isrjournals

International Journal of Advanced Research in Computer Science Engineering and Information Technology

Volume: 6, Issue: 3,Special Issue: 2 ,Apr,2023 ,ISSN_NO: 2321-3337

EFFECTIVE MULTI CLOUD AND MULTI LEVEL SECURED DATA STORAGE AND RETRIEVAL SYSTEM

Harishbabu.R.L[1], Katari Lakshmipathi[2], Pavan kumar.D[3],Priya.V[4]

UG Scholars[1 2 3]- Department of CSE, GRT Institute of Engineering And Technology, Tiruttani, India.
Priya. V[4] Asst professor- Department of CSE, GRT Institute of Engineering And Technology, Tiruttani, India.

harishbabu00000@gmail.com, *katarilakshmipathi615@gmail.com*, *dhonadhipavan838@gmail.com* , *priyaviswanath87@gmail.com*,

ABSTRACT - In accordance with more data storage needs turning over to the cloud, finding a secure and efficient data access structure has become a major research issue. In the proposed system, data is encrypted and storing it in cloud. As per our proposed system we implement the multi security levels **High security**, **Medium Security, Low Security**. In High Security it measures involve data encryption and splitting the information into two parts, which are then stored in separate cloud for added protection. In Medium Security the data is encrypted and stored in a single cloud. In Low Security For standard security, data is encrypted and stored as usual in single cloud. When the User needs to Retrieval the data, that data is rearrange and to decrypt the data.

Keywords : Encryption, Secure, Cloud.

1. INTRODUCTION

The rise of cloud computing is attributed to its capability of delivering software, infrastructure, and platform services without significant investments and operational expenses. Cloud services encompass service providers, resource providers, and users, comprising applications delivered as services and the systems facilitating such services. Collaborative cloud computing has gained traction in recent times, and it is predicted to be the next big trend in cloud computing. This technology leverages information technology as a network service to offer users with powerful computational capabilities and vast memory space at a low cost. In addition to affordability, collaborative cloud computing also addresses environmental concerns, such as carbon emissions, by advocating better resource management. However, trust computing, response speed, and automatic resource matching are some of the challenges that collaborative cloud computing faces. Addressing these issues will require new approaches such as holistic design, cooperative strategies, and distribution infrastructures.

2.PROPOSED SYSTEM

The proposed system ensures the safety of data by encrypting it before storing it in the cloud. High-level security measures are implemented by splitting the data into two parts and storing them in two separate locations. For medium-level security, the data is encrypted and stored in a single cloud, while normal security encrypts and stores the data without any additional measures.

2.1. MODULES

Modular design is a beneficial approach in software development that helps to reduce complexity and facilitate changes, which is crucial for software maintainability. It also enables parallel development of different parts of the system, making implementation easier. Effective modularity allows developers to compartmentalize functions and simplify interfaces, resulting in software that is easier to develop. Software architecture that embodies modularity involves dividing the software into separately named and addressable components known as modules, which are then integrated to meet the requirements of the problem. Software modularity is a crucial characteristic that makes programs easier to comprehend and maintain. Evaluating a design method's effectiveness in creating a useful modular design involves five critical criteria. They are decomposability, composability, understandability, continuity, and protection. To complete a project while improving the existing system and enabling future enhancements, the following modules have been planned as part of the proposed system. These modules have been designed with modularity in mind to facilitate easier implementation, maintenance, and change management.

2.2. USER INTERFACE:

This module focuses on creating a user interface that enables the data owner to upload their data onto a server. Its primary aim is to provide a means of storing and sharing data by facilitating the upload of files to a remote machine.

2.3. CLOUD SERVER:

The cloud server enables data owners to upload their data and respond to requests for specific files. The main cloud server manages both the upload of data and requests for files. When a request for a file is received, the main server communicates with the data owner to ensure that the request is authorized. Only after receiving approval from the data owner, the files are retrieved and made available to the requester.

2.4. MERKLE HASH TREE:

This module focuses on securing data before uploading it to the server. First, the data is encrypted and hashed for added security. After this process, the data is uploaded and split into multiple parts using the Merkle Hash Tree algorithm. The Merkle Tree is a data structure where each leaf node is associated with the hash value of a data block, and each non-leaf node is labeled with the cryptographic hash of its child nodes. This type of tree allows for the verification of the contents of large data structures in a secure and efficient manner. Hash trees are a more advanced version of hash lists and hash chains and are commonly used in cryptography and computer science for secure data storage and retrieval.

