

RESOLVING DATA DE-DUPLICATION ISSUES ON CLOUD

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and
Engineering

by

KURLA HARI VISWANADH (Reg. No. 37110396)

YALLA MANI DEEPAK (Reg. No. 37110426)



**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING SCHOOL OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI – 600 119

MARCH - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with “A” grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **KURLA HARI VISWANADH (Reg. No. 37110396)** and **YALLA MANI DEEPAK (Reg. No. 37110426)** who carried out the project entitled “**RESOLVING DATA DE-DUPLICATION ISSUE ON CLOUD**” under my supervision from **August 2020** to **March 2021**.

Internal Guide

DR.T. SASIKALA, M.E, PH. D.,

Head of the Department

Submitted for Viva voce Examination held on

Internal Examiner

Examiner

External

DECLARATION

I **KURLA HARI VISWANADH** hereby declare that the Project Report entitled “**RESOLVING DATA DE-DUPLICATION ISSUES ON CLOUD**” is done by me under the guidance of **DR. T. SASIKALA, M.E., PH. D.**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean**, School of Computing, **Dr. S. Vigneswari, M.E., Ph.D.,** and **Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **DR. T. SASIKALA. M.E., PH. D, DEAN** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Data deduplication is one of important data compression techniques for eliminating duplicate copies of repeating data, and has been widely used in cloud storage to reduce the amount of storage space and save bandwidth. To protect the confidentiality of sensitive data while supporting deduplication, the convergent encryption technique has been proposed to encrypt the data before outsourcing. To better protect data security, this paper makes the first attempt to formally address the problem of authorized data deduplication. Different from traditional deduplication systems, the differential privileges of users are further considered in duplicate check besides the data itself. We propose hashing technique split the data into fragments and their fragment is matched with others data, then the matched data is denoted for previous file name and extra content added as a chunk. We also present several new deduplication constructions supporting authorized duplicate check in hybrid cloud architecture. Security analysis demonstrates that our scheme is secure in terms of the definitions specified in the proposed security model. As a proof of concept, we implement a prototype of our proposed authorized duplicate check scheme and conduct test bed experiments using our prototype. We show that our proposed authorized duplicate check scheme incurs minimal overhead compared to normal operations. The security requirements of data confidentiality and tag consistency are also achieved by introducing a deterministic secret sharing scheme in distributed storage systems, instead of using convergent encryption as in previous deduplication systems.

TABLE OF CONTENT

S.NO	TITLE	PAGE.NO
	ABSTRACT	ii
	LIST OF FIGURES	viii
1	INTRODUCTION	01
	1.1 PROBLEM STATEMENT	01
	1.2 SYNOPSIS	01
	1.3 OBJECTIVE	04
	1.4 SCOPE OF THE PROJECT	04
2	LITERATURE REVIEW	05
3	SYSTEM DESIGN AND METHODOLGY	10
	3.1 EXISTING SYSTEM	10
	3.2 PROPOSED SYSTEM	10
	3.3 SOFTWARE REQUIREMENTS	11
	3.4 .NET FRAMEWORK	12
	3.6 SQL SERVER	16
	3.7 MD5 AND HASH ALGORITHMS	17
4	SOFTWARE DEVELOPMENT METHODOLOGY	20
	4.1 SYSTEM DESIGN	20
	4.2 UML DIAGRAM	20
	4.3 ARCHITECTURE	23
	4.4 DATA FLOW DIAGRAM	24
	4.5 MODULE DESCRIPTION	25
5	RESULTS AND DISCUSSION	27
	5.1 RESULT	27
6	CONCLUSION AND FUTURE ENHANCEMENT	28
	6.1 CONSLUSION	28
	6.2 FUTURE ENHANCEMENT	28

	REFERENCE	29
	APPENDIX	31
	SOURCE CODE	31
	SCREENSHOTS	61
	JOURNAL PAPER	65
	PLAGIARISM REPORT	76

LIST OF FIGURES

S.NO	TITLE	PAGE.NO
5.1.1	.NET FRAMEWORK	14
5.3.1	FLOW CHART FOR HASH VALUE	18
4.2.1	USE CASE DIAGRAM	21
4.2.2	CLASS DIAGRAM	21
4.2.3	SEQUENCE DIAGRAM	22
4.2.4	COLLABRATION DIAGRAM	22
4.2.5	ER DIAGRAM	23
4.3.1	ARCHITECTURE	23
4.3.2	SYSTEM DESIGN ARCHITECTURE	24
4.4.1	DATA FLOW DIAGRAM	24
4.5.1	USER MODULE	25
4.5.2	SERVER STARTUP	25
4.5.3	DE-DUPLICATION SYSTEM	26
4.5.4	DOWNLOAD FILE	26
7.1	HOME PAGE	61
7.2	REGISTER PAGE	61
7.3	UPLOAD PAGE	62
7.4	USER REQUEST PAGE	62
7.5	KEY GENERATION	63
7.6	FILE DOWNLOAD	63
7.7	DATABASE REGISTERED USERS	64
7.8	DATA	64

CHAPTER-1

INTRODUCTION

1.1 PROBLEM STATEMENT

One critical challenge of cloud storage services is the management of the ever-increasing volume of data. By using the cloud computing there is no de-duplication process in the existing so that we can't avoid duplication in older file or block level. This paper makes the first attempt to formally address the problem of authorized data de duplication.

1.2 SYNOPSIS:

Cloud computing provides seemingly unlimited “virtualized” resources to users as services across the whole Internet, while hiding platform and implementation details. Today's cloud service providers offer both highly available storage and massively parallel computing resources at relatively low costs. As cloud computing becomes prevalent, an increasing amount of data is being stored in the cloud and shared by users with specified *privileges*, which define the access rights of the stored data. One critical challenge of cloud storage services is the management of the ever-increasing volume of data. To make data management scalable in cloud computing, de duplication has been a well-known technique and has attracted more and more attention recently.

Data deduplication is a specialized data compression technique for eliminating duplicate copies of repeating data in storage. The technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent. Instead of keeping multiple data copies with the same content, deduplication eliminates redundant data by keeping only one physical copy and referring other redundant data to that copy. Deduplication can take place at either the file level or the block level. For file level deduplication, it eliminates duplicate copies of the same file. Deduplication can also

take place at the block level, which eliminates duplicate blocks of data that occur in non-identical files. Although data deduplication brings a lot of benefits, security and privacy concerns arise as users' sensitive data are susceptible to both insider and outsider attacks. Traditional encryption, while providing data confidentiality, is incompatible with data deduplication. Specifically, traditional encryption requires different users to encrypt their data with their own keys. Thus, identical data copies of different users will lead to different ciphertexts, making deduplication impossible.

Convergent encryption has been proposed to enforce data confidentiality while making deduplication feasible. It encrypts/decrypts a data copy with a convergent key, which is obtained by computing the cryptographic hash value of the content of the data copy. After key generation and data encryption, users retain the keys and send the ciphertext to the cloud. Since the encryption operation is Deterministic and is derived from the data content, identical data copies will generate the same convergent key and hence the same cipher text. To prevent unauthorized access, a secure proof of ownership protocol is also needed to provide the proof that the user indeed owns the same file when a duplicate is found. After the proof, subsequent users with the same file will be provided a pointer from the server without needing to upload the same file. A user can download the encrypted file with the pointer from the server, which can only be decrypted by the corresponding data owners with their convergent keys.

Thus, convergent encryption allows the cloud to perform deduplication on the cipher texts and the proof of ownership prevents the unauthorized user to access the file. However, previous deduplication systems cannot support differential authorization duplicate check, which is important in many applications. In such an authorized deduplication system, each user is issued a set of privileges during system initialization. Each file uploaded to the cloud is also bounded by a set of privileges to specify which kind of users is allowed to perform the duplicate check and access the files. Before submitting his duplicate check request for some file, the user needs to take this file and his own privileges as inputs. The user is able to find a duplicate for this file if and only if there is a copy of this file and a matched privilege stored in cloud. For example, in a company, many different privileges will be assigned to employees.

In order to save cost and efficiently management, the data will be moved to the storage server provider (SCSP) in the public cloud with specified privileges and the deduplication technique will be applied to store only one copy of the same file. Because of privacy consideration, some files will be encrypted and allowed the duplicate check by employees with specified privileges to realize the access control. Traditional de duplication systems based on convergent encryption, although providing confidentiality to some extent; do not support the duplicate check with differential privileges. In other words, no differential privileges have been considered in the deduplication based on convergent encryption technique. It seems to be contradicted if we want to realize both deduplication and differential authorization duplicate check at the same time.

1.3 OBJECTIVE:

The main goal is to enable de duplication and distributed storage of the data across multiple storage servers.

1.4 SCOPE:

Data deduplication techniques are widely employed to backup data and minimize network and storage overhead by detecting and eliminating and redundancy among data.

CHAPTER -2

LITERATURE SURVEY

Literature survey is the most important step in software development process. Before developing the tool, it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then the next step is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system, the above consideration is taken into account for developing the proposed system.

The major part of the project development sector considers and fully survey all the required needs for developing the project. For every project Literature survey is the most important sector in software development process. Before developing the tools and the associated designing it is necessary to determine and survey the time factor, resource requirement, man power, economy, and company strength. Once these things are satisfied and fully surveyed, then the next step is to determine about the software specifications in the respective system such as what type of operating system the project would require, and what are all the necessary software are needed to proceed with the next step such as developing the tools, and the associated operations.

TITLE 1: SECURE OVERLAY CLOUD STORAGE WITH ACCESS CONTROL AND ASSURED DELETION

- We can now outsource data backups off-site to third-party cloud storage services so as to reduce data management costs.
- However, we must provide security guarantees for the outsourced data, which is now maintained by third parties. We design and implement FADE, a secure overlay cloud storage system that achieves fine-grained, policy-based access control and file assured deletion.
- It associates outsourced files with file access policies, and assuredly deletes files to make them unrecoverable to anyone upon revocations of file access policies.

- To achieve such security goals, FADE is built upon a set of cryptographic key operations that are self-maintained by a quorum of key managers that are independent of third-party clouds. In particular, FADE acts as an overlay system that works seamlessly atop today's cloud storage services.
- We implement a proof-of-concept prototype of FADE atop Amazon S3, one of today's cloud storage services. We conduct extensive empirical studies, and demonstrate that FADE provides security protection for outsourced data, while introducing only minimal performance and monetary cost overhead.
- Our work provides insights of how to incorporate value-added security features into today's cloud storage services.

Advantage

A secure overlay cloud storage system that achieves fine-grained, policy-based access control and file assured deletion.

To achieve such security goals, FADE is built upon a set of cryptographic key operations that are self-maintained by a quorum of key managers that are independent of third-party clouds

Disadvantage

While we can now outsource data backup to third-party cloud storage services so as to reduce data management costs, security concerns arise in terms of ensuring the privacy and integrity of outsourced data

TITLE 2: BUILDING CONFIDENTIAL AND EFFICIENT QUERY SERVICES IN THE CLOUD WITH RASP DATA PERTURBATION

- With the wide deployment of public cloud computing infrastructures, using clouds to host data query services has become an appealing solution for the advantages on scalability and cost-saving.
- However, some data might be sensitive that the data owner does not want to move to the cloud unless the data confidentiality and query privacy are guaranteed.

- On the other hand, a secured query service should still provide efficient query processing and significantly reduce the in-house workload to fully realize the benefits of cloud computing.
- We propose the random space perturbation (RASP) data perturbation method to provide secure and efficient range query and kNN query services for protected data in the cloud.
- The RASP data perturbation method combines order preserving encryption, dimensionality expansion, random noise injection, and random projection, to provide strong resilience to attacks on the perturbed data and queries.
- It also preserves multidimensional ranges, which allows existing indexing techniques to be applied to speedup range query processing. The kNN-R algorithm is designed to work with the RASP range query algorithm to process the KNN queries.
- We have carefully analysed the attacks on data and queries under a precisely defined threat model and realistic security assumptions. Extensive experiments have been conducted to show the advantages of this approach on efficiency and security.

Advantage

The wide deployment of public cloud computing infrastructures, using clouds to host data query services has become an appealing solution for the advantages on scalability and cost-saving.

It also preserves multidimensional ranges, which allows existing indexing techniques to be applied to speedup range query processing.

Disadvantage

A new approach is needed to preserve data confidentiality and query privacy, the efficiency of query services and the benefits of using the clouds should also be preserved.

It will not be meaningful to provide slow query services as a result of security and privacy assurance. It is also not practical for the data owner to use a significant

number of in-house resources, because the purpose of using cloud resources is to reduce the need of maintaining scalable in-house infrastructures.

TITLE 3: CROWDSOURCING IN CYBER-PHYSICAL SYSTEMS: STOCHASTIC OPTIMIZATION WITH STRONG STABILITY

- Cyber-physical systems (CPSs), featuring a tight combination of computational and physical elements as well as communication networks, attracted intensive attention recently because of their wide applications in various areas.
- In many applications, especially those aggregating or processing a large amount of data over large spatial regions or long spans of time or both, the workload would be too heavy for any CPS element (or node) to finish on its own.
- How to enable the CPS nodes to efficiently collaborate with each other to accommodate more CPS services is a very challenging problem and deserves systematic research.
- In this paper, we present a cross-layer optimization framework for hybrid crowdsourcing in the CPSs to facilitate heavy-duty computation.
- Particularly, by joint computing resource management, routing, and link scheduling, we formulate an offline finite-queue-aware CPS service maximization problem to crowd source nodes' computing tasks in a CPS. We then find both lower and upper bounds on the optimal result of the problem.
- In addition, the lower bound result is proved to be a feasible result that guarantees all queues in the network are finite, i.e., network strong stability. Extensive simulations have been conducted to validate the proposed algorithms' performance.

