1. INTRODUCTION

Cloud-based data storage services have experienced a notable surge in popularity due to their efficient management and cost-effective nature. However, the openness of the network environment in which these services operate poses significant challenges regarding data security and user privacy. While encryption is commonly utilized to safeguard sensitive data, traditional methods often prove inadequate in meeting the complex requirements of data management within cloud environments. Moreover, effective access control mechanisms are essential to mitigate unauthorized access risks and potential threats such as Economic Denial of Sustainability (EDoS). In response to these challenges, this paper presents a novel dual-access control framework tailored specifically for cloud-based storage systems. By addressing both data access and download requests comprehensively, the framework aims to enhance security and efficiency while ensuring the confidentiality and integrity of user data.

This proposed framework seeks to bridge existing gaps in access control and encryption methodologies by offering a comprehensive solution that encompasses both aspects of data management in cloud environments. By integrating innovative techniques and leveraging advanced encryption methodologies, the framework aims to provide a robust defense against unauthorized access and potential security threats. Through rigorous security and experimental analyses, the effectiveness and practical viability of the proposed framework are evaluated, offering a promising avenue for enhancing the security posture of cloud-based storage services.

1.1 Motivation

The motivation for the proposed concept arises from the critical need to address the escalating security challenges confronting cloud-based data storage and access control systems. With the exponential growth in data volume and the increasing reliance on cloud services across various industries, ensuring the confidentiality, integrity, and availability of data has become paramount. However, traditional encryption methods and access control mechanisms often fall short in providing adequate protection against evolving threats, leaving organizations vulnerable to data breaches and unauthorized access. As such, there is a pressing demand for innovative solutions that can effectively mitigate these risks while ensuring seamless access to cloud resources.

Moreover, the proliferation of sophisticated cyber threats, including ransomware attacks, data exfiltration attempts, and insider threats, underscores the urgency of bolstering the security posture of cloud-based storage systems. The motivation for the proposed concept is further fueled by the need to empower organizations with robust security measures that can adapt to dynamic threat landscapes and regulatory requirements. By introducing a dual access control framework and enhancing encryption techniques, our aim is to provide organizations with the means to fortify their defences, mitigate risks, and maintain compliance with data protection regulations. Ultimately, the motivation behind this endeavour is to install confidence in cloud-based data storage solutions, enabling organizations to harness the benefits of cloud computing without compromising security.

1.2 Problem Definition

In the context of cloud-based data storage and access control, the problem at hand revolves around the need to ensure robust security measures to protect sensitive data and uphold user privacy while also addressing challenges related to scalability and efficiency. Existing systems primarily rely on encryption and access control mechanisms, such as CP-ABE schemes, to safeguard data and manage user access. However, these systems often exhibit limitations in terms of scalability, expressiveness of access policies, and computational efficiency. Moreover, there is a lack of utilization of stronger encryption techniques, such as 256 or 512 bit encrypted keys, which could enhance the overall security posture of the system. Consequently, there is a pressing need to design a comprehensive solution that not only provides effective encryption and access control but also addresses the scalability and efficiency concerns associated with cloud-based storage services.

1.3 Objectives of the Project

The primary objective of this project is to enhance the security and efficiency of cloud-based data storage and access control systems. This involves designing and implementing a comprehensive dual access control framework that regulates both data access and download requests. By integrating robust encryption techniques and access control mechanisms, the system aims to safeguard sensitive data and uphold user privacy in the face of evolving security threats. Additionally, the project seeks to address limitations in existing systems, such as scalability concerns and inefficiencies in access control policies, by introducing innovative solutions and optimizations.

A key focus of the project is to implement a dual access control system that encompasses both data access and download requests. This involves developing mechanisms for user authentication, access verification, and policy enforcement to prevent unauthorized access and data breaches. By incorporating identity key verification techniques and enhancing

Attribute-Based Encryption (ABE) schemes, the system aims to strengthen user authentication and data protection measures. Furthermore, the project aims to utilize stronger encryption keys, such as 256 or 512-bit keys, to bolster data security and mitigate the risk of unauthorized access or data breaches.

Another objective of the project is to ensure scalability and efficiency in the design and implementation of the system. This involves optimizing algorithms and data structures to minimize computational overhead and enhance system performance, thereby enabling the system to handle increasing data volumes and user demands without sacrificing performance. By conducting thorough security and experimental analyses, the project aims to validate the effectiveness and practical viability of the proposed system, identifying and addressing any potential vulnerabilities or shortcomings in the system design. Ultimately, the project seeks to provide organizations with a robust and reliable cloud-based storage solution that meets their security, privacy, and performance requirements.

2. Literature Survey

An approach of dual access control for cloud-based storage systems has explored novel cryptographic techniques and access control models to enhance data security and privacy. One concept that has garnered significant attention is Conditional Attribute-Based Encryption (CABE). CABE enables data encryption based on specific conditions or attributes, providing fine-grained control over access to sensitive information. This approach allows data owners to define access policies that dictate who can access certain data under what circumstances, thus mitigating the risk of unauthorized access and ensuring compliance with privacy regulations. By incorporating CABE into cloud-based storage systems, organizations can bolster their data security measures while maintaining flexibility and efficiency in data sharing and access management.

[1] Introduced a key-policy attribute-based proxy re-encryption (KP-ABPRE) scheme that ensures confidentiality and integrity of shared data even in the presence of adversaries with adaptive corruption capabilities. The scheme employs proxy re-encryption techniques to allow a semi-trusted proxy to transform ciphertexts from one user's key to another's, based on specified access policies. By integrating cryptographic mechanisms that are chosen specifically to withstand adaptive corruption attacks, the proposed scheme provides a high level of security assurance for Dropbox users, ensuring that their shared data remains protected against various forms of unauthorized access and manipulation.

[2] The study explores various cryptographic methods and approaches aimed at enabling efficient and secure searching of encrypted healthcare data stored in cloud platforms. By reviewing existing research and advancements in the field, the authors provide insights into the challenges, requirements, and potential solutions for implementing searchable encryption in healthcare cloud systems. This survey serves as a valuable resource for researchers, practitioners, and stakeholders involved in healthcare data security and privacy, offering a detailed analysis of the state-of-the-art techniques and future directions in this domain.

.

[3] The scheme introduces conditional identity-based broadcast proxy re-encryption (CIBPRE), which allows a sender to encrypt messages with specific access conditions based on recipients' identities. This enables selective sharing of encrypted emails, enhancing privacy

and access control. By leveraging identity-based cryptography and proxy re-encryption techniques, the scheme ensures efficient and scalable encryption and decryption operations. The authors demonstrate the practicality and effectiveness of the proposed approach by applying it to cloud-based email systems, addressing the need for secure and flexible communication in cloud environments. The paper contributes to the advancement of secure email communication in the cloud by introducing a novel cryptographic solution tailored to the unique requirements and challenges of cloud-based email systems.

3. Analysis

3.1 Existing System

The existing system consists of various proposed CP-ABE (Ciphertext-Policy Attribute-Based Encryption) schemes aimed at preserving user privacy and enhancing access control in cloud-based storage environments. Nishide et al. introduced a scheme where access policies were partially hidden by dividing attributes into value and name parts, thereby preventing adversaries from obtaining user information. Waters proposed a CP-ABE scheme with dual dual-system encryption technique to preserve privacy. Lai et al. further developed hidden access policy CP-ABE schemes, achieving full security with support for different access structures. Rao et al. introduced an efficient HP-CP-ABE scheme with constant-size secret keys and ciphertext, while Zhang et al. proposed a hierarchical HP-CP-ABE scheme with fast decryption. However, privacy leakage issues were identified in certain phases of these schemes.

DISADVANTAGES OF EXISTING SYSTEM:

The drawback of the existing system is its limited scalability as the number of users increases. As cloud-based storage environments often serve large user populations, scalability is crucial for ensuring effective access control and management of user permissions. However, the existing CP-ABE schemes may struggle to accommodate a growing user base, leading to potential performance degradation and inefficiencies in access control mechanisms.

The absence of utilization of stronger encryption keys, such as 256 or 512 bits. Stronger encryption keys enhance data security and resilience against cryptographic attacks. However, the existing schemes often rely on weaker encryption keys, potentially leaving data vulnerable to unauthorized access or breaches. By failing to incorporate stronger encryption keys, the existing system may compromise the overall security posture of cloud-based storage environments, exposing sensitive data to heightened risks.

3.2 Proposed System

Our research focuses system integrating advanced encryption standards (AES) with 256-bit keys to enhance data security in cloud-based storage environments. AES-256, a widely recognized cryptographic algorithm, offers a higher level of protection by utilizing longer key lengths, making it significantly more resistant to brute-force attacks than shorter key lengths. This ensures that sensitive data stored in the cloud remains confidential and secure, mitigating the risk of unauthorized access or data breaches. In addition to AES-256 encryption, the system introduces a novel identity key verification technique for dual access control, where each user is assigned a unique identity key to authenticate download requests. Furthermore, a modified CP-ABE technique is implemented for document encryption, optimizing the management of user attributes, and enhancing system efficiency. Through the integration of AES-256 encryption and innovative access control mechanisms, the proposed system provides a robust defense against security threats while maintaining.

