DATA DEPULICATION USING OWN CLOUD FOR EFFICIENT CLOUD STORAGE MANAGEMENT

ST. TERESA'S COLLEGE (AUTONOMOUS)

AFFILIATED TO MAHATMA GANDHI UNIVERSITY

PROJECT REPORT

In partial fulfilment of the requirements for the award of the degree of

BCA (CLOUD TECHNOLOGY AND INFORMATION SECURITY MANAGEMENT)

By

Saniya Sajo - SB22BCA031

III DC BCA (CLOUD TECHNOLOGY AND INFORMATION SECURITY MANAGEMENT)

Under the guidance of

Ms. Veena Antony

DEPARTMENT OF CYBER SECURITY AND APPLIED COMPUTING

MARCH 2025

DATA DEPULICATION USING OWN CLOUD FOR EFFICIENT CLOUD STORAGE MANAGEMENT

ST. TERESA'S COLLEGE (AUTONOMOUS)

AFFILIATED TO MAHATMA GANDHI UNIVERSITY

PROJECT REPORT

In partial fulfilment of the requirements for the award of the degree of

BCA (CLOUD TECHNOLOGY AND INFORMATION SECURITY MANAGEMENT)

By

Saniya Sajo - SB22BCA031

III DC BCA (CLOUD TECHNOLOGY AND INFORMATION SECURITY MANAGEMENT)

Under the guidance of

Ms. Veena Antony

DEPARTMENT OF CYBER SECURITY AND APPLIED COMPUTING

MARCH 2025

DECLARATION

I, undersigned, hereby declare that the project report, **'DATA DEPULICATION USING OWN CLOUD FOR EFFICIENT CLOUD STORAGE MANAGEMENT'**, submitted for partial fulfillment of the requirements for the award of degree of BCA (Cloud Technology and Information Security Management) at St. Teresa's College (Autonomous), Ernakulam (Affiliated to Mahatma Gandhi University), Kerala, is a bonafide work done by us under the supervision of Ms.Veena Antony. This submission represents our ideas in our own words and where ideas or words of others have not been included. We have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to the ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Ernakulam
March 2025

Saniya Sajo – SB22BCA031
Kripa Joseph – SB22BCA020

ST. TERESA'S COLLEGE (AUTONOMOUS), ERNAKULAM

BCA (CLOUD TECHNOLOGY AND INFORMATION SECURITY MANAGEMENT)

DEPARTMENT OF CYBER SECURITY AND APPLIED COMPUTING

CERTIFICATE

This is to certify that the report entitled "**DATA DEPULICATION USING OWN CLOUD FOR EFFICIENT CLOUD STORAGE MANAGEMENT**", submitted by Saniya Sajo to the Mahatma Gandhi University in partial fulfillment of the requirements for the award of the Degree of BCA (Cloud Computing and Information Security management) is a bonafide record of the project work carried out by them under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

VEENA ANTONY	**ARCHANA MENON P**	
Internal Supervisor	**Head of the Department**	**External Supervisor**

Department of Cyber Security and Applied Computing
St. Teresa's College (Autonomous) Ernakulam

ACKNOWLEDGEMENT

First and foremost, I thank God Almighty for his blessings. I take this opportunity to express my gratitude to all those who helped me in completing this project successfully. I wish to express my sincere gratitude to the Directors **Rev. Sr. Tessa CSST** and **Rev. Sr Francis Ann CSST** and the Principal **Dr. Alphonsa Vijiya Joseph** for providing all the facilities.

I express my sincere gratitude to the Head of the Department, **Ms. Archana Menon P**, for the support. We deeply express sincere thanks to my project guide **Ms. Veena Antony** for her proper guidance and support throughout the project work.

I are indebted to our beloved teachers whose cooperation and suggestion throughout the project which helped me a lot. I thank all my friends and classmates for their support.

I convey my hearty thanks to my parents for the moral support, suggestion and encouragement.