Advantage

It finds both lower and upper bounds on the optimal result of the problem. Moreover, the lower bound result is proved to be a feasible result which guarantees that all queues in the network are finite, i.e., network strong stability

Disadvantage

There are generally two issues associated with enabling the collaborations in a CPS: whom to collaborate with and how to collaborate, which are in correspondence to two problems: computing resource management and network design. It formulates an offline finite-queue-aware CPS service maximization problem to crowdsource nodes' computing tasks in a CPS.

TITLE 4: SECURE DE DUPLICATION WITH EFFICIENT AND RELIABLE CONVERGENT KEY MANAGEMENT

- Data de duplication is a technique for eliminating duplicate copies of data, and has been widely used in cloud storage to reduce storage space and upload bandwidth.
- Promising as it is, an arising challenge is to perform secure de duplication in cloud storage.
- Although convergent encryption has been extensively adopted for secure de duplication, a critical issue of making convergent encryption practical is to efficiently and reliably manage a huge number of convergent keys.
- This paper makes the first attempt to formally address the problem of achieving efficient and reliable key management in secure de duplication.
- We first introduce a baseline approach in which each user holds an independent master key for encrypting the convergent keys and outsourcing them to the cloud.
- However, such a baseline key management scheme generates an enormous number of keys with the increasing number of users and requires users to dedicatedly protect the master keys.
- To this end, we propose key, a new construction in which users do not need to manage any keys on their own but instead securely distribute the convergent key shares across multiple servers.
- Security analysis demonstrates that De-key is secure in terms of the definitions specified in the proposed security model.
- As a proof of concept, we implement De-key using the Ramp secret sharing scheme and demonstrate that De-key incurs limited overhead in realistic environments.

Advantage

It achieves efficient and reliable key management in secure deduplication. However, such a baseline key management scheme generates an enormous number of keys with the increasing number of users and requires users to dedicatedly protect the master keys.

To this end, we propose De-key, a new construction in which users do not need to manage any keys on their own but instead securely distribute the convergent key shares across multiple servers.

Disadvantage

Promising as it is, an arising challenge is to perform secure deduplication in cloud storage. Although convergent encryption has been extensively adopted for secure deduplication, a critical issue of making convergent encryption practical is to efficiently and reliably manage a huge number of convergent keys.

TITLE 5: A DATA ASSURED DELETION SCHEME IN CLOUD STORAGE

- In order to provide a practicable solution to data confidentiality in cloud storage service, a data assured deletion scheme, which achieves the fine-grained access control, hopping and sniffing attacks resistance, data dynamics and deduplication, is proposed.
- In our scheme, data blocks are encrypted by a two-level encryption approach, in which the control keys are generated from a key derivation tree, encrypted by an All-Or-Nothing algorithm and then distributed into DHT network after being partitioned by secret sharing.
- This guarantees that only authorized users can recover the control keys and then decrypt the outsourced data in an owner specified data lifetime.
- Besides confidentiality, data dynamics and deduplication are also achieved separately by adjustment of key derivation tree and convergent encryption.
- The analysis and experimental results show that our scheme can satisfy its security goal and perform the assured deletion with low cost.

CHAPTER-3

SYSTEM DESIGN AND METHODOLOGY

3.1 EXISTING SYSTEM

Data de duplication systems, the private cloud are involved as a proxy to allow data owner/users to securely perform duplicate check with differential privileges. Such architecture is practical and has attracted much attention from researchers. The data owners only outsource their data storage by utilizing public cloud while the data operation is managed in private cloud.

Data de duplication is a specialized data compression technique for eliminating duplicate copies of repeating data in storage. The technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent. Instead of keeping multiple data copies with the same content, de duplication eliminates redundant data by keeping only one physical copy and referring other redundant data to that copy.

Deduplication can take place at either the file level or the block level. For file level deduplication, it eliminates duplicate copies of the same file. Deduplication can also take place at the block level, which eliminates duplicate blocks of data that occur in non-identical files. Identical data copies of different users will lead to different cipher texts, making de duplication impossible.

3.1.1 DISADVANTAGE

- Traditional encryption, while providing data confidentiality, is incompatible with data de duplication.
- Identical data copies of different users will lead to different cipher texts, making de duplication impossible.

3.2 PROPOSED SYSTEM

In this paper, we enhance our system in security. Specifically, we present an advanced scheme to support stronger security by encrypting the file with differential privilege keys. In this way, the users without corresponding privileges cannot perform the duplicate check. Furthermore, such unauthorized users cannot decrypt the cipher text even collude with the S-CSP. Security analysis demonstrates that

our system is secure in terms of the definitions specified in the proposed security model.

Convergent encryption has been proposed to enforce data confidentiality while making de duplication feasible. It encrypts/decrypts a data copy with a convergent key, which is obtained by computing the cryptographic hash value of the content of the data copy. After key generation and data encryption, users retain the keys and send the ciphertext to the cloud. Since the encryption operation is deterministic and is derived from the data content, identical data copies will generate the same convergent key and hence the same cipher text. To prevent unauthorized access, a secure proof of ownership protocol is also needed to provide the proof that the user indeed owns the same file when a duplicate is found.

3.2.1 ADVANTAGES

- The user is only allowed to perform the duplicate check for files marked with the corresponding privileges.
- We present an advanced scheme to support stronger security by encrypting the file with differential privilege keys.
- Reduce the storage size of the tags for integrity check. To enhance the security of de duplication and protect the data confidentiality.

The software requirements specification is produced at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by establishing a complete information description as functional representation of system behavior, an indication of performance requirements and design constraints, appropriate validation criteria.

3.3 SOFTWARE REQUIREMENTS

Operating system	: Windows XP/7/8.1
IDE	: Visual Studio 2013
Coding Language	: .NET
Backend	: SQL Server 2012

3.4 .NET FRAMEWORK

The .NET Framework has two main parts:

1. The Common Language Runtime (CLR).
2. A hierarchical set of class libraries.

The CLR is described as the “execution engine” of .NET. It provides the environment within which programs run. The most important features are

- ◆ Conversion from a low-level assembler-style language, called Intermediate Language (IL), into code native to the platform being executed on.
- ◆ Memory management, notably including garbage collection.
- ◆ Checking and enforcing security restrictions on the running code.
- ◆ Loading and executing programs, with version control and other such features.

3.4.1 Managed Code

The code that targets .NET, and which contains certain extra Information - “metadata” - to describe itself. Whilst both managed and unmanaged code can run in the runtime, only managed code contains the information that allows the CLR to guarantee, for instance, safe execution and interoperability.

3.4.2 Managed Data

With Managed Code comes Managed Data. CLR provides memory allocation and Deal location facilities, and garbage collection. Some .NET languages use Managed Data by default, such as C#, Visual Basic.NET and JScript.NET, whereas others, namely C++, do not. Targeting CLR can, depending on the language you're using, impose certain constraints on the features available. As with managed and unmanaged code, one can have both managed and unmanaged data in .NET applications - data that doesn't get garbage collected but instead is looked after by unmanaged code.

3.4.3 Common Type System

The CLR uses something called the Common Type System (CTS) to strictly enforce type-safety. This ensures that all classes are compatible with each other,

by describing types in a common way. CTS define how types work within the runtime, which enables types in one language to interoperate with types in another language, including cross-language exception handling. As well as ensuring that types are only used in appropriate ways, the runtime also ensures that code doesn't attempt to access memory that hasn't been allocated to it.

3.4.4 Common Language Specification

The CLR provides built-in support for language interoperability. To ensure that you can develop managed code that can be fully used by developers using any programming language, a set of language features and rules for using them called the Common Language Specification (CLS) has been defined. Components that follow these rules and expose only CLS features are considered CLS-compliant.

3.4.5 CLASS LIBRARY

.NET provides a single-rooted hierarchy of classes, containing over 7000 types. The root of the namespace is called System; this contains basic types like Byte, Double, Boolean, and String, as well as Object. All objects derive from System.Object. As well as objects, there are value types. Value types can be allocated on the stack, which can provide useful flexibility. There are also efficient means of converting value types to object types if and when necessary. The set of classes is pretty comprehensive, providing collections, file, screen, and network I/O, threading, and so on, as well as XML and database connectivity. The class library is subdivided into a number of sets (or namespaces), each providing distinct areas of functionality, with dependencies between the namespaces kept to a minimum.

3.4.6 LANGUAGES SUPPORTED BY .NET

The multi-language capability of the .NET Framework and Visual Studio .NET enables developers to use their existing programming skills to build all types of applications and XML Web services. The .NET framework supports new versions of Microsoft's old favourites Visual Basic and C++ (as VB.NET and Managed C++), but there are also a number of new additions to the family.

Visual Basic .NET has been updated to include many new and improved language features that make it a powerful object-oriented programming language.

These features include inheritance, interfaces, and overloading, among others. Visual Basic also now supports structured exception handling, custom attributes and also supports multi-threading. Visual Basic .NET is also CLS compliant, which means that any CLS-compliant language can use the classes, objects, and components you create in Visual Basic .NET. Managed Extensions for C++ and attributed programming are just some of the enhancements made to the C++ language. Managed Extensions simplify the task of migrating existing C++ applications to the new .NET Framework. Microsoft Visual J# .NET provides the easiest transition for Java-language developers into the world of XML Web Services and dramatically improves the interoperability of Java-language programs with existing software written in a variety of other programming languages. Active State has created Visual Perl and Visual Python, which enable .NET-aware applications to be built in either Perl or Python. Both products can be integrated into the Visual Studio .NET environment. Visual Perl includes support for Active State's Perl Dev Kit.

Other languages for which .NET compilers are available include

- FORTRAN
- COBOL
- Eiffel

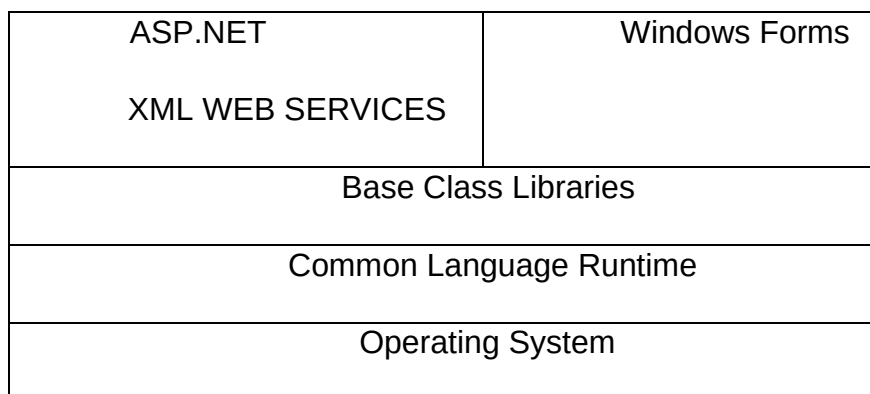


Fig 3.4.1 .Net Framework

C#.NET is also compliant with CLS (Common Language Specification) and supports structured exception handling. CLS is set of rules and constructs that are

supported by the CLR (Common Language Runtime). CLR is the runtime environment provided by the .NET Framework; it manages the execution of the code and also makes the development process easier by providing services. C#.NET is a CLS-compliant language. Any objects, classes, or components that created in C#.NET can be used in any other CLS-compliant language. In addition, we can use objects, classes, and components created in other CLS-compliant languages in C#.NET. The use of CLS ensures complete interoperability among applications, regardless of the languages used to create the application.

3.4.7 CONSTRUCTORS AND DESTRUCTORS:

Constructors are used to initialize objects, whereas destructors are used to destroy them. In other words, destructors are used to release the resources allocated to the object. In C#.NET the sub finalize procedure is available. The sub finalize procedure is used to complete the tasks that must be performed when an object is destroyed. The sub finalize procedure is called automatically when an object is destroyed. In addition, the sub finalize procedure can be called only from the class it belongs to or from derived classes.

3.4.8 OBJECTIVES OF .NET FRAMEWORK

- To provide a consistent object-oriented programming environment whether object codes is stored and executed locally on Internet-distributed, or executed remotely.
- To provide a code-execution environment to minimizes software deployment and guarantees safe execution of code.
- Eliminates the performance problems.
- There are different types of application, such as Windows-based applications and Web-based applications.

3.4.9 Features Of .Net

Microsoft .NET is a set of Microsoft software technologies for rapidly building and integrating XML Web services, Microsoft Windows-based applications, and Web solutions. The .NET Framework is a language-neutral platform for writing programs that can easily and securely interoperate. There's no language barrier

with .NET: there are numerous languages available to the developer including Managed C++, C#, Visual Basic and Java Script. The .NET framework provides the foundation for components to interact seamlessly, whether locally or remotely on different platforms. It standardizes common data types and communications protocols so that components created in different languages can easily interoperate.

“.NET” is also the collective name given to various software components built upon the .NET platform. These will be both products (Visual Studio.NET and Windows.NET Server, for instance) and services (like Passport, .NET My Services, and so on).