2.5.DATA ENCRYPTION:

The data that is uploaded to the cloud is protected by being encrypted using the AES algorithm, which ensures secure data communication. When the data is downloaded from the cloud, it can only be accessed with the correct decryption key.

2.6. DATA CHUNKING:

Data chunking is a technique used to break up large files or data into smaller parts or chunks to facilitate more secure transactions. The primary goal of data chunking is to ensure secure data storage and access, as well as make data transfer more efficient. By breaking data into smaller parts, it becomes easier to transmit and store data in a more secure manner. Furthermore, this technique can also help with data recovery as smaller chunks of data are less likely to get corrupted.

2.7. HIGH SECURE:

The user's data is encrypted before being stored in the cloud to ensure security. To achieve highly secure storage, the data is split or chunked into two parts, and each part is stored on separate cloud servers. We have used Dropbox and Google Drive as the cloud services in our implementation. This process offers a significant advantage because the two separate parts of the encrypted data are stored on two different clouds, ensuring highly secure data access.

2.8. MEDIUM SECURE:

The data uploaded by the user is subjected to encryption and then saved on the cloud. The data is split into multiple parts using chunking technique and then stored in a single cloud server for medium-level security. Dropbox is used as the cloud service in this implementation. The primary benefit of this method is to store the encrypted data in two different parts in a single cloud server for easy data access with medium-level security.

2.9.LOW SECURE:

The data uploaded by the user is encrypted and saved onto a single cloud server for low-security storage. Dropbox is the cloud service used in this implementation. The main benefit of this approach is that the file is stored in a single location in an encrypted format, providing low-security data access.

2.10. BLOCK CHAIN INTEGRATION:

The Project incorporates Block Chain technology to enhance the security of data in addition to the three levels of Secured Data Access. Block Chain employs four different algorithms - Asymmetric Key Encryption, Digital Signature, Secured Hash Algorithm, and Merkle Hash Root Algorithm - to provide highly secure access to the data. This integration of Block Chain technology is intended to strengthen the security measures and ensure the integrity of the data.

3. SYSTEM TESTING

In software development, testing is an essential process to verify whether the system has been developed as per the specified requirements and if it produces the expected results. There are two primary categories of testing methods, namely White Box Testing and Black Box Testing. Both these categories have multiple types of testing methods associated with them. The aim of testing is to identify any defects or discrepancies that may exist in the system and to ensure that it performs as expected.

3.1. TAXONOMY OF TESTING

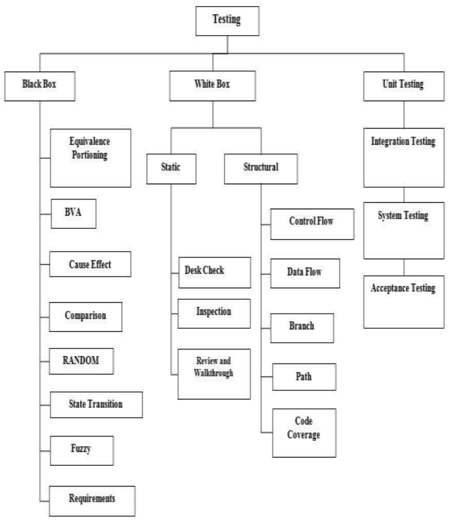


Fig 3.1 Represents taxonomy of testing

3.2 TESTING IN PARTICULAR

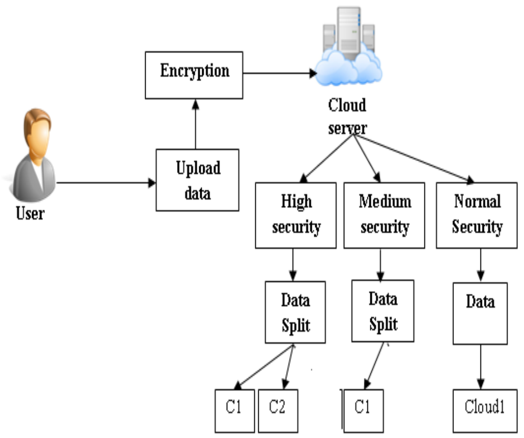
3.2.1. UNIT TESTING

Unit testing is an approach used to test individual units of a software application or program. These units include source code, computer program modules, usage procedures, control data, and operating procedures. The main objective of unit testing is to determine if each of these units is suitable for use by executing each statement properly. Each unit of the program is tested in different computer systems to ensure consistency in the project's results.

