Project Report

On

Enhancing Data Security in the Cloud with Semi-Trusted Third-Party Involvement

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Bachelor of Engineering

In

Computer Science and Engineering

By

VASPARI THARUN KUMAR 1608-20-733-022

Under the guidance of

Mrs. B. Sonal Dinanath
Assistant Professor

Department of Computer Science and Engineering
Matrusri Engineering College
Accredited by NBA & NAAC
Affiliated to Osmania University, Approved by AICTE
Saidabad, Hyderabad-500059
2023-2024

Department of Computer Science and Engineering

Matrusri Engineering College

Accredited by NBA & NAAC

(Affiliated to Osmania University, Approved by AICTE)
Saidabad, Hyderabad-500059
2023-2024

CERTIFICATE

This is to Certify that a Project report entitled **"Enhancing Data Security in the Cloud with Semi-Trusted Third-Party Involvement"** is being submitted by Vaspari Tharun Kumar **(**1608-20-733-022**)** in partial fulfillment of the requirement of the award for the degree of Bachelor of Engineering in "Computer Science and Engineering" O.U., Hyderabad during the year 2023-2024 is a record of bonafide work carried out by him under my guidance. The results presented in this thesis have been verified and are found to be satisfactory.

Project Guide	**H.O.D.**
Mrs. B. Sonal Dinanath	**Dr. P. Vijayapal Reddy**
Assistant Professor,	**Professor & Head,**
Dept of CSE	**Dept. of CSE**

External Examiner(s)

Department of Computer Science and Engineering

Matrusri Engineering College

Accredited by NBA & NAAC

(Affiliated to Osmania University, Approved by AICTE)
Saidabad, Hyderabad-500059
2023-2024

DECLARATION

I **Mr. Vaspari Tharun Kumar (1608-20-733-022)** hereby certify that the major project entitled **"Enhancing Data Security in the Cloud with Semi-Trusted Third-Party Involvement"** is submitted in the partial fulfilment of the required for the award of the degree of **Bachelor of Engineering** in **Computer Science and Engineering**.

This is a record work carried out by me under the guidance of **Mrs. B Sonal Dinanath**, Assistant Professor, CSE, Matrusri Engineering College, Saidabad. The results embodied in this report have not been reproduced/copied from any source. The results embodied in this report have not been submitted to any other university or institute for the award of any other degree or diploma.

V. THARUN KUMAR 1608-20-733-022

ACKNOWLEDGEMENT

It is our privilege and pleasure to express my profound sense of respect, gratitude and indebtedness to our guide Mrs. B Sonal Dinanath, Assistant Professor, Department of Computer Science and Engineering, Matrusri Engineering College, for her inspiration, guidance, cogent discussion, constructive criticisms and encouragement throughout this dissertation work.

We express our sincere thanks to Project Coordinator Mrs. B. J. Praveena, Assistant Professor, Department of Computer Science and Engineering, Matrusri Engineering College, for her valuable suggestions and constant helping in completing the work.

We express our sincere gratitude to Dr. P. Vijayapal Reddy, Professor & Head of the Department of Computer Science and Engineering, Matrusri Engineering College, for his precious suggestions, motivation and co-operation.

We express our sincere thanks to Dr. D. Hanumantha Rao, Principal, Matrusri Engineering College, Saidabad, Hyderabad, for his encouragement and constant help.

We extend our sincere thanks to all the teaching and non-teaching staff of Computer Science and Engineering Department for their support and encouragement.

Last but not least, we wish to acknowledge my friends and family members for giving moral strength and helping us to complete this dissertation.

ABSTRACT

This paper proposes a novel privacy-preserving mechanism that supports public auditing on shared data stored in the cloud. In particular, we exploit ring signatures to compute verification metadata needed to audit the correctness of shared data. With our mechanism, the identity of the signer on each block in shared data is kept private from public verifiers, who are able to efficiently verify shared data integrity without retrieving the entire file. In addition, our mechanism is able to perform multiple auditing tasks simultaneously instead of verifying them one by one.

The proposed system is a privacy-preserving public auditing mechanism for shared data in the cloud. We utilize ring signatures to construct homomorphism authenticators so that a public verifier is able to audit shared data integrity without retrieving the entire data. However, it cannot distinguish who the signer on each block is. We further extend our mechanism to support batch auditing to improve the efficiency of verifying multiple auditing tasks. There are two exciting problems we will continue to study for our future work. One of them is traceability, which means the group manager's ability to reveal the signer's identity based on verification metadata in some special situations.

AES is an unvarying associate degree, unlike a Feistel cipher. It has supported the 'substitution–permutation network.' It contains a series of joined operations, a number of which involve exchange inputs by specific outputs, and others involve shuffling bits around.

Interestingly, AES performs all its computations on bytes instead of bits. Hence, AES treats the 128 bits of a plaintext block as sixteen bytes. These sixteen bytes square measures are organized in four columns and four rows for processing as a matrix.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1. INTRODUCTION

1.1 PROBLEM STATEMENT

The advent of cloud computing has revolutionized data storage and sharing, enabling individuals and organizations to leverage scalable and cost-effective solutions. However, the outsourcing of data to the cloud introduces concerns about privacy and integrity, particularly when multiple parties collaborate on shared data. In response to these challenges, this report delves into a groundbreaking privacy-preserving mechanism designed for public auditing of shared data stored in the cloud. The proposed system employs innovative techniques, notably ring signatures, to compute verification metadata crucial for auditing the correctness and integrity of shared data.

The primary objective of this report is to explore and elucidate the intricacies of the privacy-preserving public auditing mechanism, emphasizing its capability to protect the identity of data signers while allowing public verifiers to efficiently audit shared data integrity without the need to retrieve the entire dataset. The mechanism's use of homomorphism authenticators, constructed from ring signatures, offers a unique approach where public verifiers can assess data integrity without distinguishing individual signers on each data block.

Furthermore, the report examines the system's extension to support batch auditing, enhancing its efficiency by simultaneously verifying multiple auditing tasks. As we delve into the technical aspects of the proposed mechanism, the report also sheds light on the underlying principles of the Advanced Encryption Standard (AES), highlighting its unconventional use of bytes instead of bits and its matrix-based computation approach.

In addition to presenting the current state of the proposed privacy-preserving public auditing mechanism, the report outlines potential avenues for future research. Notably, the exploration of traceability, wherein the group manager may reveal the identity of the signer under specific circumstances based on verification metadata, represents a key area of interest for future study. By addressing these issues, the report aims to contribute to the evolving landscape of secure data sharing in cloud environments, providing insights that may shape the development of future privacy-preserving mechanisms.

1.2 MOTIVATION

Cloud computing has become the linchpin of modern data storage and sharing, facilitating unprecedented levels of accessibility and collaboration. However, as the prevalence of cloud services continues to rise, so do concerns regarding the security and privacy of shared data. Traditional methods of ensuring data correctness, involving the retrieval of entire datasets for verification, have proven cumbersome and inefficient, particularly with the growing volumes of data hosted in the cloud. Moreover, the conventional approach exposes cryptographic signatures to public verifiers, raising critical issues related to the privacy of individual identities associated with data blocks.

Recent studies underscore a broader transformation in cloud computing, characterized by a shift towards the Internet of services. This evolution highlights the interconnectivity of diverse online services and applications, bringing forth new challenges in ensuring the security and privacy of data in cloud environments. With this backdrop, security and privacy have emerged as pivotal concerns in the cloud services ecosystem, necessitating innovative mechanisms that not only preserve data integrity but also mitigate privacy risks associated with conventional approaches.

Motivated by the limitations and privacy issues inherent in existing systems, there is a compelling need for advanced privacy-preserving solutions. The proposed mechanism addresses these challenges by introducing a novel approach that leverages ring signatures and homomorphism authenticators. This approach not only enhances the efficiency of public auditing on shared data in the cloud but also ensures that the identity of the signer for each data block remains confidential. The background sets the stage for a comprehensive exploration of the proposed privacy-preserving public auditing mechanism and its potential to redefine security practices in cloud-based data storage and sharing.

1.3 OBJECTIVES

1. Develop an Advanced Privacy-Preserving Mechanism: Create a state-of-the-art privacy-preserving mechanism that employs cutting-edge cryptographic techniques to ensure the security and privacy of shared data in cloud environments.

2. Efficient Public Auditing: Streamline the public auditing process for shared cloud data, enhancing efficiency by enabling verifiers to assess data integrity without the need to retrieve the entire file.

3. Support Simultaneous and Batch Auditing: Extend the mechanism to perform multiple auditing tasks simultaneously and introduce batch auditing capabilities, optimizing the system for handling diverse auditing scenarios concurrently.

4. Implement Traceability Features: Integrate traceability features that allow the group manager to selectively reveal the identity of signers based on verification metadata in specific situations, balancing the need for transparency with overall user data privacy.

5. Enhance Security for Confidential Data Sharing: Elevate the security measures for file sharing within the proposed mechanism, ensuring robust protection against unauthorized access and data breaches while preserving the privacy of shared information.

6. User-Centric Registration and Key Distribution: Develop a user-friendly registration process and efficient key distribution mechanism, ensuring seamless onboarding for users and simplifying the acquisition of private keys necessary for group signature generation and file decryption.

7. Scalable System Architecture: Optimize the system architecture to ensure scalability, aligning hardware and software specifications to accommodate varying workloads and large datasets, thereby enhancing overall system performance.

8. Data Dynamics and Public Risk Auditing Support: Extend the mechanism to support dynamic data scenarios, including block-level operations such as modification, deletion, and insertion, ensuring the adaptability and robustness of the system in addressing evolving data challenges.

2. LITERATURE SURVEY

2.1 Relevant Studies:

[1]. “Enabling cloud storage auditing with key-exposure resistance” by Jia Yu, Kui Ren, Cong Wang.

Cloud storage auditing is viewed as an important service to verify the integrity of the data in public cloud. Current auditing protocols are all based on the assumption that the client's secret key for auditing is absolutely secure. However, such assumption may not always be held, due to the possibly weak sense of security and/or low security settings at the client. If such a secret key for auditing is exposed, most of the current auditing protocols would inevitably become unable to work. In this paper, we focus on this new aspect of cloud storage auditing. We investigate how to reduce the damage of the client's key exposure in cloud storage auditing, and give the first practical solution for this new problem setting. We formalize the definition and the security model of auditing protocol with key-exposure resilience and propose such a protocol. In our design, we employ the binary tree structure and the preorder traversal technique to update the secret keys for the client. We also develop a novel authenticator construction to support the forward security and the property of blockless verifiability. The security proof and the performance analysis show that our proposed protocol is secure and efficient.

[2]. “Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing” by Cong Wang, Qian Wang

Cloud Computing is the long-dreamed vision of computing as a utility, where users can remotely store their data into the cloud so as to enjoy the on-demand high quality applications and services from a shared pool of configurable computing resources. By data outsourcing, users can be relieved from the burden of local data storage and maintenance. However, the fact that users no longer have physical possession of the possibly large size of outsourced data makes the data integrity protection in Cloud Computing a very challenging and potentially formidable task, especially for users with constrained computing resources and capabilities. Thus, enabling public auditability for cloud data storage security is of critical importance so that users can resort to an external audit party to check the integrity of outsourced data when needed. To securely introduce an effective third-party auditor (TPA), the following two fundamental requirements have to be met: 1) TPA should be able to efficiently audit the cloud data storage without demanding the local copy of data, and introduce no additional on-line burden to the cloud user; 2) The third-party auditing process should bring in no new vulnerabilities towards user data privacy. In this paper, we utilize the public key based homomorphic authenticator and uniquely integrate it with random mask technique to achieve a privacy-preserving public auditing system for cloud data storage security while keeping all above requirements in mind. To support efficient handling of multiple auditing tasks, we further explore the technique of bilinear aggregate signature to extend our main result into a multi-user setting, where TPA can perform multiple auditing tasks simultaneously. Extensive security and performance analysis shows the proposed schemes are provably secure and highly efficient.

[3]. "Lattice-based public-key encryption with conjunctive keyword search in multi-user setting for IIoT" by Yongli Tang, Yanpeng Ba.

Most traditional Public-Key Encryption with keyword Search (PEKS) schemes are suffering a tremendous threat occasioned by the burgeoning of quantum computing since these schemes are derived from the bilinear pairing. For the sake of preserving the security of data outsourced by the Industrial Internet of Things (IIoT), a novel efficient PEKS scheme based on lattice assumption is proffered, and it can achieve security against quantum computing attacks. Also, it supports both multi-user and conjunctive keyword search. Besides, we adopt broadcast encryption technology to address the enormous storage cost of keywords ciphertext in a multi-user setting. Our scheme only needs to generate one ciphertext for all data users, thus significantly reducing the storage cost. Lastly, its performance is analyzed theoretically and experimentally. Experimental simulation results demonstrated the superiority of our algorithms in multi-user and multi-keyword scenarios.

2.2 Existing system:

Existing privacy-preserving mechanisms for public auditing in shared cloud data exhibit vulnerabilities, notably in the realm of identity privacy. The risk of leakage to public verifiers poses a considerable threat to user confidentiality, highlighting the need for a more robust solution that ensures the utmost privacy protection. Another critical gap lies in the inefficiencies of handling multiple auditing tasks concurrently. Traditional methods often need to improve in efficiently verifying shared data integrity for various tasks simultaneously, necessitating an innovative approach to enhance overall efficiency and reduce computational burdens.

Moreover, concerns related to data loss and extended authentication times further underscore existing knowledge gaps. The reliance on conventional cryptographic primitives may not be directly applicable in cloud environments where users no longer physically possess data storage. This raises questions about the adaptability of current methods to ensure data security and protection. Additionally, the extended authentication times associated with traditional approaches hinder user experience. Addressing these gaps requires a comprehensive privacy-preserving solution that not only rectifies current limitations but also anticipates and proactively addresses emerging challenges in the dynamic landscape of cloud computing and data sharing.