ABSTRACT

With the growing reliance on cloud storage services, the problem of redundant data storage has become increasingly significant, leading to inefficiencies in storage utilization and higher costs. This project focuses on implementing a data deduplication solution using OwnCloud, an open-source cloud storage platform. The objective is to ensure that only unique data is stored on the cloud, eliminating unnecessary duplicates while maintaining a seamless user experience. In this system, when a user uploads a file to the cloud, the system performs a check for duplicate files by comparing file attributes and content. If an identical file is detected, it is not uploaded again, and only a reference to the original file is stored in the cloud, saving storage space. The deduplication process occurs transparently to the user, who interacts with the interface as if they had uploaded the file normally. This project aims to improve storage efficiency, reduce operational costs, and enhance data management on cloud platforms while ensuring that the integrity and accessibility of user data are maintained.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

Sl. No.	ABBREVIATION	FULL FORM
1.	AI	Artificial Intelligence
2.	ML	Machine Learning
3.	AES-256	Advanced Encryption Standard - 256 bit
4.	SHA-256	Secure Hash Algorithm - 256 bit
5.	CDC	Content-Defined Chunking
6.	RNN	Recurrent Neural Network
7.	MFA	Multi Factor Authentication
8.	RBAC	Role-Based Access Control
9.	VM	Virtual Machine
10.	AWS	Amzaon Web Services
11.	HDFS	Hadoop Distributed File System
12.	CNN	Convolutional Neural Network
13.	LSTM	Long Short-Term Memory
14.	GPT	Generative Pre-trained Transformer

Chapter 1
INTRODUCTION

Cloud storage has revolutionized the way data is stored and accessed, offering scalability, convenience, and cost-effectiveness. However, as data volume grows, storage redundancy becomes a major challenge, leading to inefficient use of storage resources and increased operational costs. Many cloud storage platforms lack built-in mechanisms to prevent the uploading of duplicate files, resulting in wasted storage space and unnecessary data management overhead.

Data deduplication is a technique that addresses this issue by ensuring that identical files are not stored multiple times. By implementing deduplication, storage efficiency is improved, reducing costs and optimizing resource utilization. This project proposes a cloud storage management system that integrates data deduplication with OwnCloud, an open-source cloud storage platform, to streamline file storage operations. In addition to deduplication, user engagement plays a vital role in enhancing the functionality and user experience of cloud systems. To achieve this, the proposed system incorporates a User Feedback System that allows users to submit feedback regarding their storage experience. This feedback is stored securely and managed through an admin panel, enabling administrators to analyze and implement necessary improvements based on user input.

Built using Python Django, the system architecture consists of four key modules: User, Admin, Base, and Feedback. The combination of deduplication and user feedback management provides a more efficient, structured, and user-centric cloud storage solution. By reducing redundancy and integrating feedback mechanisms, this project enhances storage efficiency while improving overall user satisfaction in cloud storage management.

