analysts for improving this critical research field method.

With the help of cloud computing, doctors and hospitals now have the power to increase patient engagement and give them anywhere anytime access to their medical data, test results, and even doctor's notes. This gives patients more power and control, as well as makes them more educated about their medical conditions. The main motivation behind cloud computing is to enable businesses to get access to data centres and manage tasks from a remote location. Cloud computing works on the pay-as-you-go pricing model, which helps businesses lower their operating cost and run infrastructure more efficiently.

- Its purpose is to maintain long-term personal records and health improvement plans. It combines a cloud computing environment to build a personal health records system to quickly collect personal information and transfer it to the back end for storage for future access.
- To provide technical support to the Department of Health and Family Welfare for achieving Universal Health Care accessible to all citizens and to prioritize special groups..
- To facilitate transparency and maintenance of standards in Counseling for medical education
- To facilitate in prevention, mitigation and elimination/eradication of communicable diseases of public health importance including emerging and re-emerging diseases.
- To facilitate in prevention, mitigation, and containment of public health emergencies due to biological (including zoonotic), chemical, radiological and nuclear hazards in disaster situations.
- To promote healthy living and to facilitate prevention, early detection and management of non-communicable diseases.



- To ensure provision of state-of-the-art Emergency Care Services, including medical, surgical (especially Trauma and Burn Care), pediatric and obstetric emergency care for all.
- To provide technical support to address climate change issues impacting health.
- To lay down specific standards and norms for safety and quality assurance of all aspects of health care including Patient Safety, Hospital Acquired Infection and Antimicrobial Resistance device.

## **1.5 PROBLEM STATEMENT**

- Use of cloud computing to support the intensities of geospatial sciences.
- Reason for need of platform like cloud computing.
- What is cloud computing?
- Spatial cloud computing (SCC).
- SSC scenarios/examples. • Opportunities and challenges.

Cloud computing has been one of the most advancing technologies recently .utilizing it in the context of Geo-spatial sciences can prove to be very useful.

# CHAPTER -2

## SYSTEM ANALYSIS

### 2. SYSTEM ANALYSIS

#### 2.1 Existing System

Accountable Proxy Re-encryption (APRE) is a cryptographic technique that allows a data owner to share encrypted data securely with a third party recipient, while also providing a way for the data owner to hold the recipient accountable for any improper use of the data. APRE accomplishes this by using a proxy re-encryption scheme, while allows a trusted proxy to convert encrypted data from one format to another, without revealing the underlying plaintext. There are several existing systems that implement APRE for secure data sharing, including:

- 1. ARX:** ARX is an implementation of APRE that was developed by researchers at the University of Waterloo. It uses a hybrid approach that combines both symmetric and asymmetric encryption to provide strong security guarantees while minimizing computational overhead.
- 2. PASC:** PASC (Privacy- and Accountability-preserving Secure data Collaboration) is another APRE system that was developed by researchers at the University of Illinois at Urbana-Champaign. It is designed to support secure and collaborative data sharing in cloud environment, and includes mechanism for enforcing access control policies and monitoring data usage.
- 3. CERA:** CERA (Cryptographically Enforced Right Amplification) is an APRE system that was developed by researchers at the University of Maryland. It is designed to support secure data sharing in the context of healthcare applications, and includes mechanisms for enforcing fine-grained access control policies and auditing data usage.

Overall, APRE provides a powerful tool for secure data sharing, and these existing systems demonstrate the potential of this technique in a variety of different applications. However, it is important to note that APRE is not a panacea, and its effectiveness depends heavily on the design and implementation of the underlying systems. Careful consideration must be given to

issues such as key management, access control, and auditing in order to ensure that APRE is used effectively and appropriately.

**Disadvantages of Existing System:**

There are several disadvantages of existing APRE systems:

**1. Key management:** APRE systems require a complex key management infrastructure to securely store and distribute the private keys of the sender and the recipient. The key management infrastructure must be designed to be robust and secure, and should be able to prevent unauthorized access to the keys.

**2. Single point of failure:** APRE systems rely on a single proxy entity to perform the re-encryption operation. If this entity is compromised, it could potentially access and modify all the data that is being shared through the systems, compromising the privacy and security of the users.

**3. Limited scalability:** APRE systems may have limited scalability due to the computational overhead associated with the re-encryption operation. As the number of users and the amount of the data being shared increases, the system may become slow and inefficient.

**4. Trust assumptions:** APRE systems rely on several trusts assumptions, such as the trustworthiness of the proxy entity and the integrity of the key management infrastructure. If any of these assumptions are violated, the security and privacy of the system may be compromised.

**5. Lack of interoperability:** Different APRE systems may use different encryption schemes and key management infrastructures, which could create interoperability issues when sharing data between different systems. This could limit the usefulness of the system in real-world scenarios where data sharing is often cross-organizational.

## **2.2 Proposed System**

The basic idea behind APRE is to use a public key infrastructure to generate public and private keys for each user. The data is encrypted using the user's public key, which can only be decrypted using the corresponding private key held by the user. The proxy re-encrypter is given the public keys of all the users and can use them to transform the encrypted data from one user to another. This transformation is done in such a way that the proxy never has access to the private keys of any of the users.

To ensure accountability in the APRE systems, a cryptographic technique called “verifiable re-encryption” is used. Verifiable re-encryption allows users to verify that their data has been re-encrypted correctly and has not been tampered with by the proxy re-encrypter. This is done by including a digital signature with the encrypted data, which can be verified by the recipient of the re-encrypted data. The APRE system has several advantages over traditional encryption methods. It allows for secure data sharing between multiple parties without compromising the privacy of the data. It also provides accountability by allowing users to verify that their data has been re-encrypted correctly. Additionally, it does not require users to share their private keys with anyone else, which reduces the risk of the keys being compromised.

### **Advantages of proposed system**

**1. Enhanced security:** APRE offers a higher level of security than traditional data sharing methods, such as using a third-party cloud services provider. With APRE, the data remains encrypted and is only accessible to authorized users. The proxy re-encrypts the data using a different key each time it is shared, which reduces the risk of data breaches and unauthorized access.

**2. Accountability:** APRE provides accountability for the actions of the proxy. The proxy is responsible for managing the re-encryption keys and is accountable for any unauthorized access or data breaches. This ensures that the data owner has complete control over their data and can hold the proxy accountable for any mishandling of the data.

**3. Flexibility:** APRE offers flexibility in data sharing. The data owner can share their encrypted data with a third party without revealing the decryption key, which allows them to maintain control over their data. The proxy can re-encrypt the data multiple times, allowing for multiple levels of access and sharing.

**4. Scalability:** APRE is scalable and can be used in a public cloud environment, making it ideal for large-scale data sharing. It can also be used in a variety of applications, including healthcare, finance, and government, where secure data sharing is essential’

**5. Cost-effective:** APRE is cost-effective compared to other traditional data sharing methods. It eliminates the need for expensive hardware and infrastructure, reducing the cost of data sharing while maintaining security and privacy.

## **2.3 Feasibility Study**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICALFEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

### **2.3.1 ECONOMICAL FEASIBILITY:-**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### **2.3.2 TECHNICAL FEASIBILITY:-**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

### **2.3.3 SOCIAL FEASIBILITY:-**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His

level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

## **2.4 System Requirement Specification**

To ensure secure sharing of personal health records in cloud computing, the following system requirements specifications should be considered:

**Authentication and Access Control:** The system should implement strong authentication mechanisms to ensure that only authorized users have access to the personal health records. Access control policies should be implemented to ensure that only authorized individuals can view, modify, or delete records.

**Data Encryption:** The system should use strong encryption algorithms to protect the personal health records during transmission and storage. This ensures that even if the data is intercepted or stolen, it cannot be accessed without the appropriate decryption key.

**Data Integrity:** The system should ensure that the personal health records are not tampered with or altered during transmission or storage. This can be achieved by implementing cryptographic mechanisms that provide data integrity protection.

**Data Availability and Reliability:** The system should ensure that the personal records are available and accessible to authorized users at all times. The system should also ensure that data is not lost or corrupted due to hardware failures or other technical issues.

**Compliance with Privacy Regulations:** The system should comply with all relevant privacy regulations and standards, such as the Health Insurance Portability and Accountability Act (HIPAA), the General Data Protection Regulation (GDPR), and others.

**Audit Trail:** The system should maintain an audit trail of all activities related to personal health records. This allows for accountability and traceability, as well as the ability to detect and investigate any suspicious activities.

**Disaster Recovery:** The system should have a disaster recovery plan in place to ensure that personal health records can be restored in the event of a catastrophic event, such as a natural disaster or cyber-attack.

**User Interface:** The system should have an intuitive and user-friendly interface to enable easy access and use of personal health records by authorized users. The interface should be designed with the principles of usability and accessibility in mind.

**Scalability:** The system should be scalable to accommodate an electronic health record (EHR) systems to ensure seamless integration and increasing number of personal health records and users. It should be able to handle large amounts of data and high traffic volumes without compromising on performance or security.

**Compatibility with Existing systems:** The system should be compatible with existing healthcare information systems and interoperability. This will ensure that the personal health records can be shared securely and efficiently between different systems and organizations.

## 2.4.1 FUNCTIONAL REQUIREMENTS

Functional requirements describe what the system should do, i.e. the services provided for the users and for other systems. The functional requirements can be further categorized as follows:

**Input Requirements:** It describe what inputs the system should accept and under what conditions. The following are the input requirements for the present system.

- Uploading documents into cloud server by Data Owner.
- Encryption is done to uploaded documents.
- Searching documents by Filename, Keyword and ranked.

**Output Requirements:** It describes what outputs the system should produce and under what conditions. Outputs can be to the screen or printed. The following are the output requirements for the present system.

- Decryption is done by generating dynamic multi-keywords for downloading the encrypted documents.
- Parallel search process is possible to reduce the time cost.



**Storage Requirements:** It describes the system should store that other systems might use. The following are the storage requirements for present system.

- The system stores the encrypted documents in the cloud server.

**Computation Requirements:** It describes what computations the system should perform. The computations should be described at a level all the readers can understand. The following are the computation requirements for present system.