2.3 Limitations of Existing System

- As users no longer physically possess the storage of their data, traditional cryptographic primitives for the purpose of data security protection cannot be directly adopted.
- They do not perform the multiple auditing tasks in simultaneously.
- Loss of data.
- Does not provide any privacy for private data.
- Authentication time takes too long.

3. SYSTEM REQUIREMENT SPECIFICATION

3.1 Software Requirements

- Language – Java (JDK 1.7)
- OS - Windows 7 32bit
- MySql Server
- NetBeans IDE 7.1.2

3.2 Hardware Requirements

- 1 GB RAM
- 80 GB Hard Disk
- Above 2GHz Processor
- Data Card

3.3 Non- Functional Requirements

- Confidentiality
- Integrity
- Availability
- Performance
- Flexibility
- Maintainability
- Scalability
- Security

4. DESIGN

4.1 System Architecture

The system architectural design is the design process for identifying the subsystems making up the system and framework for subsystem control and communication. The goal of the architectural design is to establish the overall structure of the software system.

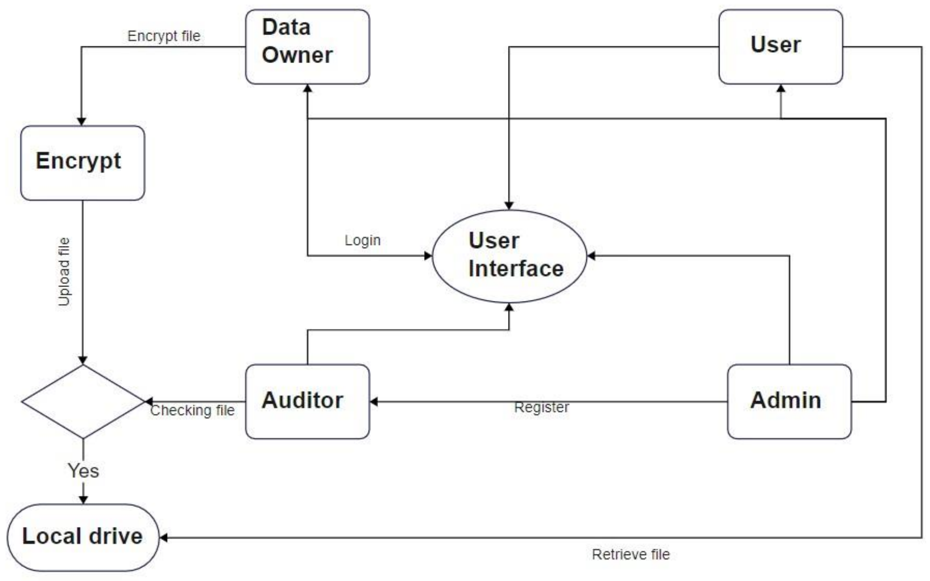
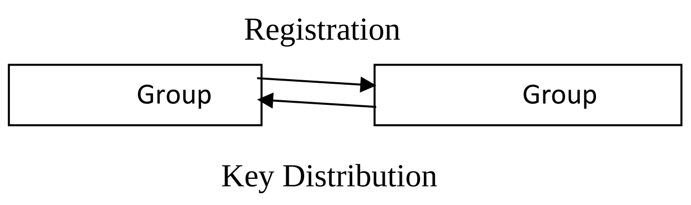


Figure- 4.1 System Architecture

4.2 Module Specification

4.2.1 User Registration module

For the registration of user with identity ID the group manager randomly selects a number. Then the group manager adds into the group user list which will be used in the traceability phase. After the registration, user obtains a private key which will be used for group signature generation and file decryption.



4.2.2 Auditing module

Homomorphic authenticators are unforgeable verification metadata generated from individual data blocks, which can be securely aggregated in such a way to assure an auditor that a linear combination of data blocks is correctly computed by verifying only the aggregated authenticator. Overview to achieve privacy-preserving public auditing, we propose to uniquely integrate the Homomorphic authenticator with random mask technique. In our protocol, the linear combination of sampled blocks in the server's response is masked with randomness generated by a pseudo random function (PRF). The proposed scheme is as follows:

- Setup Phase
- Audit Phase

4.2.3 Sharing Data module

The canonical application is data sharing. The public auditing property is especially useful when we expect the delegation to be efficient and flexible. The schemes enable a content provider to share her data in a confidential and selective way, with a fixed and small ciphertext expansion, by distributing to each authorized user a single and small aggregate key.

4.2.4 Integrity Checking module

Hence, supporting data dynamics for privacy-prng public risk auditing is also of paramount importance. Now we show how our main scheme can be adapted to build upon the existing work to support data dynamics, including block level operations of modification, deletion and insertion. We can adopt this technique in our design to achieve privacy-preserving public risk auditing with support of data dynamics. The user downloads the particular file not download entire file.

4.3 UML Diagrams

Any complex system is best understood by making some kind of diagrams or pictures. These diagrams have a better impact on our understanding. If we look around, we will realize that the diagrams are not a new concept but it is used widely in different forms in different industries. We prepare UML diagrams to understand the system in a better and simple way. A single diagram is not enough to cover all the aspects of the system. UML defines various kinds of diagrams to cover most of the aspects of a system. You can also create your own set of diagrams to meet your requirements. Diagrams are generally made in an incremental and iterative way. There are two broad categories of diagrams and they are again divided into subcategories –

- Structural Diagrams
- Behavioral Diagrams

Structural Diagrams: The structural diagrams represent the static aspect of the system. These static aspects represent those parts of a diagram, which forms the main structure and are therefore stable. These static parts are represented by classes, interfaces, objects, components, and nodes. The four structural diagrams are –

- Class diagram
- Object diagram
- Component diagram
- Deployment diagram

Behavioral Diagrams: Any system can have two aspects, static and dynamic. So, a model is considered as complete when both the aspects are fully covered. Behavioral diagrams basically capture the dynamic aspect of a system. Dynamic aspect can be further described as the changing/moving parts of a system. UML has the following five types of behavioral diagrams –

- Use case diagram
- Sequence diagram
- Collaboration diagram
- State chart diagram
- Activity diagram

4.3.1 Use case Diagram:

Use case diagrams represent the overall scenario of the system. A scenario is nothing but a sequence of steps describing an interaction between a user and a system. Thus, a use case is a set of scenarios tied together by some goal. The use case diagrams are drawn for exposing the functionalities of the system.

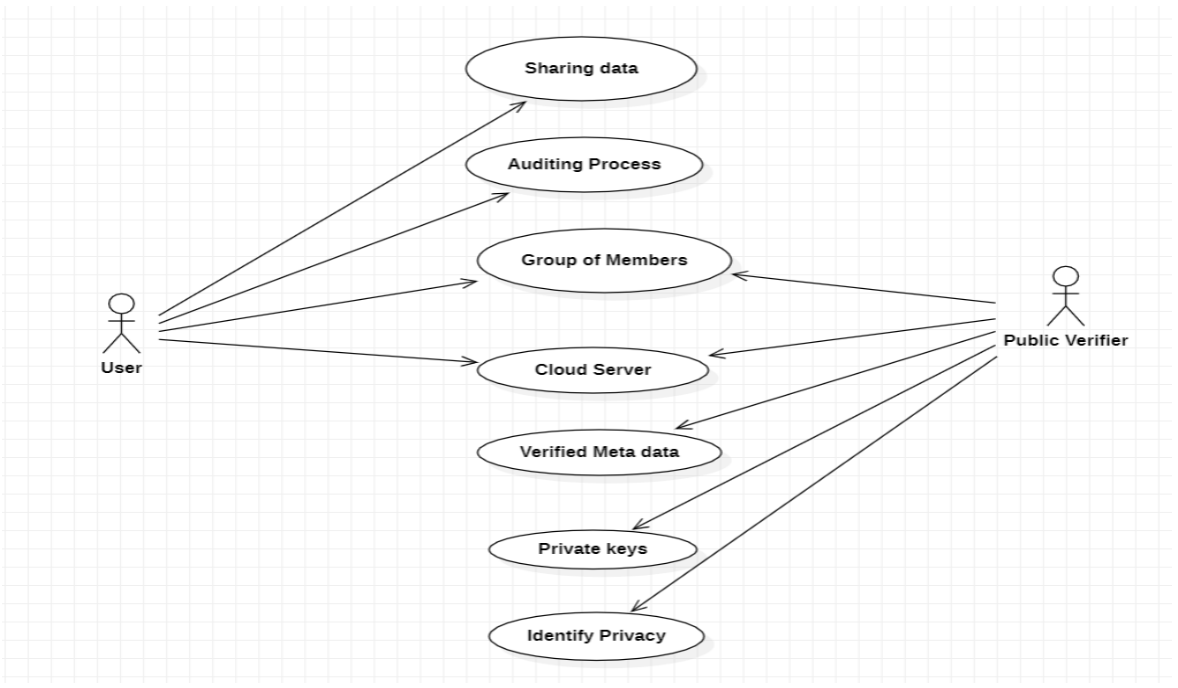


Figure-4.3.1 Use Case diagram

4.3.2 CLASS DIAGRAM:

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modelling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity. Class diagrams are useful in all forms of object-oriented programming (OOP). The concept is several years old but has been refined as OOP modelling paradigms have evolved.

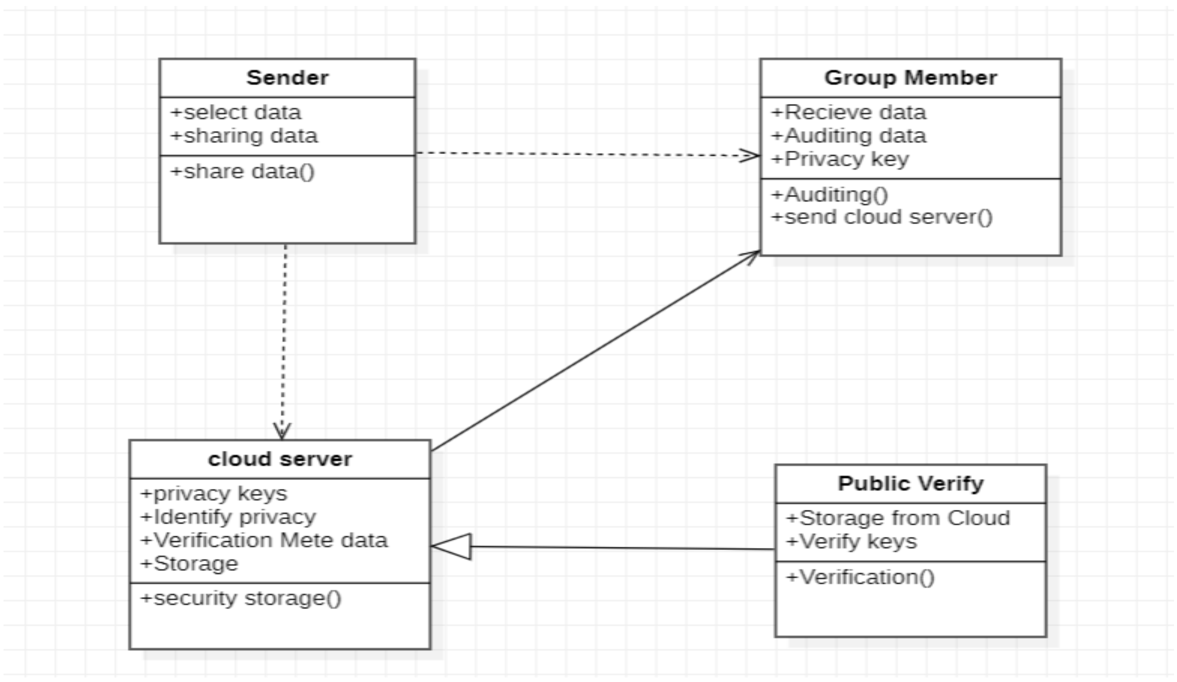


Figure-4.3.2 Class diagram

4.3.3 SEQUENCE DIAGRAM:

IT shows the interaction between a set of objects, through the messages that may be dispatched between them. The diagrams consist of interacting objects and actors, with messages in-between them it is common to focus the model on scenarios specified by use-cases. It is also often useful input to the detailed class diagram to try to model the specified.