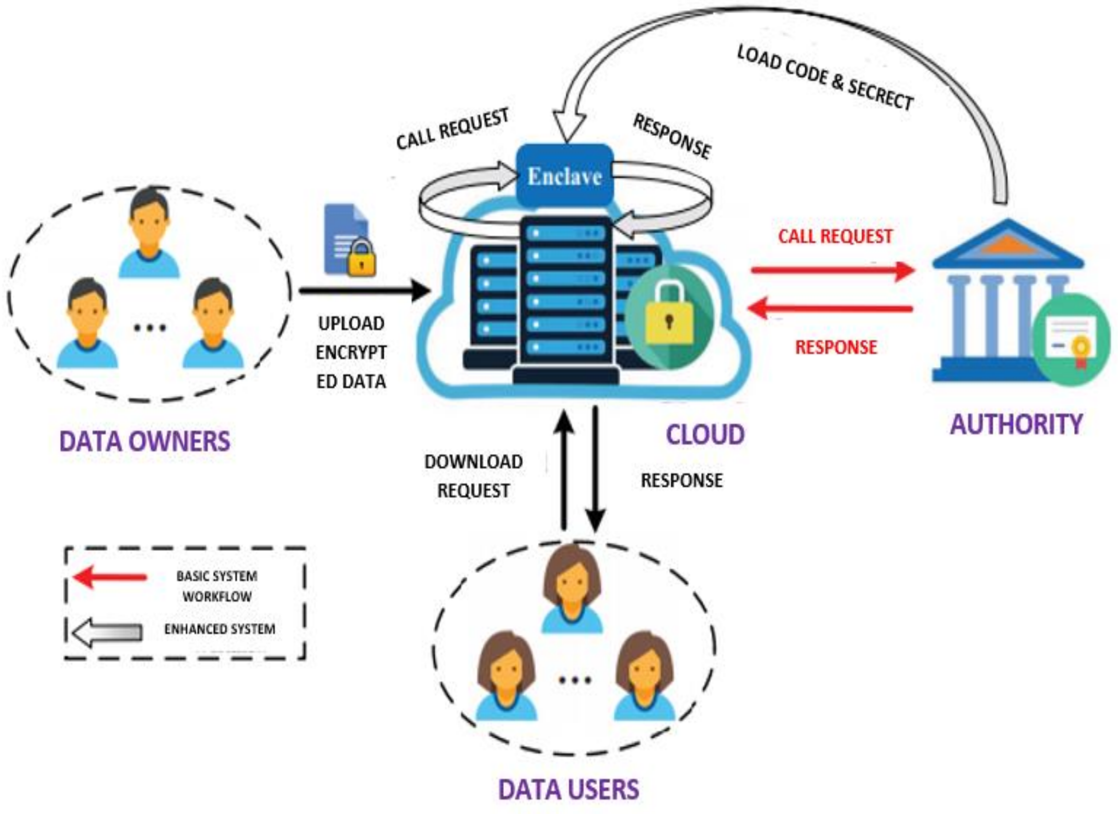


Figure 3.2 Architecture

ADVANTAGES OF PROPOSED SYSTEM:

The proposed system enhances data security and access control in cloud-based storage environments by integrating AES-256 encryption, a highly secure cryptographic algorithm, with advanced access control mechanisms. This combination ensures that sensitive data remains confidential and secure, even in the face of potential security threats or unauthorized access attempts. Additionally, the system introduces a novel identity key verification technique and a modified CP-ABE encryption scheme, which further strengthens access control measures and optimizes the management of user attributes. By leveraging these advanced techniques, the proposed system provides robust protection for sensitive data while maintaining user privacy and data integrity, making it an ideal solution for secure cloud-based storage.

3.3 Software Requirement Specification

3.3.1 Purpose

The purpose of this project is to address the critical need for enhancing data security and access control within cloud-based storage environments. With the increasing adoption of cloud services across various industries, ensuring the confidentiality, integrity, and availability of data has become paramount. The project aims to design and implement a comprehensive solution that leverages advanced encryption standards, such as AES-256, to provide robust protection for sensitive data stored in the cloud. By integrating AES-256 encryption with innovative access control mechanisms, the project seeks to mitigate the risk of unauthorized access and potential security breaches, thereby safeguarding valuable information from malicious actors. Additionally, the project aims to overcome limitations in existing access control techniques by introducing novel identity key verification techniques and modifying CP-ABE encryption schemes. These enhancements optimize attribute management and access control policies, ensuring that only authorized users with the necessary credentials can access sensitive data. Ultimately, the project endeavors to establish a secure and reliable cloud-based storage environment that upholds user privacy and data integrity, meeting the evolving security needs of organizations in today's digital landscape.

3.3.2 Scope

The scope of this project encompasses the design, development, and implementation of a comprehensive solution for enhancing data security and access control in cloud-based storage environments. This includes integrating advanced encryption standards like AES-256 with innovative access control mechanisms to provide robust protection for sensitive data. Additionally, the project aims to address limitations in existing access control techniques by introducing novel identity key verification techniques and modifying CP-ABE encryption schemes. The project will involve rigorous testing and validation procedures to ensure the effectiveness, scalability, and efficiency of the proposed solution. Furthermore, the project may explore potential extensions, such as integration with existing cloud platforms or adoption by various industries, to broaden its impact and relevance in real-world scenarios. Ultimately, the project seeks to establish a secure and reliable cloud-based storage system that meets the evolving security needs of organizations while upholding user privacy and data integrity.

3.3.3 Overall Description

H/W System Configuration:-

➢ System	:	Pentium i3 Processor
➢ Hard Disk	:	500 GB
➢ Ram	:	2 GB

Software Requirements:

➢ Operating system	:	Windows XP.
➢ Coding Language	:	JAVA
➢ Front End	:	J2EE
➢ Database	:	MYSQL

4.Design

4.1 UML DIAGRAMS

UML (Unified Modelling Language) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML comprises two major components: a Meta-model and a notation The Unified Modelling Language is used for business modelling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects Various UML Diagrams are:

- ❖ Use Case Diagram
- ❖ Class Diagram
- ❖ Activity Diagram
- ❖ Sequence Diagram
- ❖ State Chart Diagram
- ❖ Collaboration Diagram
- ❖ Component Diagram
- ❖ Deployment Diagram

4.1.1 USE CASE DIAGRAM

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

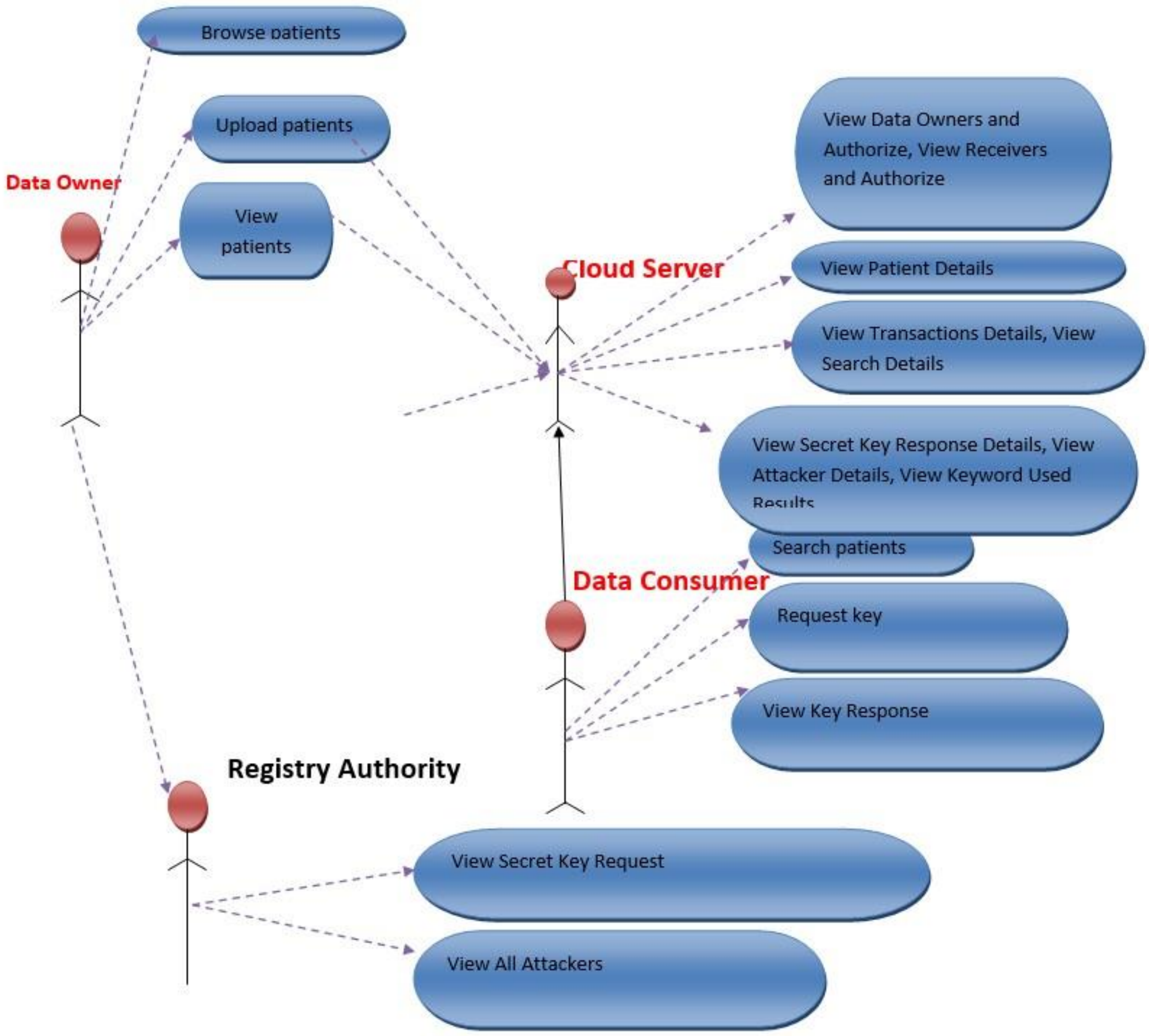


Figure 4.1.1 Use Case Diagram

4.1.2 SEQUENCE DIAGRAM

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".