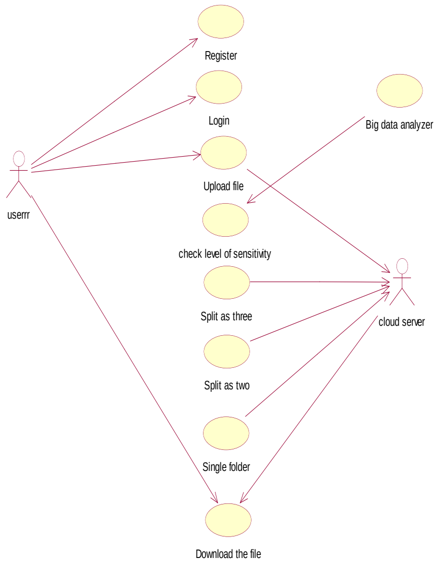
3.2.2. INTEGRATION TESTING

Integration testing is a critical phase of software testing that comes after unit testing and before validation testing. In this phase, individual software modules are combined and tested as a group. The process involves taking the modules that have already passed unit testing, grouping them into larger aggregates, and applying tests outlined in an integration test plan to those aggregates. The outcome of integration testing is a fully integrated system that is then ready for system testing. This testing approach ensures that the software modules work well together and that the integrated system functions as expected.

4. Architecture diagram

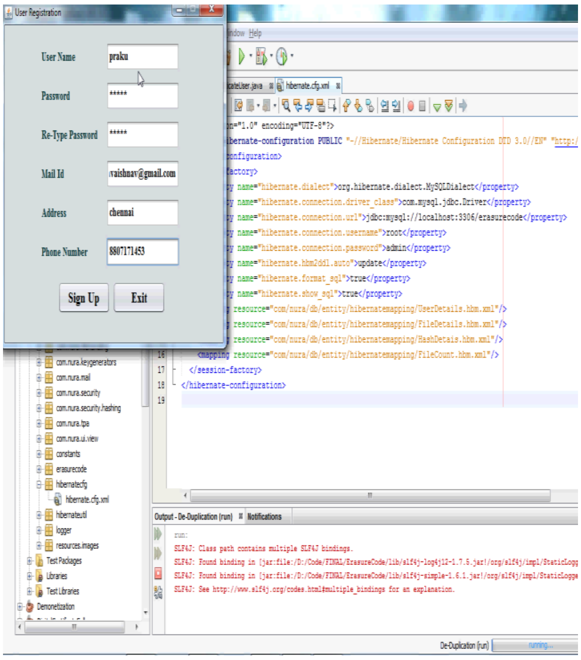


4.1. UML CASE DIAGRAMS



4.RESULTS AND DISCUSSION

SCREENSHOTS :

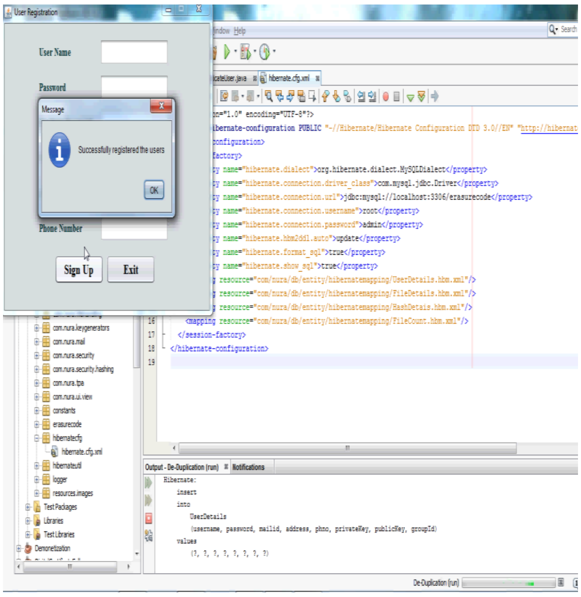


User Name praku
Password *****
Re-Type Password *****
Mail Id vaishnav@gmail.com
Address chennai
Phone Number 8807171453

```
n="1.0" encoding="UTF-8"?>
ibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http:/
configuration>
factory>
y name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
y name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
y name="hibernate.connection.url">jdbc:mysql://localhost:3306/erasurecode</property>
y name="hibernate.connection.username">root</property>
y name="hibernate.connection.password">admin</property>
y name="hibernate.hbm2ddl.auto">update</property>
y name="hibernate.format_sql">true</property>
y name="hibernate.show_sql">true</property>
 resource="com/nura/db/entity/hibernatemapping/UserDetails.hbm.xml"/>
 resource="com/nura/db/entity/hibernatemapping/FileDetails.hbm.xml"/>
 resource="com/nura/db/entity/hibernatemapping/HashDetais.hbm.xml"/>
    <mapping resource="com/nura/db/entity/hibernatemapping/FileCount.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