3.5 SQL-SERVER

The OLAP Services feature available in SQL Server version 7.0 is now called SQL Server 2000 Analysis Services. The term OLAP Services has been replaced with the term Analysis Services. Analysis Services also includes a new data mining component. The Repository component available in SQL Server version 7.0 is now called Microsoft SQL Server 2000 Meta Data Services. References to the component now use the term Meta Data Services. The term repository is used only in reference to the repository engine within Meta Data Services. SQL-SERVER database consists of five type of objects, they are,

- 1 TABLE
- 2 QUERY
- 3 FORM
- 4 REPORT
- 5 MACRO

3.5.1 TABLE:

A database is a collection of data about a specific topic. We can work with a table in two types,

1. Design View
2. Datasheet View

Design View

To build or modify the structure of a table we work in the table design view. We can specify what kind of data will be hold.

Datasheet View

To add, edit or analyses the data itself we work in tables datasheet view mode.

3.5.2 QUERY:

A query is a question that has to be asked the data. Access gathers data that answers the question from one or more table. The data that make up the answer is either dynast (if you edit it) or a snapshot (it cannot be edited). Each time we run query, we get latest information in the dataset. Access either displays the dataset or snapshot for us to view or perform an action on it, such as deleting or updating.

3.6 MD5 AND HASH ALGORITHM

These algorithms are used for encryption and decryption. Hash algorithm is used to compare two strings

3.6.1 Hash algorithm:

A hash function is any function that can be used to map data of arbitrary size to data of a fixed size. The values returned by a hash function are called hash values, hash codes, digests, or simply hashes. Hash functions are often used in combination with a hash table, a common data structure used in computer software for rapid data lookup. Hash functions accelerate table or database lookup by detecting duplicated records in a large file. One such application is finding similar stretches in DNA sequences. They are also useful in cryptography.

In this hashing technique, the hash of a string is calculated as:

$$\text{Hash} = (s[0]*p^0 + s[1]*p^1 + \dots + s[m]*p^m) \bmod M$$

Where P and M are some positive numbers. And s[0], s[1], s[2] ... s[n-1] are the values assigned to each character in English alphabet (a->1, b->2, ... z->26).

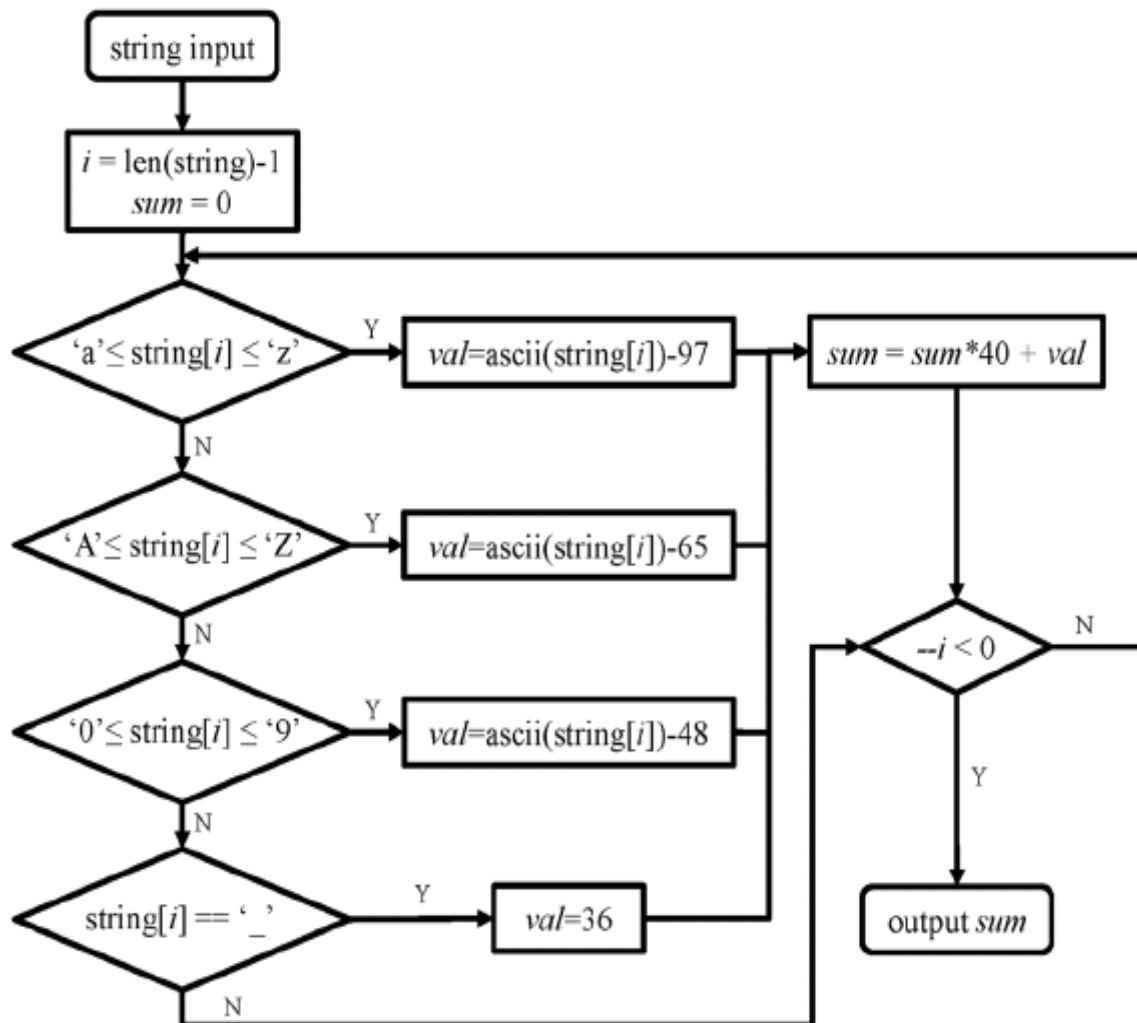


FIG 3.6.1 FLOW CHAR FOR HASH VALUE

3.6.2 Finding duplicate records:

When storing records in a large unsorted file, one may use a hash function to map each record to an index into a table T , and to collect in each bucket $T[i]$ a list of the numbers of all records with the same hash value i . Once the table is complete, any two duplicate records will end up in the same bucket. The duplicates can then be found by scanning every bucket $T[i]$ which contains two or more members, fetching those records, and comparing them. With a table of appropriate size, this method is likely to be much faster than any alternative approach (such as sorting the file and comparing all consecutive pairs).

3.6.3 Finding similar records

Hash functions can also be used to locate table records whose key is similar, but not identical, to a given key; or pairs of records in a large file which have similar keys. For that purpose, one needs a hash function that maps similar keys to hash values that differ by at most m , where m is a small integer (say, 1 or 2). If one builds a table T of all record numbers, using such a hash function, then similar records will end up in the same bucket, or in nearby buckets. Then one need only check the records in each bucket

- $T[i]$ against those in buckets $T[i+k]$ where k ranges between $-m$ and m .
- Hashing variable-length data
- $S \leftarrow S_0$; // Initialize the state.
- for k in 1, 2, ..., m do // Scan the input data units:
- $S \leftarrow F(S, b[k])$; // Combine data unit k into the state.
- return $G(S, n)$ // Extract the hash value from the state.

3.6.4 MD5 ALGORITHM

Md5 (Message Digest 5) is a cryptographic function that allows you to make a 128-bits (32 characters) "hash" from any string taken as input, no matter the length (up to 2^{64} bits). The only way to decrypt your hash is to compare it with a database using decrypter. The data in the file will be encrypted so that others can not view the data in that file. To view the file the owner should give the permissions and a key will be generated to the email id of the user.

CHAPTER-4

SOFTWARE DEVELOPMENT METHODOLOGY

4.1 SYSTEM DESIGN

System Design involves identification of classes their relationship as well as their collaboration. In objector, classes are divided into entity y classes and control classes. The Computer Aided Software Engineering (CASE) tools that are available commercially do not provide any assistance in this transition. CASE tools take advantage of Meta modelling that is helpful only after the construction of the class diagram. In the FUSION method some object-oriented approach likes Object Modelling Technique (OMT), Classes, and Responsibilities. Collaborators (CRC), etc, are used. Objector used the term” agents” to represent some of the hardware and software system. In Fusion method, there is no requirement phase, where a user will supply the initial requirement document. Any software project is worked out by both the analyst and the designer. The analyst creates the user case diagram. The designer creates the class diagram. But the designer can do this only after the analyst creates the use case diagram. Once the design is over, it is essential to decide which software is suitable for the application

4.2 UML DIAGRAM

UML is a standard language for specifying, visualizing, and documenting of software systems and created by Object Management Group (OMG) in 1997. There are three important type of UML modelling are Structural model, Behavioural model, and Architecture model. To model a system the most important aspect is to capture the dynamic behaviour which has some internal or external factors for making the interaction. These internal or external agents are known as actors. It consists of actors, use cases and their relationships. In this fig we represent the Use Case diagram for our project.

4.2.1 Use case diagram:

A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components a user or another system that will interact with the

system modelled. A use case is an external view of the system that represents some action the user might perform in order to complete a task.

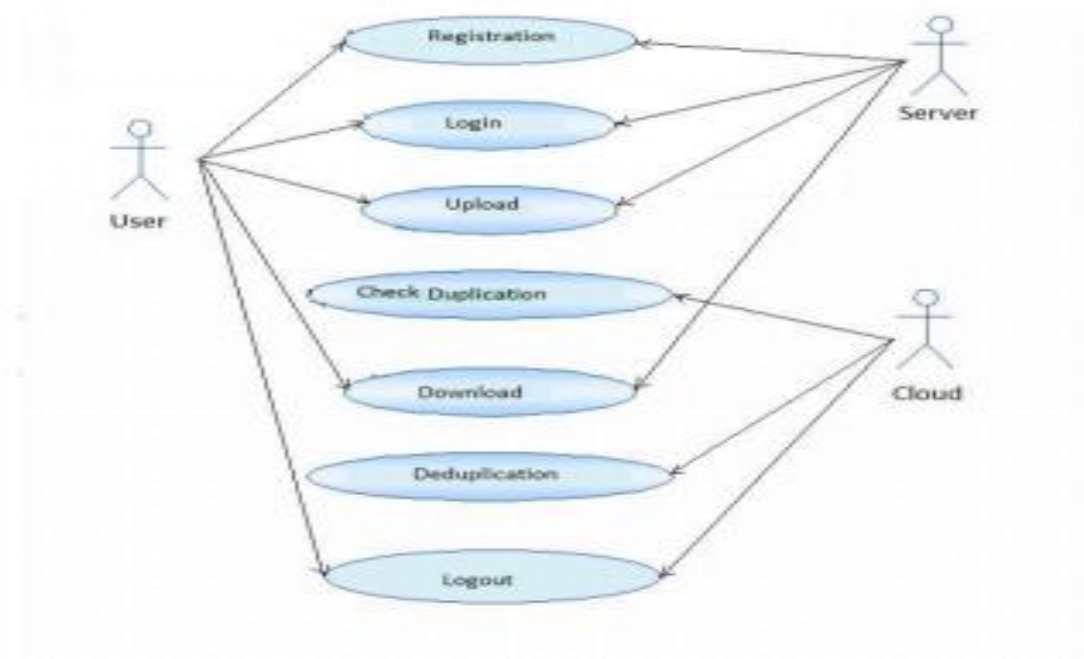


FIG 4.2.1 UML DIAGRAM

4.2.2 Class Diagram:

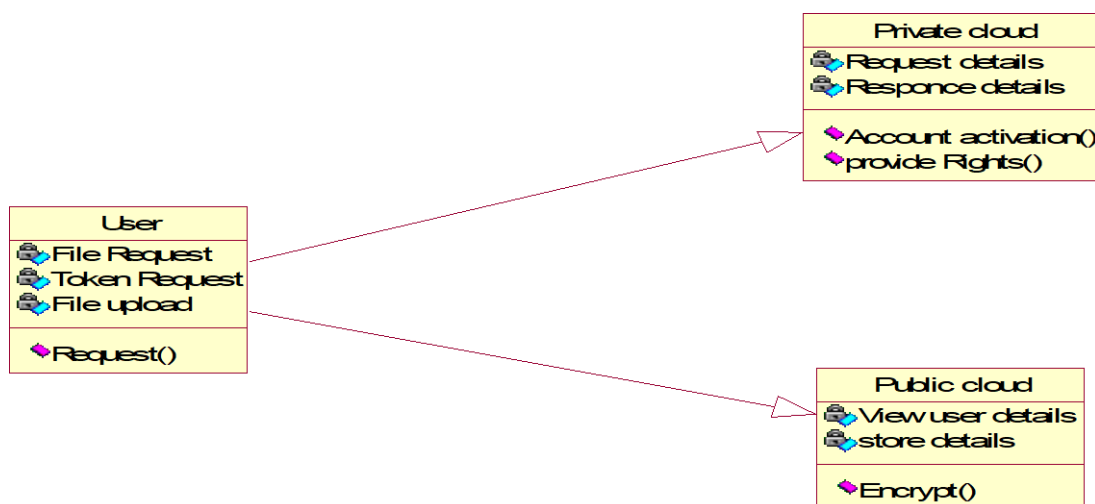


FIG 4.2.2 CLASS DIAGRAM

A class is the one which represents the structure of the system by showing the system classes their attributes, operations and the relationships among objects. All the main activities will be held in the private cloud like account activation, account requests. The details will be stored in the public cloud. And the data will be encrypted