- Encryption and decryption is done by using Commutative RSA algorithm.
- Parallel search process is done by Parallel search process algorithm.

### 2.4.2 Non-Functional Requirements

Non-functional requirements as name suggests are those requirements which are not directly concerned with the specific functions delivered by the system. They may relate to emergent system properties such as reliability, response time, performance and store occupancy.

Many non-functional requirements relate to the system as a whole rather than to individual system features. This means that they are often more critical than individual functional requirements.

While failure to meet an individual functional requirement may degrade the system, failure to meet a non-functional system requirement may make the whole system unusable. These requirements are usually constraints that must be considered when designing the solutions.

### Production Requirements:

These are requirements that specify product behavior.

- Performance requirements on how fast the system must execute and how much memory it requires.
- Reliability requirements that set out the acceptable failure rate.
- Portability requirements and usability requirements.

## **Organizational Requirements:**

These are derived from policies and procedures in the customer's and developer's organization.

- Process standards which must be used.
- Implementation requirements such as the programming language or design method used in this work we use JAVA software.
- Delivery requirements which specify when the product and its documentation are to be delivered.

## **External Requirements:**

These covers all requirements which are derived from factors external to the system and its development process.

- Interoperability requirements which define how the system interacts with systems in other organizations.
- Legislative requirements which must be followed to ensure that the system operates within the law.

## **CHAPTER-3**

### **SOFTWARE REQUIREMENTS**

#### **3.SOFTWARE REQUIREMENTS**

##### **3.1 Hardware Requirements**

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Monitor : 15 VGA Colour.
- Mouse : Logitech.
- Ram : 512 Mb.

##### **3.2 Software Requirements**

- Operating system : Windows XP/7.
- Coding Language : JAVA/J2EE.
- IDE : Net beans 8.2.
- Database : MYSQL.

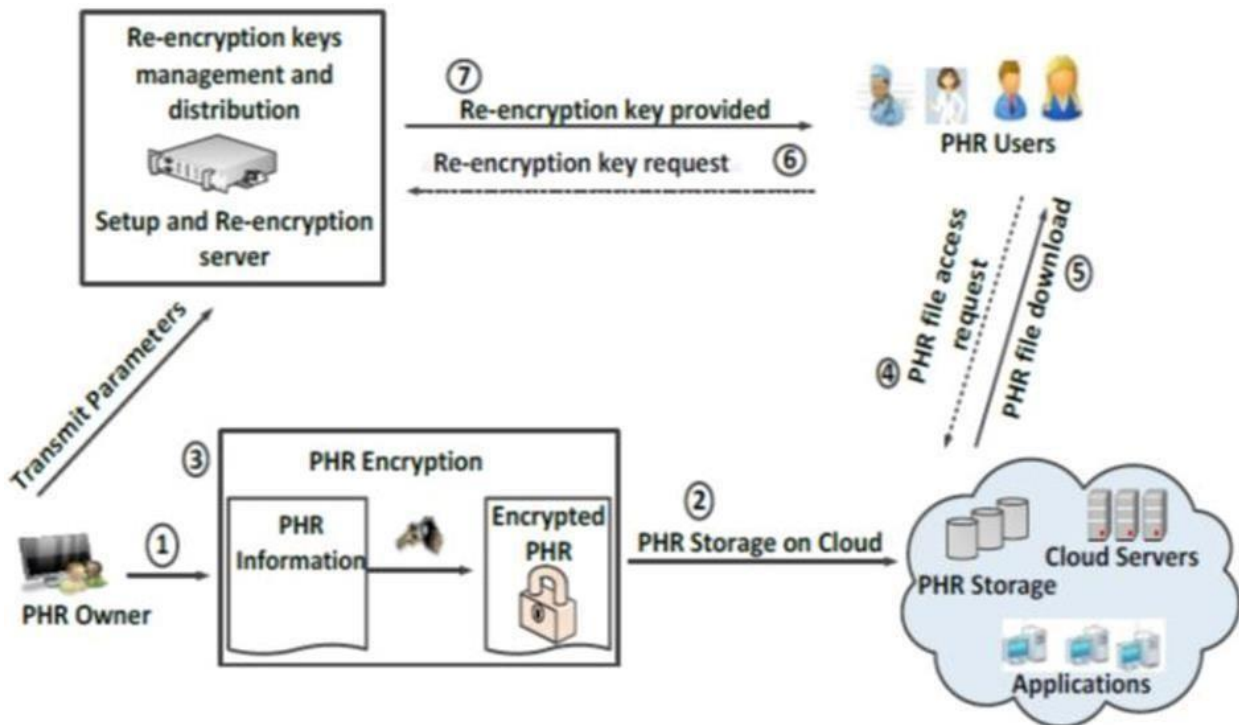
## CHAPTER -4

### SYSTEM DESIGN

#### 4.System Design

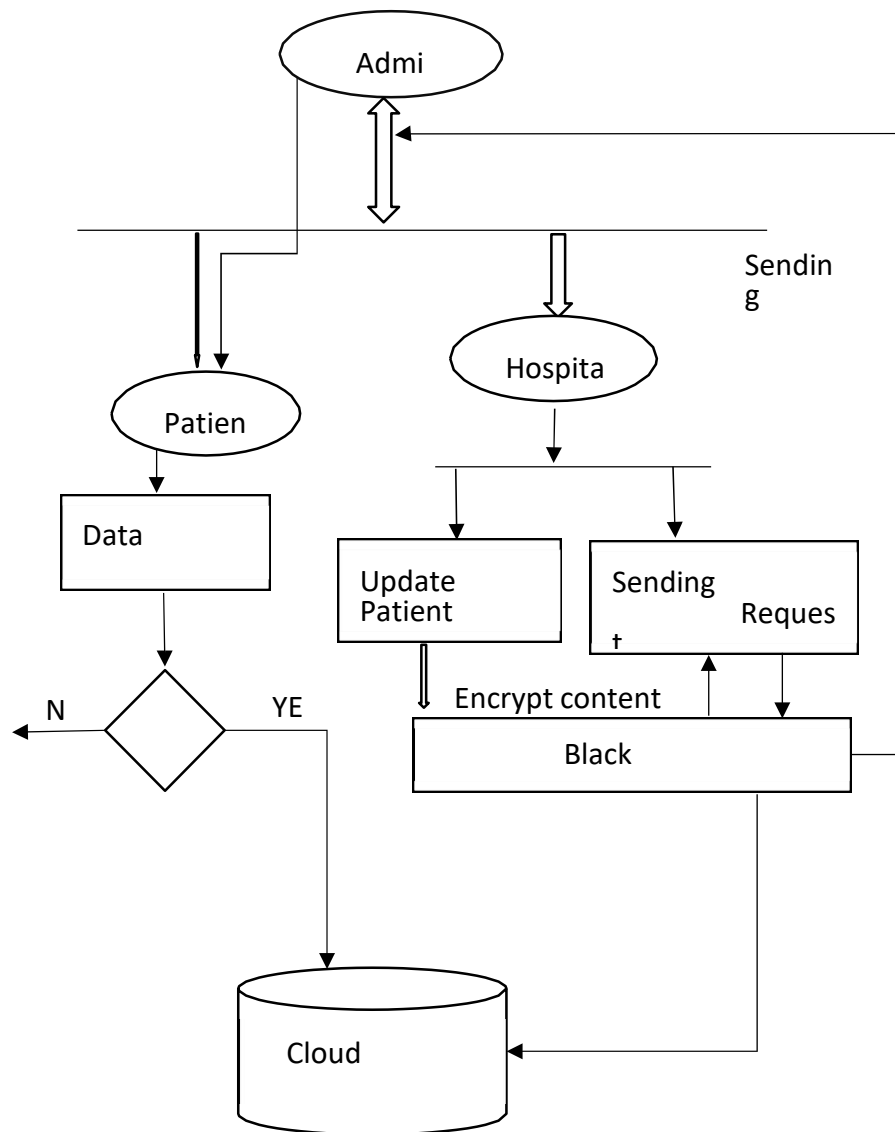
##### 4.1 System Architecture:

The architecture of a secure personal health record (PHR) system in cloud computing typically consists of several components, including:



Architecture of the proposed SeSPHR methodology

- **User Interface:** This component provides an interface for patients and healthcare providers to interact with the PHR system. It may include a web-based portal or mobile application.
- **PHR Application Layer:** This layer contains the logic for managing the PHR data, including storage, retrieval, and sharing. It also includes data encryption and decryption mechanisms.
- **Cloud Storage:** PHR data is stored in the cloud. The cloud storage component provides scalable and reliable storage for the PHR data. The data is stored in encrypted form to ensure confidentiality.
- **Access Control:** Access control is used to regulate access to the PHR data. It includes mechanisms for authentication, authorization, and audit.
- **Secure Communication:** PHR data is transmitted between the user interface and the PHR application layer and between the PHR application layer and the cloud storage using secure communication protocols such as HTTPS and TLS.
- **Privacy Preservation:** The PHR system must ensure the privacy of patient data. Privacy preservation techniques such as anonymization, de-identification, and encryption are used to protect patient privacy.
- **Backup and Recovery:** Backup and recovery mechanisms are used to ensure the availability and reliability of the PHR data.



**Fig. 4.1 System Architecture**

## 4.2 System MODULES:

- Registration
- Upload files
- ABE for Fine-grained Data Access Control
- Setup and Key Distribution
- Break-glass

## **Registration:**

In this module normal registration for the multiple users. There are multiple owners, multiple AAs, and multiple users. The attribute hierarchy of files – leaf nodes is atomic file categories while internal nodes are compound categories. Dark boxes are the categories that a PSD's data reader has access to.

Two ABE systems are involved: for each PSD the revocable KP-ABE scheme is adopted for each PUD, our proposed revocable MA-ABE scheme.

- PUD – public domains
- PSD – personal domains
- AA – attribute authority
- MA-ABE – multi-authority ABE
- KP-ABE – key policy ABE

## **Upload files:**

In this module, users upload their files with secure key probabilities. The owners upload ABE-encrypted PHR files to the server. Each owner's PHR file encrypted both under a certain fine grained model.