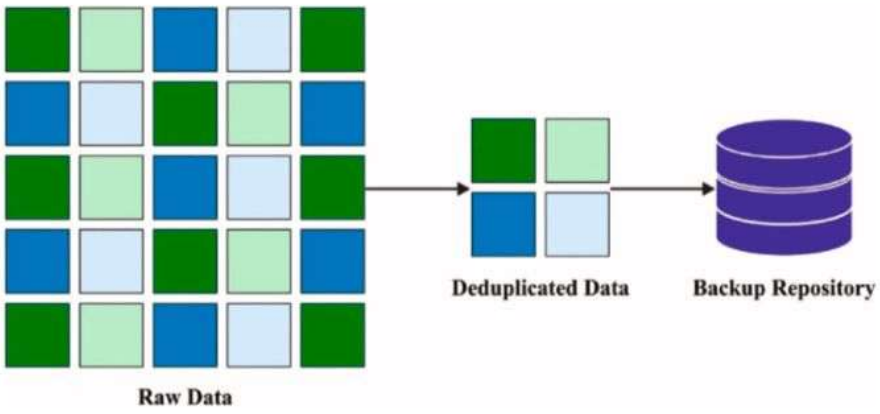

figure 1.1 Deduplication process

1.1 History of deduplication

Data deduplication has evolved significantly over the years, transforming from manual data management practices into highly sophisticated AI-driven storage optimization techniques. Initially, in the early days of computing, storage was expensive and limited, requiring IT professionals to manually identify and remove redundant files, a time-consuming and inefficient process. As digital data grew exponentially, the need for automation led to the development of file-level deduplication in the 1980s and 1990s, where duplicate files were detected based on metadata such as file names and sizes, allowing only a single copy to be stored. While effective for basic redundancy elimination, file-level deduplication was unable to detect duplicate content within files, leading to the emergence of block-level deduplication in the late 1990s and early 2000s, which divided files into fixed-size or variable-size blocks and stored only unique blocks. This approach was widely adopted by companies such as EMC, NetApp, and Veritas in enterprise storage and backup systems, significantly optimizing storage use in backup and disaster recovery solutions. By the mid-2000s, deduplication evolved further with the introduction of inline and post-process techniques, where inline deduplication eliminated redundancy in real-time as data was written to storage, reducing storage needs immediately but demanding high computational power, while post-process deduplication scanned and removed redundant data after storage, lowering CPU load but requiring temporary extra space. As cloud computing gained traction, major providers such as Amazon Web Services (AWS), Google Cloud, and Microsoft Azure incorporated deduplication to optimize cloud storage, reducing storage costs and bandwidth consumption. The 2010s saw the rise of AI and machine learning in deduplication, enabling pattern recognition, intelligent data

compression, and predictive analytics to enhance efficiency and reduce computational overhead. At the same time, hybrid cloud deduplication emerged as a crucial innovation, ensuring sensitive data remained secure in private clouds while non-sensitive data benefited from cost-efficient deduplication in public clouds. More recently, blockchain-based deduplication has been explored to create tamper-proof and secure deduplication records, while federated deduplication enables multiple cloud platforms to share deduplicated data without compromising security or data sovereignty. Today, deduplication is an integral part of cloud storage, enterprise backup, and data management systems, continually evolving to meet the demands of ever-growing data volumes while improving efficiency, security, and scalability.

1.2 Importance of Deduplication in Cloud Storage

Data deduplication is a critical process in cloud storage that enhances storage efficiency, reduces costs, and improves data management by eliminating redundant copies of data. As cloud storage adoption continues to grow, enterprises and individuals generate vast amounts of data, much of which contains duplicate files or blocks. Without deduplication, cloud providers would require significantly more storage space, leading to increased infrastructure costs and inefficient resource utilization. Deduplication ensures that only unique data is stored while duplicate instances are replaced with references to the original copy, significantly reducing storage consumption. This optimization not only lowers storage costs for cloud service providers but also benefits end-users by offering cost-effective storage plans. Another key advantage of deduplication in cloud storage is bandwidth efficiency. When data is uploaded or backed up to the cloud, deduplication minimizes the amount of data that needs to be transferred, resulting in faster upload speeds and lower network congestion. This is particularly beneficial for enterprises that rely on cloud-based disaster recovery and backup solutions, as deduplication reduces backup windows and improves recovery times. Additionally, deduplication enhances security and compliance by reducing the attack surface of sensitive data. By storing fewer redundant copies, organizations can better manage access controls and encryption policies, mitigating risks associated with data breaches. In hybrid and multi-cloud environments, deduplication also optimizes data synchronization across multiple

storage locations, ensuring efficient replication without unnecessary redundancy. As cloud storage demands continue to increase, deduplication remains an essential technology for scalability, performance, and cost savings.

Applications of Deduplication

Deduplication technology is essential in cloud storage, data management, and enterprise applications, significantly reducing storage costs and improving efficiency. Cloud service providers like Amazon Web Services (AWS), Google Cloud, and Microsoft Azure use deduplication to eliminate redundant data, optimizing storage utilization. It is widely used in backup and disaster recovery solutions, such as Veeam and Veritas NetBackup, to store only unique data blocks, minimizing storage requirements and accelerating recovery processes. Network bandwidth optimization is another key application, as deduplication reduces redundant data transfers, improving remote synchronization, cloud backups, and replication. In virtualized environments, platforms like VMware ,vSAN and Microsoft Hyper-V eliminate duplicate virtual machine (VM) images, enhancing storage efficiency. Email and document management systems, including Microsoft Exchange and Google Workspace, also benefit from deduplication by removing redundant attachments and duplicate documents. In big data analytics, deduplication helps manage large datasets efficiently, reducing redundant storage in Hadoop-based systems. The technology also enhances security and compliance by minimizing redundant files, simplifying encryption and access control enforcement. In healthcare, it optimizes electronic health records (EHRs) and medical imaging storage. Additionally, blockchain-based deduplication ensures data integrity, making it valuable in finance, legal records, and supply chain management.