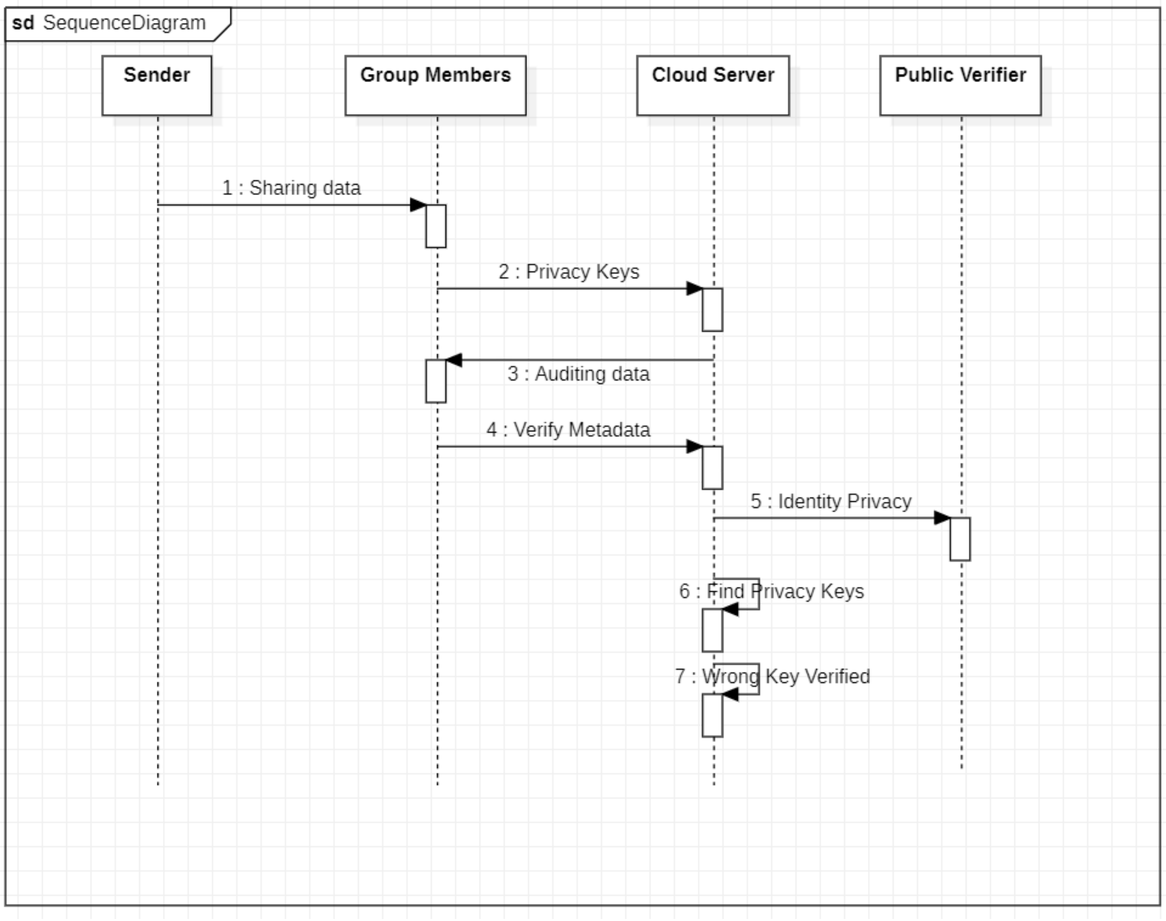


Figure-4.3.3 Sequence diagram

4.3.4 COLLABORATION DIAGRAM:

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modelling Language (UML). A collaboration diagram resembles a flowchart that portrays the roles, functionality and behaviour of individual objects as well as the overall operation of the system in real time.

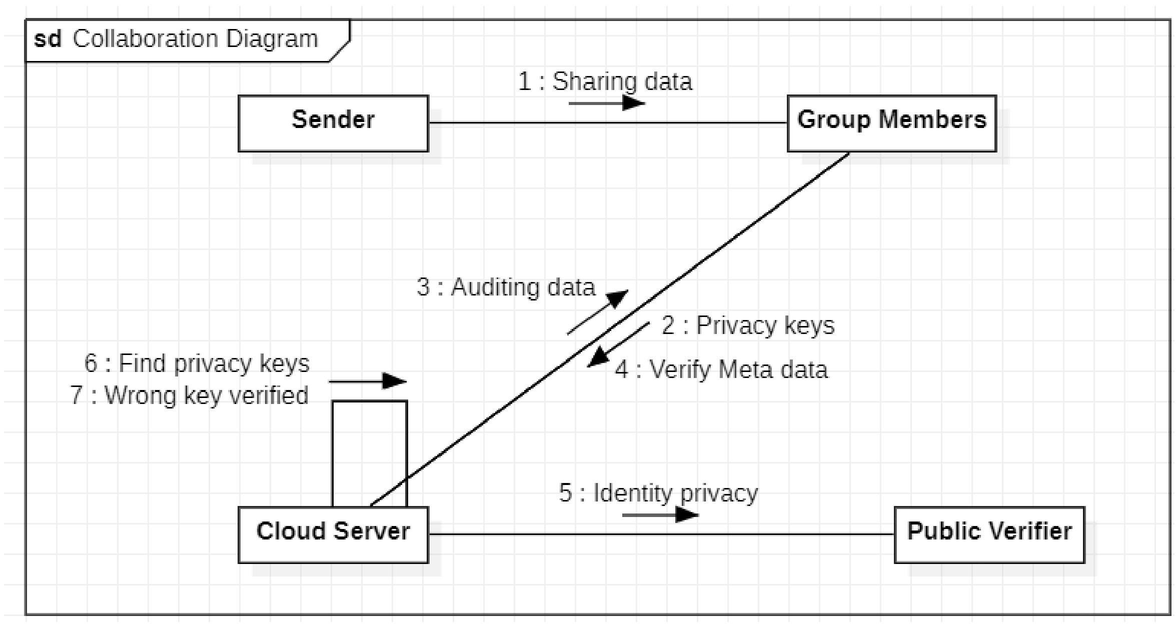


Figure-4.3.4 Collaboration diagram

4.3.5 ACTIVITY DIAGRAM:

It shows the flow through a program from a defined start point to an end point. Activity diagrams describe the workflow behaviour of a system. Activity diagrams are similar to state diagrams because activities are the state of doing something. The diagrams describe the state of activities by showing the sequence of activities performed. Activity diagrams can show activities that are conditional or parallel. Basic elements in activity diagrams are activities, branches (conditions or selections), transitions, forks and joins.

Activity diagrams should be used in conjunction with other modelling techniques such as interaction diagrams and state diagrams. The main reason to use activity diagrams is to model the workflow behind the system being designed. Activity Diagrams are also useful for: analysing a use case by describing what actions need to take place and when they should occur; describing a complicated sequential algorithm; and modelling applications with parallel processes. Activity diagrams do not give detail about how objects behave or how objects collaborate.

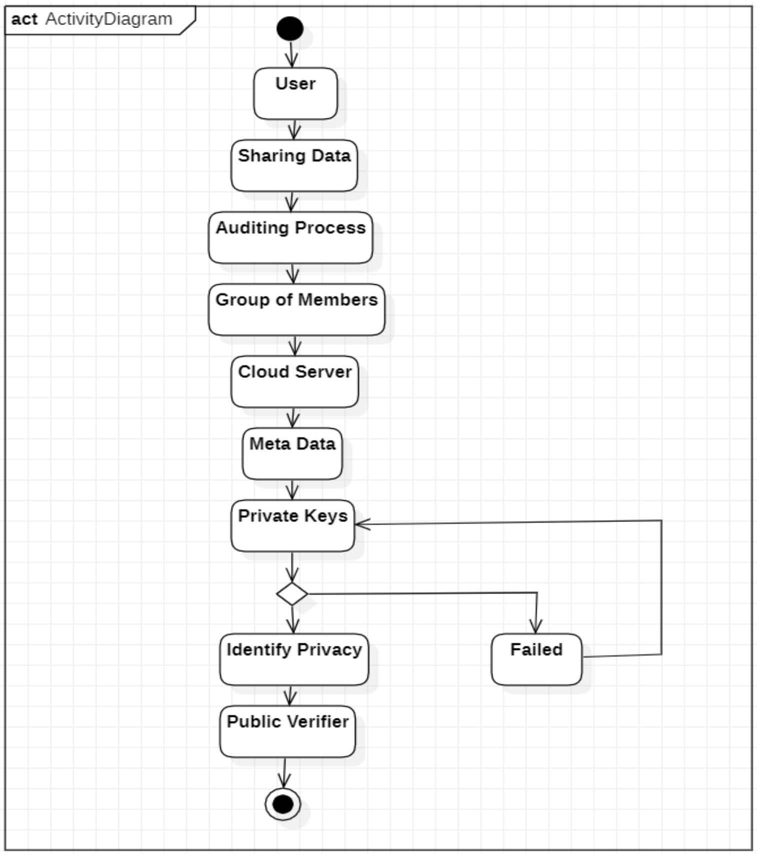


Figure-4.3.5 Activity diagram

4.3.6 Data Flow Diagram:

DFD level 0:

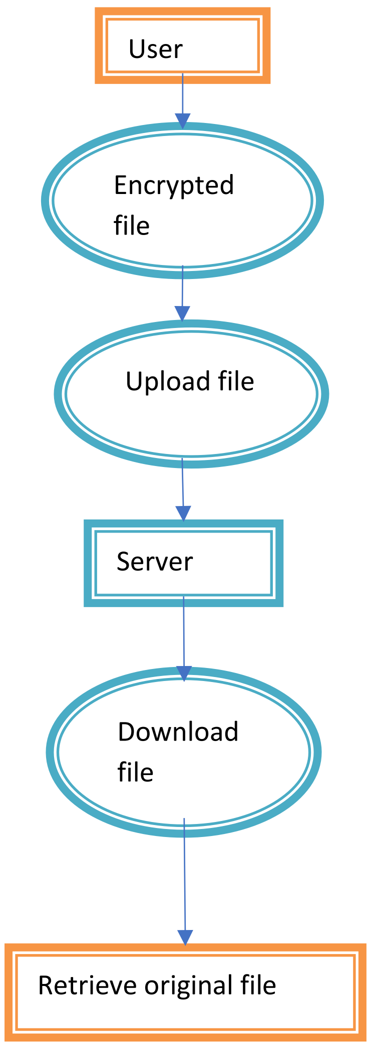


Figure-4.3.6.1 DFD level 0

DFD level 1:

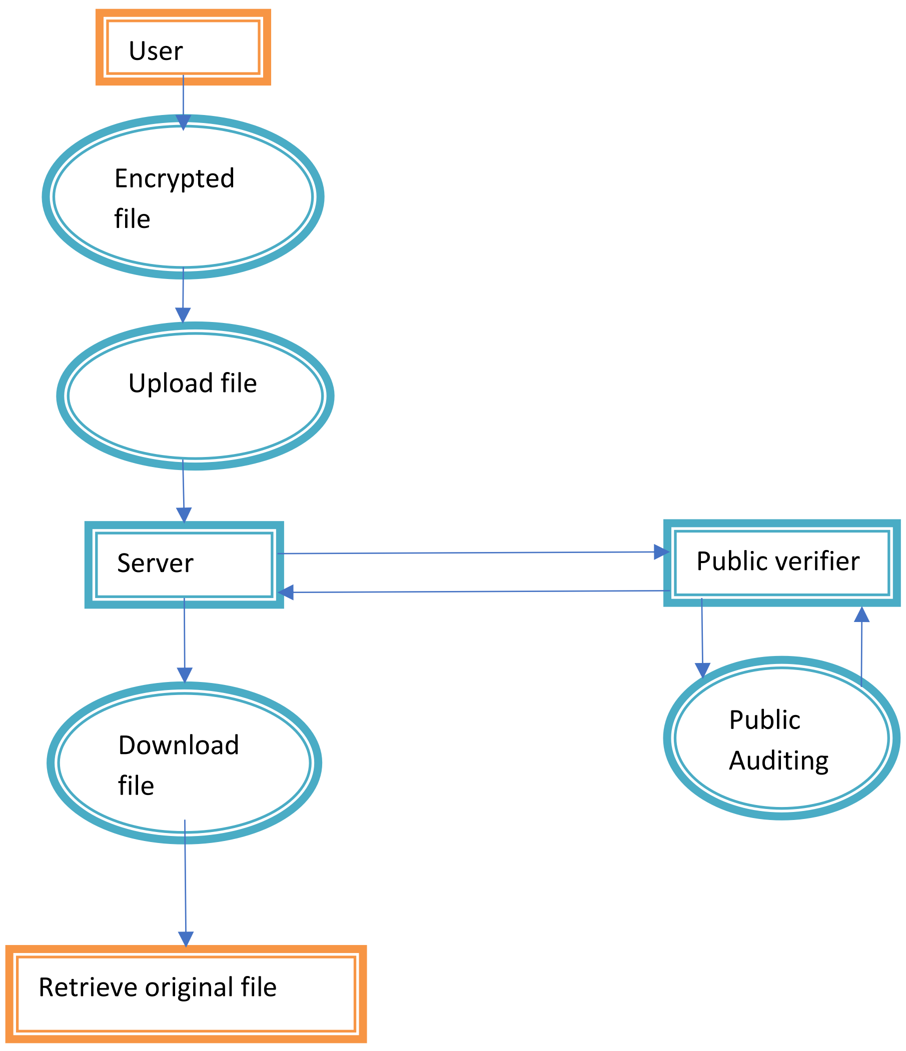


Figure-4.3.6.2 DFD level 1

DFD level 2:

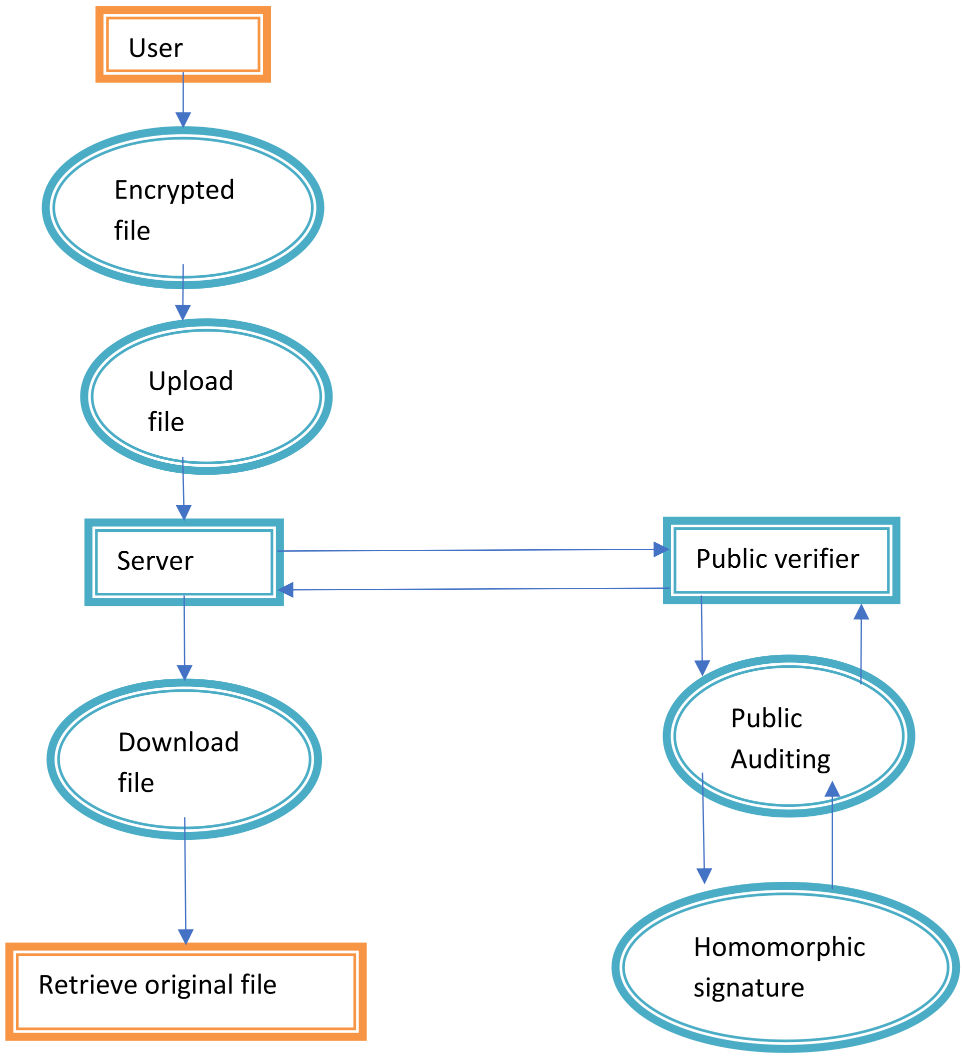


Figure-4.3.6.3 DFD level 2

4.4 ALGORITHMS

Advanced Encryption Standard (AES):