## **ABE for Fine-grained Data Access Control:**

In this module ABE to realize fine-grained access control for outsourced data especially, there has been an increasing interest in applying ABE to secure electronic healthcare records (EHRs). An attribute-based infrastructure for EHR systems, where each patient's EHR files are encrypted using a broadcast variant of CP-ABE that allows direct revocation. However, the cipher text length grows linearly with the number of un revoked users. In a variant of ABE that allows delegation of access rights is proposed for encrypted EHRs applied cipher text policy ABE (CP-ABE) to manage the sharing of PHRs, and introduced the concept of social/professional domains investigated using ABE to generate self-protecting EMRs, which can either be stored on cloud servers or cell phones so that EMR could be accessed when the health provider is offline.

**Setup and Key Distribution :** In this module the system first defines a common universe of data attributes shared by every PSD, such as “basic profile”, “medical history”, “allergies”, and “prescriptions”. An emergency attribute is also defined for break-glass access.

Each PHR owner’s client application generates its corresponding public/master keys. The public keys can be published via user’s profile in an online healthcare social-network (HSN)

There are two ways for distributing secret keys.

- First, when first using the PHR service, a PHR owner can specify the access privilege of a data reader in her PSD, and let her application generate and distribute corresponding key to the latter, in a way resembling invitations in GoogleDoc.
- Second, a reader in PSD could obtain the secret key by sending a request (indicating which types of files she wants to access) to the PHR owner via HSN, and the owner will grant her a subset of requested data types. Based on that, the policy engine of the application automatically derives an access structure, and runs keygen of KP-ABE to generate the user secret key that embeds her access structure.

### **Break-glass module:**

In this module when an emergency happens, the regular access policies may no longer be applicable. To handle this situation, break-glass access is needed to access the victim’s PHR. In our framework, each owner’s PHR’s access right is also delegated to an emergency department ED to prevent from abuse of break-glass option, the emergency staff needs to contact the ED to verify her identity and the emergency situation, and obtain temporary read keys. After the emergency is over, the patient can revoke the emergent access via the ED.

## **4.3 Introduction To UML Diagrams:**

Uml diagrams of a methodology for secure sharing of personal health records in cloud computing Sure, I can provide you with an example set of UML diagrams for a methodology for secure sharing of personal health records in cloud computing.

The methodology consists of three main components: the patient, the cloud service provider (CSP), and the healthcare provider. The patient stores their personal health records on the CSP's servers and grants access to healthcare providers as needed. The methodology includes security measures to protect the privacy and confidentiality of the patient's data.

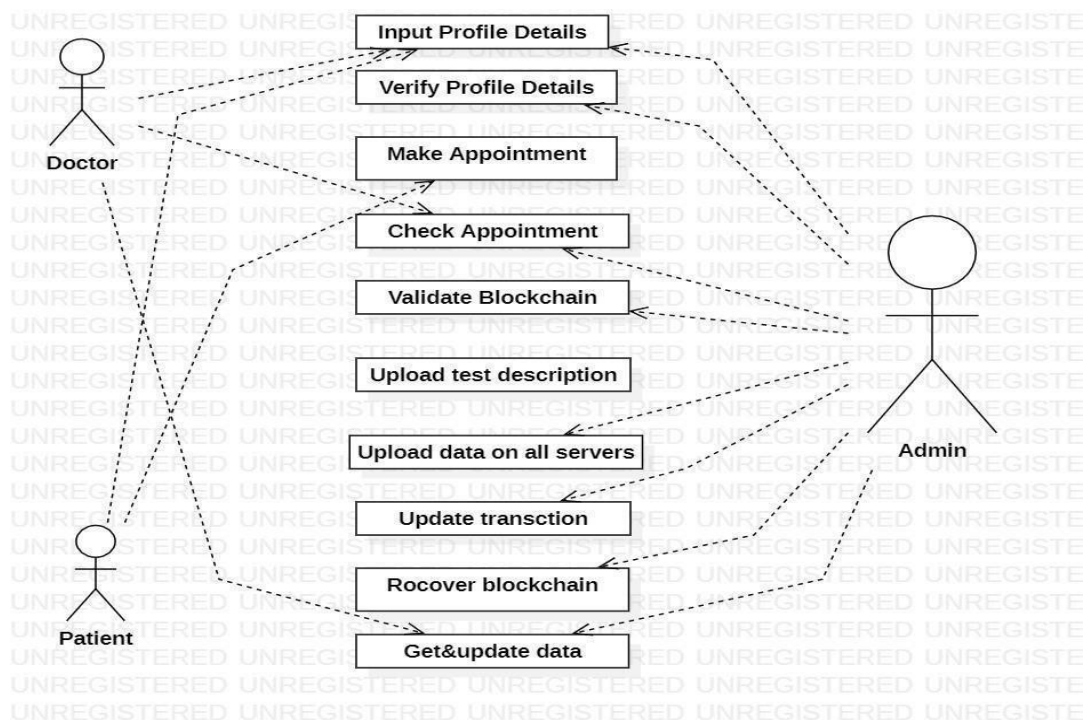


Here are three UML diagrams that illustrate the methodology:

- ‡ Use Case Diagram
- ‡ Class Diagram
- ‡ Sequence Diagram
- ‡ Activity Diagram
- ‡ Component Diagram
- ‡ Deployment Diagram

#### 4.3.1 Use Case Diagram:

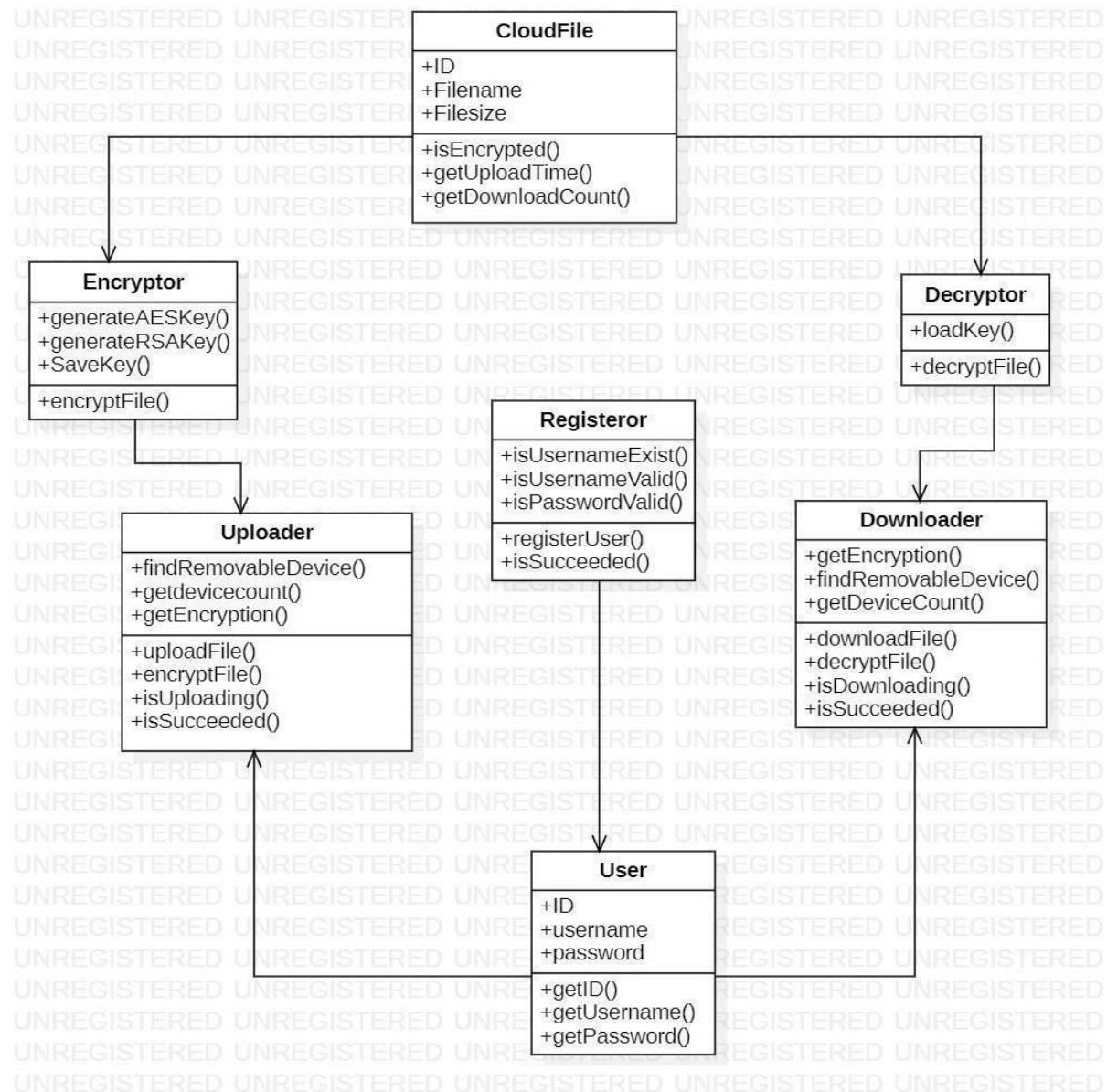
The use case diagram below shows the actors and their interactions with the system. The patient can create, read, update, and delete their health records. The healthcare provider can read the records with the patient's permission. The CSP manages the storage and access to the records.