1.3 Technologies used in Deduplication

Deduplication in cloud storage relies on several advanced technologies to identify and eliminate redundant data efficiently while maintaining security and performance. One of the fundamental technologies is cryptographic hashing algorithms, such as SHA-256, MD5, and SHA-1, which generate unique hash values for data blocks or files, allowing the system to detect and remove duplicates quickly. Another essential technology is content-defined chunking (CDC), used in variable-size block deduplication, where intelligent algorithms identify natural breakpoints within data to optimize deduplication accuracy. Machine learning (ML) and artificial intelligence (AI) have also enhanced deduplication by enabling pattern

recognition, anomaly detection, and predictive storage optimization. AI-powered deduplication systems analyze historical data usage patterns to improve deduplication efficiency while minimizing processing overhead.

Modern deduplication systems use data compression techniques, such as lossless compression algorithms (e.g., LZ77, Huffman coding, and Zstandard), to further reduce storage requirements by minimizing the size of unique data blocks. Inline and post-process deduplication leverage real-time processing or scheduled batch operations, depending on performance requirements, while global deduplication enables deduplication across multiple cloud environments, optimizing storage at a large scale. Additionally, blockchain-based deduplication is emerging as a secure method for maintaining immutable records of deduplicated data, ensuring integrity and preventing unauthorized modifications. Hybrid cloud deduplication integrates private and public cloud resources to optimize storage allocation while ensuring security compliance. As cloud storage demands increase, these technologies continue to evolve, making deduplication more efficient, scalable, and secure.

Chapter 2
LITERATURE SURVEY

The literature survey has been conducted on various research papers, including technical papers and review articles, published in leading publications, journals, and conferences. Keywords such as data deduplication, Django, OwnCloud, cloud storage optimization, and file redundancy detection were used to filter relevant studies. The focus was on understanding existing techniques, frameworks, and algorithms for detecting and eliminating duplicate data, particularly in cloud-based storage systems like OwnCloud.

Data deduplication is a widely accepted technique for minimizing storage costs by eliminating duplicate data across various storage systems, including archives, backups, primary storage, SSDs, and RAM. Despite its broad application, challenges and trade-offs differ for each storage type, often leading to misconceptions about the relevance of new research and development. This article classifies deduplication systems based on six key design criteria: granularity, locality, timing, indexing, technique, and scope. Each criterion represents essential design decisions, with different approaches employed depending on the type of storage system. The article also explores how specific combinations of these design decisions address the unique challenges associated with each storage type. Furthermore, the study identifies significant research challenges and unexplored design points, providing insights into areas where improvements can be made. This classification framework serves as a foundational reference for developing more efficient deduplication systems and offers guidance for future research aimed at enhancing deduplication techniques across diverse storage environments.[1]

DD techniques aim to eliminate redundant data, optimizing storage utilization and reducing data access time. DD allows the storage of unique data copies while mapping duplicates to the original data through pointers. Despite various DD methods being developed, no single approach effectively manages all redundancy types. Each method differs in design, performance, and overhead, tailored to specific redundancy scenarios. When datasets contain

numerous duplicates, DD compares files without examining content, resulting in faster processing. However, for similar but not identical files, DD analyzes file content to identify matching portions within previously stored data, enhancing storage efficiency.This paper categorizes existing DD approaches based on granularity, deduplication location, and deduplication timing. Effective redundancy detection is achieved through techniques like hashing (chunk indexing) and bloom filters. The study further explains how these approaches function, highlighting their strengths and limitations. Understanding the diversity of DD techniques is essential for developing efficient, comprehensive solutions that balance performance and storage optimization in various applications.[2]