```
run:
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/D:/Code/FINAL/ErasureCode/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLogg
SLF4J: Found binding in [jar:file:/D:/Code/FINAL/ErasureCode/lib/slf4j-simple-1.6.1.jar!/org/slf4j/impl/StaticLogge
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
```

4.1 Register page

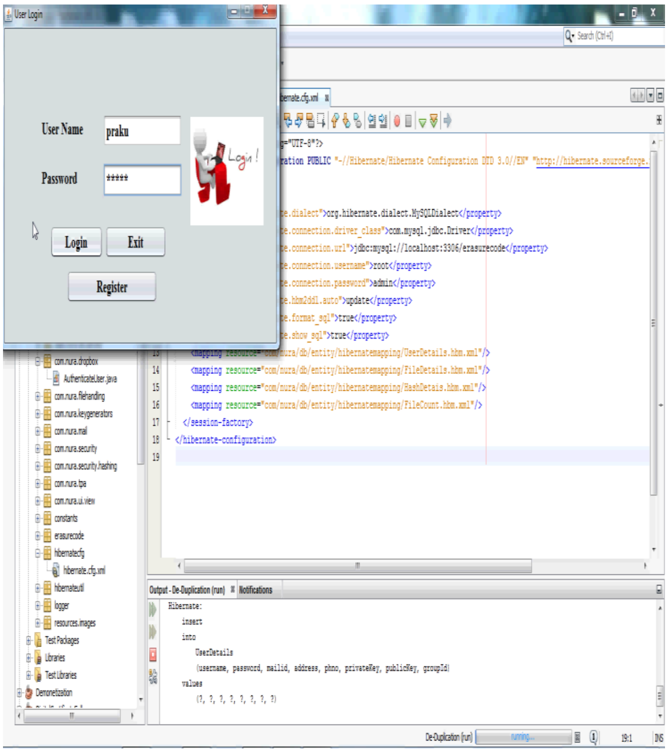


Successfully registered the users

```
n="1.0" encoding="UTF-8"?>
ibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hibernat
configuration>
factory>
y name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
y name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
y name="hibernate.connection.url">jdbc:mysql://localhost:3306/erasurecode</property>
y name="hibernate.connection.username">root</property>
y name="hibernate.connection.password">admin</property>
y name="hibernate.hbm2ddl.auto">update</property>
y name="hibernate.format_sql">true</property>
y name="hibernate.show_sql">true</property>
 resource="com/nura/db/entity/hibernatemapping/UserDetails.hbm.xml"/>
 resource="com/nura/db/entity/hibernatemapping/FileDetails.hbm.xml"/>
 resource="com/nura/db/entity/hibernatemapping/HashDetais.hbm.xml"/>
    <mapping resource="com/nura/db/entity/hibernatemapping/FileCount.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

```
Hibernate:
    insert
    into
        UserDetails
        (username, password, mailid, address, phno, privateKey, publicKey, groupId)
    values
        (?, ?, ?, ?, ?, ?, ?, ?)
```