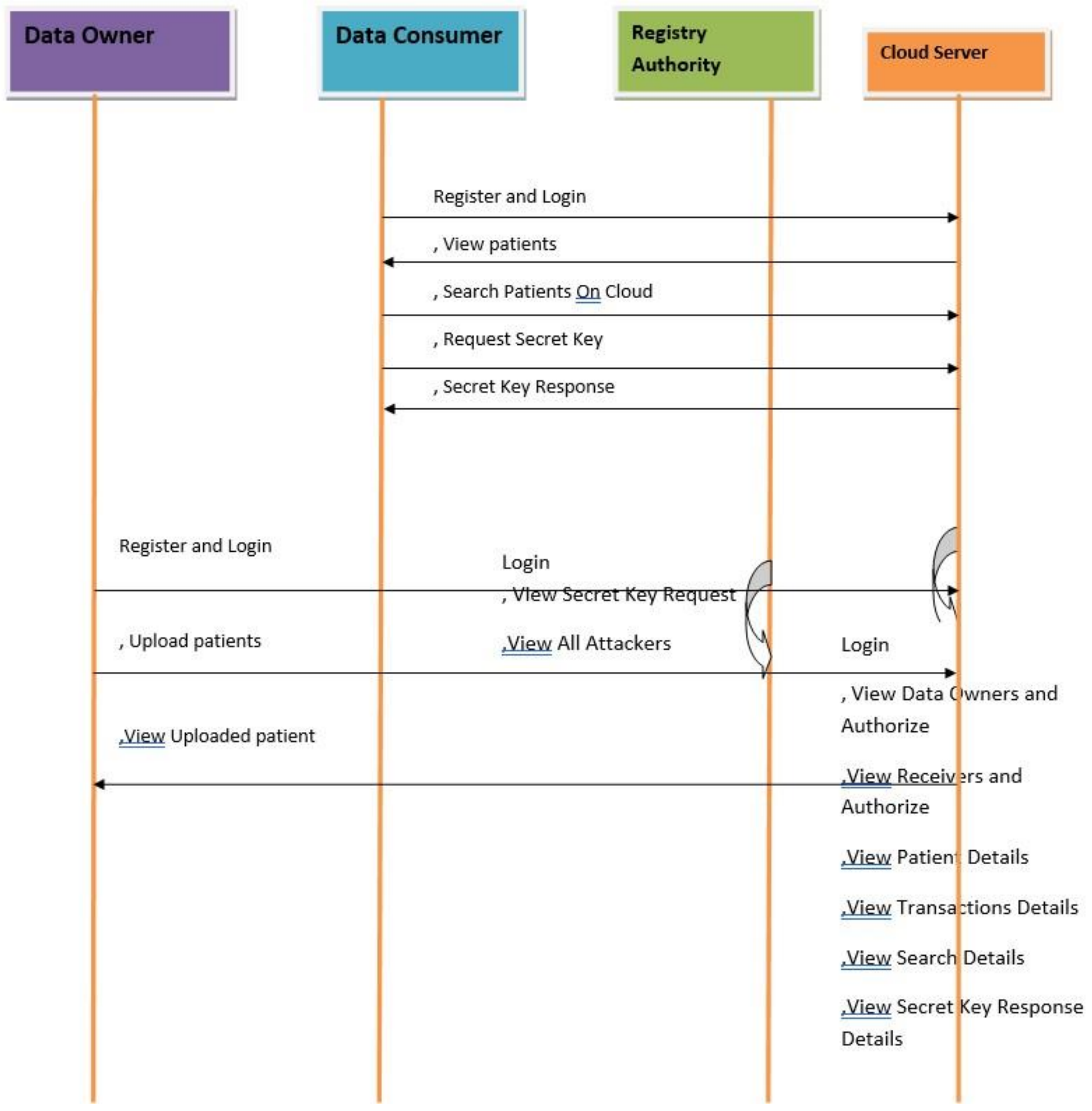


Figure 4.1.2 Sequence Diagram

5. IMPLEMENTATION

5.1 MODULES:

- Data Owner
- Data Consumer
- Registry Authority
- Cloud

5.2 MODULES DESCRIPTION:

Data Owner

As a data owner, one of the primary responsibilities is to upload files to the designated storage systems and ensure their security through encryption. This involves selecting appropriate encryption algorithms and techniques to protect the confidentiality and integrity of the data. Upon uploading files, the data owner initiates the encryption process, which involves converting the plaintext data into ciphertext using encryption keys. By encrypting the files, the data owner mitigates the risk of unauthorized access and data breaches, thereby safeguarding sensitive information from potential threats. Additionally, the data owner may establish access controls and permissions to regulate who can decrypt and access the encrypted files, further enhancing data security and privacy. Overall, uploading and encrypting files are critical tasks performed by the data owner to protect the organization's data assets and ensure compliance with security standards and regulations.

Data Consumer

As a data consumer, the primary task involves decrypting encrypted files to access the underlying data. Once the encrypted files are obtained from the storage systems, the data consumer initiates the decryption process using the appropriate decryption keys or credentials. Decrypting the files reverses the encryption process, converting the ciphertext back into its original plaintext format. With access to the decrypted data, the data consumer can then analyse, interpret, and utilize the information for decision-making, reporting, or other operational purposes. It is essential for data consumers to adhere to access controls and permissions established by the data owner to ensure the secure and authorized access of data assets.

Registry Authority

The registry authority serves as a centralized entity responsible for managing and overseeing the security of data assets within an organization. Its primary role involves maintaining a registry or database that records details of potential security breaches, including information about attackers and compromised files. This registry acts as a repository of security incidents, providing insights into unauthorized access attempts, breaches, and other security threats. Additionally, the registry authority plays a crucial role in facilitating secure data access by providing decryption keys to authorized data consumers. These decryption keys enable data consumers to decrypt encrypted files, allowing them to access and utilize the data for various purposes. By managing security incidents and providing decryption keys, the registry authority helps to ensure the integrity, confidentiality, and availability of data assets while mitigating the risks associated with unauthorized access and breaches.

Cloud

A cloud service provider (CSP) is a company that offers cloud computing services and infrastructure to businesses and individuals. These services typically include computing resources, storage, networking, and software applications delivered over the internet on a pay-as-you-go basis. CSPs manage and maintain the underlying hardware, software, and infrastructure required to deliver these services, allowing customers to access computing resources and scale their operations without the need for significant upfront investment in physical infrastructure. With a wide range of service offerings, CSPs enable organizations to streamline operations, increase efficiency, and innovate rapidly by leveraging the flexibility and scalability of cloud

Front End and User Interface Design:

The user interface, a critical component, is designed within a browser-specific environment with an emphasis on an Intranet-Based Architecture for distributed concepts. HTML standards lay the foundation for browser-specific components, while Java Server Pages (JSP) add dynamism to the design. Communication architecture hinges on Servlets and Enterprise Java Beans, with Java Database Connectivity (JDBC) acting as the bridge for database connectivity. The three-tier architecture prioritizes higher cohesion and limited coupling for operational effectiveness.

Java Language Choice:

Java, selected as the language for the project, stands out due to its platform independence, cohesiveness, and consistency. Originally named "Oak," Java emerged in 1995 with a primary motivation for a platform-independent, architecture-neutral language suitable for embedded software in various consumer electronic devices. Java offers full control to programmers within the constraints of the Internet environment. Its significance in Internet programming is likened to C's role in system programming.

Java's Impact on the Internet:

Java has left an indelible mark on the Internet, expanding the universe of objects that can move freely in cyberspace. In networked environments, two categories of objects—passive information and dynamic active programs—are transmitted between servers and personal computers. Java applets, small programs dynamically downloaded across the network, introduced a new form of executable programs in cyberspace. Java addresses security concerns associated with downloading executable programs, ushering in a new era of program development known as Applets.

Java Architecture:

The architecture of Java provides a portable, robust, and high-performance environment for development. Java's uniqueness lies in its bytecode, an optimized set of instructions executed by the Java Virtual Machine (JVM). The JVM, integral to Java technology, acts as an interpreter for bytecode, ensuring portability across various platforms. The development process involves writing Java source code, compilation into bytecode using the Java compiler (javac), and execution in the JVM.

Java Database Connectivity (JDBC):

JDBC, short for Java Database Connectivity, represents a critical Java API designed for executing SQL statements. More than just a set of classes and interfaces, JDBC provides a standard API for tool and database developers, enabling the creation of database applications using pure Java. The three primary functions of JDBC include establishing a connection with a database, sending SQL statements, and processing the results.

Two-Tier and Three-Tier Models:

JDBC supports both the two-tier and three-tier models for database access. In the two tier model, a Java applet or application communicates directly with the database. This requires a JDBC driver capable of interfacing with the specific database management system. The three-tier model introduces a middle tier, where commands are sent to services that communicate with the database. The three-tier architecture is favored for its enhanced control over access and updates to corporate data, offering performance advantages and a simplified API.

JDBC Driver Types:

Several JDBC driver types exist, categorized into four groups: JDBC-ODBC bridge plus ODBC driver, Native-API partly-Java driver, JDBC-Net pure Java driver, and Native protocol pure Java driver. Each type caters to specific requirements, with considerations for security, robustness, and ease of use. The choice of JDBC driver depends on the project's needs and the target database.

Java Server Pages (JSP):

Java Server Pages, a technology for creating dynamic-content web pages, emerges as a simple yet powerful tool based on the Java programming language. This architecture facilitates the separation of content generation from presentation, allowing different team members to focus on their expertise. Java Server Pages offer proven portability, open standards, and a mature re-usable component model, making them a preferred choice for building dynamic web applications.

Tomcat 6.0 web server

Tomcat is an open source web server developed by Apache Group. Apache Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and Java Server Pages technologies. The Java Servlet and Java Server Pages specifications are developed by Sun under the Java Community Process. Web Servers like Apache Tomcat support only web components while an application server supports web components as well as business components (BEAs Web logic, is one of the popular application server).

5.3 List of program files

Dbconnection:

This Java code defines a class called `SQLconnection` inside the `DBconnection` package, which is responsible for establishing a connection to a MySQL database. Overall, this class provides a static method `getconnection()` that can be called to obtain a connection to the specified MySQL database. However, note that this code should ideally handle exceptions more gracefully, such as by logging them or propagating them to the caller for handling. Additionally, it's recommended to close the database connection when it's no longer needed to prevent resource leaks.

Cloud.java:

Overall, this servlet handles user authentication for a cloud application by checking the provided username and password against expected values and redirecting the user accordingly. However, it should ideally handle exceptions more gracefully and provide appropriate error messages or logging.

Decription.java:

Overall, this class provides a method `decrypt()` to decrypt text using the AES algorithm with a provided secret key. It's worth noting that using `sun.misc.BASE64Decoder` and `sun.misc.BASE64Encoder` is not recommended because they are internal APIs and might not be available on all Java implementations. It's better to use standard libraries like `java.util.Base64`. Additionally, handling exceptions more gracefully and securely managing the secret key would improve the robustness and security of this code.

Download.java:

Overall, this servlet facilitates file downloads by retrieving encrypted files from the database, decrypting them, and serving them to users for download. It also records download information in the database for auditing purposes.

Web.Xml:

Overall, this `web.xml` file provides essential configuration information for servlet mappings and session management in the Java web application. It specifies which servlets handle which URL patterns and sets session timeout values.

5.4 List of Libraries

Java Database Connectivity (JDBC):

JDBC is used to connect to a MySQL database. It provides a standard interface for Java applications to interact with relational databases.

Apache Tomcat:

Tomcat is a servlet container and web server that implements the Java Servlet and JavaServer Pages (JSP) technologies. It provides a runtime environment for Java web applications.

Java Servlet API:

The Java Servlet API provides the means to define HTTP-specific servlet classes. Servlets are used to extend the capabilities of servers, such as Tomcat, to respond to network requests.