Data deduplication has become a cornerstone in optimizing cloud storage systems by eliminating redundant data, thereby enhancing storage efficiency and reducing costs. The author provides a comprehensive survey titled " A Review on Secure Data Deduplication: Cloud Storage Security Issue" where they delve into various secure deduplication techniques. The authors categorize these techniques into client-side, server-side, and hybrid approaches, each addressing specific security challenges inherent in cloud storage environments. A significant focus is placed on privacy-preserving mechanisms, notably convergent encryption, which ensures that identical data blocks generate the same ciphertext, facilitating deduplication while maintaining confidentiality. The survey also highlights potential vulnerabilities, such as data leakage and unauthorized access, underscoring the necessity for robust security measures. By systematically analyzing existing schemes, the authors provide valuable insights into the strengths and limitations of current methodologies, paving the way for future research directions in secure data deduplication.[3]

In the realm of integrated cloud-edge networks, the authors propose a novel approach in their paper "A Secure Data Deduplication System for Integrated Cloud-Edge Networks." They introduce a system that combines Convergent Encryption (CE) with Modified Elliptic Curve Cryptography (MECC) to identify data redundancy at the block level. This dual-layered encryption ensures that data remains secure during the deduplication process, addressing the challenges of implementing deduplication over encrypted data in dynamic cloud-fog environments. The proposed method not only reduces storage space but also enhances computational efficiency, making it particularly suitable for environments where data security

and storage optimization are paramount. The study's findings suggest that this integrated approach effectively balances the trade-offs between security and efficiency, offering a viable solution for secure data deduplication in modern cloud architectures.[4]

The paper "A Review on Secure Data Deduplication: Cloud Storage Security Issue" delves into various proposed approaches for secure deduplication techniques in cloud storage. The authors conduct a literature review on the subject, discussing the challenges and solutions associated with implementing secure deduplication. They emphasize the importance of balancing storage efficiency with data security, highlighting the need for robust encryption methods and access control mechanisms. The review serves as a valuable resource for understanding the current state of secure deduplication practices and the ongoing efforts to enhance data security in cloud storage systems.[5]

In "Secure and Efficient Deduplication for Cloud Storage with Dynamic Ownership Management," the authors propose a deduplication protocol that addresses the challenges of dynamic ownership management in cloud storage. They introduce a server-aided encryption scheme that supports secure deduplication while allowing dynamic changes in data ownership. The protocol ensures that only authorized users can access deduplicated data, enhancing data security and privacy. The study demonstrates that the proposed scheme achieves both storage efficiency and secure data management, making it a promising solution for cloud storage providers.[6]

Private data deduplication protocols aim to enhance data storage efficiency while ensuring privacy. Unlike public data deduplication protocols, which focus on eliminating redundant public data storage, private data deduplication addresses the challenges associated with secure storage and ownership verification of private data. Halevi et al. [7] pioneered public data deduplication protocols, but their approach lacks privacy guarantees for sensitive information. Building on this, the concept of private data deduplication protocols was introduced to ensure that clients can prove ownership of private data to a server without disclosing the data itself. The security of these protocols is framed within the simulation-based paradigm in two-party computations, emphasizing protection against malicious adversaries. The proposed protocols

rely on standard cryptographic assumptions, including collision-resistant hash functions, the hardness of the discrete logarithm problem, and erasure coding algorithms capable of handling α-fraction of data loss. This approach offers a novel solution for privacy-preserving deduplication, making it a significant advancement in secure data storage systems.[7]

The paper *"Web-Based Feedback Supervision System"* by K. Vinothini, V. Kiruthika, M. Abilash Hariram, K. Gayathri, and S. Keerthana presents a web-based application designed to improve feedback management within organizations. This system aims to streamline the process of collecting, storing, managing, and updating feedback related to employee and client performance. By effectively handling feedback, the proposed system enhances productivity, encourages innovation, and promotes a healthy work environment. It provides a structured approach to feedback management, allowing organizations to document and monitor performance efficiently. Additionally, the system contributes to building a culture of continuous improvement and growth, irrespective of an individual's position within the organizational hierarchy. The authors highlight that top-performing organizations actively seek ways to enhance their processes, making this feedback system valuable for sustaining high-performance standards. Overall, the study emphasizes that a digital feedback management system is essential for organizations to stay competitive and foster personal and professional growth among employees.[8]