**Fig 4.2 Use Case Diagram**

### 4.3.2 Class Diagram:

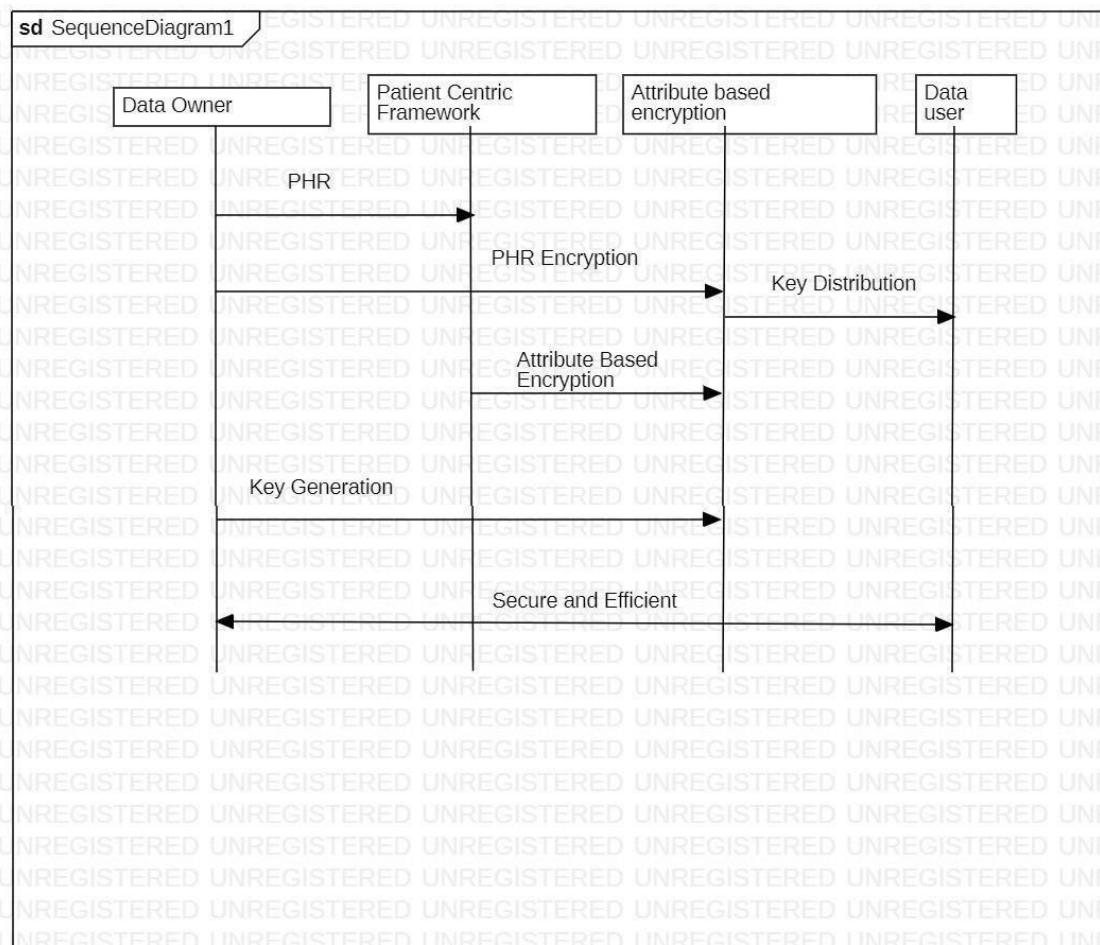
The class diagram below shows the classes involved in the system and their relationships. The Patient class contains information about the patient and their health records. The Healthcare Provider class contains information about the healthcare provider. The CSP class manages the storage and access to the records. The Security class provides security measures to protect the patient's data.



**Fig 4.3 Class Diagram**

### 4.3.3 Sequence Diagram:

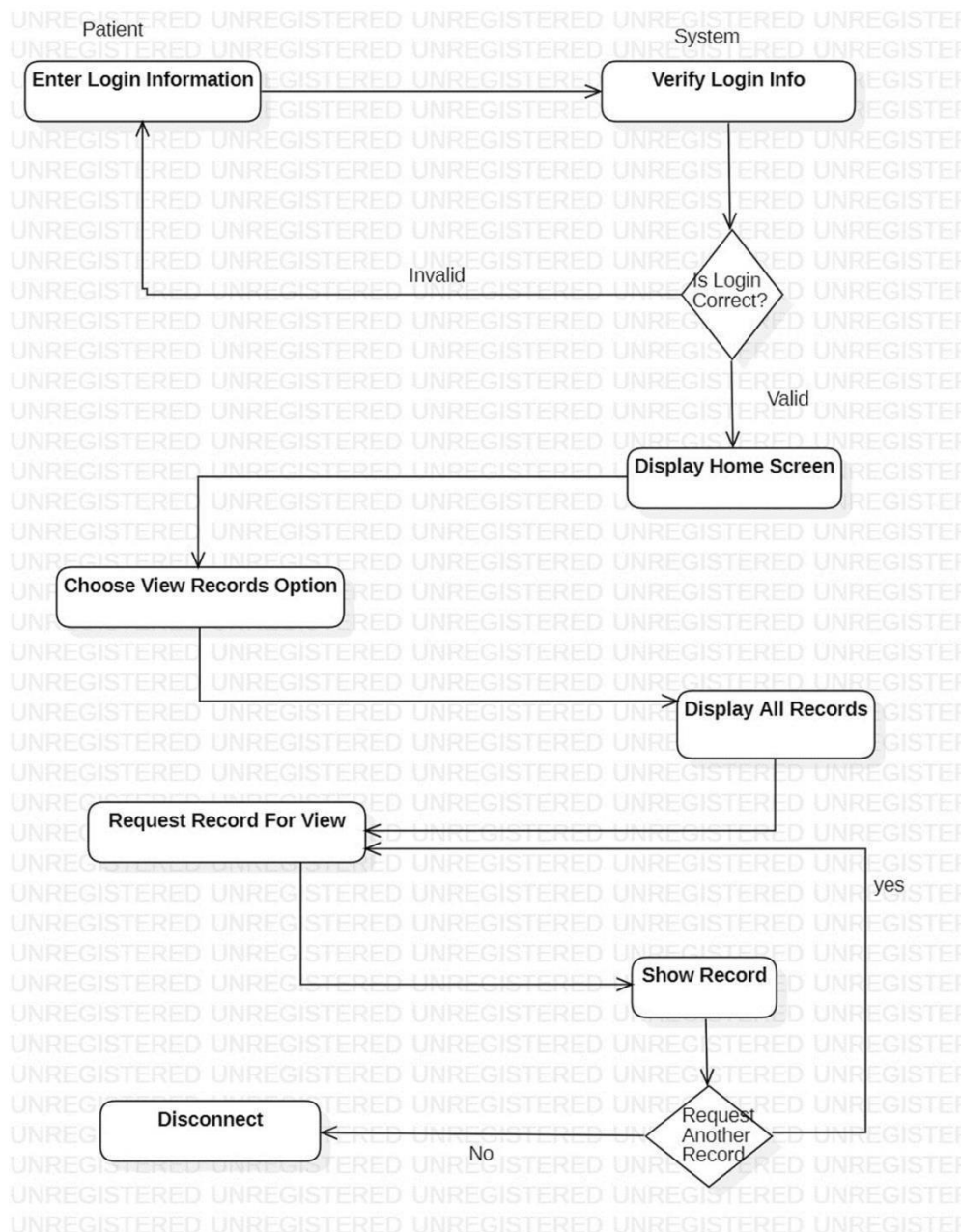
The sequence diagram below shows the sequence of interactions between the patient, the healthcare provider, and the CSP. The patient creates and stores their health records on the CSP's servers. When the healthcare provider requests access to the records, the patient grants permission. The CSP verifies the patient's permission and grants access to the healthcare provider. The healthcare provider can then read the records.



**Fig 4.4 Sequence Diagram**

#### 4.3.4 Activity diagram:

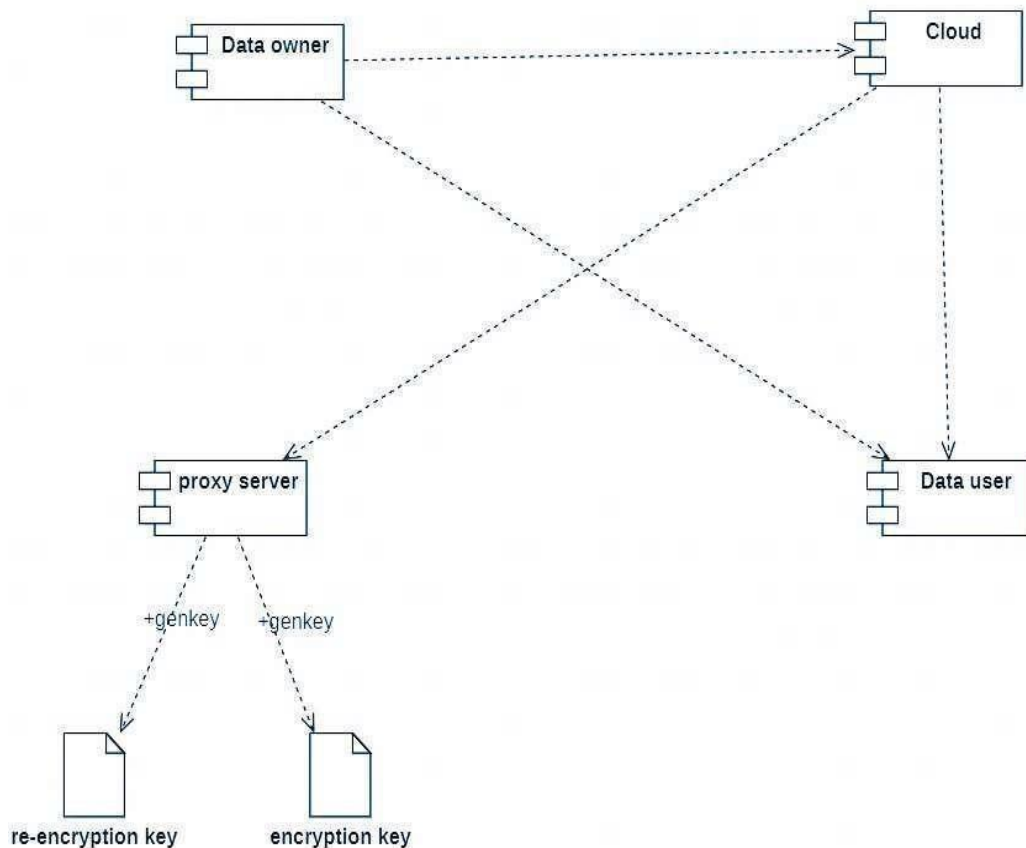
Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another.