The Advanced Encryption Standard (AES) algorithm serves as a cornerstone cryptographic technique, ensuring the security and confidentiality of shared data. Utilized within the AESEncrypter Java class, AES is a symmetric encryption algorithm known for its substitution-permutation network structure. Unlike Feistel ciphers, AES operates on entire blocks of data simultaneously, enhancing efficiency and robustness in encryption. Within AES, each plaintext block's 128 bits are treated as sixteen bytes, organized into a four-column, four-row matrix format. This matrix-based computation approach forms the foundation of AES's cryptographic operations, facilitating the transformation of plaintext input into ciphertext output.

The AESEncrypter class implements AES encryption and decryption functionalities, leveraging the Cipher Block Chaining (CBC) mode with PKCS5Padding for cryptographic operations. The constructor initializes the encryption and decryption ciphers, requiring a secret key to perform encryption and decryption operations securely. An 8-byte initialization vector (IV) is utilized to enhance security during encryption. Through interconnected operations, AES performs a combination of substitution and permutation operations, ensuring data confidentiality and integrity.

One notable feature of AES is its ability to support various key lengths, including 128-bit, 192-bit, and 256-bit keys. This flexibility enables AES to adapt to diverse security requirements, making it suitable for a wide range of applications. In the context of the project, AES is employed to encrypt and decrypt shared data, safeguarding sensitive information from unauthorized access. By utilizing AES as a foundational encryption mechanism, the project enhances the privacy and integrity of shared data, contributing to the overall security of the proposed privacy-preserving mechanism for public auditing in cloud storage systems.

The AESEncrypter class provides methods for encrypting and decrypting data streams, ensuring secure communication and storage of sensitive information. The encrypt method accepts an input stream containing cleartext data and encrypts it using AES encryption, while the decrypt method decrypts encrypted data streams. Exception handling mechanisms are incorporated to handle IO-related errors during file processing, enhancing the reliability and robustness of the encryption and decryption processes.

By leveraging AES encryption within the AESEncrypter class, the project aims to fortify the privacy and integrity of shared data in cloud storage systems. AES encryption ensures that sensitive information remains confidential and secure, mitigating the risks of unauthorized access and data breaches. Through the adoption of AES as a foundational encryption mechanism, the project aligns with best practices in data security, contributing to the overall trustworthiness and security of cloud-based applications and services.

Simple Mail Transfer Protocol (SMTP):

SMTP (Simple Mail Transfer Protocol) plays a vital role in facilitating seamless communication between devices within specific networks, serving as the backbone for transmitting messages securely and efficiently. SMTP email service providers enable businesses to connect effectively with customers, partners, and stakeholders by ensuring reliable email delivery. Leveraging protocols like SMTP, these service providers enable users to send and receive messages with ease, enhancing communication across various platforms and devices.

To utilize an SMTP service provider for sending emails, businesses typically follow a straightforward process. They gather essential information such as the host name, port, SSL authentication details, and account credentials. Subsequently, they configure the SMTP settings by entering this information into designated forms provided by the service provider. Once the setup is complete, businesses can access the SMTP interface to add and authenticate their accounts, paving the way for seamless email communication.

Within the provided Java servlet code (UserRegServlet), SMTP is integrated to facilitate email notifications during user registration processes. Upon successful registration, the servlet initiates an SMTP session to send activation emails to newly registered users. This process involves setting up SMTP properties such as the host, port, authentication, and security details to establish a secure connection with the SMTP server. Through this integration, businesses can enhance user experience by providing timely notifications and account activation instructions.

The servlet code utilizes JavaMail API to interact with the SMTP server for sending activation emails. It establishes a secure connection using SSL/TLS protocols and authenticates with the SMTP server using valid credentials. Once authenticated, the servlet constructs and sends email messages containing essential registration details, such as the username, password, and activation key. By leveraging SMTP and JavaMail API, the servlet ensures reliable email delivery and enhances the registration process's efficiency and effectiveness.

By combining SMTP functionality with Java servlets, businesses can streamline their user registration processes and improve communication with their audience. The integration of SMTP facilitates automated email notifications, ensuring that users receive timely updates and activation instructions. This seamless communication approach contributes to a positive user experience, fostering trust and engagement with the platform. Additionally, leveraging SMTP within Java servlets aligns with industry best practices for secure and efficient email delivery in web-based applications.

Hashing:

The mdhashing Java class offers functionalities for computing checksums of files, specifically MD5 and SHA-1 algorithms. These checksums serve to verify file integrity and detect any unauthorized modifications. The class is designed to provide a straightforward interface for developers to incorporate checksum calculations into their Java applications seamlessly.

The main method, main, serves as the entry point for the program. It showcases the usage of the checkSum method to compute the MD5 checksum of a specified file. Upon execution, it prints the calculated MD5 checksum to the console, demonstrating how to utilize the mdhashing class for checksum generation.

The checkSum method is responsible for computing the MD5 checksum of a file identified by its path. Leveraging the MessageDigest class, it performs cryptographic hashing to generate the checksum. Upon completion, it returns the computed MD5 checksum as a hexadecimal string, ready for further processing or validation within the application.

For SHA-1 checksum computation, the createSha1 method is provided. This method accepts a File object representing the file for which the checksum is to be calculated. It employs the MessageDigest class to perform the SHA-1 hashing operation on the file's contents. The resulting SHA-1 checksum is returned as a byte array, facilitating integrity verification and comparison with expected values.

An additional SHA-1 checksum method, is available for calculating checksums based on file paths. Similar to createSha1, this method utilizes the MessageDigest class to compute the SHA-1 checksum of the specified file. The computed checksum is returned as a hexadecimal string, suitable for various validation and security purposes within Java applications.

To handle exceptions and errors gracefully, the mdhashing class employs logging capabilities using the java.util.logging.Logger class. Any exceptions encountered during file processing, such as IO errors or cryptographic algorithm issues, are logged for diagnostic purposes. This logging mechanism enhances the reliability and maintainability of applications utilizing the mdhashing class for checksum calculations.

In summary, the mdhashing class provides a robust solution for computing MD5 and SHA-1 checksums of files in Java applications. By offering clear and concise methods for checksum generation, developers can seamlessly integrate file integrity verification into their projects. Whether for data auditing, file transfer validation, or security enforcement, the mdhashing class offers a reliable and efficient approach to checksum computation in Java environments.

5. IMPLEMENTATION

5.1 Methodology

The propose system, a privacy-preserving public auditing mechanism for shared data in the cloud. We utilize ring signatures to construct homomorphism authenticators, so that a public verifier is able to audit shared data integrity without retrieving the entire data, yet it cannot distinguish who is the signer on each block. To improve the efficiency of verifying multiple auditing tasks, we further extend our mechanism to support batch auditing. There are two interesting problems we will continue to study for our future work. One of them is traceability, which means the ability for the group manager to reveal the identity of the signer based on verification metadata in some special situations.

SMTP stands for "Simple Mail Transfer Protocol." this can be the protocol used for causation e-mail over the web. Your e-mail shopper uses SMTP to send a message to the mail server, and also the mail server uses SMTP to relay that message to the proper receiving mail server. Basically, SMTP could be a set of commands that certify and direct the transfer of electronic message. Once configuring the settings for your e-mail program, you always ought to set the SMTP server to your native net Service Provider's SMTP settings. However, the incoming mail server (IMAP or POP3) ought to be set to your mail account's server, which can differ than the SMTP server.

ADVANTAGES:

- The proposed system can perform multiple auditing tasks simultaneously
- They improve the efficiency of verification for multiple auditing tasks.
- High security provides for file sharing.
- Admin has control deleting users
- Users can send request to auditor.

5.2 Executable Code

Database:

```
create database if not exists `erasurecode`;

USE `erasurecode`;

DROP TABLE IF EXISTS `admin`;

CREATE TABLE `admin` (

  `username` varchar(30) default NULL,

  `password` varchar(30) default NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

insert into `admin` (`username`,`password`) values ('deva','deva');

DROP TABLE IF EXISTS `gender`;

CREATE TABLE `gender` (

  `Gender` varchar(20) default NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

insert into `gender` (`Gender`) values ('Male');

insert into `gender` (`Gender`) values ('Female');

DROP TABLE IF EXISTS `ownerprofile`;

CREATE TABLE `ownerprofile` (

  `UserName` varchar(35) default NULL,

  `groupname` varchar(35) default NULL,

  `filename` varchar(700) NOT NULL,

  `server1` varchar(100) default NULL,

  `server2` varchar(100) default NULL,

  `server3` varchar(100) default NULL,
  `server4` varchar(100) default NULL,

  PRIMARY KEY  (`filename`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;


Insert into `ownerprofile`
(`UserName`,`groupname`,`filename`,`server1`,`server2`,`server3`,`server4`) values
('ram','ram','3.jpg','ef3cad6eee251f670014365fde992889e73f1a8e','ef3cad6eee251f670014
365fde992889e73f1a8ekey1','ef3cad6eee251f670014365fde992889e73f1a8ekey2','ef3cad6
eee251f670014365fde992889e73f1a8ekey3');

DROP TABLE IF EXISTS `registration`;

CREATE TABLE `registration` (

  `username` varchar(30) default NULL,

  `password` varchar(30) default NULL,

  `gender` varchar(20) default NULL,

  `email` varchar(50) default NULL,

  `phoneno` varchar(50) default NULL,

  `userproductkey` varchar(50) default NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

insert into `registration`
(`username`,`password`,`gender`,`email`,`phoneno`,`userproductkey`) values
('ram','ram','male','deva@gamil.com','1234','asdfasdfasdfasdf');

insert into `registration`
(`username`,`password`,`gender`,`email`,`phoneno`,`userproductkey`) values ('deva
',NULL,NULL,NULL,NULL,NULL);

insert into `registration`
(`username`,`password`,`gender`,`email`,`phoneno`,`userproductkey`) values
('kumar',NULL,NULL,NULL,NULL,NULL);
```

Java Code:

```
package com.ErasureCode;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

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 java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

public class AdminServlet extends HttpServlet {

    private Object out;

    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)

            throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");

        try (PrintWriter out = response.getWriter()) {

            out.println("<!DOCTYPE html>");

            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet AdminServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet AdminServlet at " + request.getContextPath() + "</h1>");
            out.println("</body>");
            out.println("</html>");
        }
    }
    @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 {
        HttpSession session1=request.getSession();
        Connection con=null;
        Statement st=null;
        ResultSet rs=null;
        try(PrintWriter out = response.getWriter()) {
            String Username=request.getParameter("username");
            String Password=request.getParameter("password");
```

```
            System.out.println("this line my cheking======="+Username);

            System.out.println("this line my cheking======="+Password);

            Class.forName("com.mysql.jdbc.Driver");
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/imageselectionerasureco
de","root","password");

            st=con.createStatement();

            rs=st.executeQuery("select * from admin where username='"+Username+"' and
password='"+Password+"'");

            if(rs.next()) {

                response.sendRedirect("UserReg.jsp");

            } else {

                RequestDispatcher rd=request.getRequestDispatcher("Admin1.jsp");

                rd.include(request, response);

                out.print("<br><br><br><h1><center>Sorry UserName or Password
Error!"+"</h1>");

            }

        } catch(Exception ex) {

            ex.printStackTrace();

            }

      }

}
```

6. TESTING

TESTING DEFINATION:

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

6.1 UNIT TESTING

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

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.

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

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.

6.2 Test Cases

A test case is the individual unit of testing. It checks for a specific response to a particular set of inputs.

Outcomes Possible:

There are three types of possible test outcomes:

- OK – This means that all the tests are passed.
- FAIL – This means that the test did not pass and an AssertionError exception is raised.
- ERROR – This means that the test raises an exception other than AssertionError.