The study "A Survey on DE–Duplication Schemes in Cloud Servers for Secured Data Storage" provides an in-depth analysis of deduplication schemes aimed at securing data storage in cloud servers. The authors compare various deduplication techniques in terms of their performance, security features, and implementation challenges. They discuss the pros and cons of each scheme, offering insights into their suitability for different cloud storage scenarios. The survey serves as a guide for selecting appropriate deduplication strategies that align with specific security requirements and storage efficiency goals in cloud environments.[9]

Data deduplication is a widely accepted technique for minimizing storage costs by eliminating duplicate data across various storage systems, including archives, backups, primary storage, SSDs, and RAM. Despite its broad application, challenges and trade-offs differ for each storage

type, often leading to misconceptions about the relevance of new research and development. This article classifies deduplication systems based on six key design criteria: granularity, locality, timing, indexing, technique, and scope. Each criterion represents essential design decisions, with different approaches employed depending on the type of storage system. The article also explores how specific combinations of these design decisions address the unique challenges associated with each storage type. Furthermore, the study identifies significant research challenges and unexplored design points, providing insights into areas where improvements can be made. This classification framework serves as a foundational reference for developing more efficient deduplication systems and offers guidance for future research aimed at enhancing deduplication techniques across diverse storage environments.[10]

Chapter 3
Existing Systems

Data deduplication is an essential technique in cloud storage and computing environments to optimize data management by identifying and eliminating redundant information. Traditional and modern deduplication systems vary in their implementation based on the level at which deduplication occurs, the methods used for identifying duplicates, and how security is maintained.

3.1 Traditional Deduplication Techniques

Traditional deduplication techniques have been widely used across different storage environments, from local systems to cloud-based platforms. These techniques aim to reduce the storage footprint and optimize data transfer efficiency. The most common methods include:

3.1.1 File-Level Deduplication

How it Works:

- This technique detects duplicate files by comparing entire file contents, names, or metadata (e.g., file size, checksum).
- If a duplicate file is detected, only one copy is stored, and all redundant instances are replaced with a reference to the original file.

Advantages:

- Simple and efficient for environments where large duplicate files are common.
- Reduces storage requirements significantly in organizations with shared documents and media files.
- Faster implementation compared to more granular techniques.

Disadvantages:

- Ineffective when dealing with files that have minor modifications (e.g., different versions of the same document).
- Large files with minor differences still take up redundant space.
- Cannot detect partial similarities within files.

Real-World Example:

- **Google Drive** and **Dropbox** implement file-level deduplication by storing a single copy of the same uploaded file across multiple accounts. If a user uploads an identical file, the system only references the existing copy instead of storing another one.

3.1.2 Block-Level Deduplication

How it Works:

- Instead of comparing entire files, this method breaks data into smaller, fixed-size or variable-size blocks.
- If a block is identical to another stored block, it is referenced instead of storing duplicate copies.

Types of Block-Level Deduplication:

- **Fixed-Size Block Deduplication:** Splits files into uniform-sized blocks (e.g., 4KB, 8KB). Efficient for structured data but may lead to misalignment issues.
- **Variable-Size Block Deduplication:** Uses intelligent algorithms to determine optimal block sizes based on content patterns. More effective for unstructured data.

Advantages:

- More effective than file-level deduplication as it eliminates redundancy even within files.
- Works well for backups and cloud storage solutions with minor changes between versions.
- Reduces bandwidth usage when transferring data.

Disadvantages:

- Computationally expensive due to the need for block-by-block comparison.
- Requires more processing power and memory to manage indexing of stored blocks.

Real-World Example:

- **AWS S3 Glacier** and **Azure Blob Storage** implement block-level deduplication to reduce redundancy in backup and disaster recovery solutions.

3.1.3 Inline Deduplication

How it Works:

- Deduplication occurs in real-time as data is being written to storage.
- The system checks for duplicates before committing data to disk, ensuring that redundant data is never written.

Advantages:

- Reduces storage space usage immediately.
- Eliminates unnecessary write operations, improving efficiency.
- Ideal for high-performance cloud environments and enterprise backup systems.

Disadvantages:

- Requires powerful computing resources to perform real-time analysis.
- May slow down data ingestion processes if the deduplication algorithm is complex.

Real-World Example:

- **VMware vSAN** uses inline deduplication to optimize storage efficiency in enterprise virtualized environments.