**Fig 4.5 Activity diagram**

### 4.3.5 Component diagram :

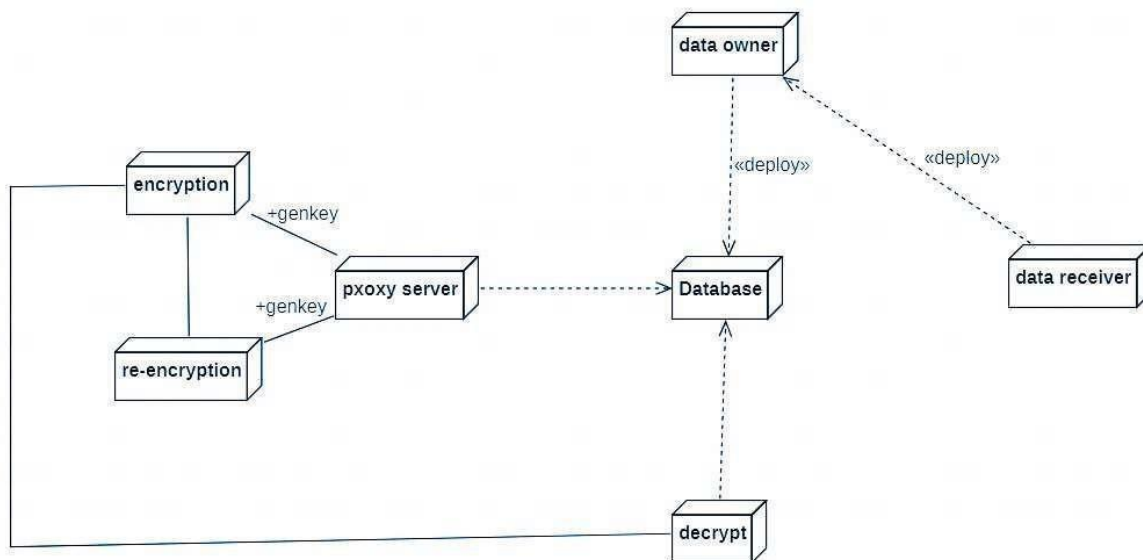
The component diagram's main purpose is to show the structural relationships between the components of a system. The more common use of the term component," which refers to things such as COM components. component diagrams offer architects a natural format to begin modelling a solution. component diagrams are useful communication tools for various groups. Developers find the component diagram useful because it provides them with a high- level, architectural view of the system that they will be building, which helps developers begin formalizing a roadmap for the implementation.



**Fig 4.6 Component diagram**

### 4.3.6 Deployment diagram:

A deployment diagram falls under the structural diagramming family and describes an aspect of the system itself. In this case, the deployment diagram describes the physical deployment of information generated by the software program on hardware components. The information that is generated by the software is called an **artifact**. This shouldn't be confused with the use of the term in other modelling approaches like BPMN. In UML, the hardware where the software is deployed is called a **node**.



**Fig4.7 Deployment diagram**

### 4.3.7 Data Base Design

Designing a secure database for sharing personal health records (PHRs) in cloud computing involves several considerations, including data privacy, security, and compliance with healthcare regulations. Here are some steps you can follow:

**Identify the stakeholders:** Identify the stakeholders who will access the database and their respective roles. This could include patients, healthcare providers, insurance companies, and government agencies.

**Determine the data elements:** Identify the types of data that will be stored in the database, such as patient demographics, medical history, diagnosis, treatment plans, and test results.

**Determine the database schema:** Based on the data elements identified in the previous step, determine the database schema, which includes the tables, fields, and relationships.

**Implement security controls:** Implement security controls to ensure that only authorized users can access the database. This could include access controls, encryption, and auditing.

**Comply with regulations:** Ensure that the database design complies with healthcare regulations, such as the Health Insurance Portability and Accountability Act (HIPAA) in the United States.

**Design for scalability:** Design the database to be scalable, so that it can handle a large number of users and data volumes.

**Implement backup and recovery:** Implement backup and recovery procedures to ensure that the data is recoverable in the event of a disaster.

**Test and validate:** Test and validate the database design to ensure that it meets the requirements of all stakeholders.

### 4.3.8 Normalization

Normalization is an essential aspect of database design, particularly in the context of secure sharing of personal health records (PHRs) in cloud computing. Normalization helps to eliminate redundancy and ensure data consistency, which are crucial for maintaining data privacy and security. Here are some steps you can follow to normalize a database for secure sharing of PHRs in cloud computing:

1. **Identify the entities:** Identify the entities involved in the PHR system, such as patients, healthcare providers, medical facilities, and insurers.
2. **Identify the data elements:** Identify the data elements associated with each entity, such as patient name, address, medical history, diagnosis, and treatment plans.
3. **Determine the functional dependencies:** Determine the functional dependencies between the data elements to identify the primary key and the non-key attributes for each entity.
4. **Normalize the database:** Normalize the database by applying the rules of normalization, such as first normal form (1NF), second normal form (2NF), and third normal form (3NF). For example, to achieve 1NF, ensure that each attribute has a single value, and to achieve 2NF, ensure that all non-key attributes are functionally dependent on the primary key.
5. **Create tables:** Create tables for each entity, using the primary key and non-key attributes identified in the previous steps.
6. **Create relationships:** Create relationships between the tables using foreign keys to ensure data consistency and integrity.
7. **Validate the design:** Validate the database design by testing it with sample data to ensure that it meets the requirements of all stakeholders and is secure.



# CHAPTER -5

## SYSTEM IMPLEMENTATION

### 5.System Implementation

#### 5.1 FRONTEND IMPLIMENTATION:

##### Java Technology

Java technology is both a programming language and a platform.

##### The Java Programming Language

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- High performance
- Interpreted
- Multithreaded
- Robust
- Dynamic Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java byte codes* —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.

You can think of Java byte codes as the machine code instructions for the *Java Virtual Machine* (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make “write once, run anywhere” possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java

VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.

## **The Java Platform**

A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and Mac OS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

- The *Java Virtual Machine* (JVM)
- The *Java Application Programming Interface* (Java API)

You have already been introduced to the Java VM. It's the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces. These libraries are known as *packages*. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware. Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

## **What Can Java Technology Do?**

The most common types of programs written in the Java programming language are applets and applications. If you've surfed the Web, you're probably already familiar with

applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java

programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a *server* serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a *servlet*. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs?

It does so with packages of software components that provides a wide range of functionality.

Every full implementation of the Java platform gives you the following features:

- **The essentials:** Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
- **Applets:** The set of conventions used by applets.
- **Networking:** URLs, TCP (Transmission Control Protocol), UDP (User Datagram Protocol) sockets, and IP (Internet Protocol) addresses.
- **Internationalization:** Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
- **Security:** Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
- **Software components:** Known as JavaBeans<sup>TM</sup>, can plug into existing component architectures.

- **Object serialization:** Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBC™):** Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.

## How Will Java Technology Change My Life?

We can't promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

- **Get started quickly:** Although the Java programming language is a powerful object-oriented language, it's easy to learn, especially for programmers already familiar with C or C++.
- **Write less code:** Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.
- **Write better code:** The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people's tested code and introduce fewer bugs.
- **Develop programs more quickly:** Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.
- **Avoid platform dependencies with 100% Pure Java:** You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure Java™ Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.
- **Write once, run anywhere:** Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.

- **Distribute software more easily:** You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded “on the fly,” without recompiling the entire program.

## **5.2 BACKEND IMPLEMENTATION:**

### **ODBC**

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a de facto standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources. From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn't change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources.

The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn't as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

## **JDBC**

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of "plug-in" database connectivity modules, or drivers. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC's framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after. The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

## JDBC Goals

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

### 1. SQL Level API

The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to “generate” JDBC code and to hide many of JDBC’s complexities from the end user.

#### SQL Conformance:

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

#### **JDBC must be implemental on top of common database interfaces**

The JDBC SQL API must “sit” on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.

#### **Provide a Java interface that is consistent with the rest of the Java system**

Because of Java’s acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

##### **a. Keep it simple**

This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.



**b. Use strong, static typing wherever possible**

Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime.

**c. Keep the common cases simple**

Because more often than not, the usual SQL calls used by the programmer are simple SELECT's, INSERT's, DELETE's and UPDATE's, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible. Finally we decided to proceed the implementation using Java [Networking](#). And for dynamically updating the cache table we go for MS [Access](#) database.

Java has two things: A programming language and A platform.

Java is a high-level programming language that is all of the following :

Simple	Architecture
Object-oriented	neutral
Distributed	Portable
Interpreted	High-performance
Robust	multithreaded
Secure	Dynamic

Java is also unusual in that each Java program is both compiled and interpreted. With a compile you translate a Java program into an intermediate language called Java byte codes the platform-independent code instruction is passed and run on the computer.

Compilation happens just once; interpretation occurs each time the program is executed. The figure illustrates how this works.

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware. Java byte codes help make "write once, run anywhere" possible. You can compile your Java program into byte codes on my platform that has a Java compiler.

# **CHAPTER -6**

## **SYSTEM TESTING**

### **6 . System Testing**

System testing for secure sharing of personal health records in cloud computing involves evaluating the system's ability to protect the privacy, confidentiality, and security of personal health information while it is being stored, processed, and transmitted in the cloud.

Here are some steps that can be taken to perform system testing for secure sharing of personal health records in cloud computing:

- Test access controls: Verify that only authorized users are able to access personal health records. Test the authentication and authorization mechanisms to ensure that they are functioning correctly and that access to personal health records is restricted only to authorized individuals.
- Test data encryption and decryption: Ensure that personal health records are encrypted both at rest and in transit. Test the system's ability to encrypt and decrypt personal health records and verify that the encryption is strong enough to prevent unauthorized access.
- Test disaster recovery and business continuity: Test the system's ability to recover from disasters such as hardware failures, power outages, and natural disasters. Verify that the system can restore data quickly and accurately and that it can continue to operate even during an outage.
- Test auditing and logging: Verify that the system is capable of auditing all activities related to personal health records, including access attempts, modifications, and deletions. Ensure that the system is logging all relevant data and that the logs are secure and tamper-evident.

By following these steps, you can ensure that your system for secure sharing of personal health records in cloud computing is functioning as intended and that it is capable of protecting the privacy, confidentiality, and security of personal health information.