6.EXPERIMENTAL RESULTS

6.1 Experiment setup

Setting up the experimental environment for a project involves preparing the necessary software, databases, and dependencies.

Prerequisites:

Java Development Kit (JDK):

Install the latest version of JDK on the server where your web application will run. You can download it from the official Oracle or OpenJDK website.

Apache Tomcat:

Download and install Apache Tomcat, which will serve as the servlet container for your web application. Configure Tomcat to work with the installed JDK.

MySQL Database:

Install and set up MySQL database server. Create a database named 'EHR' and a user with appropriate privileges. Update the database connection details in the project's JDBC code.

Application Files:

Deploy your web application files (JSP pages, HTML, CSS, JavaScript) into the appropriate directories in the Tomcat web apps folder.

Configuration:

Database Connection Configuration:

Update the database connection details in the JSP files where JDBC connections are established. This includes the URL, username, and password.

Tomcat Configuration:

Adjust Tomcat configuration files if necessary. Specifically, you may need to configure the context for your web application in the server.xml file.

Security Configuration:

Implement necessary security measures such as HTTPS, user authentication, and authorization based on the requirements of your project.

Running the Experiment:

Start MySQL Database:

Ensure that the MySQL database server is running.

Deploy Web Application:

Start the Tomcat server and deploy your web application by placing the WAR file or the project directory in the web apps folder.

Access the Application:

Open a web browser and navigate to the URL where Tomcat is running (e.g., http://localhost:8090/DataTrustFramework/). Access the different roles in the application (Owner, End User, Cloud Service Provider) and perform actions as specified in your project.

Monitoring and Logging:

Implement logging mechanisms to capture relevant events and errors. Monitor Tomcat logs, database logs, and application-specific logs to track the behavior of the system.

Performance Testing (Optional):

If performance evaluation is part of your experiment, consider using tools like Apache JMeter to simulate concurrent users and analyze the system's performance under different loads.

Security Auditing:

Perform security audits using tools like AES or other security scanners to identify vulnerabilities in your application.

6.2 Parameters with Formulas

The Advanced Encryption Standard (AES) is a symmetric key encryption algorithm that is widely used to secure electronic data. It was established as a standard by the U.S. National Institute of Standards and Technology (NIST) in 2001. AES operates on blocks of data, each consisting of 128 bits, and uses keys of 128, 192, or 256 bits in length.

Key Expansion:

KeyExpansion(Key, RoundKeys) // Input: Key (128, 192, or 256 bits),

Output: RoundKeys (derived from Key)

RoundKeys[0] = Key

For i from 1 to Nr (Nr is the total number of rounds):

RoundKeys[i] = KeySchedule(RoundKeys[i-1], i)

Initial Round:

AddRoundKey(State, RoundKeys[0])

Rounds:

For round = 1 to Nr-1:

SubBytes(State)

ShiftRows(State)

MixColumns(State)

AddRoundKey(State, RoundKeys[round])

Final Round:

SubBytes(State)

ShiftRows(State)

AddRoundKey(State, RoundKeys[Nr])

Where the operations are defined as:

SubBytes(State):

Replace each byte of the State matrix with the corresponding byte from a fixed substitution table, S-box.

State[i][j] = S-box[State[i][j]]

ShiftRows(State):

Shift each row of the State matrix cyclically to the left.

Row i is shifted left by i bytes.

This operation ensures that each byte of the input affects multiple bytes of the output.

MixColumns(State):

Treat each column of the State matrix as a polynomial over the finite field GF(2^8).

Multiply each column with a fixed polynomial modulo an irreducible polynomial.

This operation provides diffusion throughout the block.

AddRoundKey(State, RoundKey):

Perform a bitwise XOR operation between each byte of the State matrix and the corresponding byte of the RoundKey. The encryption process takes the plaintext as input and generates the ciphertext, while the decryption process takes the ciphertext as input and generates the plaintext. The decryption process is essentially the same as encryption, but with the key schedule used in reverse order, and the MixColumns operation omitted in the final round.

6.3 Sample Code

SQL Connection:

```
Package DBconnection;
import java.sql.Connection;
import java.sql.DriverManager;
public class SQLconnection
{
static Connection con;
public static Connection getconnection()
{
      try {

         Class.forName("com.mysql.jdbc.Driver");
         con = DriverManager.getConnection("jdbc:mysql://localhost:3306/ehr", "root",
         "root");
         }
         catch (Exception e) {
      }

      return con;

   }

}
```

Cloud.java:

```
package EHR;
import java.io.IOException;

import java.io.PrintWriter;
 import javax.servlet.ServletException;
 import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
 import javax.servlet.http.HttpServletResponse;
 public class Cloud extends HttpServlet
 {
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
        String name = request.getParameter("name");

        String pass = request.getParameter("pass");

        System.out.println("========================================"    +name
+pass);

        if    (name.equalsIgnoreCase("cloud")    &&    pass.equalsIgnoreCase("cloud"))    {
response.sendRedirect("Cloud_Home.jsp?Success");
        } else {

            response.sendRedirect("Cloud_login.jsp?Failed");

        }

      } catch (Exception ex) {

      }

   }
```

```
    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on
the left to edit the code.">

    @Override                  protected    void    doGet(HttpServletRequest    request,
HttpServletResponse response)             throws ServletException, IOException {

        processRequest(request, response);

    }


    @Override                  protected    void    doPost(HttpServletRequest    request,
HttpServletResponse response)                throws ServletException, IOException {
processRequest(request, response);
    }

    @Override     public String
getServletInfo() {         return
"Short description";
    }// </editor-fold>


}
```

Decryption.java:

```
package EHR;
import
com.sun.org.apache.xerces.internal.impl.dv.util.Base64;
import java.io.ByteArrayOutputStream;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.util.Scanner;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import
javax.crypto.spec.SecretKeySpec;
```

```
import          javax.swing.JOptionPane;
import       sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;
public class Decryption

{


public String decrypt(String txt,String skey)

{

    String decryptedtext = null;

            try

            {



            //converting   string   to   secretkey
byte[] bs=Base64.decode(skey);
            SecretKey sec=new SecretKeySpec(bs, "AES");

            System.out.println("converted string to seretkey:"+sec);



        System.out.println("secret key:"+sec);



        Cipher aesCipher = Cipher.getInstance("AES");//getting AES instance

            aesCipher.init(Cipher.ENCRYPT_MODE,sec);//initiating    ciper    encryption
using secretkey



         byte[] byteCipherText =new BASE64Decoder().decodeBuffer(txt); //encrypting data



          //  System.out.println("ciper text:"+byteCipherText);
        aesCipher.init(Cipher.DECRYPT_MODE,sec,aesCipher.getParameters());//initiating
ciper decryption

       byte[] byteDecryptedText = aesCipher.doFinal(byteCipherText);

            decryptedtext = new String(byteDecryptedText);



         System.out.println("Decrypted Text:"+decryptedtext);

        }




            catch(Exception e)

            {

                    System.out.println(e);

            }

    return decryptedtext;

}



}
```

Download.java:

```
package EHR;
import
DBconnection.SQLconnection;
import static
com.sun.org.apache.xerces.internal.xinclude.XIncludeHandler.BUFFER_SIZE;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.sql.Blob;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
public class Download extends HttpServlet
{
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html;charset=UTF-8");
try (PrintWriter out = response.getWriter())
{
        String fileid = request.getParameter("fid");

        String dkey = request.getParameter("dkey");

        System.out.println("Fileid===" + fileid);



        HttpSession user = request.getSession();

        String mid = user.getAttribute("mid").toString();

        String mname = user.getAttribute("mname").toString();

        String mmail = user.getAttribute("mmail").toString();

        String mrole = user.getAttribute("mrole").toString();
        Connection conn = SQLconnection.getconnection();

        Statement st = conn.createStatement();

        Statement     st1     =     conn.createStatement();
Statement st2 = conn.createStatement();


        ResultSet rs = st2.executeQuery(" Select * from request where fid ='" + fileid + "' AND
status='Approved' AND dkey='" + dkey + "'");

        if (rs.next()) {

          String filename = rs.getString("filename");

          ResultSet rs1 = st.executeQuery(" Select * from data_files where id ='" + fileid + "'
AND filename ='" + filename + "'  AND dkey='" + dkey + "'");

          if (rs1.next()) {

            String doid = rs1.getString("mid");

            String doname = rs1.getString("mname");

            String file = rs1.getString("enc_data");

            String file1 = rs1.getString("data");



            System.out.println("dkey--  "  +  dkey);
long aTime = System.nanoTime();
            Decryption d1 = new Decryption();
String decrypted = d1.decrypt(file, dkey);
long bTime = System.nanoTime();                  float
decryptTime = (bTime - aTime) / 1000;
System.out.print("\nPlain Text          --------  " +
file);

            System.out.print("\nDecrypted Text          --------  " + decrypted);

            System.out.println("Time  taken  to  Decrypt  File:  "  +  decryptTime  +  "
microseconds.");
```

```
                System.out.println("filename,fileid==" + filename + fileid);


                String is = decrypted;


                response.setHeader("Content-Disposition", "attachment;filename=\"" + filename
+
"\"");
out.write(file1);
out.close();
                int i = st1.executeUpdate("insert into download (mid, mname, filename, time ,
fileid , doname ,doid, decrypt_time, mrole)values('" + mid + "','" + mname + "','" + filename
+ "',now(),'" + fileid + "','" + doname + "','" + doid + "','" + decryptTime + "','"+ mrole +"')");


                if (i != 0) {

                    System.out.println("Download success...");

                } else {

                    System.out.println("error  while updating information...");

                }

            } else {

                System.out.println("file not found...");

            }

        } else {

            response.sendRedirect("Download_file.jsp?failed");

        }


    }       catch       (SQLException       ex)       {
ex.printStackTrace();
```

```
            response.sendRedirect("Download_file.jsp?download_failed");

        }       catch       (IOException       ex)       {
ex.printStackTrace();
            response.sendRedirect("Download_file.jsp?download_failed");

        }

    }



    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on
the left to edit the code.">

    @Override                protected    void    doGet(HttpServletRequest    request,
HttpServletResponse response)              throws ServletException, IOException {
processRequest(request, response);
    }

    @Override                protected    void    doPost(HttpServletRequest    request,
HttpServletResponse response)              throws ServletException, IOException {
processRequest(request, response);
    }

    @Override     public String
getServletInfo() {        return
"Short description";
    }// </editor-fold>

}
```