3.1.4 Post-Process Deduplication

How it Works:

- Data is written to storage first, and deduplication is performed periodically in the background.
- The system scans stored data for duplicate blocks or files and replaces redundant instances with references.

Advantages:

- Less impact on data write speeds.
- Useful for environments where immediate deduplication is not necessary.
- Allows batch processing of large datasets.

Disadvantages:

- Requires additional temporary storage until deduplication is completed.
- Does not prevent redundant data from being written, which may increase initial storage usage.

Real-World Example:

- **Microsoft Azure Backup** employs post-process deduplication to optimize storage after data has been backed up.

3.2 Limitations of Existing Deduplication Systems

Despite their effectiveness, traditional deduplication systems face several challenges, including:

3.2.1 High Computational Overhead

- Block-level and content-level deduplication require intensive computing power to compare and process large amounts of data.
- Inline deduplication can slow down data ingestion, especially in high-transaction environments.

3.2.2 Inefficient Resource Utilization

- Some deduplication methods fail to balance security, storage efficiency, and performance.
- Deduplication processes can consume excessive CPU and memory resources.

3.2.3 Security Risks

- Deduplicated data stored on public cloud platforms may be vulnerable to unauthorized access.
- Many traditional deduplication methods do not fully implement user-specific access controls.

3.2.4 Single Point of Failure

- Many centralized deduplication systems store metadata in a single location. If the central repository fails, the entire deduplication mechanism may collapse.

3.3 Cloud-Based Deduplication Systems

Most cloud service providers implement deduplication at either the file level or block level. These solutions help optimize storage costs but still have some limitations.

3.3.1 Deduplication in Commercial Cloud Storage Services

- Google Drive & Dropbox: File-level deduplication ensures that identical files uploaded by different users are stored only once.
- Amazon S3 & Azure Blob Storage: Block-level deduplication reduces storage footprint for backups and large data transfers.
- IBM Cloud Object Storage: Implements a combination of inline and post-process deduplication for cost-effective storage.

3.3.2 Challenges in Cloud Deduplication

1. Computational Load: Large-scale deduplication operations require significant processing power.
2. Data Privacy Issues: Deduplication techniques that compare user-uploaded data can lead to privacy concerns.
3. Security & Access Control: Implementing secure deduplication without exposing sensitive data remains a challenge.

Chapter 4

Proposed system

4.1 Overview of the Proposed System

The proposed system introduces an advanced cloud data deduplication mechanism leveraging a hybrid cloud architecture. It enhances traditional deduplication techniques by implementing both file-level deduplication and content-level deduplication while ensuring data security, efficiency, and minimal computational overhead.

4.1.1 Hybrid Cloud Architecture

Hybrid cloud architecture integrates both private and public cloud resources to provide a flexible and cost-effective solution for deduplication.

Private Cloud:

1. Stores and processes sensitive or confidential data, ensuring compliance with security policies.
2. Performs local deduplication to eliminate duplicate files before data is uploaded to the public cloud.
3. Provides greater control over data access and enhances data privacy.

Public Cloud:

1. Stores non-sensitive or less critical data, minimizing storage costs.
2. Supports global deduplication, ensuring that duplicate files across multiple users are eliminated.
3. Reduces the bandwidth required for uploads by storing only unique data.

By combining private and public cloud resources, this approach balances security, performance, and cost optimization while ensuring efficient data storage.

4.2 Key Features of the Proposed System

1. Hybrid Cloud Architecture

- Implements a two-tiered storage system where sensitive data remains protected in a private cloud, while non-sensitive data is efficiently managed in a public cloud.
- Reduces storage redundancy across multiple cloud environments while maintaining security compliance.

2. User-Specific Tokens and Access Control

- User Authentication:
  - Each user is assigned a unique authentication token to prevent unauthorized access.
  - Tokens verify user identity and grant role-based access to stored files.
- Access Control Mechanism:
  - The system enforces granular access control, ensuring that users can only upload, retrieve, or modify their authorized data.
  - Prevents unauthorized file modification, reducing security risks in multi-user environments.