## 6.1 TESTING CONCEPTS:

### ➤ Unit testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

### ➤ Integration testing:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

### ➤ Functional test:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input: Identified classes of valid input must be accepted.
- Invalid Input: Identified classes of invalid input must be rejected.
- Functions: Identified functions must be exercised.
- Output: Identified classes of application outputs must be exercised.
- Systems/Procedures: interfacing systems or procedures must be invoked.

➤ System testing:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

➤ **White Box Testing:**

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose.

It is purpose. It is used to test areas that cannot be reached from a black box level.

➤ Black Box Testing:

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .

you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

## **6.2 Test strategy and approach:**

Field testing will be performed manually and functional tests will be written in detail.

### **Test objectives**

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

### **Features to be tested**

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

## **Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

## **Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

### **6.3 Test cases:**

#### **I. Authentication and Authorization:**

- Verify that only authorized users can access personal health records.
- Test login credentials with both valid and invalid login attempts.
- Test that users cannot access data that they are not authorized to view.

#### **II. Data Encryption:**

- Test that personal health records are encrypted in transit and at rest.
- Verify that the encryption is strong enough to prevent unauthorized access.
- Test that data is decrypted correctly when authorized users access it.

#### **III. Access Logging and Monitoring:**

- Verify that access to personal health records is logged and monitored.
- Test that logs accurately record all access attempts, modifications, and deletions.
- Test that the logs are secure and tamper-evident.

#### **IV. Disaster Recovery and Business Continuity:**

- Test the system's ability to recover from hardware failures, power outages, and natural disasters.
- Verify that the system can restore data quickly and accurately.
- Test that the system can continue to operate even during an outage.

#### **V. User Interface and Usability:**

- Test the user interface for ease of use and clarity.
- Verify that the user interface meets accessibility standards.
- Test that users can easily find and access the personal health records they need.

#### **VI. Performance:**

- Test the system's performance with different amounts of data and different numbers of users.
- Verify that the system can handle peak loads without slowing down or crashing.
- Test that the system responds quickly to user requests.

#### **VII. Security:**

- Test the system for vulnerabilities that could potentially compromise the security of personal health information.
- Verify that the system is protected against common attacks such as SQL injection and cross-site scripting.
- Test that the system has appropriate measures in place to prevent data breaches.

## CHAPTER-7

### SOURCE CODE

#### 7.SOURCE CODE

##### 7.1 ABE.JAVA:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

import java.security.spec.KeySpec;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.DESedeKeySpec;
import org.apache.commons.codec.binary.Base64;

public class ABE {
    private static final String UNICODE_FORMAT = "UTF8";
    private static final String DESEDE_ENCRYPTION_SCHEME = "DESede";
    private KeySpec ks;
    private SecretKeyFactory skf;
    private Cipher cipher;
    byte[] arrayBytes;
    private String myEncryptionKey;
    private String myEncryptionScheme;
    SecretKey key;

    public ABE() throws Exception {
        myEncryptionKey = "ThisIsSpartaThisIsSparta";
        myEncryptionScheme = DESEDE_ENCRYPTION_SCHEME;
        arrayBytes = myEncryptionKey.getBytes(UNICODE_FORMAT);
        ks = new DESedeKeySpec(arrayBytes);
        skf = SecretKeyFactory.getInstance(myEncryptionScheme);
        cipher = Cipher.getInstance(myEncryptionScheme);
        key = skf.generateSecret(ks);
    }

    public String encrypt(String unencryptedString) {
        String encryptedString = null;
```

```

    Try{
        cipher.init(Cipher.ENCRYPT_MODE, key);
        byte[] plainText = unencryptedString.getBytes(UNICODE_FORMAT);
        byte[] encryptedText = cipher.doFinal(plainText);
        encryptedString = new String(Base64.encodeBase64(encryptedText));
    } catch (Exception e) {
        e.printStackTrace();
    }
    return encryptedString;
}

public String decrypt(String encryptedString) {
    String decryptedText=null;
    try {
        cipher.init(Cipher.DECRYPT_MODE, key);
        byte bb[]=encryptedString.getBytes();
        byte[] encryptedText = Base64.decodeBase64(encryptedString.getBytes());
        byte[] plainText = cipher.doFinal(encryptedText);
        decryptedText= new String(plainText);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return decryptedText;
}
}

```

## 7.2 CONTENT EXTRACTOR.JAVA:

```

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

/**
 *
 * @author jp8
 */
public class ContentExtractor {
    public static String getStringFromInputStream(InputStream is) {
        BufferedReader br = null;
        StringBuilder sb = new StringBuilder();
        String line;
        try {
            br = new BufferedReader(new InputStreamReader(is));
            while ((line = br.readLine()) != null) {
                sb.append(line + "\n");
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {

```



```

        if (br != null) { try {
            br.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
return sb.toString();
}
}

```

### 7.3 DB CONN.JAVA:

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

import java.sql.Connection;
import java.sql.DriverManager;

/**
 *
 * @author java2
 */
public class Dbconn {

    public static Connection getConnection() {
        Connection con = null;
        try {
            Class.forName("com.mysql.jdbc.Driver");
            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/sesphr", "root", "");
            System.out.println("Connected");
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return con;
    }
}

```

### 7.4 FTPCON.JAVA:

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

```

```

import java.sql.Connection;
import java.sql.DriverManager;

/**
 *
 * @author java2
 */
public class Dbconn {

    public static Connection getConnection() {
        Connection con = null;
        try {
            Class.forName("com.mysql.jdbc.Driver");
            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/sesphr", "root", "");
            System.out.println("Connected");
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return con;
    }
}

```

## 7.5 UPLOAD.JAVA:

```

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.util.Iterator;
import java.util.List;
import java.util.Random;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.swing.JOptionPane;
import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.FileItemFactory;
import org.apache.commons.fileupload.FileUploadException;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;

/**
 *
 * @author pravash
 */
public class Upload extends HttpServlet {

```

```

private static java.sql.Date getCurrentDate() {
    java.util.Date today = new java.util.Date();
    return new java.sql.Date(today.getTime());
}

protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        /* TODO output your page here. You may use following sample code. */
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet Upload</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Servlet Upload at " + request.getContextPath() + "</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}

// <editor-fold defaultstate="collapsed" desc="HttpServletRequest methods. Click on the + sign on the left to
// edit the code.">
/**
 * Handles the HTTP <code>GET</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Handles the HTTP <code>POST</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");

```

```

PrintWriter out = response.getWriter();
try{

    int key=1234;

    HttpSession ssn=request.getSession();
    String id=ssn.getAttribute("id").toString();
    //String tid=ssn.getAttribute("tokenid").toString();
    String uname=ssn.getAttribute("uname").toString();
        System.out.println("ID is "+id);
        // System.out.println("ID is "+id);
        System.out.println("Username is "+uname);
    /* TODO output your page here. You may use following sample code. */
    Connection con;
    PreparedStatement pstmt = null;
    File f = null;
    DiskFileItemFactory diskFile = new DiskFileItemFactory();
    diskFile.setSizeThreshold(1 * 1024 * 1024);
    diskFile.setRepository(f);
    ServletFileUpload sfu = new ServletFileUpload(diskFile);
    List item = sfu.parseRequest(request);
    Iterator itr = item.iterator();

    FileItem items = (FileItem) itr.next();
    String str = ContentExtractor.getStringFromInputStream(items.getInputStream());
    //FileItem items1 = (FileItem) itr.next();
    System.out.println("file:"+str);
    String str1 = ContentExtractor.getStringFromInputStream(items.getInputStream());
    //System.out.println("file1 :"+str1);
    //rsa.main(key);
    String encstr = new ABE().encrypt(str);
    InputStream input = items.getInputStream();
    int size=input.available();
    System.out.println("file name is " +items.getName());

    con = Dbconn.getConnection();
    Random r=new Random();
    int skey=0;
    while(true)
    {
        skey=r.nextInt(10000);
        if(skey>1000)
            break;
    }
    pstmt = con.prepareStatement("insert into files (file_name, size, file, data,skey,pid,username)
values(?,?,?,?,?,?,?)");
    pstmt.setString(1, items.getName());
    pstmt.setInt(2, size);
    pstmt.setBinaryStream(3, items.getInputStream(),items.getInputStream().available());
    pstmt.setString(4, encstr);
    pstmt.setInt(5, skey);

```

```

        pstmt.setString(6, id);
//        pstmt.setString(7, tid);
        pstmt.setString(7, uname);
        /*Cloud Start
        File f1 = new File("E:\\"+items.getName());
        //File f1 = new File(items.getName());
        FileWriter fw = new FileWriter(f1);
        fw.write(encstr);
        fw.close();
        Ftpcon ftpcon = new Ftpcon();
        ftpcon.upload(f1, items.getName());
        Cloud End*/

        int i = pstmt.executeUpdate();
        if(i!=0){
            response.sendRedirect("PhrOwnerHome.jsp?q=fileuplodedsusscessfully...");
        }
        else{
            response.sendRedirect("PhrOwnerHome.jsp?q=Uploadingfailed...");
        }
    } catch(Exception ex){
        //response.sendRedirect("f_upload.jsp?q=File_name_already_exit");
        ex.printStackTrace();
    }
    finally{
        out.close();
    }
}

/**
 * Returns a short description of the servlet.
 *
 * @return a String containing servlet description
 */
@Override
public String getServletInfo() {
    return "Short description";
} // </editor-fold>