Test cases

Admin login:

Table 6.2.1 Admin Login

Use case ID	Admin-login
Use case name	Admin page
Description	Admin has to login in to the User Interface using their login credentials, and the OTP generated in the database.
Primary actor	Admin
Precondition	Open the project folder in NetBeans IDE 8.0.2 and run the project.
Post condition	Admin login into the application and registers all the data owners, auditors, and users.
Frequency of use case	Many times
Alternative use case	N/A

Data Owner:

Table 6.2.2 Data Owner Login

Use case ID	Data owner login
Use case name	Data owner page
Description	Data owner has to encrypt the data file using the AES algorithm and upload it in the cloud storage.
Primary actor	Data Owner
Precondition	Admin registers the Data owner and sends username, password and product key to the owner using SMTP protocol.
Post condition	Data owner encrypts the data file using AES algorithm with the help of product key and then uploads it in the cloud storage.
Frequency of use case	Many times
Alternative use case	N/A

Auditor:

Table 6.2.3 Auditor Login

Use case ID	Auditor-login
Use case name	Auditor page
Description	Auditor has to login in to the User Interface using their login credentials and verify the integrity of the uploaded file.
Primary actor	Auditor
Precondition	Admin registers the Auditor and sends username and password to the auditor using SMTP protocol.
Post condition	Auditor uploads the same file uploaded by the data owner and then verifies the integrity of the file by using Hashing.
Frequency of use case	Many times
Alternative use case	N/A

User:

Table 6.2.4 User Login

Use case ID	User-login
Use case name	User page
Description	User has to login in to the User Interface using their login credentials and with the help of secret key, encrypted key is decrypted and accessed by the user.
Primary actor	User
Precondition	Admin registers the User and sends username, password and secret key to the user using SMTP protocol.
Post condition	User has to enter the secret key and decrypt the encrypted file uploaded by the data owner and access the data.
Frequency of use case	Many times
Alternative use case	N/A

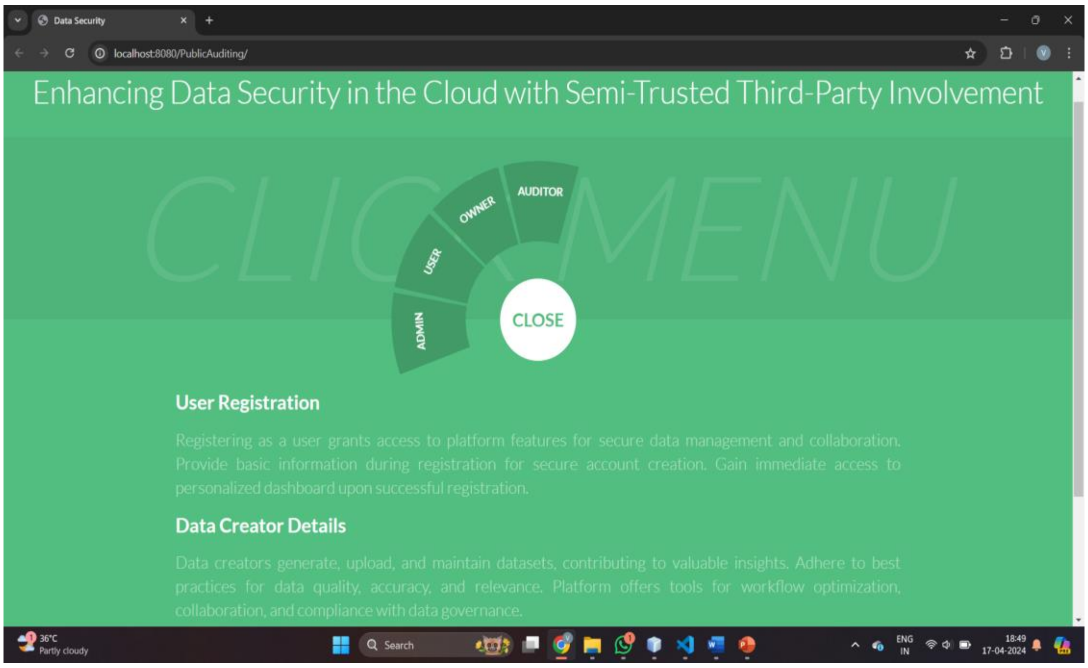
User Revocation:

Table 6.2.5 User Revocation

Use case ID	User-Revocation
Use case name	User deletion page
Description	Admin has to login in to the User Interface using their login credentials and delete the users.
Primary actor	Admin
Precondition	Open the project folder in NetBeans IDE 8.0.2 and run the project.
Post condition	Select user revocation method and delete the users who are not authorized or who do not maintain security.
Frequency of use case	Many times
Alternative use case	N/A

7. RESULTS

- Open the project folder in NetBeans IDE 8.0.2 and run the project. Then, the user interface will open as shown in the image below.



Enhancing Data Security in the Cloud with Semi-Trusted Third-Party Involvement

User Registration

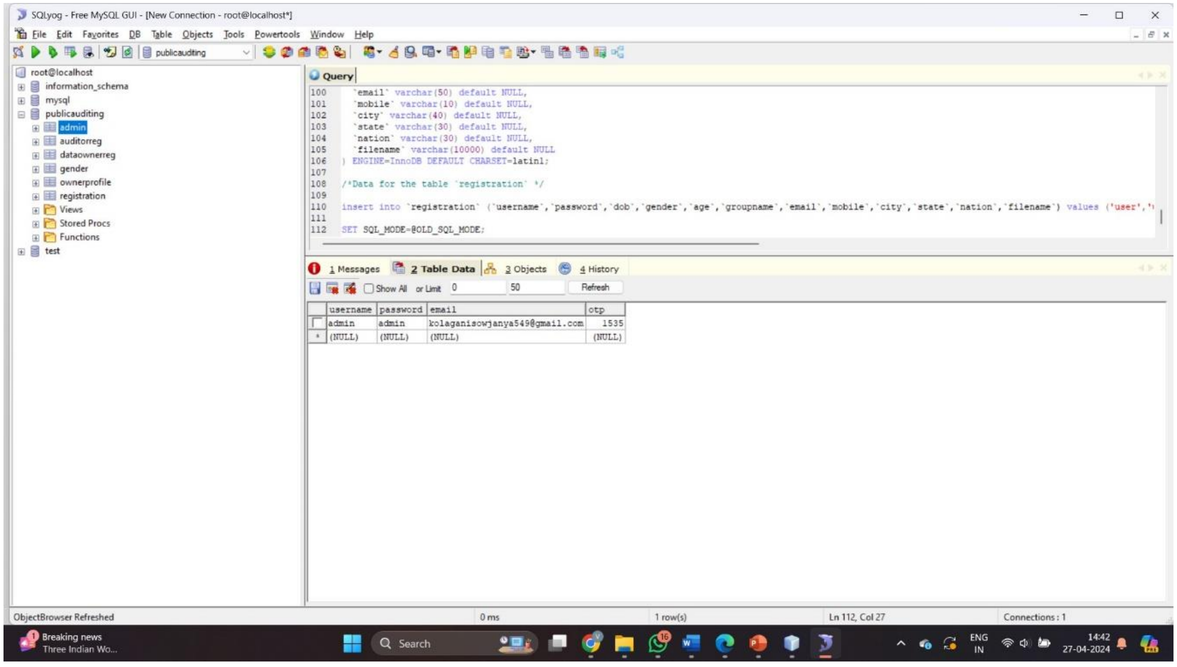
Registering as a user grants access to platform features for secure data management and collaboration. Provide basic information during registration for secure account creation. Gain immediate access to personalized dashboard upon successful registration.

Data Creator Details

Data creators generate, upload, and maintain datasets, contributing to valuable insights. Adhere to best practices for data quality, accuracy, and relevance. Platform offers tools for workflow optimization, collaboration, and compliance with data governance.

Figure-7.1 User Interface

- Next, we need to connect the frontend user interface with the backend database using SQLyog. Once this connection is established, both the frontend and backend will be connected to each other.



```
  `email` varchar(50) default NULL,
  `mobile` varchar(10) default NULL,
  `city` varchar(40) default NULL,
  `state` varchar(30) default NULL,
  `nation` varchar(30) default NULL,
  `filename` varchar(10000) default NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Data for the table `registration` */

insert into `registration` (`username`,`password`,`dob`,`gender`,`age`,`groupname`,`email`,`mobile`,`city`,`state`,`nation`,`filename`) values ('user','

SET SQL_MODE=@OLD_SQL_MODE;
```

	username	password	email	otp
	admin	admin	kolaganisowjanya549@gmail.com	1535
*	(NULL)	(NULL)	(NULL)	(NULL)

Figure-7.2 Admin Details

- Admin login details are stored in the database as shown in the above figure.

Admin Login Page:

Admin has to login in to the User Interface using their login credentials, and the OTP generated in the database.

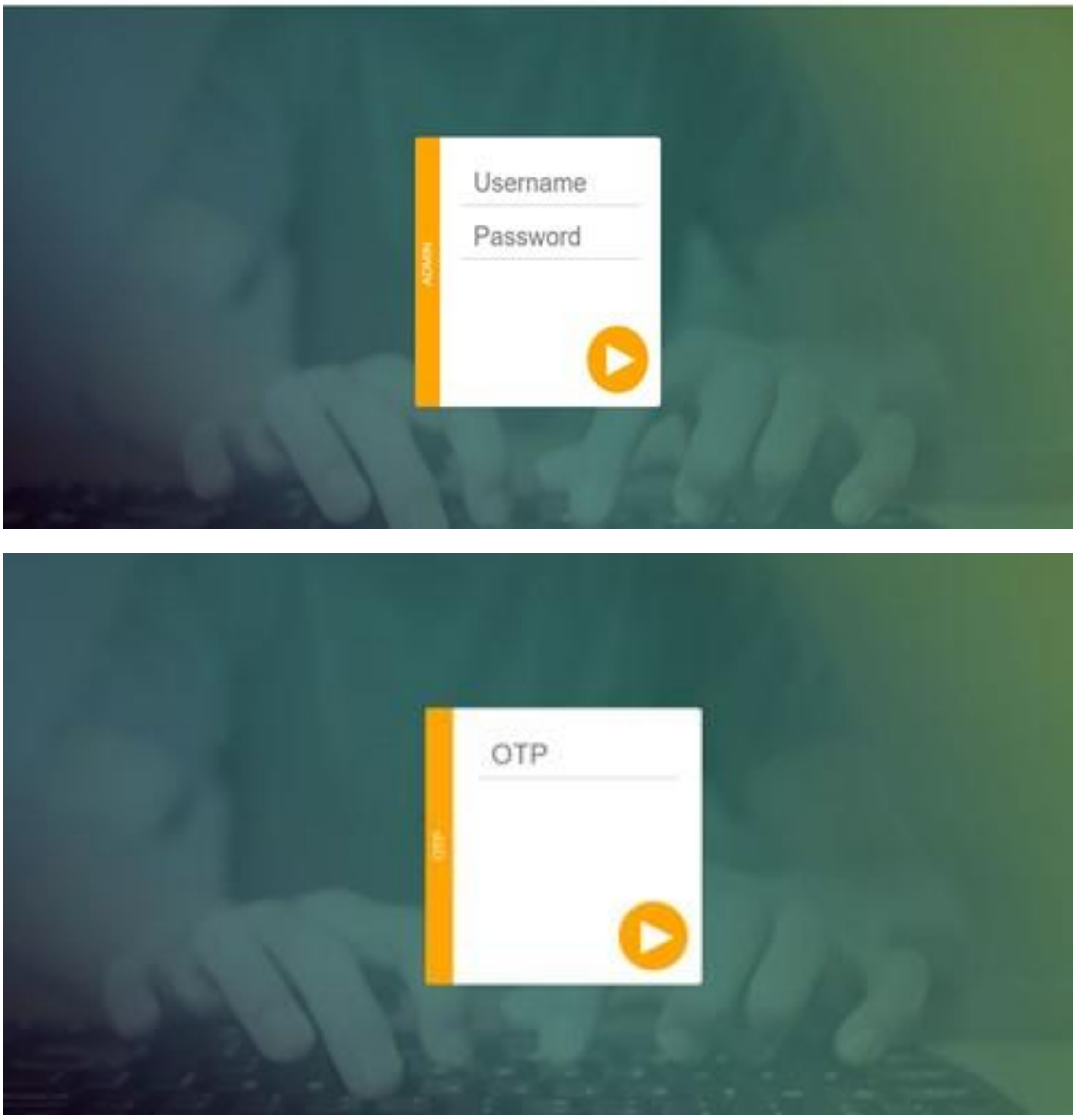


Figure-7.3 Admin Login

Registration Forms:

- After the admin logs into the User Interface, they must register all the users, data creators, and auditors who perform data transfer operations in the cloud.