4.2 Successfully registered

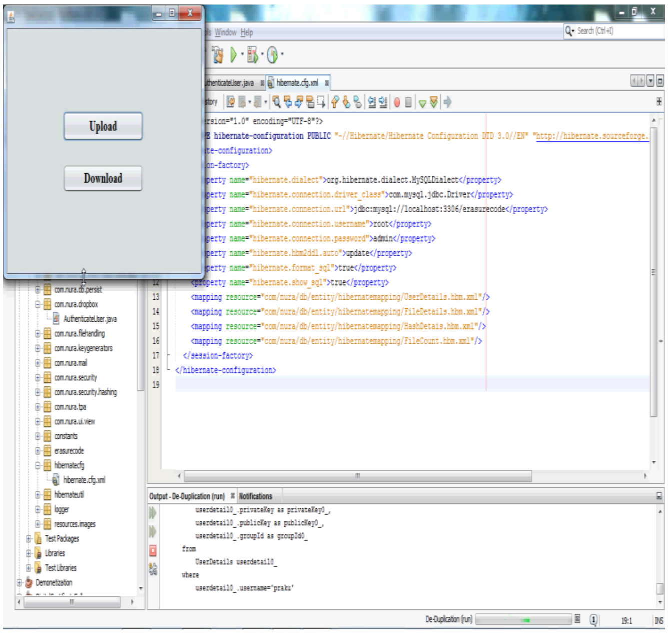


User Name praku
Password *****

```
g="UTF-8"?>
ration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hibernate.sourceforge.
e.dialect">org.hibernate.dialect.MySQLDialect</property>
e.connection.driver_class">com.mysql.jdbc.Driver</property>
e.connection.url">jdbc:mysql://localhost:3306/erasurecode</property>
e.connection.username">root</property>
e.connection.password">admin</property>
e.hbm2ddl.auto">update</property>
e.format_sql">true</property>
e.show_sql">true</property>
    <mapping resource="com/nura/db/entity/hibernatemapping/UserDetails.hbm.xml"/>
    <mapping resource="com/nura/db/entity/hibernatemapping/FileDetails.hbm.xml"/>
    <mapping resource="com/nura/db/entity/hibernatemapping/HashDetais.hbm.xml"/>
    <mapping resource="com/nura/db/entity/hibernatemapping/FileCount.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

```
Hibernate:
    insert
    into
        UserDetails
        (username, password, mailid, address, phno, privateKey, publicKey, groupId)
    values
        (?, ?, ?, ?, ?, ?, ?, ?)
```

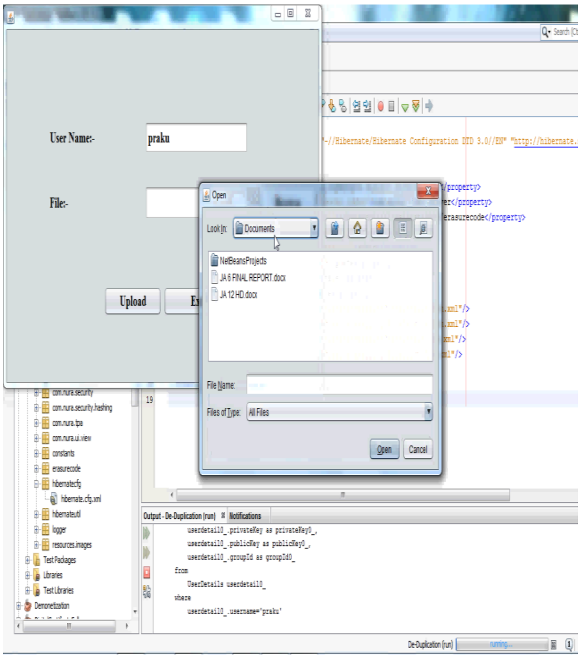
4.3 Login page



```
rsion="1.0" encoding="UTF-8"?>
E hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hibernate.sourceforge.
te-configuration>
on-factory>
perty name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
perty name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
perty name="hibernate.connection.url">jdbc:mysql://localhost:3306/erasurecode</property>
perty name="hibernate.connection.username">root</property>
perty name="hibernate.connection.password">admin</property>
perty name="hibernate.hbm2ddl.auto">update</property>
perty name="hibernate.format_sql">true</property>
<property name="hibernate.show_sql">true</property>
    <mapping resource="com/nura/db/entity/hibernatemapping/UserDetails.hbm.xml"/>
    <mapping resource="com/nura/db/entity/hibernatemapping/FileDetails.hbm.xml"/>
    <mapping resource="com/nura/db/entity/hibernatemapping/HashDetais.hbm.xml"/>
    <mapping resource="com/nura/db/entity/hibernatemapping/FileCount.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

```
        userdetail0_.privateKey as privateKey0_,
        userdetail0_.publicKey as publicKey0_,
        userdetail0_.groupId as groupId0_
    from
        UserDetails userdetail0_
    where
        userdetail0_.username='praku'
```