4.2.3 Sequence Diagram

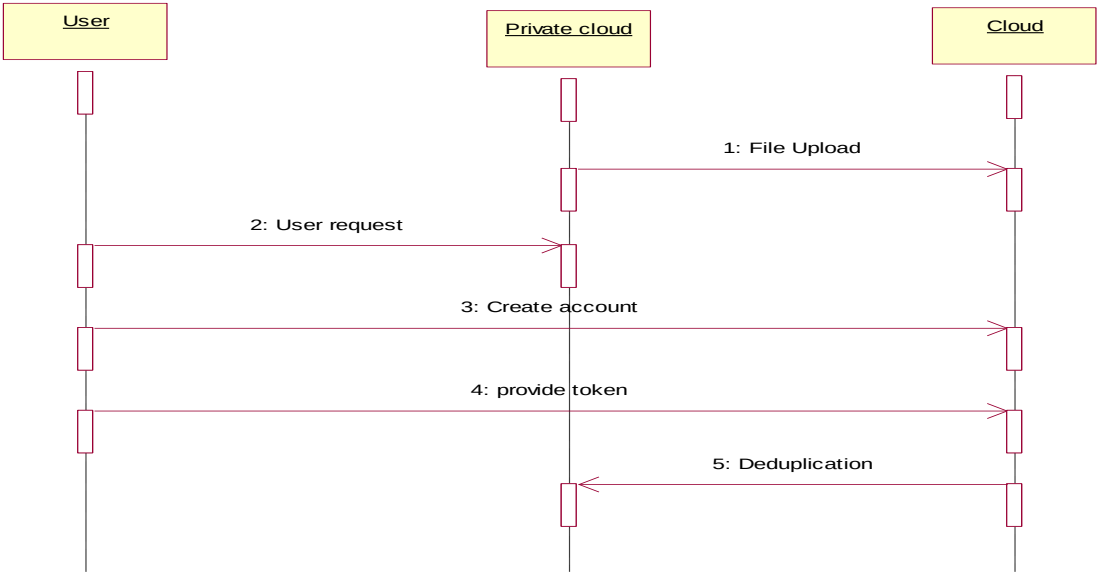


FIG 4.2.3 SEQUENCE DIAGRAM

4.3.4 Collaboration Diagram

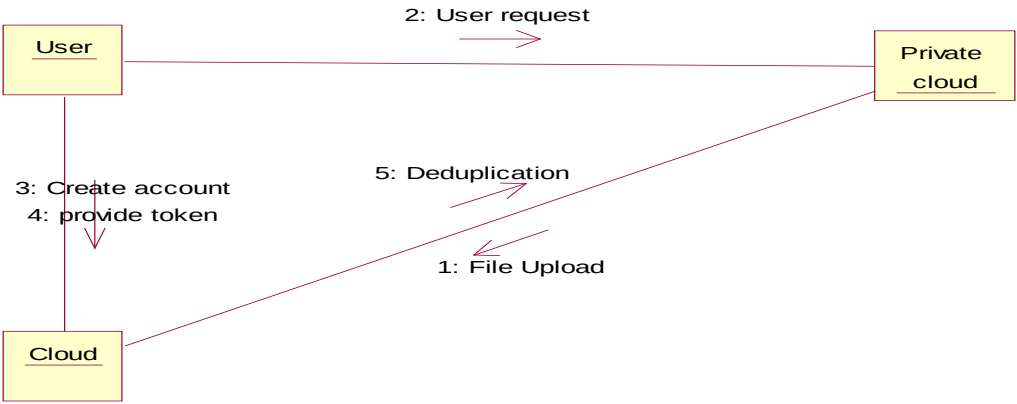


FIG 4.2.4 COLLABRATION DIAGRAM

4.2.5 ER DIAGRAM

An entity–relationship model describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types and specifies relationships that can exist between entities.

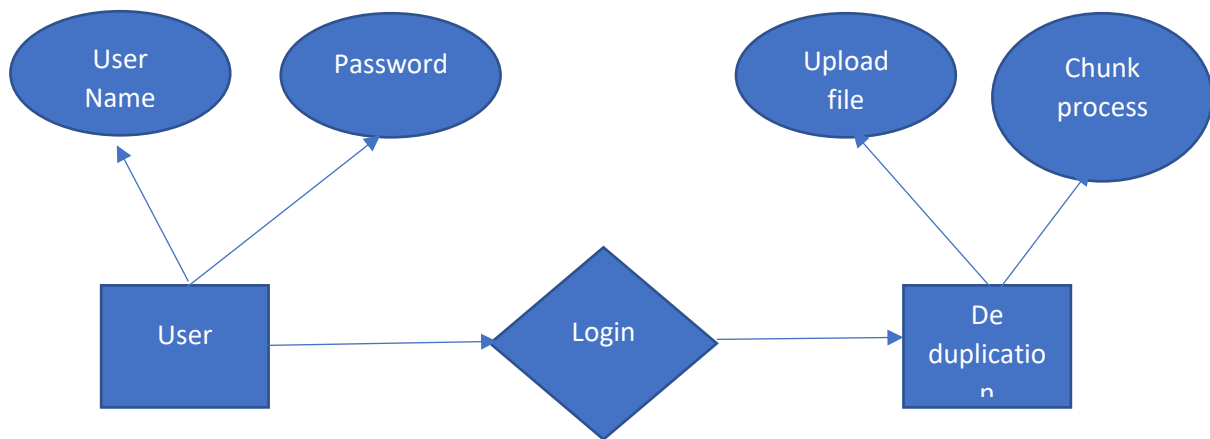


FIG 4.2.5 ER DIAGRAM

4.3 ARCHITECTURE

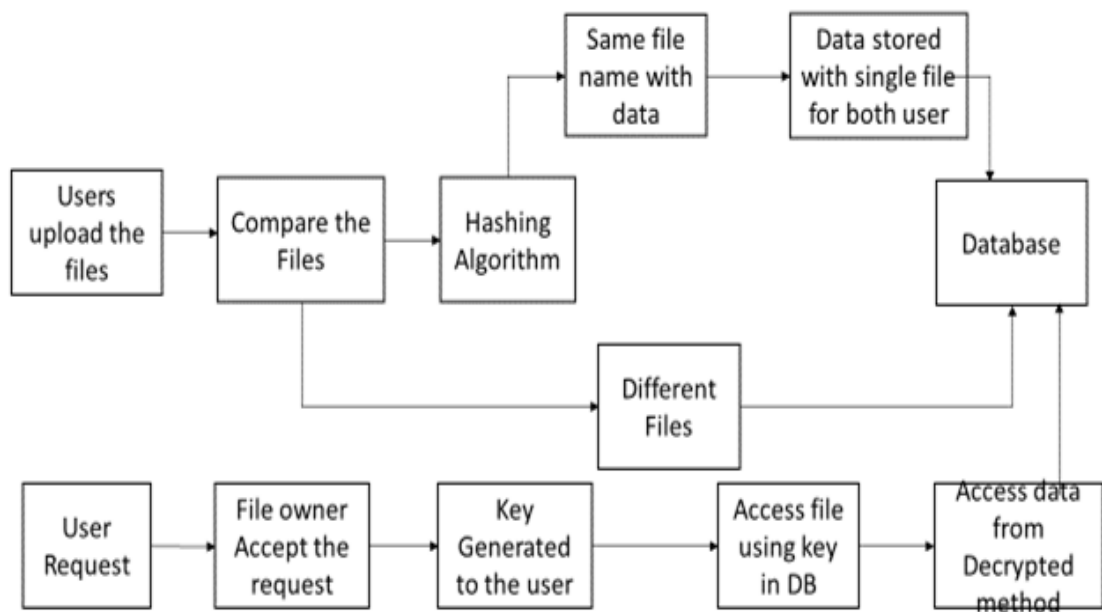


FIG 4.3.1 ARCHITECTURE

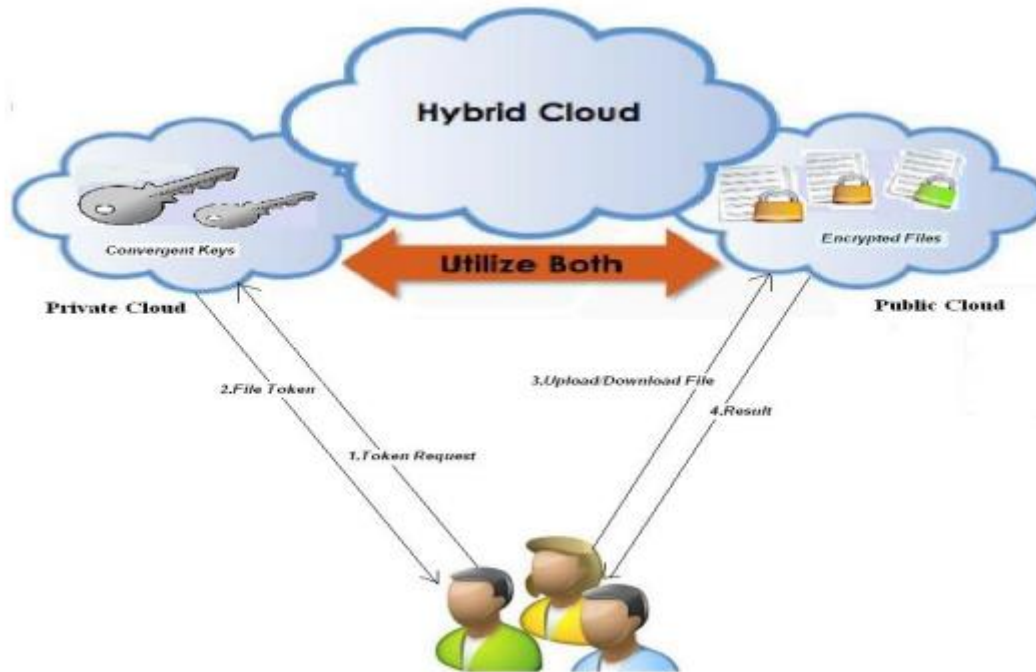


FIG 4.3.2 SYSTEM DESIGN ARCHITECTURE

4.4 DATA FLOW DIAGRAM

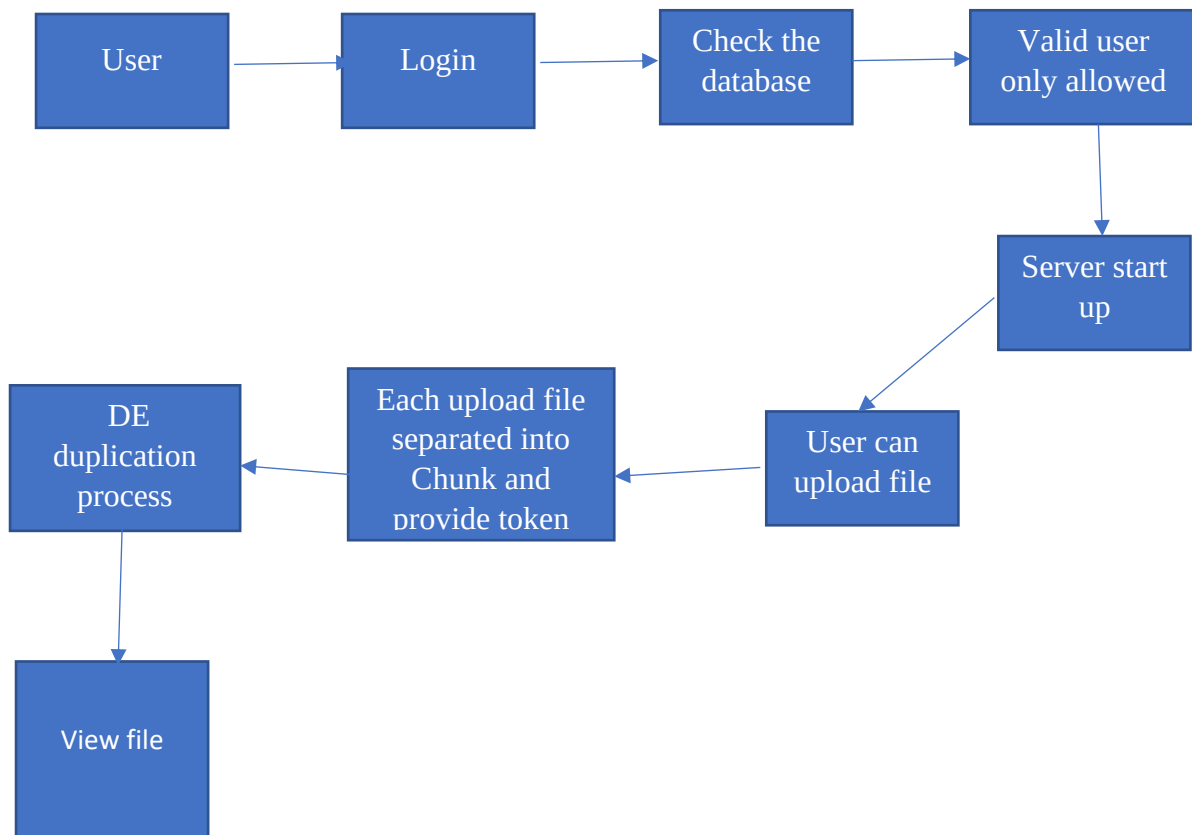


FIG 4.4.1: DATA FLOW DIAGRAM

The Data Flow diagram is a graphic tool used for expressing system requirements in a graphical form. The DFD also known as the “bubble chart” has the purpose of clarifying system requirements and identifying major transformations that to become program in system design. Thus, DFD can be stated as the starting point of the design phase that functionally decomposes the requirements specifications down to the lowest level of detail. The DFD consist of series of bubbles joined by lines. The bubbles represent data transformations and the lines represent data flows in the system. A DFD describes what that data flow in rather than how they are processed. So, it does not depend on hardware, software, data structure or file organization.