Encryption.java:

```
package EHR;

import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import
sun.misc.BASE64Encoder;
public class Encryption
```

```
{   public   String   encrypt(String   text,SecretKey
secretkey)
{

   String plainData=null,cipherText=null;

         try

         {

          plainData=text;




      Cipher aesCipher = Cipher.getInstance("AES");//getting AES instance

         aesCipher.init(Cipher.ENCRYPT_MODE,secretkey);//initiating    ciper    encryption
using secretkey


      byte[] byteDataToEncrypt = plainData.getBytes();

         byte[] byteCipherText = aesCipher.doFinal(byteDataToEncrypt);//encrypting data


       //  System.out.println("ciper text:"+byteCipherText);


      cipherText  =  new  BASE64Encoder().encode(byteCipherText);//converting  encrypted
data to string


      System.out.println("\n Given text : "+plainData+" \n Cipher Data : "+cipherText);


      }

         catch(Exception e)
```

```
          {
                    System.out.println(e);
          }
   return cipherText;
}
}
```

Authentication :

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@page import="java.util.*"%>

<%@ include file="connect.jsp"%>

<%@page
import="java.util.*,java.security.Key,java.util.Random,javax.crypto.Cipher,javax.crypto.spec
.SecretKeySpec,org.bouncycastle.util.encoders.Base64"%>

<%@ page
import="java.sql.*,java.util.Random,java.io.PrintStream,java.io.FileOutputStream,java.io.Fil
eInputStream,java.security.DigestInputStream,java.math.BigInteger,java.security.MessageDi
gest,java.io.BufferedInputStream"%>

<%@ page
import="java.security.Key,java.security.KeyPair,java.security.KeyPairGenerator,javax.crypto
.Cipher"%>

<%@page
import="java.util.*,java.text.SimpleDateFormat,java.util.Date,java.io.FileInputStream,java.i
o.FileOutputStream,java.io.PrintStream"%>


<%

String name = request.getParameter("userid");
String pass = request.getParameter("pass");


try {

application.setAttribute("cname", name);


String sql = "SELECT * FROM cloud where name='" + name+ "' and pass='" + pass + "' ";

Statement stmt = connection.createStatement();

ResultSet rs = stmt.executeQuery(sql);


if (rs.next()==true)

{


            response.sendRedirect("c_CloudMain.jsp");


}

 else

{


            response.sendRedirect("wronglogin.html");

}


}

catch (Exception e)

{
```

```
out.print(e);

e.printStackTrace();

}

%>
```

Data Owner:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Data Owner Main</title>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link href="css/style.css" rel="stylesheet" type="text/css" />

<link rel="stylesheet" type="text/css" href="css/coin-slider.css" />

<script type="text/javascript" src="js/cufon-yui.js"></script>

<script type="text/javascript" src="js/cufon-aller-700.js"></script>

<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>

<script type="text/javascript" src="js/script.js"></script>

<script type="text/javascript" src="js/coin-slider.min.js"></script>

<style type="text/css">

<!--

.style3 {font-size: 36px}

.style10 {font-weight: bold}

.style11 {	color: #FF0000;
font-weight: bold;

}

-->

</style>

</head>

<body>

<div class="main">

  <div class="header">

    <div class="header_resize">

      <div class="searchform"></div>

      <div class="logo">

        <h1><a href="index.html"><span class="style3">Dual Access Control for Cloud-Based
Data Storage and Sharing</span></a></h1>

      </div>

      <div class="clr"></div>

      <div class="menu_nav">

        <ul>

          <li class="active"><a href="o_DataOwnerMain.jsp"><span>Data
Owner</span></a></li>

          <li><a href="o_DataOwnerLogin.jsp">Logout</a></li>

        </ul>

      </div>

      <div class="clr"></div>

      <div class="slider">
```

```
        <div id="coin-slider">

          <div align="justify"><a href="#"><img src="images/slide1.jpg" width="960"
height="360" alt="" /></a><a href="#"><img src="images/slide2.jpg" width="960"
height="360" alt="" /></a><a href="#"><img src="images/slide3.jpg" width="960"
height="360" alt="" /></a></div>

        </div>

        </div>

      <div class="clr"></div>

    </div>

  </div>

  <div class="content">

    <div class="content_resize">

      <div class="mainbar">

<div class="article">

  <h2><span class="pages">Welcome ::<%=application.getAttribute("dname")%>
</span></h2>

  <p class="infopost">&nbsp;</p>

  <div class="clr"></div>

  <div class="img"><img src="images/img1.jpg" width="213" height="331" alt="" class="fl"
/></div>

  <div class="post_content">

    <p align="justify"><span class="style11">With the rapid development of cloud
computing, an increasing number of individuals and organizations are sharing data in the
public cloud. To protect the privacy of data stored in the cloud, a data owner usually encrypts
his data in such a way that certain designated data users can decrypt the data. This raises a
serious problem when the encrypted data needs to be shared to more people beyond those
initially designated by the data owner. To address this problem, we introduce and formalize
```

```
an identity-based encryption transformation (IBET) model by seamlessly integrating two
well-established encryption mechanisms, namely identity-based encryption (IBE) and
identity-based broadcast encryption (IBBE). In IBET, data users are identified and authorized
for data access based on their recognizable identities, which avoids complicated certificate
management in usual secure distributed systems. More importantly, IBET provides a
transformation mechanism that converts an IBE ciphertext into an IBBE ciphertext so that a
new group of users not specified during the IBE encryption can access the underlying data.
We design a concrete IBET scheme based on bilinear groups and prove its security against
powerful attacks. Thorough theoretical and experimental analyses demonstrate the high
efficiency and practicability of the proposed scheme..</span></p>


  </div>

  <div class="clr"></div>

</div>

</div>

     <div class="sidebar">

       <div class="gadget">

         <h2 class="star">Menu</h2>

         <div class="clr"></div>

         <ul class="sb_menu style10">

           <li><a href="o_DataOwnerMain.jsp">Home</a></li>

           <li><a href="o_UploadPatient.jsp">Upload Patient Details</a></li>

           <li><a href="O_View_Patient_Details.jsp">View Uploaded Patient Details</a></li>


           <li><a href="o_DataOwnerLogin.jsp">Logout</a></li>

         </ul>
```

```
        </div>
      </div>
      <div class="clr"></div>
    </div>
  </div>
  <div class="fbg"></div>
  <div class="footer"></div>
</div>
<div align=center></div>
</body>
</html>
```

7.Testcases

Test case ID	**INPUT**	**Expected Output**	**Actual Output**	**Rate**
1.	Data Owner Registration	Data Owner Registered successfully.	Data Owner Registered successfully.	success
2.	Data Owner login	Login Success	Login success.	success
3.	Data Owner Login.	Login success.	Login Fails due to invalid member details.	Failure
4.	Data Consumer Registration	Data Consumer Registered Successfully	Data Consumer Registered Successfully	success
5.	Data Consumer Login	Login Success.	Login Success.	Success
6.	Data Consumer Login	Login Success	Login Fails due to invalid member details	Failure
7.	Registry Authority login	Login Success	Login Success	Success

8.	Registry Authority Login	Login Success	Login Fails due to invalid member details	Failure

8.Screenshots

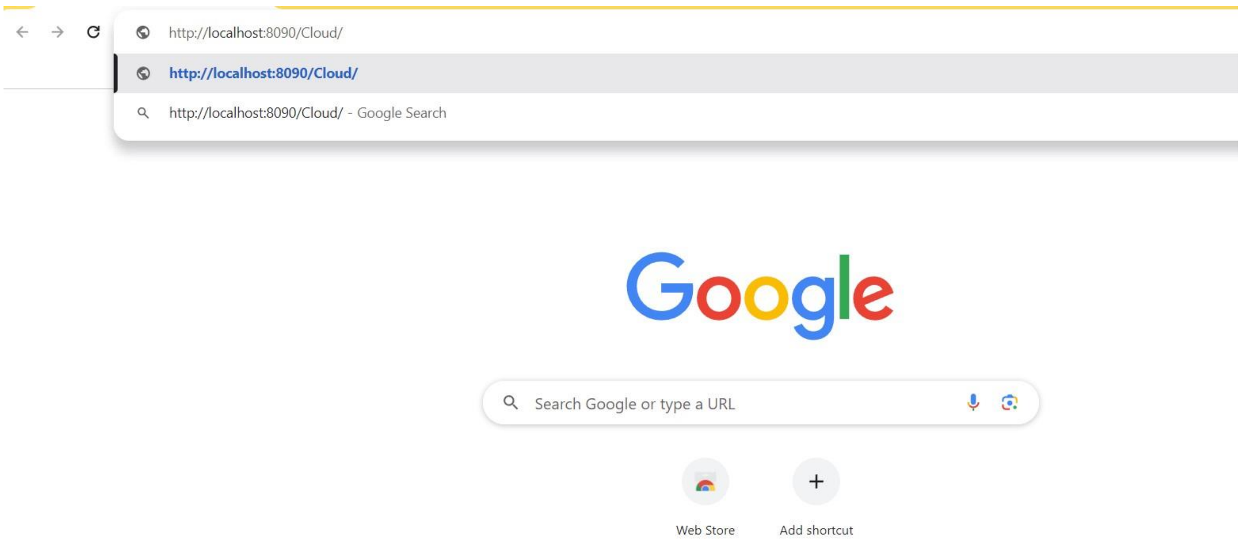
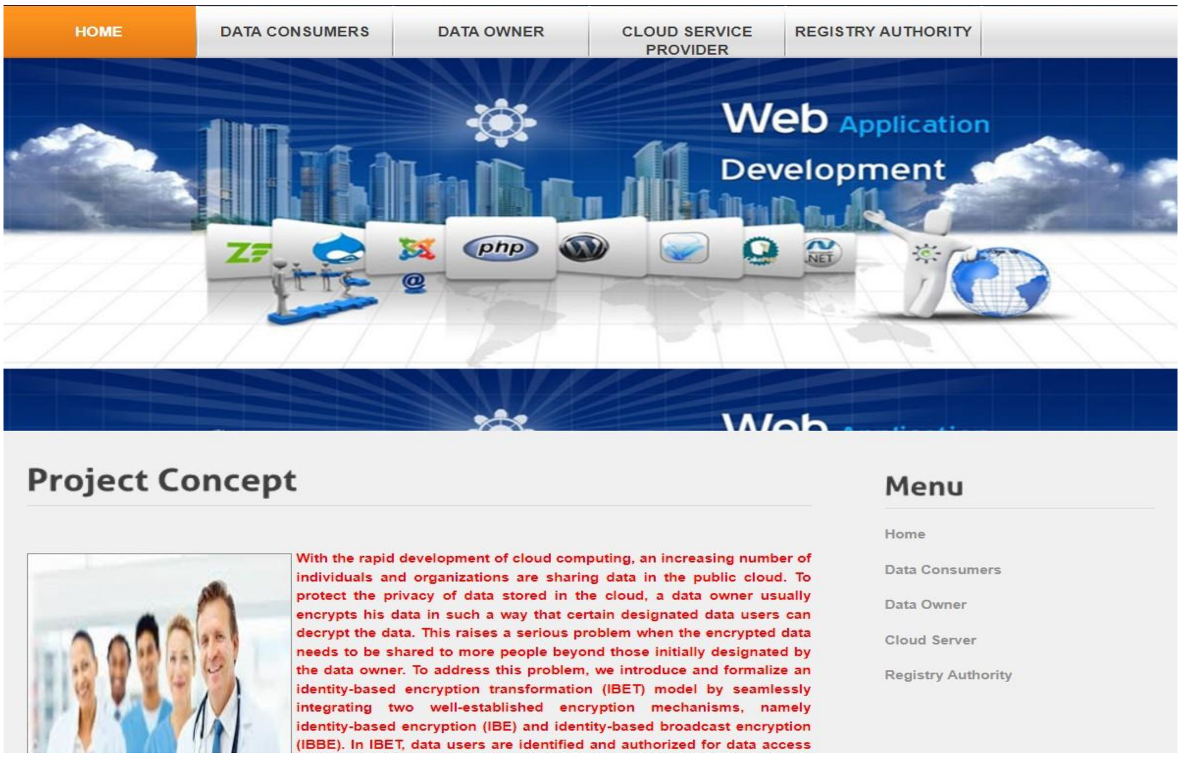