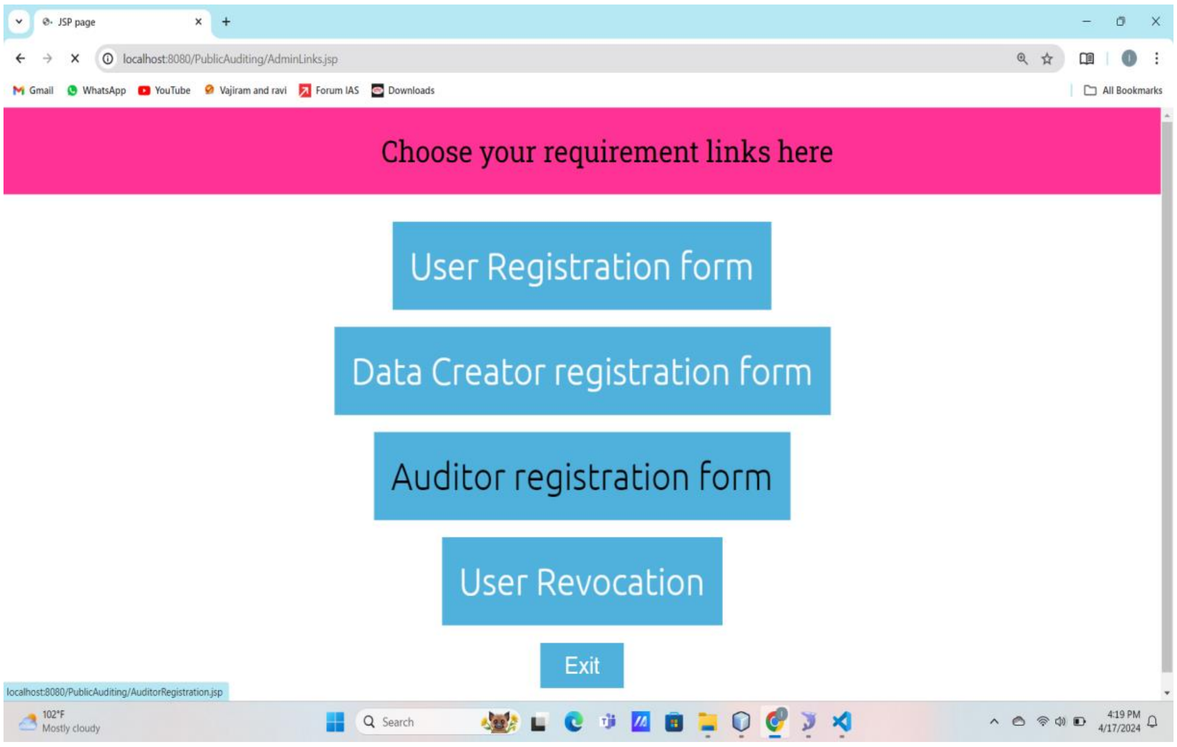

JSP page
localhost:8080/PublicAuditing/AdminLinks.jsp
Choose your requirement links here
User Registration form
Data Creator registration form
Auditor registration form
User Revocation
Exit
localhost:8080/PublicAuditing/AuditorRegistration.jsp

Figure-7.4 Registration Forms

Data Owner Registration:

- The admin must register all the data owners in the cloud who can store data by providing them with a username, password, and product key. These credentials are then transferred to the data owner's registered email address using the SMTP protocol.

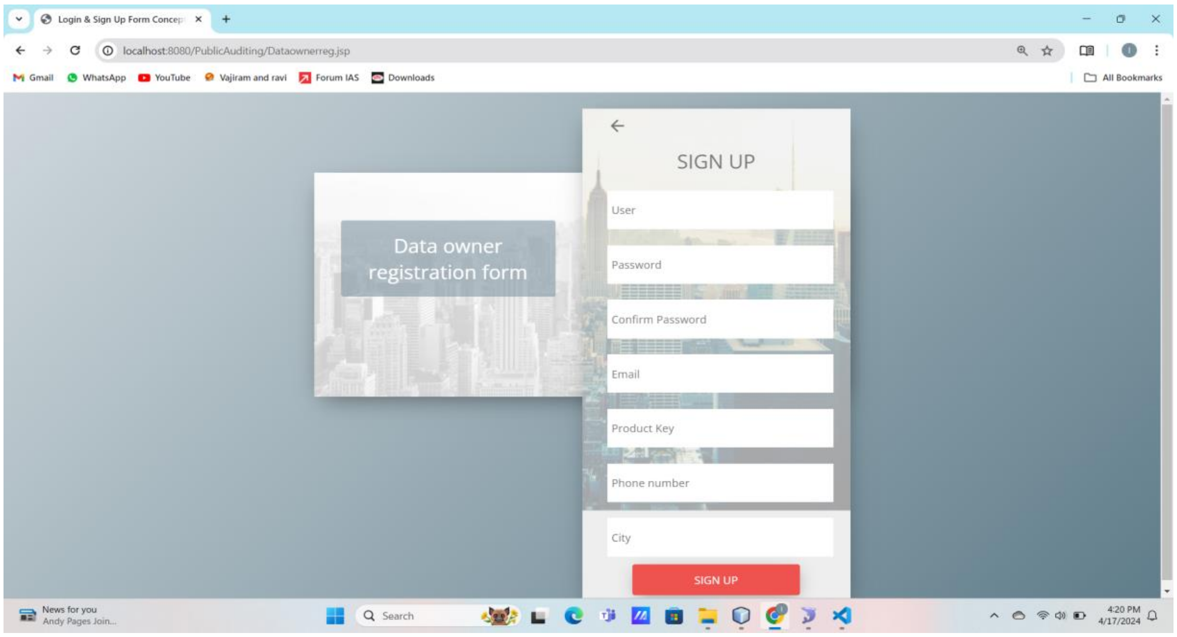
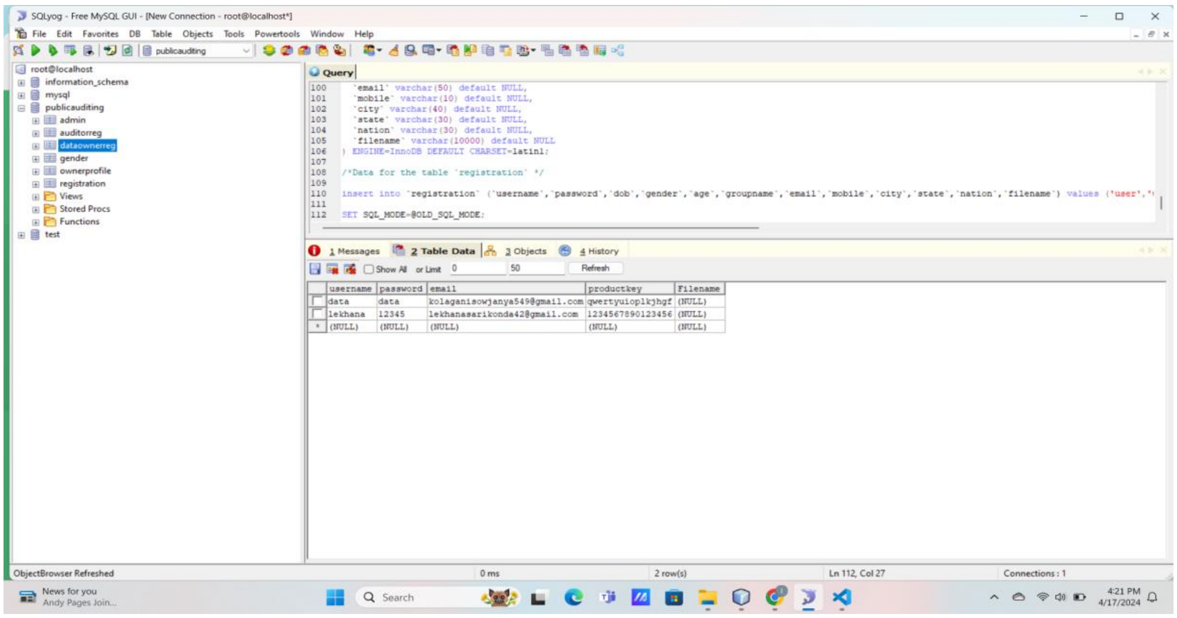

Login & Sign Up Form Concep
localhost:8080/PublicAuditing/Dataownerreg.jsp
Data owner registration form
SIGN UP
User
Password
Confirm Password
Email
Product Key
Phone number
City
SIGN UP

Figure-7.5 Data Owner Registration

Data Owner details in the Database:

All the data owner’s details are stored in the database as shown in the below image.



```
  `email` varchar(50) default NULL,
  `mobile` varchar(10) default NULL,
  `city` varchar(40) default NULL,
  `state` varchar(30) default NULL,
  `nation` varchar(30) default NULL,
  `filename` varchar(10000) default NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Data for the table `registration` */

insert into `registration` (`username`,`password`,`dob`,`gender`,`age`,`groupname`,`email`,`mobile`,`city`,`state`,`nation`,`filename`) values ('user','

SET SQL_MODE=@OLD_SQL_MODE;
```

username	password	email	productkey	Filename
data	data	kolaganisowjanya549@gmail.com	qwertyuioplkjhgf	(NULL)
lekhana	12345	lekhanasarikonda42@gmail.com	1234567890123456	(NULL)
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

Figure-7.6 Data Owner details

Auditor Registration:

- The admin must register all the auditors in the cloud who can verify the integrity of the data uploaded by the data owner by providing them with a username and password. These credentials are then transferred to the auditor's registered email address using the SMTP protocol.

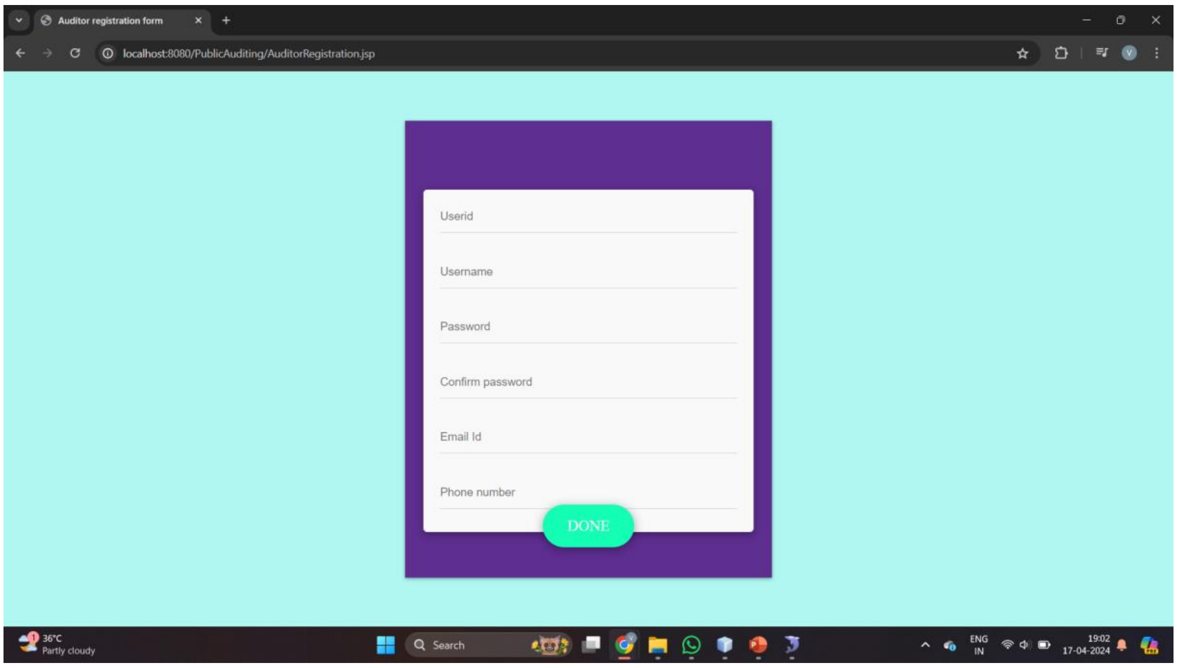
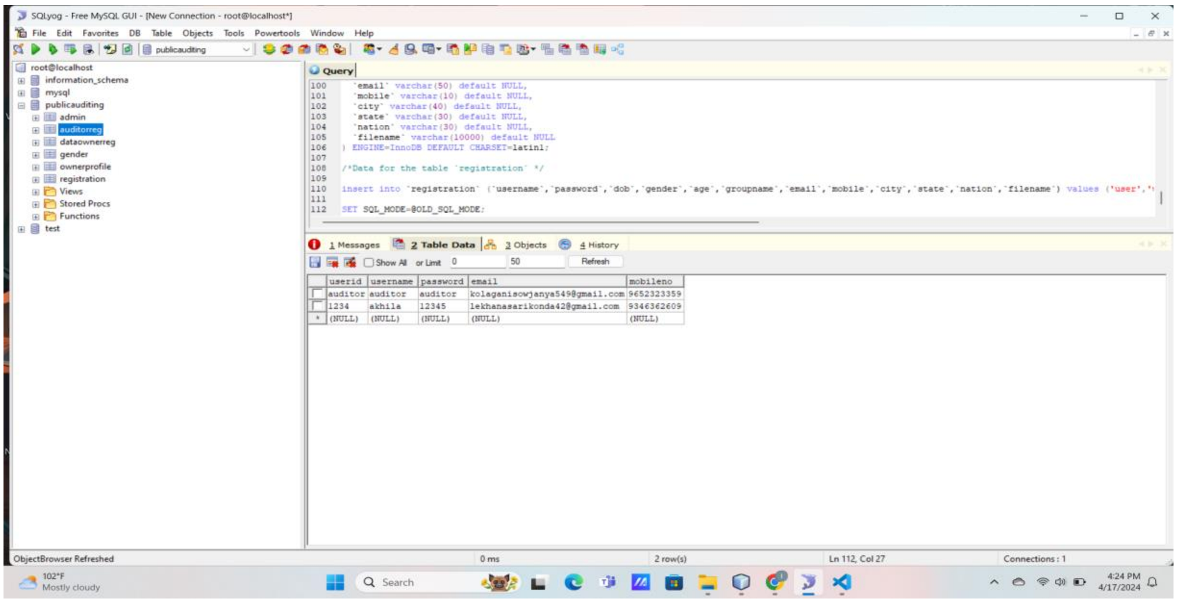

Auditor registration form
localhost:8080/PublicAuditing/AuditorRegistration.jsp
Userid
Username
Password
Confirm password
Email Id
Phone number
DONE

Figure-7.7 Auditor Registration

All the auditor's details are stored in the database as shown in the below image.



```
  `email` varchar(50) default NULL,
  `mobile` varchar(10) default NULL,
  `city` varchar(40) default NULL,
  `state` varchar(30) default NULL,
  `nation` varchar(30) default NULL,
  `filename` varchar(10000) default NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Data for the table `registration` */

insert into `registration` (`username`,`password`,`dob`,`gender`,`age`,`groupname`,`email`,`mobile`,`city`,`state`,`nation`,`filename`) values ('user','

SET SQL_MODE=@OLD_SQL_MODE;
```

	userid	username	password	email	mobileno
	auditor	auditor	auditor	kolaganisowjanya549@gmail.com	9652323359
	1234	akhila	12345	lekhanasarikonda42@gmail.com	9346362609
*	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

Figure-7.8 Auditor details

User Registration:

- The admin must register all the users in the cloud who access data by providing them with a username, password, and secret key. These credentials are then transferred to the user's registered email address using the SMTP protocol.