}

```

## CHAPTER-8

### OUTPUT SCREEN

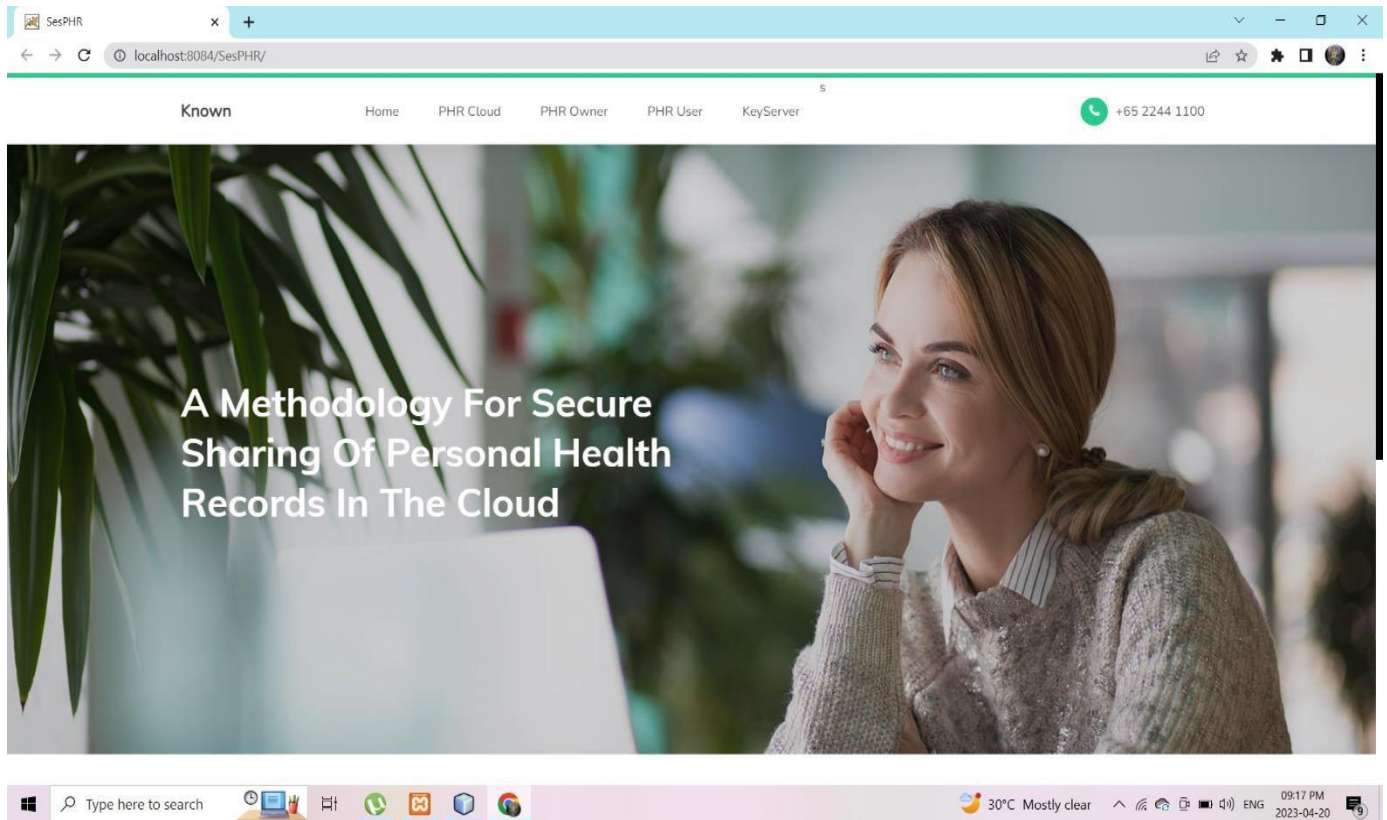


Fig 8.1. Home page

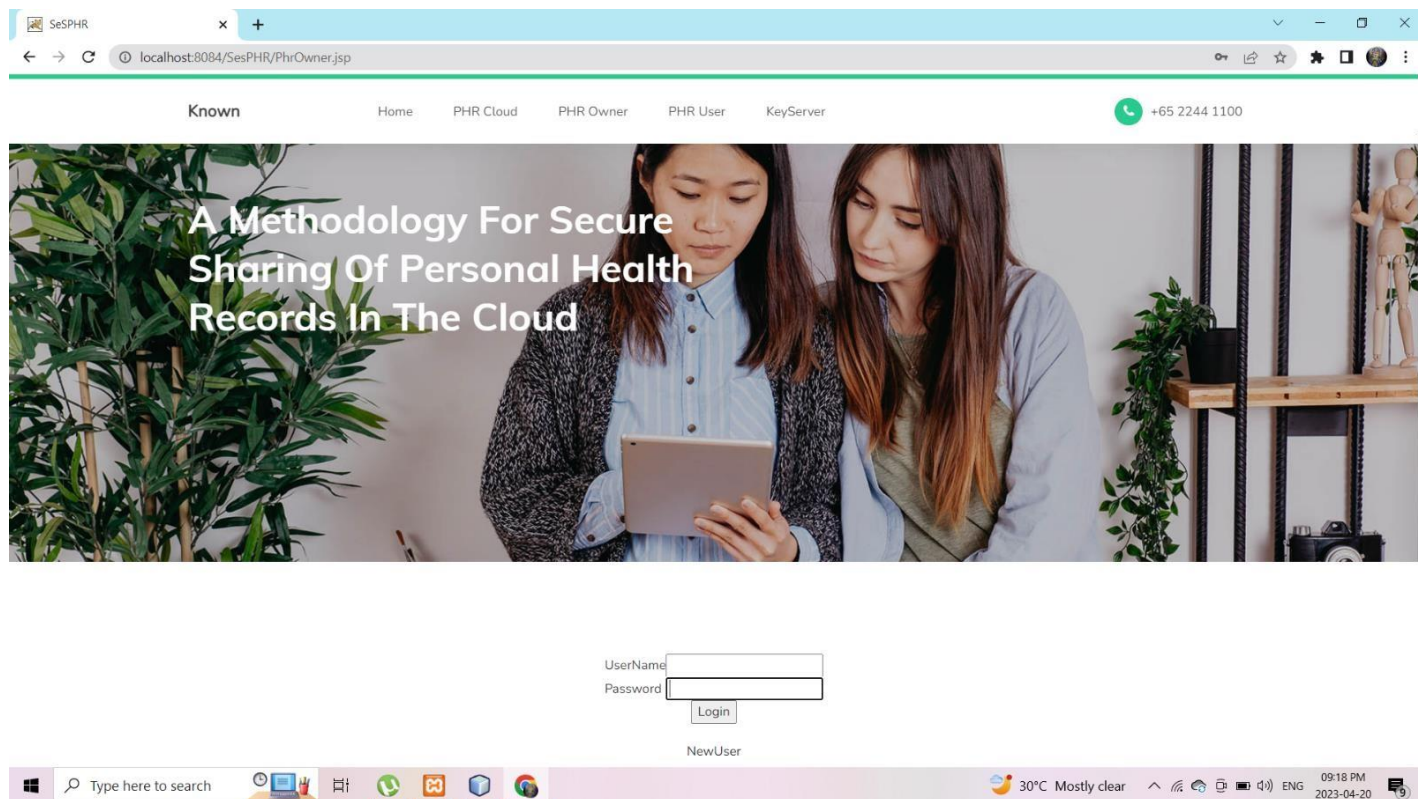


Fig 8.2 Login Page

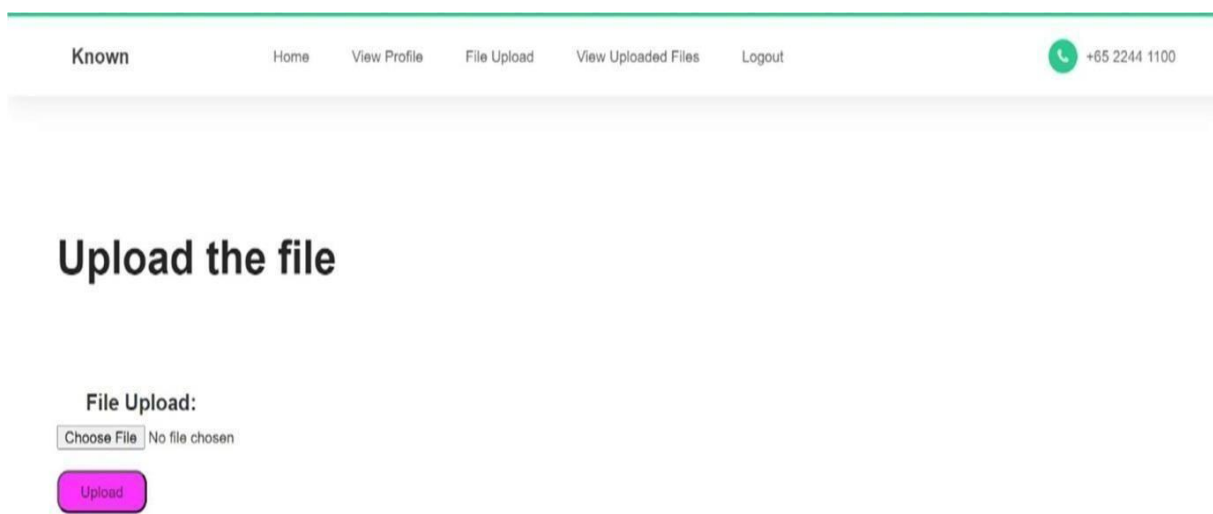


Fig 8.3 Owner File upload

JSP Page

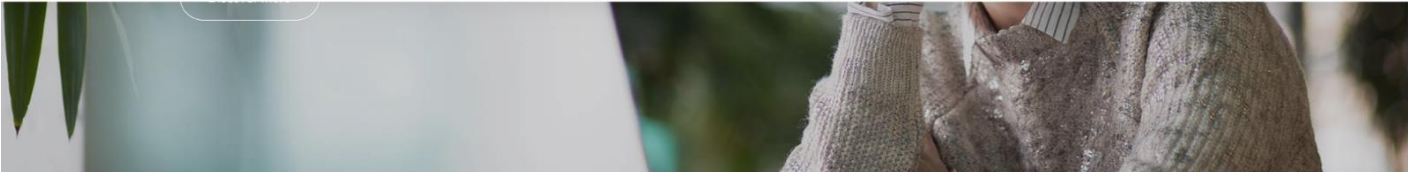
localhost:8084/SesPHR/ViewUsers.jsp

Known

[Home](#)
[PHRUsers](#)
[PHRDataOwners](#)
[Uploaded Files](#)
[Downloaded Files](#)
[Request Accepted](#)
[Request Denied](#)

Logout

+65 2244 1100



id	username	name	password	mail	phoneno	qualification	authorized
1	pravash	pravash	12345	pravashranjansahoo@gmail.com	9030196130	MCA	yes Authorize
2	Bishesh	Bishesh	12345	bishesh@gmail.com	9123456789	Mtech	No Authorize
3	Vaddi Sravan	Vaddi Sravan	Sravani@12345	sravanivaddi308@gmail.com	9618019884	MBBS	yes Authorize

Type here to search

30°C Mostly clear
09:19 PM 2023-04-20

Fig 8.4 User List



JSP Page x +


localhost:8084/SesPHR/ViewDataOwner.jsp

Known

Home PHRUsers PHRDataOwners Uploaded Files Downloaded Files Request Accepted Request Denied

Logout

+65 2244 1100



	id	username	name	password	mail	phoneno	qualification	authorized
1	pravash	pravash		12345	pravashranjansahoo@gmail.com	9030196130	MCA	No
3	hello	pravash		12345	pravashranjansahoo@gmail.com	9030196130	mca	yes
4	sravani	Dunna sravani	sravani@123		Sravanidunna95@gmail.com	9390423908	b.tech	yes
5	Vijaya	Vijaya Pothula	Vijaya@123		lakshmivijayapothula@gmail.com	9390254030	B.tech	yes
6	sudheer	sudheer kumar	sudheer@123		kanchisudheerkumar777@gmail.com	7416702699	B.tech	yes

Type here to search

30°C Mostly clear 09:19 PM 2023-04-20

Fig 8.5 Data Owners


Known					Home	View Profile	View Uploaded Files	Download	Logout	 +65 2244 1100
Requestid	file_name	size	date	data						
Send Request	1 cloud_page.jsp	1671	2020-03-11 18:07:35.0	bx2skqCXiO7fUeOUx/rTRCewEZDptsFgerwoVPAm4VUGM9aNT+seJEwO+1pZqG7r5s2XpGQGJBC7PG1kglyO62xHjKBT3ecjQDY5tlC8b+kJ4SZ						
Send Request	2 Home.jsp	10341	2020-03-13 06:30:25.0	bx2skqCXiO7fUeOUx/rTRAjlOgidFjUVdVbTrG4pKYL0W5VPEXny1A7U/LMKemB4Zfp2Ki/zHNxWTXqomGj35h7PNVCMec6qUhW+LY84C26R9n						
Send Request	3 storage.jsp	11640	2020-03-13 06:34:27.0	bx2skqCXiO7fUeOUx/rTRIUEPrH93bYmfomYcNlakRDyyqJByZ+7h/RblU8TGfLUzMWPLvKNqvlhLDN03Cn4fkz7RymdBVx81XjKywchytvHaySo						
Send Request	4 Register.jsp	8059	2020-03-13 06:38:52.0	OU7Z/vDPE3K6f2C+p3BjldLWwLjnL1XKSFBjAANcEi/2ZM5iiBnMRciLEF9G/h0z6or2OsxPjg5HI7g0uxReGUgoU3gH81MJCDliTolREY6KcgsKF6ej						
Send Request	5 index.jsp	6805	2020-03-13 07:07:52.0	bx2skqCXiO4JvdPIBfuqb0e8Sp1G4K2OZprrqQceakdXtlJeU1IC0U+ah2GdrSYUCBQG3CcOlzlf5Q28YmvNWqPkl9s1HLP3dMua5lZNxx6NxSIHL						
Send Request	6 1st apr.txt	3039	2020-04-30 14:24:01.0	AtZ04YDKveeLLcbY6aQKzulgaZyMle4hddpzYeSKCw04Z4vfU153BAcSGpReJk7Dujs+HAG90YPucYF+CH0dMnpjVjXQ9o78InU2s02+oPsGjgqir						
Send Request	7 MS OFFICE LOG IN DETAILS.txt	57	2023-03-03 22:54:36.0	GuSRbsgE1izaYOSply3sQXSrp5E90/2vbDSkK1q3fIUtzxhxhvia0FNHHiHp5w7OPfYRH9frn0E=						
Send Request	8 Dbconn.java	717	2023-03-04 17:17:05.0	xkVaYzffsruaOe6DtlGigtidMYuZyZvdty+B4BF778mFDREEINK4aBp0QOyITBLy1wTJdFiUqmaxsH9GVVwKWE+z5N6Hk1Y3a7W30nuVP5E9LMn						

Fig 8.6 User Send Request

id	file_name	date	name	mail	
3	MS OFFICE LOG IN DETAILS.txt	2023-03-03 22:54:36.0	pravash	pravashranjansahoo@gmail.com	Accept
5	ABE.java	2023-03-04 17:37:47.0	pravash	pravashranjansahoo@gmail.com	Accept
6	ABE.java	2023-03-04 17:37:47.0	pravash	pravashranjansahoo@gmail.com	Accept
10	ABE.java	2023-03-04 17:37:47.0	pravash	pravashranjansahoo@gmail.com	Accept

Fig 8.7 Key Accept

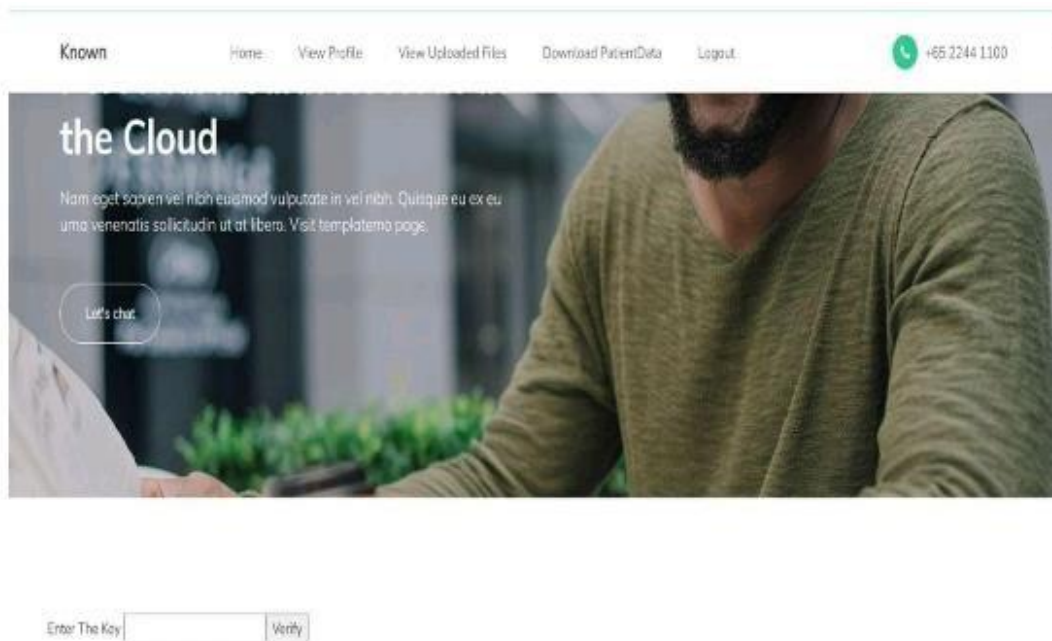


Fig 8.8 Encrypted Key