http://localhost:8090/Cloud/
http://localhost:8090/Cloud/
http://localhost:8090/Cloud/ - Google Search
Google
Search Google or type a URL
Web Store
Add shortcut

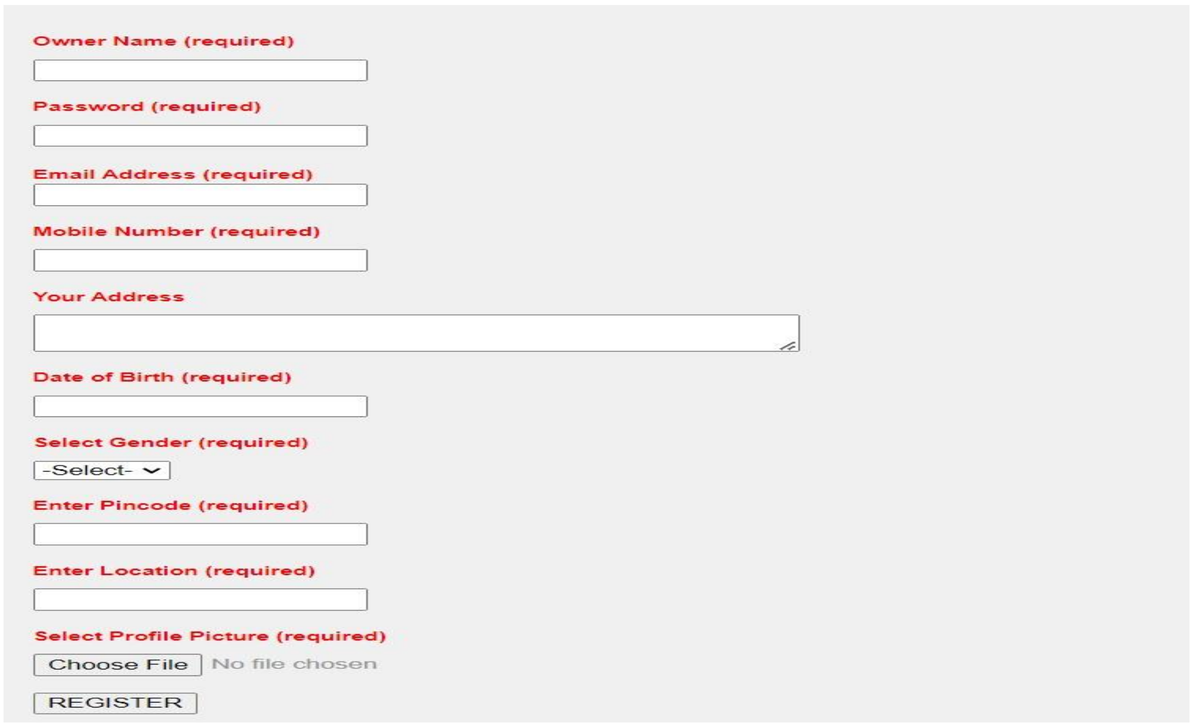
Figure 8.1 Local Host URL



Project Concept

With the rapid development of cloud computing, an increasing number of individuals and organizations are sharing data in the public cloud. To protect the privacy of data stored in the cloud, a data owner usually encrypts his data in such a way that certain designated data users can decrypt the data. This raises a serious problem when the encrypted data needs to be shared to more people beyond those initially designated by the data owner. To address this problem, we introduce and formalize an identity-based encryption transformation (IBET) model by seamlessly integrating two well-established encryption mechanisms, namely identity-based encryption (IBE) and identity-based broadcast encryption (IBBE). In IBET, data users are identified and authorized for data access

Figure 8.2 Home Page


Owner Name (required)

Password (required)

Email Address (required)

Mobile Number (required)

Your Address

Date of Birth (required)

Select Gender (required)

-Select-

Enter Pincode (required)

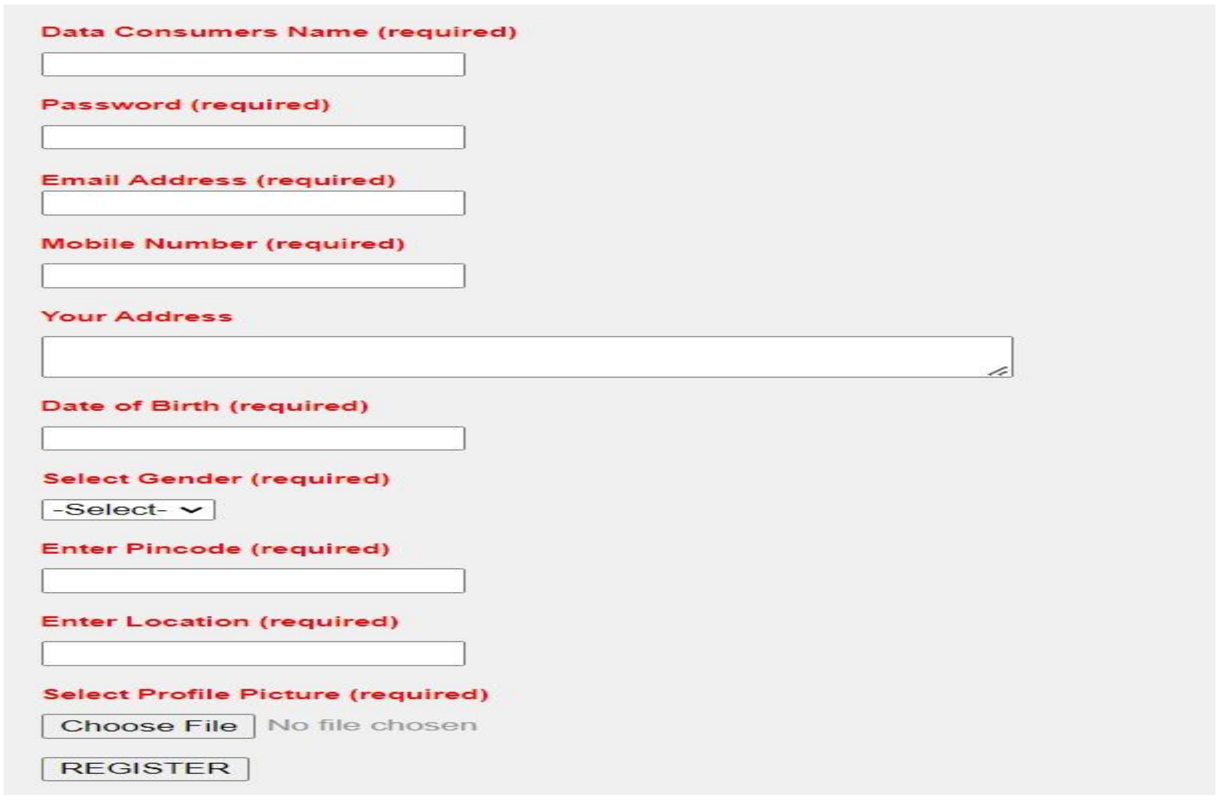
Enter Location (required)

Select Profile Picture (required)

Choose File No file chosen

REGISTER

Figure 8.3 Data Owner Registration


Data Consumers Name (required)

Password (required)

Email Address (required)

Mobile Number (required)

Your Address

Date of Birth (required)

Select Gender (required)

-Select-

Enter Pincode (required)

Enter Location (required)

Select Profile Picture (required)