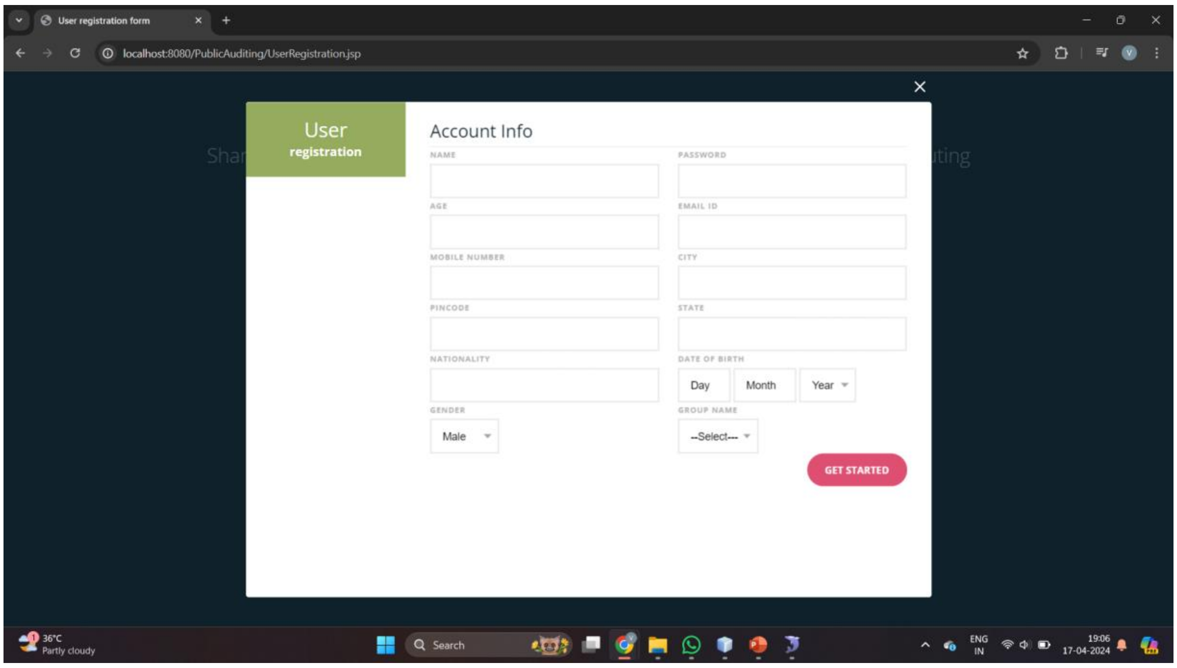
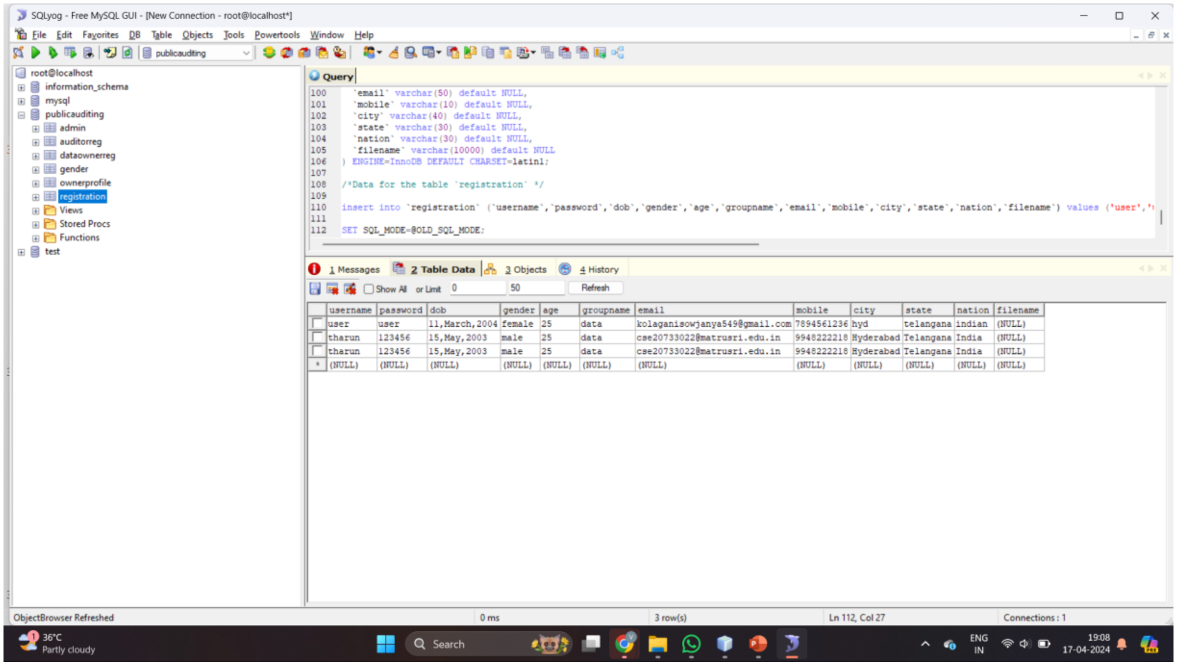


Figure-7.9 User Registration

All the user’s details are stored in the database as shown in the below image.



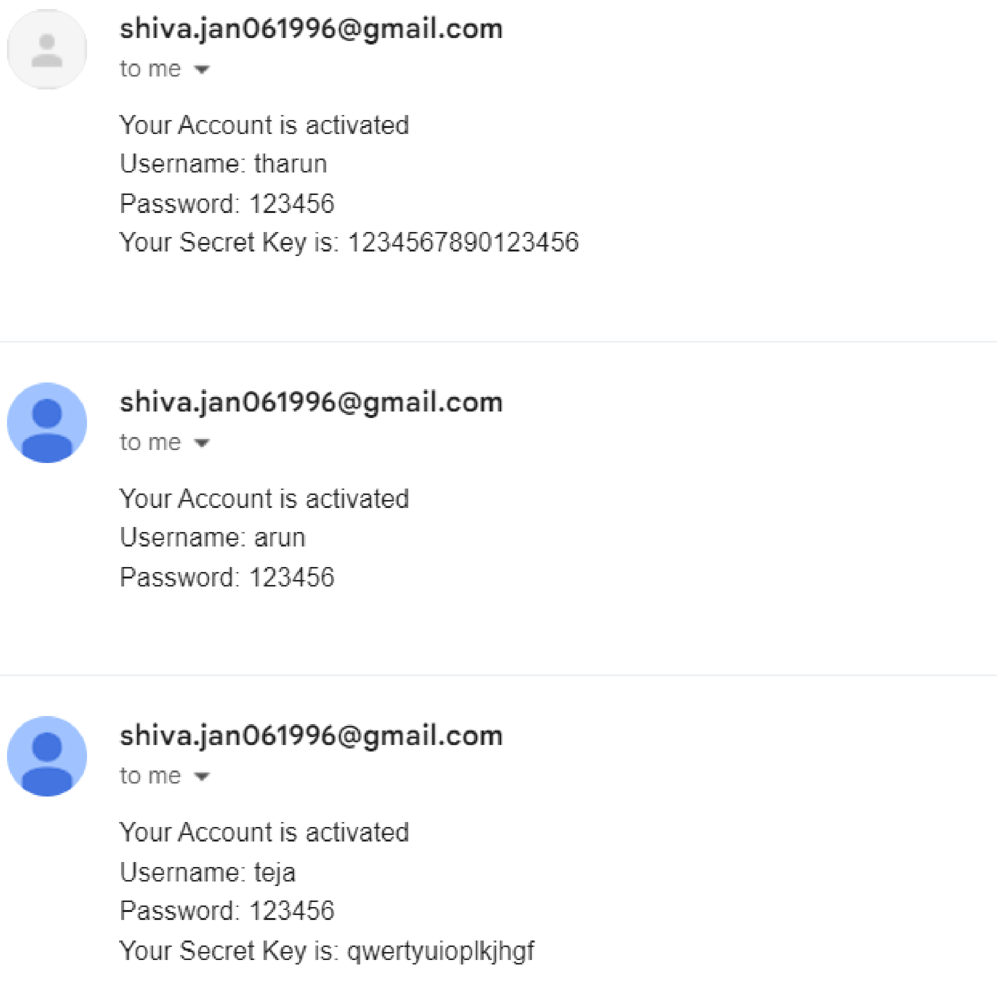
```
  `email` varchar(50) default NULL,
  `mobile` varchar(10) default NULL,
  `city` varchar(40) default NULL,
  `state` varchar(30) default NULL,
  `nation` varchar(30) default NULL,
  `filename` varchar(10000) default NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Data for the table `registration` */

insert into `registration` (`username`,`password`,`dob`,`gender`,`age`,`groupname`,`email`,`mobile`,`city`,`state`,`nation`,`filename`) values ('user','

SET SQL_MODE=@OLD_SQL_MODE;
```

	username	password	dob	gender	age	groupname	email	mobile	city	state	nation	filename
	user	user	11,March,2004	female	25	data	kolaganisowjanya549@gmail.com	7894561236	hyd	telangana	indian	(NULL)
	tharun	123456	15,May,2003	male	25	data	cse20733022@matrusri.edu.in	9948222218	Hyderabad	Telangana	India	(NULL)
	tharun	123456	15,May,2003	male	25	data	cse20733022@matrusri.edu.in	9948222218	Hyderabad	Telangana	India	(NULL)
*	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

Figure-7.10 User details

These are the mails received for every registration by the admin.

i. This mail is received at the time of Data Owner registration.
ii. This mail is received at the time of Auditor registration.
iii. This mail is received at the time of User registration.



shiva.jan061996@gmail.com
to me

Your Account is activated
Username: tharun
Password: 123456
Your Secret Key is: 1234567890123456

shiva.jan061996@gmail.com
to me

Your Account is activated
Username: arun
Password: 123456

shiva.jan061996@gmail.com
to me

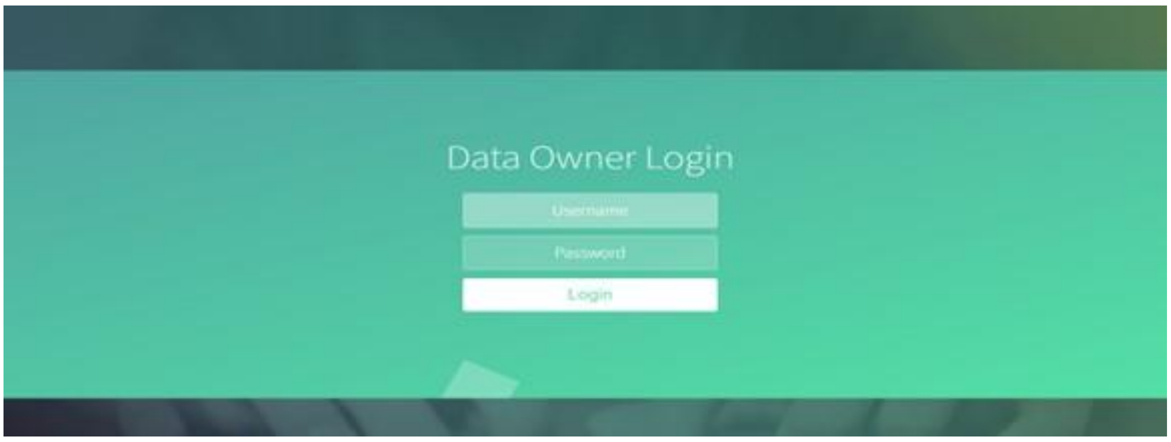
Your Account is activated
Username: teja
Password: 123456
Your Secret Key is: qwertyuioplkjhgf

Figure-7.11Mails Received

Data Owner Login:

The Data Owner has to login using the username and password received in the mail and upload the encrypted file. This encryption is done using AES algorithm and it is stored in the cloud which is accessible only to authorized users.

The file can be of any format like pdf, jpg, image, doc, etc. and the file can be uploaded from any location in the system i.e. desktop, drives etc.



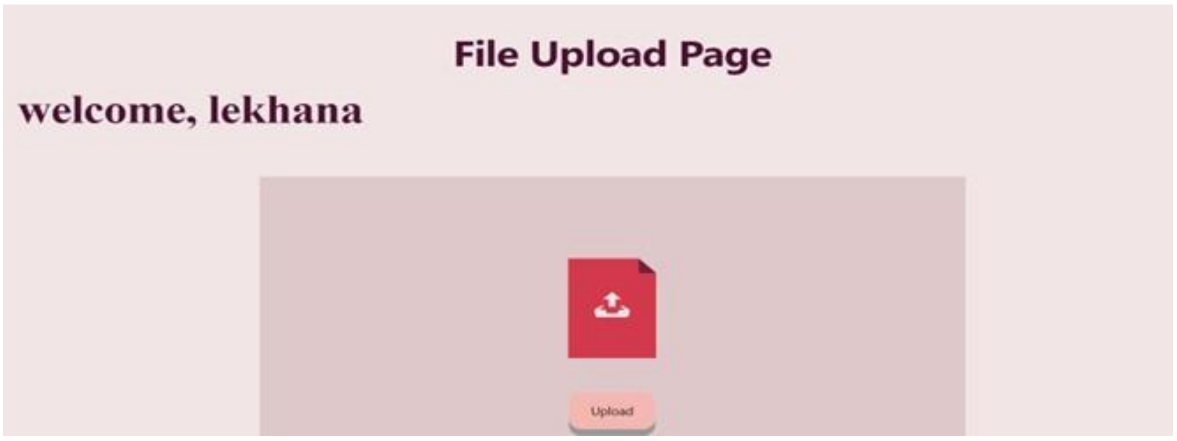
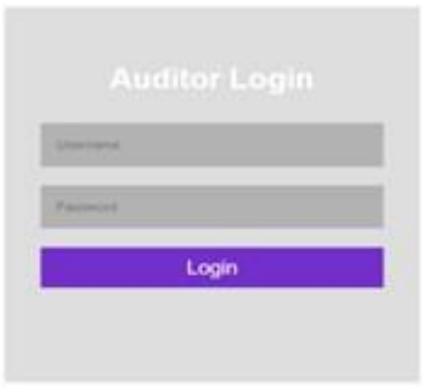


Figure-7.12 Data Owner login and Uploading file

Auditor login:

The auditor has to login using the username and password received in the mail and verify the integrity of the uploaded file by uploading the same file as uploaded the data owner. This verification is done using Hashing algorithm. The auditor who verifies the data has a unique id to maintain security in the cloud.