4.5 MODULE DESCRIPTION

4.5.1 USER MODULE

In this module, Users are having authentication and security to access the detail which is presented in the ontology system. Before accessing or searching the details user should have the account in that otherwise they should register first. At the very least, you need to provide an email address, username, password, display name, and whatever profile fields you have set to required. The display name is what will be used when the system needs to display the proper name of the user.



FIG 4.5.1 USER MODULE

4.5.2 SERVER START UP AND UPLOAD FILE

The user can start up the server after cloud environment is opened. Then the user can upload the file to the cloud.



FIG 4.5.2 SERVER STARTUP

4.5.3 SECURE DE DUPLICATION SYSTEM

To support authorized de duplication the tag of a file F will be determined by the file F and the privilege. To show the difference with traditional notation of tag, we call it file token instead. To support authorized access a secret key KP will be bounded with a privilege p to generate a file Token. De duplication exploits identical content, while encryption attempts to make all content appear random; the same content encrypted with two different keys results in very different cipher text. Thus, combining the space efficiency of de duplication with the secrecy aspects of encryption is problematic.

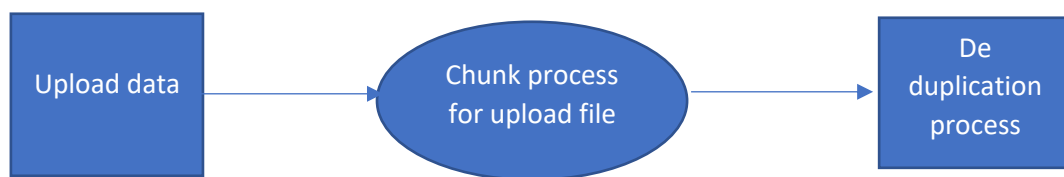


FIG 4.5.3 DE-DUPLICATION SYSTEM

4.5.4 DOWNLOAD FILE

After the cloud storage, the user can download the file based on key or token. Once the key request was received, the sender can send the key or he can decline it. With this key and request id which was generated at the time of sending key request the receiver can decrypt the message.

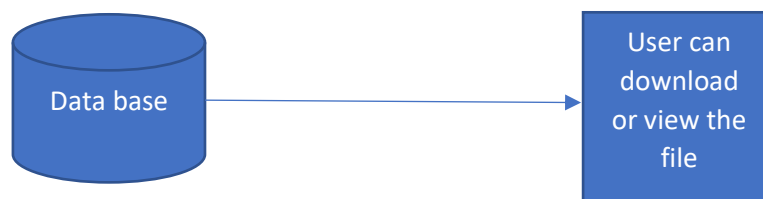


FIG 4.5.4 DOWNLOAD FILE

CHAPTER-5

RESULTS AND DISCUSSION

Our project checks whether there are any duplicate files in the system if exist, they eliminate the files. The data which has been uploaded by the data owners will be encrypted and stored with a key value. to view the user will be sending the data request to owner and owner will be accepting the request then the user will be receiving a token to their respective mail id. by using the token the user can view and download the file. There may be existing systems for laptops and other cloud sources but few lack the security issues. So, our encryption and decryption may be useful to store the data in the cloud so that no other user can be able to view the data of other users. It allows for the storage of one unique instance of all the data within a database, without any copies needlessly taking up space. Once the redundant copies of data are removed, data deduplication gives you the option to compress the single copies of data that are stored to save even more space

CHAPTER-6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

In this paper, the notion of authorized data de duplication was proposed to protect the data security by including differential privileges of users in the duplicate check. We also presented several new de duplication constructions supporting authorized duplicate check in hybrid cloud architecture, in which the duplicate-check tokens of files are generated by the private cloud server with private keys. Security analysis demonstrates that our schemes are secure in terms of insider and outsider attacks specified in the proposed security model. As a proof of concept, we implemented a prototype of our proposed authorized duplicate check scheme and conduct test bed experiments on our prototype. We showed that our authorized duplicate check scheme incurs minimal overhead compared to convergent encryption and network transfer.

6.2 FUTURE ENHANCEMENT

Finally, we believe that cloud data storage security is still full of challenges and of paramount importance, and many research problems remain to be identified.

REFERENCES

1. A.Anusha,K Kiran, and J Dayanika “compressing the data secure authorized deduplication checker in hybrid cloud” published on semantic scholar
2. Chaoling li, Yue Chen and Yanzhou Zhou “a data assured deletion scheme in clous storage” published in china communication (volume 11, issue: 4, April 2014)
3. G.uma and L.Jayasimman “enhanced convergent encryption key generation for secured data deduplication in cloud storage” published in journals of physics conference series
4. Huiqi Xu, Shumin Guo and Keke Chen “Building confidential and efficient query services in the cloud with RASP data perturbation” published in IEEE transaction on knowledge and data engineering.
5. J. Li, X. Chen, M. Li, J. Li, P. Lee, and W. Lou. Secure deduplication with efficient and reliable convergent key management. In IEEE Transactions on Parallel and Distributed Systems, 2013.
6. J. Stanek, A. Sorniotti, E. Androulaki, and L. Kencl. A secure data deduplication scheme for cloud storage. In Technical Report, 2013.
7. Ming Li and Pan Li “crowdsourcing in cyberphysical systems: stochastic optimization with strong ability” published in IEEE transactions on emerging topics in computing.
8. M. Bellare and A. Palacio. Gq and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In CRYPTO, pages 162–177, 2002.
9. S. Halevi, D. Harnik, B. Pinkas, and A. ShulmanPeleg. Proofs of ownership in remote storage systems. In Y. Chen, G. Danezis, and V. Shmatikov, editors, ACM Conference on Computer and Communications Security, pages 491–500. ACM, 2011
10. W. K. Ng, Y. Wen, and H. Zhu. Private data deduplication protocols in cloud storage. In S. Ossowski and P. Lecca, editors, Proceedings of the 27th Annual ACM Symposium on Applied Computing, pages 441–446. ACM, 2012.

11. Yang Tang, Patrick P.C Lee, John C.S.Lui and Radia Perlman “Secure overlay cloud storage with access control and Assured Deletion” published in IEEE transactions on Dependable and secure Computing

APPENDIX

SOURCE CODE

HOME PAGE:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Home.aspx.cs"
Inherits="Home" %>
```

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

```
<head runat="server">
```

```
    <meta name="keywords" content="" />
```

```
<meta name="description" content="" />
```

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

```
<title>De-Duplication</title>
```

```
<link href="http://fonts.googleapis.com/css?family=Abel" rel="stylesheet"
```

```
type="text/css" />
```

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

```
<script type="text/javascript"
```

```
src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"></script>
```

```
<script type="text/javascript" src="jquery.dropotron-1.0.js"></script>
```

```
<style type="text/css">
```

```
    .auto-style2 {
```

```
        height: 54px;
```

```
        width: 976px;
```

```
    }
```

```
    .auto-style5 {
```

```
        height: 293px;
```

```
    }
```

```
    .auto-style6 {
```

```
        width: 100%;
```

```

    }
    .auto-style7 {
        width: 276px;
    }
    .auto-style8 {
        width: 50px;
    }
    .auto-style9 {
        width: 133px;
    }
    .auto-style10 {
        width: 276px;
        text-align: center;
    }
    .auto-style11 {
        text-align: center;
        width: 250px;
    }
    .auto-style12 {
        width: 250px;
    }
</style>
</head>
<body>
    <form id="form2" runat="server">
        <div id="wrapper">
            <div id="header-wrapper">
                <div id="header" class="auto-style6">
                    <br />
                    <br />
                    <br />
                    <br />
                </div>
            </div>
        </div>
    </form>

```

```

        <h1 class="auto-style2">DATA DEDUPLICATION</h1>
    </div>
</div>
</div>

<div id="splash"></div>
    <div id="page">

<table class="auto-style6">
    <tr>
        <td colspan="5">&nbsp;</td>
    </tr>
    <tr>
        <td class="auto-style9">&nbsp;</td>
        <td class="auto-style7">
            <asp:Image ID="Image1" runat="server" Height="210px"
ImageUrl="~/images/LoginRed.jpg" Width="248px" />
        </td>
        <td class="auto-style8">&nbsp;</td>
        <td class="auto-style12">
            <asp:Image ID="Image2" runat="server" Height="195px"
ImageUrl="~/images/cpt-registration.gif" Width="204px" />
        </td>
        <td>&nbsp;</td>
    </tr>
    <tr>
        <td class="auto-style9">&nbsp;</td>
        <td class="auto-style10">
            <br />
            <asp:Button ID="Button1" runat="server" OnClick="Button1_Click"
Text="LOGIN" />

```

```

        <br />
    </td>
    <td class="auto-style8">&nbsp;</td>
    <td class="auto-style11">
        <asp:Button ID="Button2" runat="server" OnClick="Button2_Click"
Text="REGISTER" />
    </td>
    <td>&nbsp;</td>
</tr>
</table>

</div>
</form>
</body>
</html>

```

REGISTER:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.SqlClient;

public partial class REGISTER : System.Web.UI.Page
{
    SqlConnection conn;
    SqlCommand cmd;
    //string con = "Data Source=VISWANADH\\SQLEXPRESS;Initial
Catalog=deduplication;Integrated Security=True";
    Class1 c = new Class1();

```

```

protected void Page_Load(object sender, EventArgs e)
{

}

protected void Button1_Click(object sender, EventArgs e)
{
    conn = new SqlConnection(c.constr);
    conn.Open();
    string q = "insert into reg values('" + TextBox1.Text + "','" + TextBox2.Text +
    "','" + TextBox3.Text + "','" + TextBox4.Text + "','" + TextBox5.Text + "','" +
    DropDownList1.Text + "')";
    cmd = new SqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = q;
    cmd.ExecuteNonQuery();
    TextBox1.Text = "";
    TextBox2.Text = "";
    TextBox3.Text = "";
    TextBox4.Text = "";
    TextBox5.Text = "";
    conn.Close();
    string script = "alert(\"Sucessfully Registered\")";
    ScriptManager.RegisterStartupScript(this, GetType(), "ServerControlScript",
script, true);
    Response.Redirect("Home.aspx");
}

protected void Button2_Click(object sender, EventArgs e)
{
    Response.Redirect("Home.aspx");
}
}

```

LOGIN:

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data.SqlClient;

public partial class LOGIN : System.Web.UI.Page

{

    SqlConnection conn;

    SqlCommand cmd;

    SqlDataReader rs;

    string con = "Data Source=DESKTOP-GQM99F3;Initial
Catalog=deduplication1;Integrated Security=True";

    string type;

    protected void Page_Load(object sender, EventArgs e)

    {

    }

    protected void Button1_Click(object sender, EventArgs e)
```

```

{

    if(TextBox1.Text=="||TextBox2.Text=="")

    {

        string script = "alert(\"Enter all details\");";

        ScriptManager.RegisterStartupScript(this, GetType(), "ServerControlScript", script,
true);

    }

    if (TextBox1.Text == "admin" && TextBox2.Text == "admin")

    {

        Response.Redirect("cloud1.aspx");

    }

    else

    {

        conn = new SqlConnection(con);

        conn.Open();

        string q = "select type from reg where u_name=" + TextBox1.Text + " and pwd="
+ TextBox2.Text + " ";

        cmd = new SqlCommand();

        cmd.Connection = conn;

        cmd.CommandText = q;

        rs = cmd.ExecuteReader();

        if (rs.Read())

```

```

{

    type = Convert.ToString(rs.GetValue(0))

}

rs.Close();

if (type == "Customer")

{

    Session["u_id"] = TextBox1.Text.ToString();

    conn = new SqlConnection(con);

    conn.Open();

    string s = "select mail from reg where u_name='" + TextBox1.Text + "' and
pwd='" + TextBox2.Text + "' ";

    cmd = new SqlCommand();

    cmd.Connection = conn;

    cmd.CommandText = s;

    rs = cmd.ExecuteReader();

    if (rs.Read())

    {

        type = Convert.ToString(rs.GetValue(0));

        string emailid = rs.GetValue(0).ToString();

        Session["emailid"] = emailid;

```

```

    }

    rs.Close();

    Response.Redirect("cust1.aspx");

}

else if (type == "Data Owner")

{

    Session["u_id"] = TextBox1.Text.ToString();

    Response.Redirect("owner1.aspx");

}

else

{

    string script = "alert(\"invalid User\")";

    ScriptManager.RegisterStartupScript(this,GetType(),"ServerControlScript", script,
true);

}

conn.Close();

}

}

protected void Button2_Click(object sender, EventArgs e)

{

```

```

        Response.Redirect("Home.aspx");

    }

}

```

DATA OWNER:

```

<%@ Page Title="" Language="C#" MasterPageFile="~/data.master"
AutoEventWireup="true" CodeFile="owner1.aspx.cs" Inherits="owner1" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder2"
Runat="Server">
    <table class="auto-style6">
    <tr>
    <td style="width: 174px">&nbsp;</td>
    <td>
        <table class="auto-style6" style="width: 71%">
        <tr>
            <td colspan="2" style="text-align: center"><strong>
                <br />
                FILE UPLOAD<br />
            </strong></td>
        </tr>
        <tr>
            <td style="width: 185px; height: 40px;">KEYWORD</td>
            <td style="height: 40px">
                <asp:TextBox ID="TextBox1" runat="server" Height="30px"
Width="151px"></asp:TextBox>
            </td>
        </tr>
        <tr>
            <td style="width: 185px">FILE NAME</td>
            <td>
                <asp:FileUpload ID="FileUpload1" runat="server" Height="34px"
Width="242px" />
            </td>
        </tr>
        <tr>
            <td style="width: 185px">
                <br />
            </td>
            <td>
                <asp:Button ID="Button1" runat="server" OnClick="Button1_Click"
Text="UPLOAD" style="height: 26px" />
            </td>
        </tr>
    </table>
    </td>
    </tr>
    </table>

```

```

        </tr>
    </table>
</td>
<td>&nbsp;</td>
</tr>
</table>
</asp:Content>

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.SqlClient;
using System.Security.Cryptography;
using System.IO;

public partial class owner1 : System.Web.UI.Page
{
    string filename = string.Empty;
    string path2;
    SqlConnection conn;
    SqlCommand cmd;
    SqlDataReader rs;
    string status1;
    string file2, org, dup, path1, file3;
    string con = "Data Source=VISWANADH\\SQLEXPRESS;Initial
Catalog=deduplication;Integrated Security=True";
    int a;
    Class2 c1 = new Class2();
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        path2 = "temp/" + FileUpload1.FileName;
        filename = Path.GetFileName(FileUpload1.PostedFile.FileName);
        FileUpload1.SaveAs(Server.MapPath("~/temp/" + filename));
        string status = validate(path2);
        if (status == "dup")
        {
            conn = new SqlConnection(con);
            conn.Open();
            dup = FileUpload1.FileName;
            string q = "select count(*) from index1";

```

```

        cmd = new SqlCommand();
        cmd.Connection = conn;
        cmd.CommandText = q;
        a = Convert.ToInt32(cmd.ExecuteScalar());
        a += 1;
        q = "insert into index1 values('" + a + "','" + dup + "','" + org + "','" + file3 +
        "','" + Convert.ToString(Session["u_id"]) + "','" + TextBox1.Text + "','" + file2 + "')";
        cmd.CommandText = q;
        cmd.ExecuteNonQuery();
        conn.Close();
    }
    else if (status == "new")
    {
        upload();
        conn = new SqlConnection(con);
        conn.Open();
        org = FileUpload1.FileName;
        string q = "select count(*) from index1";
        cmd = new SqlCommand();
        cmd.Connection = conn;
        cmd.CommandText = q;
        a = Convert.ToInt32(cmd.ExecuteScalar());
        a += 1;
        q = "insert into index1 values('" + a + "','" + org + "','" + org + "','" + path2 +
        "','" + Convert.ToString(Session["u_id"]) + "','" + TextBox1.Text + "','" + path1 + "')";

        cmd.CommandText = q;
        cmd.ExecuteNonQuery();
        conn.Close();
    }
}

public void upload()
{
    filename = Path.GetFileName(FileUpload1.PostedFile.FileName);
    FileUpload1.SaveAs(Server.MapPath("~/Files/" + filename));
    Response.Write("File uploaded sucessfully.");

    c1.encrypt(Server.MapPath("~/Files/" + filename),
    Server.MapPath("~/Files/Encrypted/" + filename));
    path2 = "Files/Encrypted/" + FileUpload1.FileName;
    path1 = "Files/" + FileUpload1.FileName;
}

public string validate(string path1)
{
    string file1 = Server.MapPath(path1);

    conn = new SqlConnection(con);

```

```

conn.Open();
string q = "select path2,org,path from index1 ";
cmd = new SqlCommand();
cmd.Connection = conn;
cmd.CommandText = q;
rs = cmd.ExecuteReader();
while (rs.Read())
{
    file2 = Convert.ToString(rs.GetValue(0));
    string file4 = Server.MapPath(Convert.ToString(rs.GetValue(0)));
    file3 = Convert.ToString(rs.GetValue(2));

    HashAlgorithm hashAlgo = HashAlgorithm.Create();

    FileStream stream1 = new FileStream(file1, FileMode.Open);
    FileStream stream2 = new FileStream(file4, FileMode.Open);

    byte[] hash1;
    byte[] hash2;

    hash1 = hashAlgo.ComputeHash(stream1);
    hash2 = hashAlgo.ComputeHash(stream2);
    stream1.Close();
    stream2.Close();
    string text = System.IO.File.ReadAllText(file1);
    string text2 = System.IO.File.ReadAllText(file4);
    if (Convert.ToBase64String(hash1) == Convert.ToBase64String(hash2))
    {
        status1 = "dup";
    }

    int aa = Compute(text, text2);
    if (aa == 0)
    {
        status1 = "dup";
        org = Convert.ToString(rs.GetValue(1));
        break;
    }
}

rs.Close();
conn.Close();
if (status1 == "dup")
{
    return "dup";
}

```

```

else
{
    return "new";
}

}

public static int Compute(string s, string t)
{
    if (string.IsNullOrEmpty(s))
    {
        if (string.IsNullOrEmpty(t))
            return 0;
        return t.Length;
    }

    if (string.IsNullOrEmpty(t))
    {
        return s.Length;
    }

    int n = s.Length;
    int m = t.Length;
    int[,] d = new int[n + 1, m + 1];

    // initialize the top and right of the table to 0, 1, 2, ...
    for (int i = 0; i <= n; d[i, 0] = i++) ;
    for (int j = 1; j <= m; d[0, j] = j++) ;

    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= m; j++)
        {
            int cost = (t[j - 1] == s[i - 1]) ? 0 : 1;
            int min1 = d[i - 1, j] + 1;
            int min2 = d[i, j - 1] + 1;
            int min3 = d[i - 1, j - 1] + cost;
            d[i, j] = Math.Min(Math.Min(min1, min2), min3);
        }
    }
    return d[n, m];
}
}

```

UPLOAD:

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data.SqlClient;

using System.Security.Cryptography;

using System.IO;


public partial class owner1 : System.Web.UI.Page

{

    string filename = string.Empty;

    string path2;

    SqlConnection conn;

    SqlCommand cmd;

    SqlDataReader rs;

    string status1;

    string file2, org, dup,path1,file3;
```

```

string con = "Data Source=DESKTOP-GQM99F3;Initial
Catalog=deduplication1;Integrated Security=True";

int a;

Class2 c1 = new Class2();

protected void Page_Load(object sender, EventArgs e)

{

}

protected void Button1_Click(object sender, EventArgs e)

{

    path2 = "temp/" + FileUpload1.FileName;

    filename = Path.GetFileName(FileUpload1.PostedFile.FileName);

    FileUpload1.SaveAs(Server.MapPath("~/temp/" + filename));

    string status = validate(path2);

    if (status == "dup")

    {

        conn = new SqlConnection(con);

        conn.Open();

        dup = FileUpload1.FileName;

        string q = "select count(*) from index1";

        cmd = new SqlCommand();

```

```

cmd.Connection = conn;

cmd.CommandText = q;

a = Convert.ToInt32(cmd.ExecuteScalar());

a += 1;

q = "insert into index1 values('" + a + "','" + dup + "','" + org + "','" + file3 + "','" +
Convert.ToString(Session["u_id"]) + "','" + TextBox1.Text + "','" + file2 + "')";

cmd.CommandText = q;

cmd.ExecuteNonQuery();

conn.Close();

}

else if (status == "new")

{

upload();

conn = new SqlConnection(con);

conn.Open();

org = FileUpload1.FileName;

string q = "select count(*) from index1";

cmd = new SqlCommand();

cmd.Connection = conn;

cmd.CommandText = q;

a = Convert.ToInt32(cmd.ExecuteScalar());

```

```

        a += 1;

        q = "insert into index1 values('" + a + "','" + org + "','" + org + "','" + path2 + "','" +
Convert.ToString(Session["u_id"]) + "','" + TextBox1.Text + "','" + path1 + "')";

        cmd.CommandText = q;

        cmd.ExecuteNonQuery();

        conn.Close();

    }

}

public void upload()

{

    filename = Path.GetFileName(FileUpload1.PostedFile.FileName);

    FileUpload1.SaveAs(Server.MapPath("~/Files/" + filename));

    Response.Write("File uploaded sucessfully.");


    c1.encrypt(Server.MapPath("~/Files/" + filename),
Server.MapPath("~/Files/Encrypted/" + filename));

    path2 = "Files/Encrypted/" + FileUpload1.FileName;

    path1 = "Files/" + FileUpload1.FileName;

}

public string validate(string path1)

```

```
{
```

```
string file1 =Server.MapPath( path1);
```

```
conn = new SqlConnection(con);
```

```
conn.Open();
```

```
string q = "select path2,org,path from index1 ";
```

```
cmd = new SqlCommand();
```

```
cmd.Connection = conn;
```

```
cmd.CommandText = q;
```

```
rs = cmd.ExecuteReader();
```

```
while (rs.Read())
```

```
{
```

```
file2 = Convert.ToString(rs.GetValue(0));
```

```
string file4 = Server.MapPath(Convert.ToString(rs.GetValue(0)));
```

```
file3 = Convert.ToString(rs.GetValue(2));
```

```
HashAlgorithm hashAlgo = HashAlgorithm.Create();
```

```
FileStream stream1 = new FileStream(file1, FileMode.Open);
```

```

FileStream stream2 = new FileStream(file4, FileMode.Open);

byte[] hash1;

byte[] hash2;

hash1 = hashAlgo.ComputeHash(stream1);

hash2 = hashAlgo.ComputeHash(stream2);

stream1.Close();

stream2.Close();

string text = System.IO.File.ReadAllText(file1);

string text2 = System.IO.File.ReadAllText(file4);

if (Convert.ToBase64String(hash1) == Convert.ToBase64String(hash2))

{

    status1 = "dup";

}

int aa = Compute(text, text2);

if (aa == 0)

{

    status1 = "dup";

```

```

        org = Convert.ToString(rs.GetValue(1));

        break;
    }
}

rs.Close();

conn.Close();

if (status1 == "dup")
{
    return "dup";
}

else
{
    return "new";
}

}

public static int Compute(string s, string t)
{

```

```

if (string.IsNullOrEmpty(s))
{
    if (string.IsNullOrEmpty(t))
        return 0;

    return t.Length;
}

```

```

if (string.IsNullOrEmpty(t))
{
    return s.Length;
}

```

```

int n = s.Length;

int m = t.Length;

int[,] d = new int[n + 1, m + 1];

```

```

// initialize the top and right of the table to 0, 1, 2, ...

```

```

for (int i = 0; i <= n; d[i, 0] = i++) ;

```

```

for (int j = 1; j <= m; d[0, j] = j++) ;

```

```

for (int i = 1; i <= n; i++)

```

```

{
    for (int j = 1; j <= m; j++)
    {
        int cost = (t[j - 1] == s[i - 1]) ? 0 : 1;

        int min1 = d[i - 1, j] + 1;

        int min2 = d[i, j - 1] + 1;

        int min3 = d[i - 1, j - 1] + cost;

        d[i, j] = Math.Min(Math.Min(min1, min2), min3);
    }
}

return d[n, m];
}
}

```

User module:

```

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

```

```

public partial class cust1 : System.Web.UI.Page
{
    string filename = string.Empty;

    string path2;

    SqlConnection conn;

    SqlCommand cmd;

    SqlDataReader rs;

    string status1;

    string file2, org, dup;

    string con = "Data Source=DESKTOP-GQM99F3;Initial
Catalog=deduplication1;Integrated Security=True";

    int a;

    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        SqlConnection con = new SqlConnection("Data Source=DESKTOP-GQM99F3;Initial
Catalog=deduplication1;Integrated Security=True");

        string strSQL = "Select file_id,org,u_name from index1 where org like'" +
        TextBox1.Text + "%' or Keywords like'" + TextBox1.Text + "%'";

        SqlDataAdapter dt = new SqlDataAdapter(strSQL, con);

        DataSet ds = new DataSet();

        dt.Fill(ds, "file_id");
    }
}

```

```

        con.Close();

        GridView2.DataSource = ds;

        GridView2.DataBind();
    }

    protected void GridView2_SelectedIndexChanged(object sender, EventArgs e)
    {
        conn = new SqlConnection(con);

        conn.Open();

        string q = "select count(*) from req";

        cmd = new SqlCommand();

        cmd.Connection = conn;

        cmd.CommandText = q;

        a = Convert.ToInt32(cmd.ExecuteScalar());

        a += 1;

        q = "insert into req values('" + Convert.ToString(Session["u_id"]) + "','" +
        Convert.ToString(GridView2.SelectedRow.Cells[1].Text) + "','" +
        Convert.ToString(GridView2.SelectedRow.Cells[2].Text) + "','" + '0' + "','" +
        Convert.ToString(GridView2.SelectedRow.Cells[3].Text) + "','" + a + "','" +
        Convert.ToString(Session["emailid"]) + "')";

        cmd.CommandText = q;

        cmd.ExecuteNonQuery();

        conn.Close();
    }
}