Choose File No file chosen

REGISTER

Figure 8.4 Data Consumer Registration

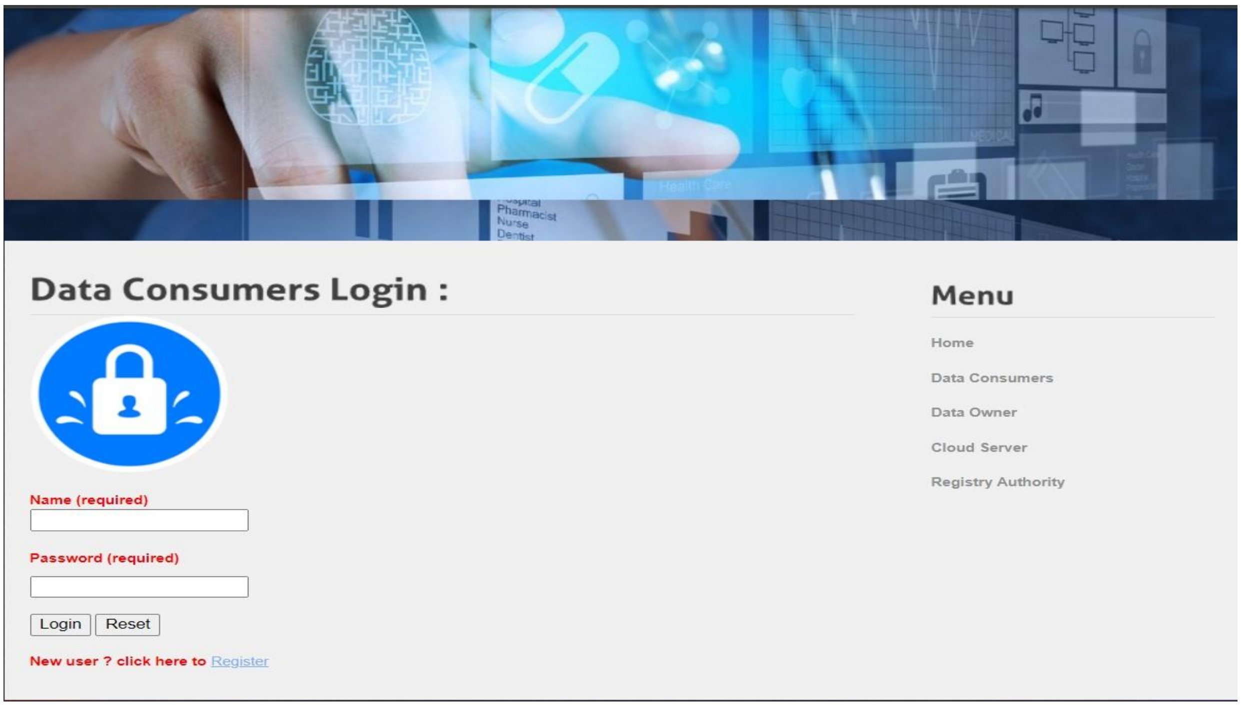


Figure 8.5 Data Consumer Login

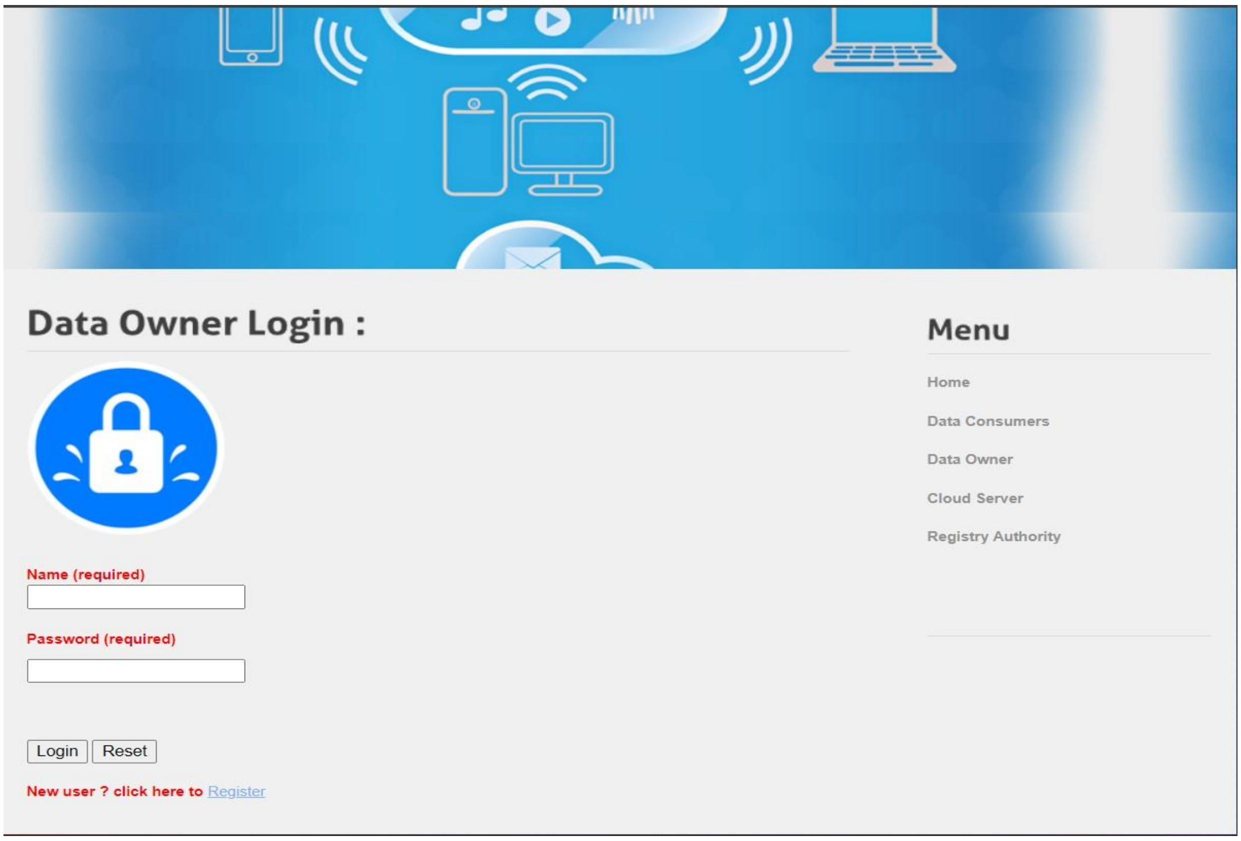


Figure 8.6 Data Owner Login

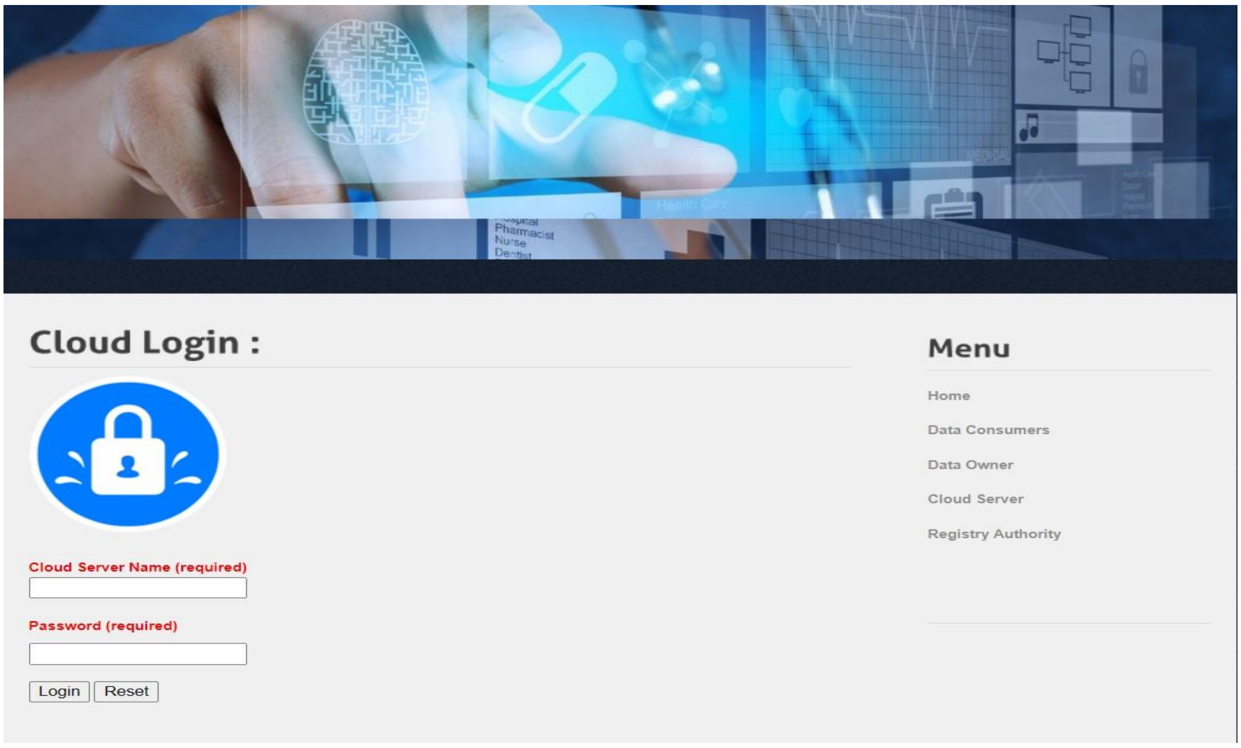


Figure 8.7 Data Cloud Login

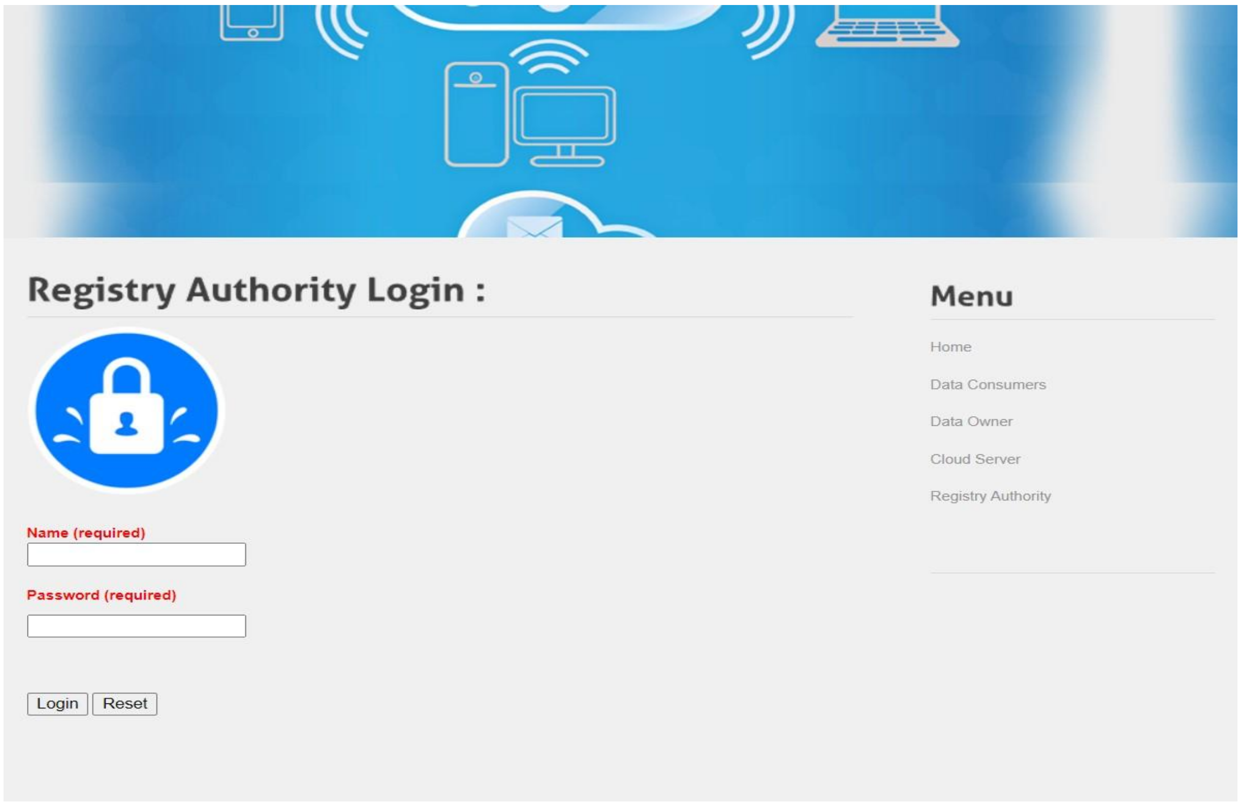
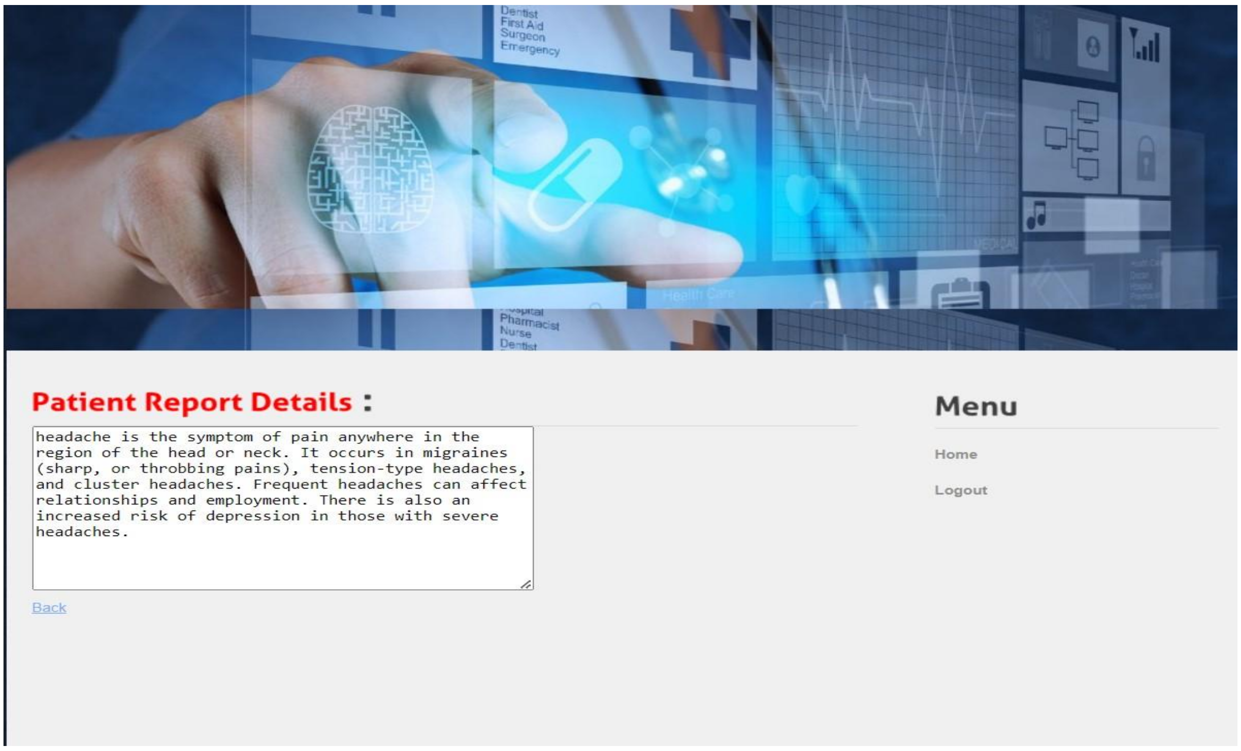


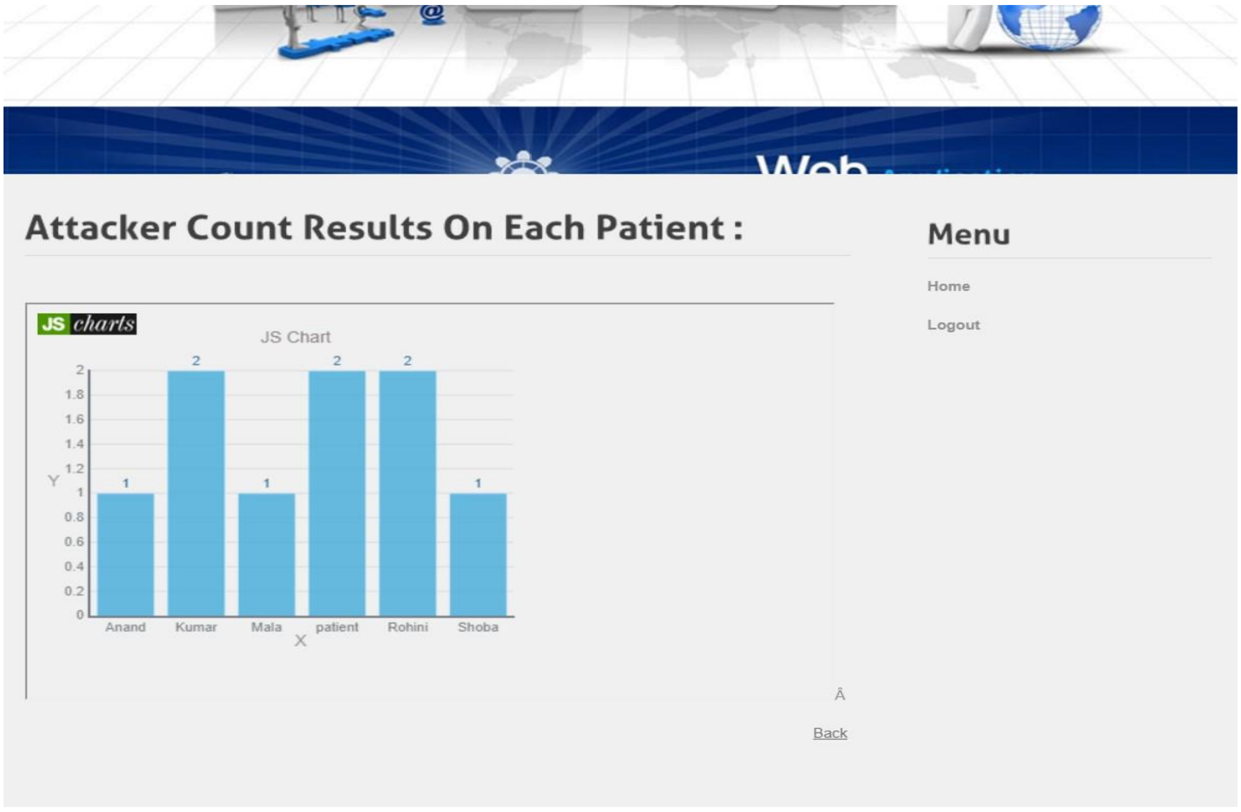
Figure 8.8 Registry Authority Login



Patient Report Details :

headache is the symptom of pain anywhere in the region of the head or neck. It occurs in migraines (sharp, or throbbing pains), tension-type headaches, and cluster headaches. Frequent headaches can affect relationships and employment. There is also an increased risk of depression in those with severe headaches.

Figure 8.9 Patient Report Details



Attacker Count Results On Each Patient :

Anand	Kumar	Mala	patient	Rohini	Shoba
1	2	1	2	2	1

Figure 8.10 Attacker Count Results

9.Conclusion

The concept of enhancing data security and access control within cloud-based storage environments addresses the imperative need to safeguard sensitive information amidst the proliferation of digital data and widespread cloud adoption. This project aimed to develop a comprehensive solution integrating advanced encryption standards like AES-256 and innovative access control mechanisms to fortify data protection. By leveraging cutting-edge technologies and industry best practices, the project sought to establish a robust framework for securing data assets stored in the cloud. Rooted in the recognition of the critical importance of data security in today's interconnected world, the project addressed the multifaceted challenges posed by unauthorized access and breaches. Through meticulous research and development efforts, the project strived to deliver a solution that not only meets stringent security standards but also aligns with organizations' evolving security needs. Ultimately, the project's outcomes will serve as a cornerstone for bolstering data security practices and safeguarding organizations' most valuable assets in an increasingly digital landscape.