## 9.CONCLUSION

A personal health records is one type of medical record which is ability add information in it. Using cloud computing the data distribute and manage various locations without using it infrastructure.persnal health record generated by patients, doctors, hosipitals. Electronic health records are controlled by only doctors by using personal health records patients better to manage their health care personal health care records are maintained to computers or papers. By using cloud in health care discus about clinical policies to receive at right time. Offered to a cloud early detection can be taken. Health cloud provides that can be used to a review and medical care services. Medical records provide documentary evidence of a patient's health care information. Frame work of secure sharing of personal health records in cloud computing. Considering partially trust worthy cloud servers, argue that to fully realize the patient-centric concept, patients shall have complete control of their own privacy through encrypting their PHR files to allow fine-grained access. Cloud facilitates technologies that are used in healthcare like electronic medical records, mobile apps, patient portals, devices with IOT.

Medical records provide documentary evidence of a patient's health care information. Health cloud provides objects you can use to review and evaluate medical care services, communicate about clinical policies, and health plan members ensure they receive the right care in the right setting at the right time. We proposed a methodology to securely store and transmission of the PHRs to the authorized entities in the cloud. The methodology preserves the confidentiality of the PHRs and enforces a patient-centric access control to different portions of the PHRs based on the access provied by the patients. We implemented a fine-grained access control method in such a way that even the valid system users cannot access those portions of the PHR for which they are not authorized.

## REFERENCE

- [1] K. Gai, M. Qiu, Z. Xiong, and M. Liu, "Privacy-preserving multi-channel communication in Edge-of-Things," *Future Generation Computer Systems*, 85, 2018, pp. 190-200.
- [2] K. Gai, M. Qiu, and X. Sun, "A survey on FinTech," *Journal of Network and Computer Applications*, 2017, pp. 1-12.
- [3] A. Abbas, K. Bilal, L. Zhang, and S. U. Khan, "A cloud based health insurance plan recommendation system: A user centered approach," *Future Generation Computer Systems*, vols. 43- 44, pp. 99-109, 2015.
- [4] A. N. Khan, ML M. Kiah, S. A. Madani, M. Ali, and S. Shamshir band, "Incremental proxy re-encryption scheme for mobile cloud computing environment," *The Journal of Super computing*, Vol. 68, No. 2, 2014, pp. 624-651.
- [5] R. Wu, G.-J. Ahn, and H. Hu, "Secure sharing of electronic health records in clouds," In *8th IEEE International Conference on Collaborative Computing: Networking, Applications and Work-* 2168-7161 (c) 2018 IEEE. This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2854790, *IEEE Transactions on Cloud Computing* 14 sharing (Collaborate Com), 2012, pp. 711-718.
- [6] A. Abbas and S. U. Khan, "A Review on the State-of-the-Art Privacy Preserving Approaches in E-Health Clouds," *IEEE Journal of Biomedical and Health Informatics*, vol.18, no. 4, pp. 1431-1441, 2014.
- [7] M. H. Au, T. H. Yuen, J. K. Liu, W. Susilo, X. Huang, Y. Xiang, and Z. L. Jiang, "A general framework for secure sharing of personal health records in cloud system," *Journal of Computer and System Sciences*, vol. 90, pp. 46-62, 2017.
- [8] J. Li, "Electronic personal health records and the question of privacy," *Computers*, 2013, DOI: 10.1109/MC.2013.225.