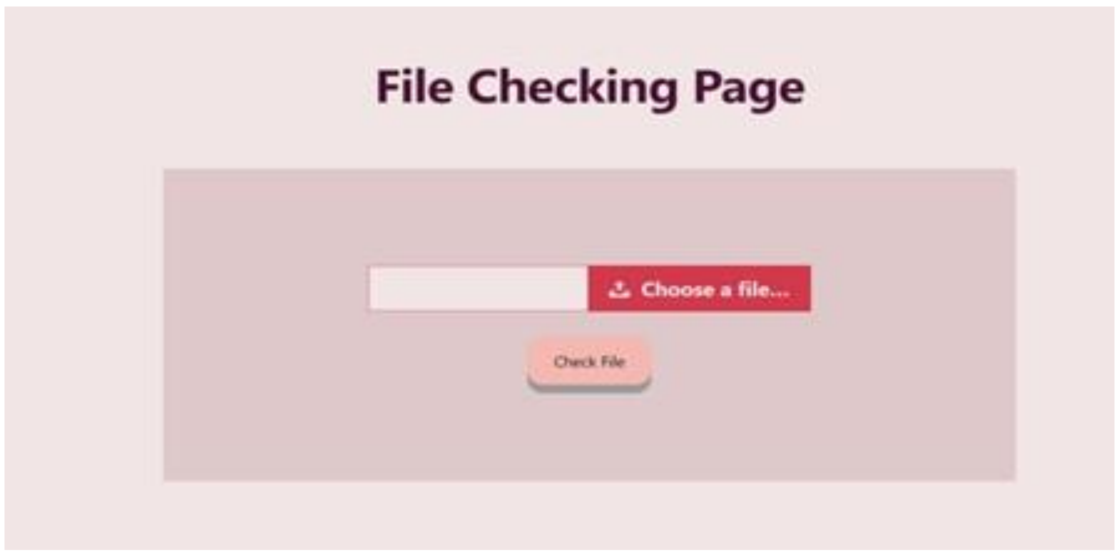


Figure-7.13 Auditor login and checking file

User Login:

The user has to login using the username and password received in the mail and enter the secret key to retrieve the file uploaded by the data owner.

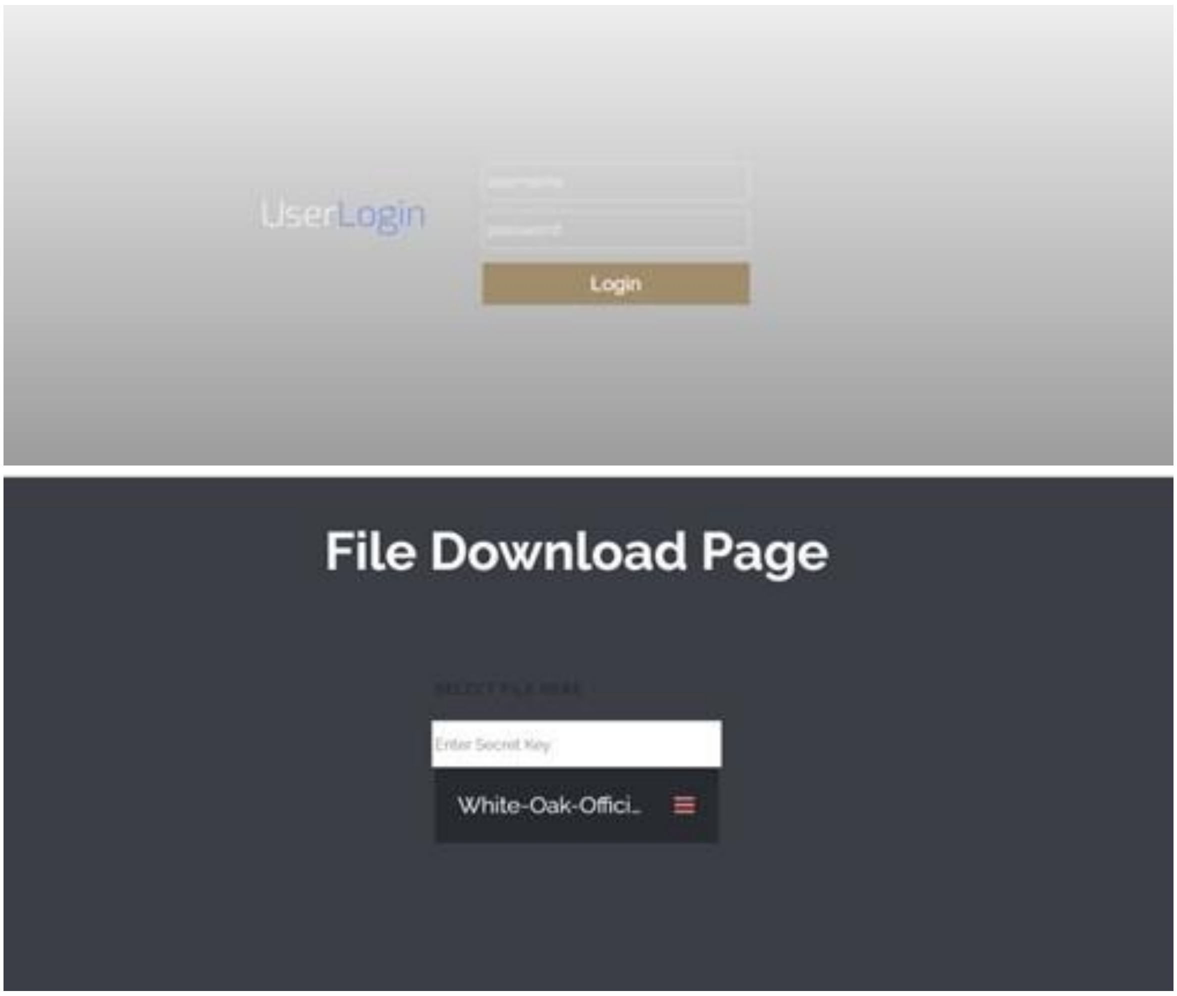


Figure-7.14 User login and Downloading file

User Revocation:

To revoke a user's access to the cloud, the admin logs into the user interface, navigates to the user revocation section, selects the respective user's name, and proceeds to delete the user's access privileges.

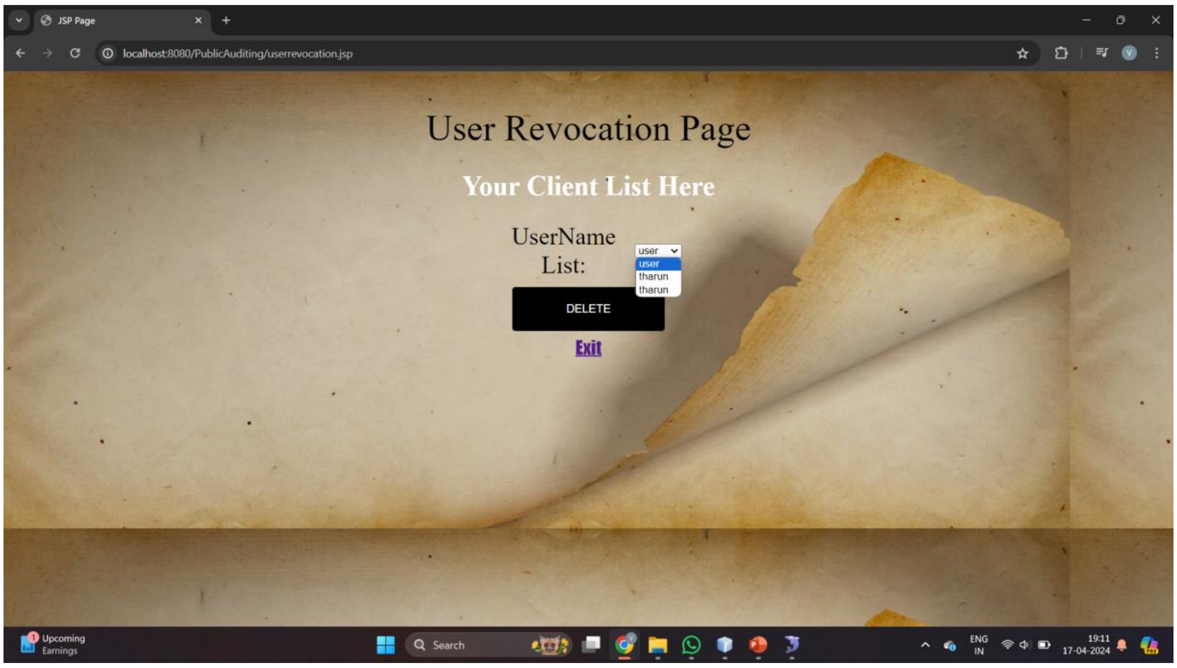


Figure-7.15 User Revocation

8. CONCLUSION & FUTURE SCOPE

8.1 Conclusion

In the rapidly evolving landscape of cloud computing, the proposed privacy-preserving public auditing mechanism stands as a pivotal advancement in fortifying the security of shared data. By harnessing cutting-edge technologies, this system not only addresses current challenges but also lays the groundwork for a resilient solution in the face of future threats. The integration of innovative cryptographic techniques ensures a heightened level of security, mitigating the risk of identity privacy leakage that has been a prevalent concern in existing mechanisms. This forward-looking approach positions the system as a robust safeguard for user identities, aligning with the dynamic nature of data sharing in contemporary cloud environments.

Furthermore, the system's commitment to efficiency is evident in its support for batch auditing, a feature that streamlines the verification process for multiple tasks concurrently. This not only enhances the overall responsiveness of the system but also tackles one of the critical limitations of traditional methods. Additionally, the exploration of traceability opens avenues for future enhancements, allowing for a deeper understanding of signer identities in specific scenarios. In essence, the proposed mechanism not only meets the current demands for secure shared data but also lays the groundwork for an adaptive and forward-thinking solution that anticipates and addresses the evolving challenges in cloud-based information management.

8.2 Future Scope

As a response, erasure coding as an alternative to backup has emerged as a method of protecting against drive failure. Raid just does not cut it in the age of high-capacity HDDs. The larger a disk's capacity, the greater the chance of bit error. And when a disk fails, the Raid rebuild process begins, during which there is no protection against a second (or third) mechanism failure. So not only has the risk of failure during normal operation grown with capacity, it is much higher during Raid rebuild, too. Also, rebuild times were once measured in minutes or hours, but disk transfer rates have not kept pace with the rate of disk capacity expansion, so large Raid rebuilds can now take days or even longer.

REFERENCES

- [1]. C. Wang, Q. Wang, K. Ren, and W. Lou, “Privacy-preserving public auditing for data storage security in cloud computing,” in 2010 proceedings ieee infocom. IEEE, 2010, pp. 1–9.

- [2]. H. Yuan, X. Chen, J. Li, T. Jiang, J. Wang, and R. Deng, “Secure cloud data deduplication with efficient re-encryption,” IEEE Transactions on Services Computing, 2019.

- [3]. J. Yu, K. Ren, C. Wang, and V. Varadharajan, “Enabling cloud storage auditing with key-exposure resistance,” Information Forensics and Security, IEEE Transactions on, vol. 10, no. 6, pp. 1167–1179, 2015.

- [4]. M. Ali, R. Dhamotharan, E. Khan, S. Khan, A. Vasilakos, K. Li, and A. Zomaya, “Sedasc: Secure data sharing in clouds,” IEEE Systems Journal, vol. 11, no. 2, pp. 395–404, 2017.

- [5]. R. S. Bali and N. Kumar, “Secure clustering for efficient data dissemination in vehicular cyberphysical systems,” Future Generation Computer Systems, pp. 476–492, 2016.

- [6]. S. Challa, A. K. Das, V. Odelu, N. Kumar, S. Kumari, M. K. Khan, and A. V. Vasilakos, “An efficient ecc-based provably secure threefactor user authentication and key agreement protocol for wireless healthcare sensor networks,” Computers and Electrical Engineering, vol. 69, pp. 534–554, 2018.

- [7]. S. Gordon, X. Huang, A. Miyaji, C. Su, K. Sumongkayothin, and K. Wipusitwarakun, “Recursive matrix oblivious ram: An oram construction for constrained storage devices,” IEEE Transactions on Information Forensics and Security, vol. 12, no. 12, pp. 3024–3038, 2017.

- [8]. S. Zarandioon, D. D. Yao, and V. Ganapathy, “K2c: Cryptographic cloud storage with lazy revocation and anonymous access,” in International Conference on Security and Privacy in Communication Systems. Springer, 2011, pp. 59–76.

- [9]. X. Chen, X. Huang, J. Li, J. Ma, W. Lou, and D. S. Wong, “New algorithms for secure outsourcing of large-scale systems of linear equations,” IEEE transactions on information forensics and security, vol. 10, no. 1, pp. 69–78, 2014.

- [10]. X. Zhang, Y. Tang, H. Wang, C. Xu, Y. Miao, and H. Cheng, “Lattice-based proxy-oriented identity-based encryption with keyword search for cloud storage,” Information Sciences, vol. 494, pp. 193–207, 2019.