10.Future Enhancement

- Multi-factor authentication (MFA), requiring users to provide multiple forms of verification before accessing sensitive data. Our scheme supports dynamic group member operations which include join and revocation.

- Advanced threat detection mechanisms, such as machine learning algorithms, can be integrated to identify and mitigate potential security threats in real time.

- Enhancing access control mechanisms to provide more granular control over user permissions and privileges can further restrict unauthorized access to sensitive data.

- Continuous monitoring and auditing tools can provide real-time visibility into data access patterns, allowing organizations to detect and respond to security incidents promptly.

- Exploring the integration of blockchain technology for transparent and immutable data transactions could enhance data integrity and trustworthiness in cloud-based storage environments.

11.BIBLIOGRAPHY

- C. Ge, W. Susilo, L. Fang, J. Wang, and Y. Shi, “A cca-secure key-policy attribute-based proxy re-encryption in the adaptive corruption model for Dropbox data sharing system,” Designs, Codes and Cryptography, pp. 1–17, 2023.
- R. Zhang, R. Xue, and L. Liu, “Searchable encryption for healthcare clouds: a survey,” Services Computing, vol. 11, no. 6, pp. 978–996, 2022.
- H. Yin, Z. Qin, J. Zhang, L. Ou, and K. Li, “Achieving secure, universal, and fine-grained query results verification for secure search scheme over encrypted cloud data,” Cloud Computing, 2022.
- H. Guo, Z. Zhang, J. Xu, N. An, and X. Lan, “Accountable proxy re-encryption for secure data sharing,” Dependable and Secure Computing, 2023.
- K. Li, W. Zhang, C. Yang, and N. Yu, “Security analysis on one-to-many order preserving encryption-based cloud data search,” Information Forensics and Security, vol. 10, no. 9, pp, 2020.