4.4 Represents to upload


User Name:- praku

File:-

Open

Look In: Documents

NetBeansProjects

JA 6 FINAL REPORT.docx

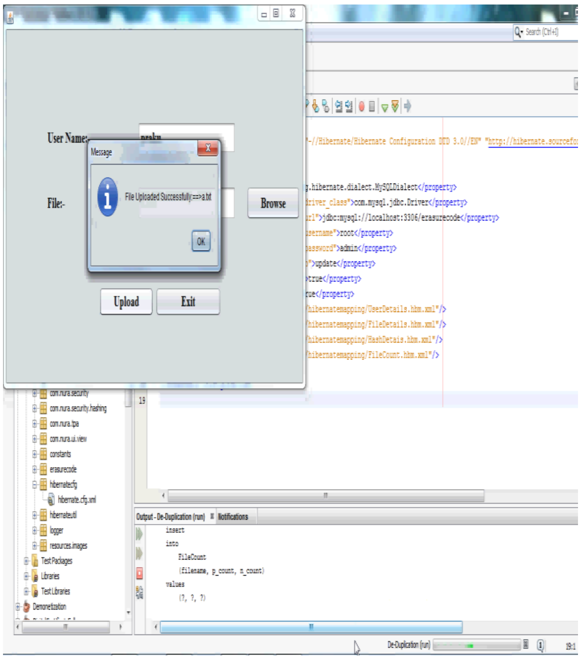
JA 12 HD.docx

File Name:

Files of Type: All Files

```
Output - De-Duplication (run)
        userdetail0_.privateKey as privateKey0_,
        userdetail0_.publicKey as publicKey0_,
        userdetail0_.groupId as groupId0_
    from
        UserDetails userdetail0_
    where
        userdetail0_.username='praku'
```

4.5 Searching the document


User Name:- praku

File:-

Message

File Uploaded Successfully==>a.txt

```
-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hibernate.sourcefo
.hibernate.dialect.MySQLDialect</property>
river_class">com.mysql.jdbc.Driver</property>
rl">jdbc:mysql://localhost:3306/erasurecode</property>
sername">root</property>
assword">admin</property>
">update</property>
>true</property>
rue</property>
hibernatemapping/UserDetails.hbm.xml"/>
hibernatemapping/FileDetails.hbm.xml"/>
hibernatemapping/HashDetais.hbm.xml"/>
hibernatemapping/FileCount.hbm.xml"/>
```

```
Output - De-Duplication (run)
    insert
    into
        FileCount
        (filename, p_count, n_count)
    values
        (?, ?, ?)
```

4.6 Uploaded successfully

5. CONCLUSION:

Hence, the article concludes that the storage of a user's data on the cloud will depend on the level of sensitivity of the information. Although users tend to store a large amount of data on the cloud, the security measures for that data are insufficient. Therefore, we have implemented a three-tiered approach to ensure the security of the stored data. In future we can add or update multi cloud for medium security level. In high security level we can add or update multiple clouds.

6.REFERENCES:

[1] A. Sahai and B. Waters, “Fuzzy identity-based encryption,” in Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 457–473, Springer, 2005.

[2] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 89–98, Acm, 2006.

[3] L. Cheung and C. Newport, “Provably secure ciphertext policy ABE,” in Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 456–465, ACM, 2007.

[4] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext - policy attributebased encryption,” in 2007 IEEE Symposium on Security and Privacy (SP’07), pp. 321–334, IEEE, 2007.

[5] D. F. Ferraiolo and D. R. Kuhn, "Role-based access controls," arXiv preprint arXiv:0903.2171, 2009.

[6] A. C. OConnor and R. J. Loomis, "2010 economic analysis of role-based access control," NIST, Gaithersburg, MD, vol. 20899, 2010.

[7] A. Elliott and S. Knight, "Role explosion: Acknowledging the problem.," in Software Engineering Research and Practice, pp. 349–355, 2010.

[8] G. Wang, Q. Liu, J. Wu, and M. Guo, "Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers," Computers & Security, vol. 30, no. 5, pp. 320–331, 2011.

[9] P. Mell and T. Grance, "The NIST definition of cloud computing," 2011.

[10] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," in Theory of Cryptography Conference, pp. 253–273, Springer, 2011.

[11] M. Lichman, "UCI machine learning repository," 2013.

[12] S. Wang, J. Zhou, J. K. Liu, J. Yu, J. Chen, and W. Xie, "An efficient file hierarchy attribute-based encryption scheme in cloud computing," IEEE Transactions on Information Forensics and Security, vol. 11, no. 6 pp. 1265–1277, 2016.

[13] P. Institute, "Sixth annual benchmark study on privacy and security of healthcare data," tech. rep., Ponemon Institute LLC, 2016.

[14] R. Cohen, "The cloud hits the mainstream: More than half of U.S. businesses now use cloud computing." http://www.forbes.com, April 2013. Online; posted 10-January-2017.

[15] E. Zaghloul, T. Li, and J. Ren, "An attribute-based distributed data sharing scheme," in IEEE Globeocm 2019, (Abu Dhabi, UAE.), 9-13 December 2018.