```

Apply key:

```

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.IO;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;


public partial class cust2 : System.Web.UI.Page
{
    string p,p1;

    SqlConnection conn;

    SqlCommand cmd;

    SqlDataReader rs;

    string con = "Data Source=DESKTOP-GQM99F3;Initial
Catalog=deduplication1;Integrated Security=True";

    string type, s2;

    Class1 c = new Class1();

    Class2 c1 = new Class2();

    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)

```

```

{
    conn = new SqlConnection(con);

    conn.Open();

    string q = "select file_id from temp where keyvalue='" + TextBox1.Text + "' ";

    cmd = new SqlCommand();

    cmd.Connection = conn;

    cmd.CommandText = q;

    rs = cmd.ExecuteReader();

    if (rs.Read())
    {

        type = Convert.ToString(rs.GetValue(0));

        s2 = "yes";

    }

    rs.Close();

    if (s2 == "yes")
    {

        q = "select org from index1 where file_id='" + type + "' ";

        cmd.CommandText = q;

        rs = cmd.ExecuteReader();

        if (rs.Read())
        {

            p = Convert.ToString(rs.GetValue(0));

```

```

    }

    rs.Close();

    c1.decrypt(Server.MapPath("~/Files/Encrypted/" + p),
Server.MapPath("~/Files/Decrypted/" + p));

    p1= Server.MapPath("~/Files/Decrypted/" + p);

    if (p1 != string.Empty)
    {
        if (p1.EndsWith(".txt"))
        {
            Response.ContentType = "application/txt";
        }
        else if (p1.EndsWith(".pdf"))
        {
            Response.ContentType = "application/pdf";
        }
        else if (p1.EndsWith(".docx"))
        {
            Response.ContentType = "application/docx";
        }
        else
        {
            Response.ContentType = "image/jpg";
        }

        string filePath = p1;

```

```

        //Response.AddHeader("Content-Disposition", "attachment;filename=\" +
filePath + "\"");

        //Response.TransmitFile(Server.MapPath(filePath));

        //Response.End();

    Response.Clear();

    Response.ContentType = "application/octet-stream";

    Response.AppendHeader("Content-Disposition", "attachment; filename=" +
Path.GetFileName(p1));

    Response.WriteFile(p1);

    Response.End();

}

Label4.Text = "";

//Response.Redirect("customer1.aspx");

}

else

{

    Label4.Text = "Enter correct key";

}

}

protected void UpdateTimer_Tick(object sender, EventArgs e)

{

    conn = new SqlConnection(con);

    conn.Open();

    string q = "delete from temp ";

```

```
cmd = new SqlCommand();  
cmd.Connection = conn;  
cmd.CommandText = q;  
cmd.ExecuteNonQuery();  
Response.Redirect("timeout.aspx");  
}  
}
```

SCREEN SHOTS

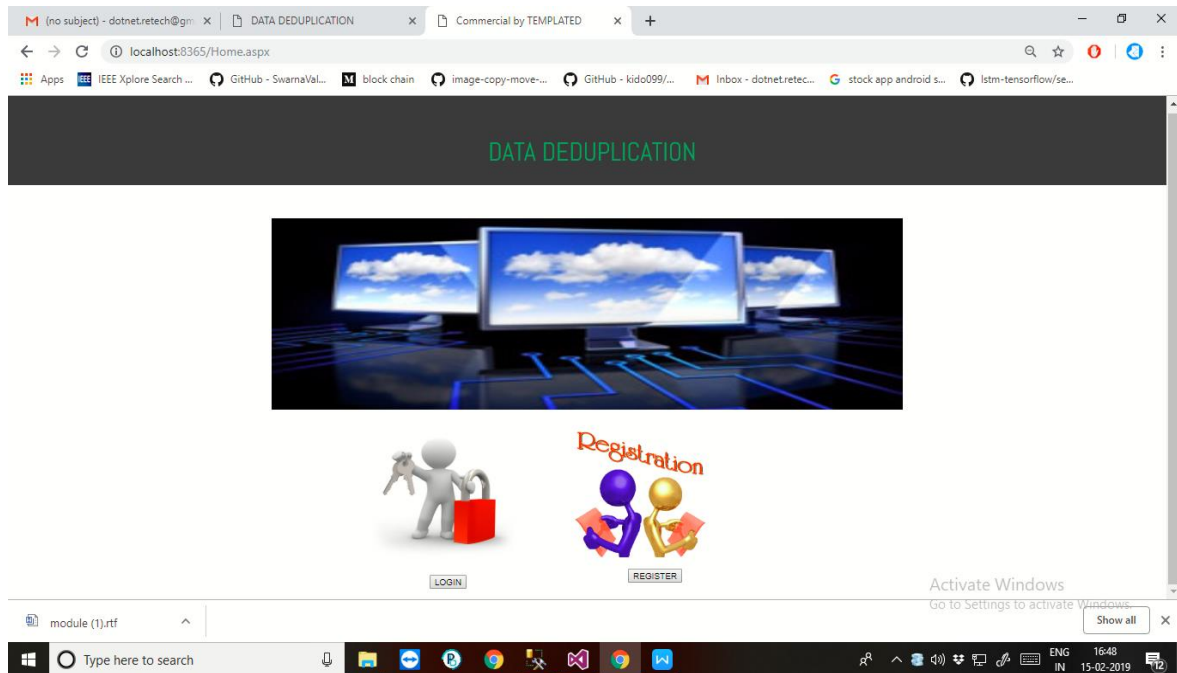


FIG:7.1 HOME PAGE

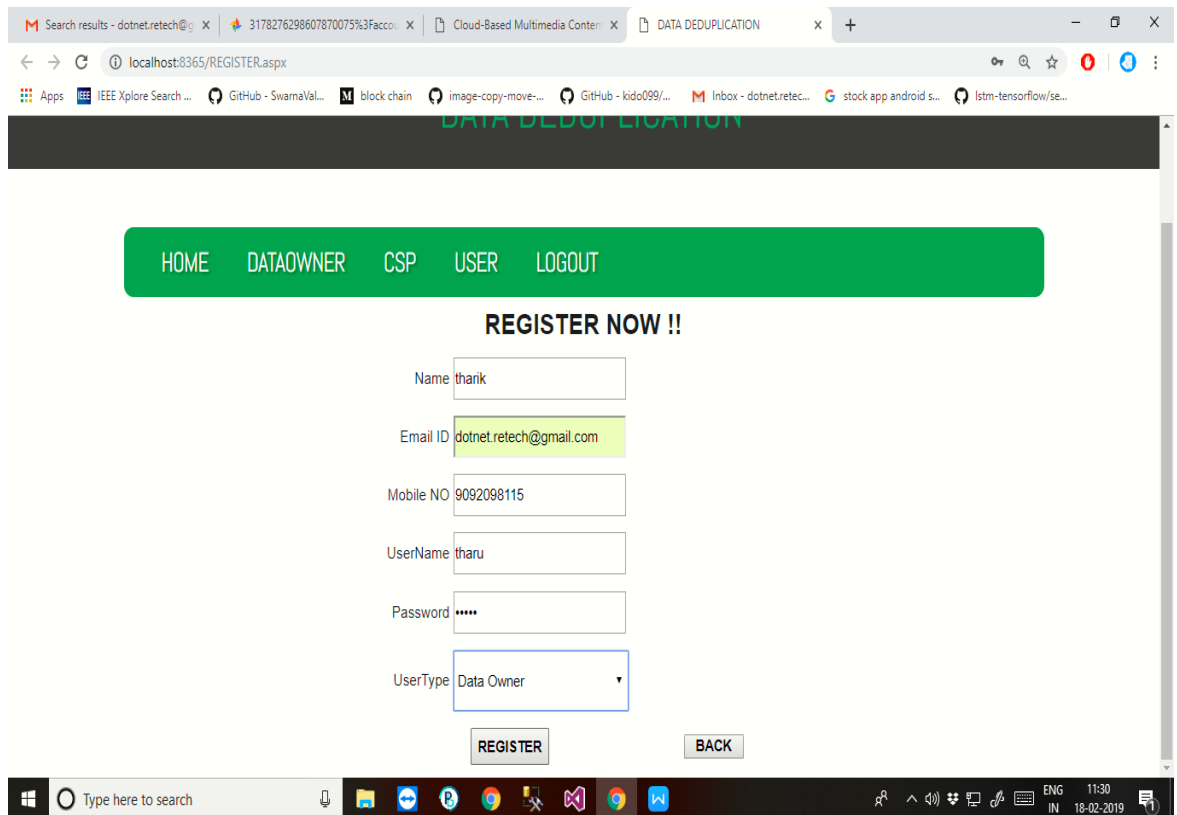


FIG:7.2 REGISTER PAGE

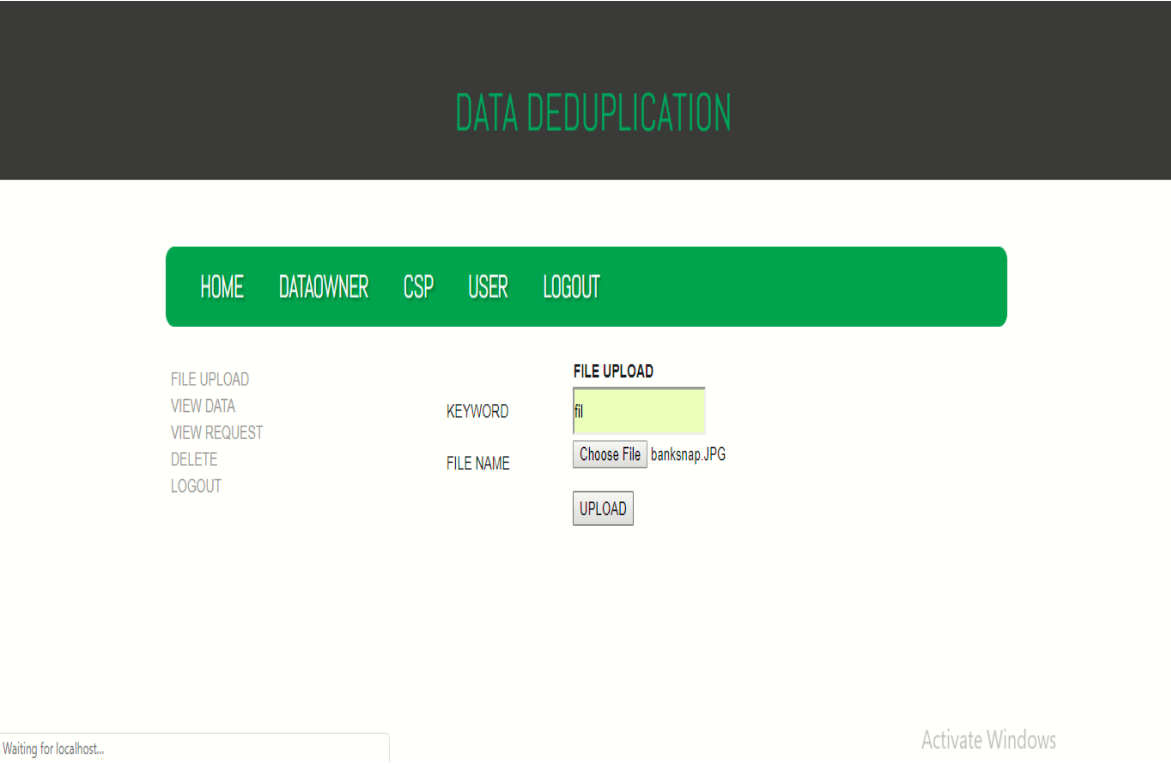


FIG: 7.3 UPLOAD PAGE

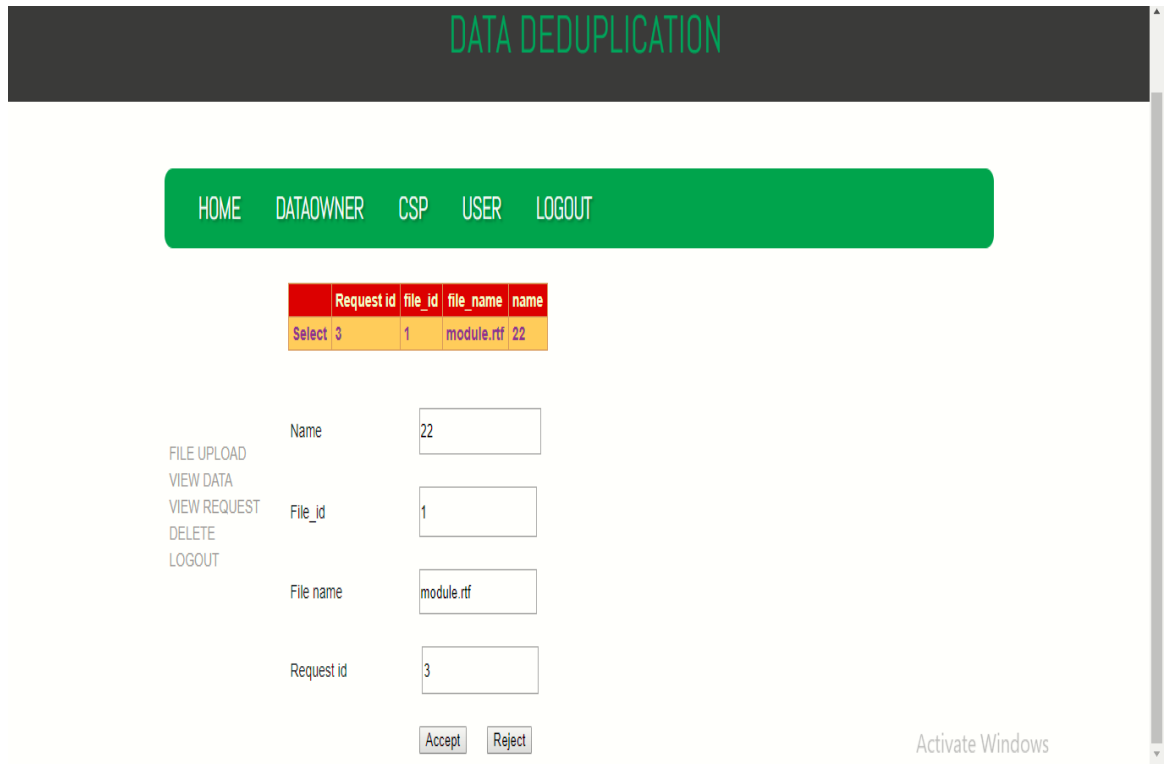


FIG: 7.4 USER REQUEST PAGE

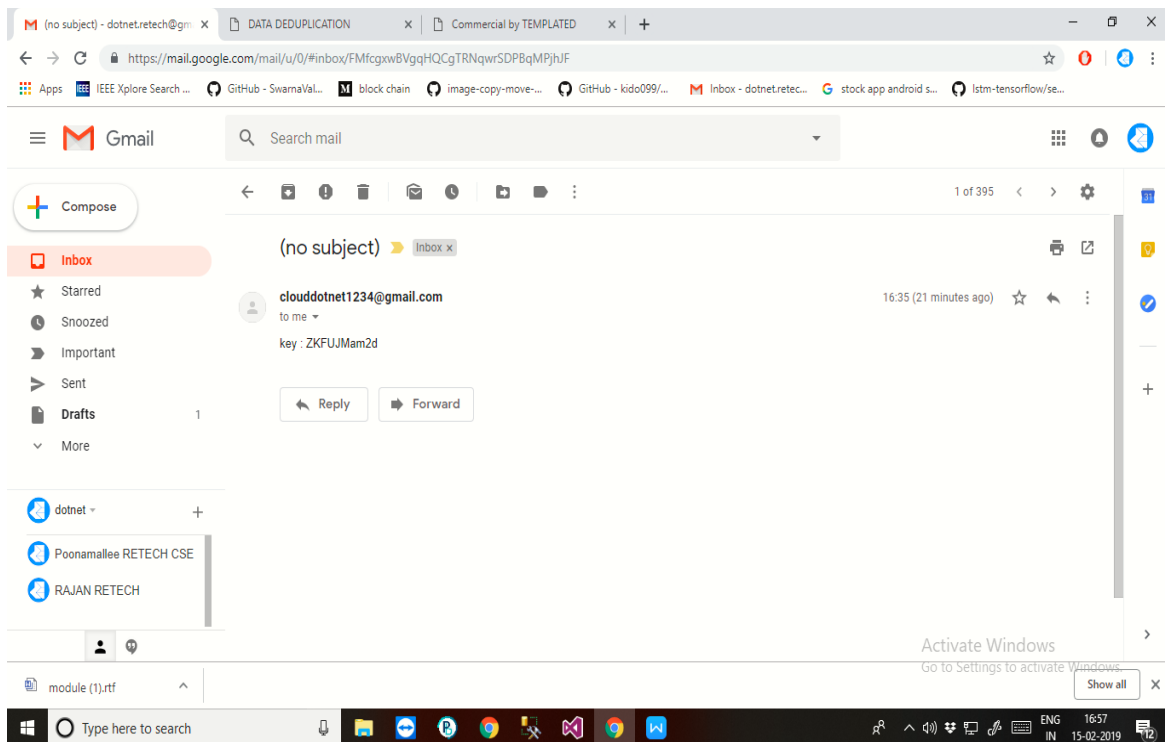


FIG:7.5 KEY GENERATION

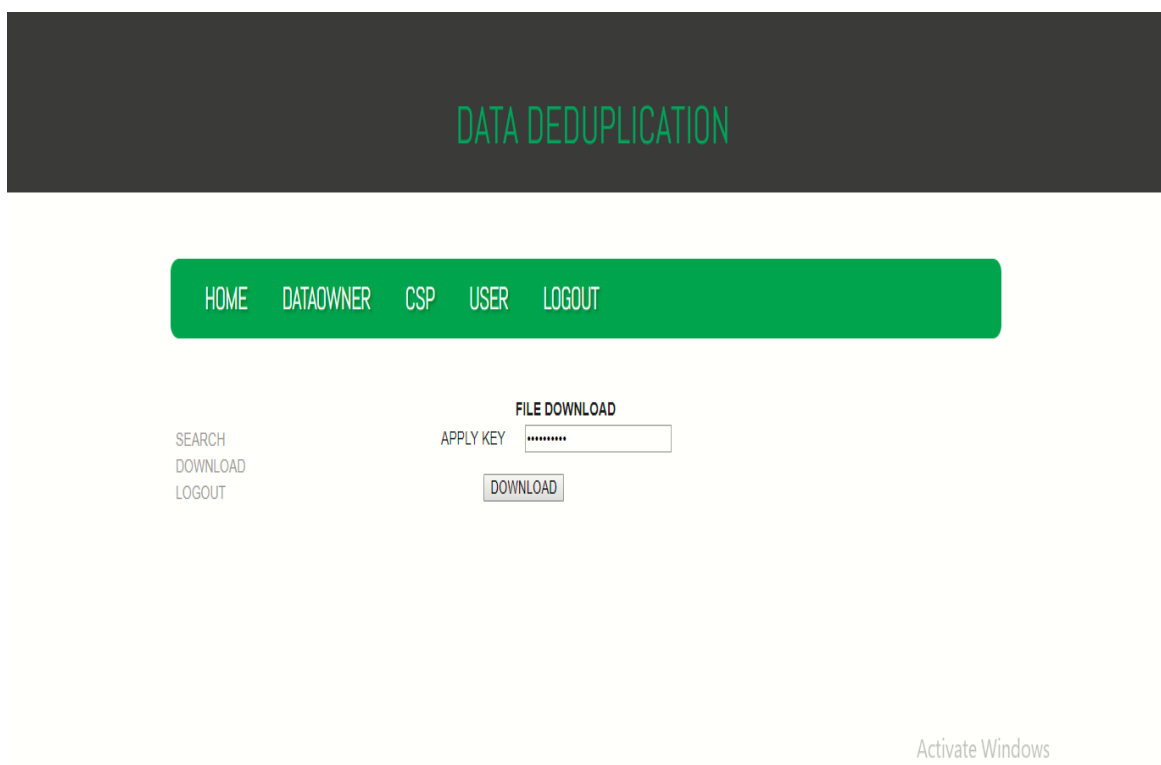


FIG:7.6 FILE DOWNLOAD

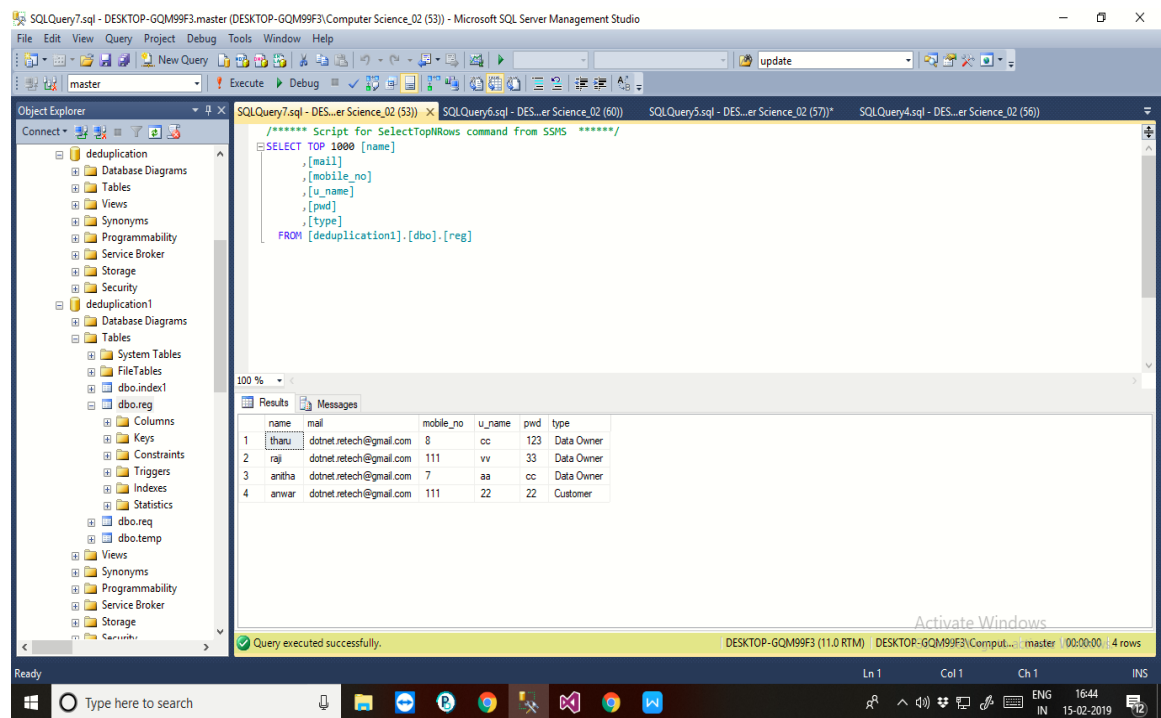


FIG:7.7 DATABASE REGISTERED USERS

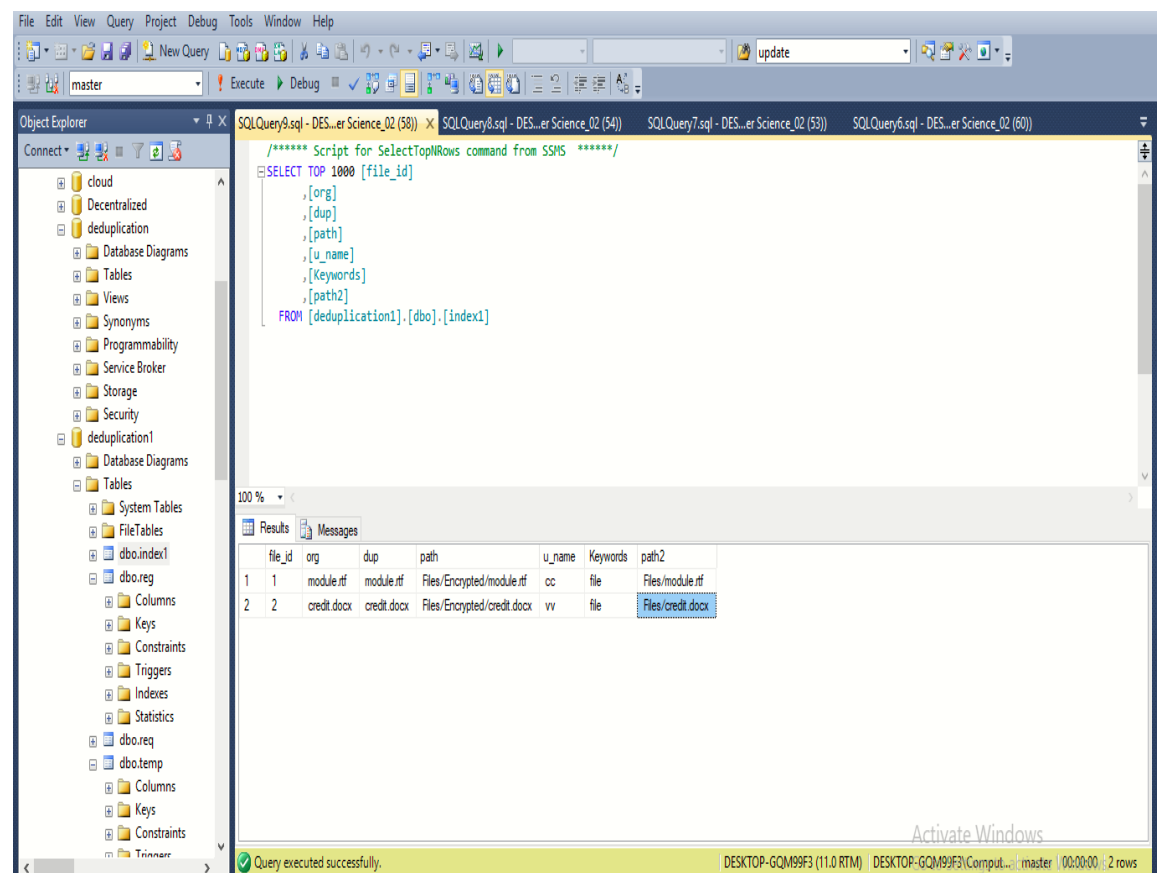


FIG:7.8 DATA

JOURNAL PAPER