3. File-Level and Content-Level Deduplication

- File-Level Deduplication:
  - Detects and eliminates duplicate files by comparing their file names, sizes, and metadata.
  - Ensures that identical files are stored only once, saving storage space.
- Content-Level Deduplication:
  - Examines file contents using cryptographic hashing (e.g., SHA-256).
  - Stores only unique file segments, even if different versions of the same file exist.

- Ensures maximum redundancy elimination by comparing content rather than just file names.

4. Minimal Overhead

- Optimizes CPU and memory usage by efficiently managing deduplication requests.
- Uses intelligent indexing mechanisms to reduce unnecessary computations.
- Processes only relevant data, ensuring faster deduplication without excessive system resource consumption.

5. Security and Privacy

- End-to-End Encryption:
  - Data is encrypted before uploading using AES-256 encryption to ensure security.
  - Prevents unauthorized access, even if a breach occurs.
- Data Integrity Verification:
  - Utilizes hash-based integrity checks to ensure that stored data remains unaltered.
  - Detects any tampering or unauthorized changes to the data.
- Multi-Factor Authentication (MFA):
  - Adds an additional layer of security by requiring multiple authentication steps before granting access.

6. Scalability

- Designed to handle increasing data volumes efficiently using auto-scaling cloud resources.
- Dynamic Load Balancing distributes deduplication tasks across multiple servers to avoid bottlenecks.

- Supports parallel processing to speed up deduplication in high-traffic cloud environments.

4.3 Deduplication Process

The deduplication process follows a structured workflow to ensure data integrity, efficiency, and security.

Step 1: File Upload and Hash Generation

- When a user uploads a file, the system computes a cryptographic hash of its contents (e.g., SHA-256 or MD5).
- This hash serves as a unique identifier, allowing the system to detect duplicate files efficiently.

Step 2: Hash Comparison with Existing Data

- The computed hash value is checked against an existing hash database.
- The system determines if an identical file or block already exists in storage.

Step 3: Deduplication Decision

1. If a duplicate hash exists:
    - The system prevents re-uploading of the file.
    - Instead of storing the file again, the user gets a pointer to the existing file, saving storage space and bandwidth.
2. If no duplicate is found:
    - The system uploads the file to the appropriate cloud environment (private or public).
    - The newly computed hash is added to the hash database for future reference.

Step 4: Storage and Access Optimization

- The stored data is encrypted before being uploaded to the cloud.
- Files are compressed to further reduce storage consumption.
- Users retrieve deduplicated files via unique references, eliminating redundant downloads.

Advantages of the Proposed System

The proposed system overcomes the limitations of traditional deduplication methods by combining hybrid cloud storage, advanced security, and optimized performance.

1. Improved Storage Efficiency

- Reduces redundant data storage across multiple cloud servers.
- Supports both global (cross-user) and local (single-user) deduplication.

2. Enhanced Performance

- Lowers network bandwidth consumption by eliminating unnecessary uploads.
- Uses parallel deduplication techniques to handle large datasets quickly.

3. Stronger Security and Privacy Controls

- Protects sensitive files with end-to-end encryption.
- Ensures secure user authentication using token-based access control.

4. Cost Optimization

- Minimizes cloud storage costs by avoiding redundant file storage.
- Reduces compute resource consumption, leading to lower operational expenses.

Chapter 5

SYSTEM REQUIREMENTS

This specifies the hardware and the support software required to carry out the development.

Hardware requirement

SOFTWARE REQUIREMENTS

One of the most difficult task is selecting software for the system, once the system requirements is found out then we have to determine whether a particular software package fits for those system requirements. The application requirement:

Operating System : Microsoft windows 8 or Above

Cloud Platform : OwnCloud
Front End : HTML,CSS, Bootstrap
Back End :Django, Python
IDE : VS code

HARDWARE COMPONENTS

The selection of hardware is very important in the existence and proper working of any software. Then selection hardware, the size and capacity requirements are also important.

Processor : Intel core i3 or above
Main Memory : 4GB or Above
Hard Disk Capacity : 40GB or Above
Mouse : Any compatible
Keyboard : Any compatible

Chapter 6

SYSTEM DESIGN AND ARCHITECTURE

6.1 Data flow diagrams

The Data Flow Diagrams (DFDs) provided represent the flow of data in a hybrid cloud system for both users and administrators. Here are the steps for each:

User DFD

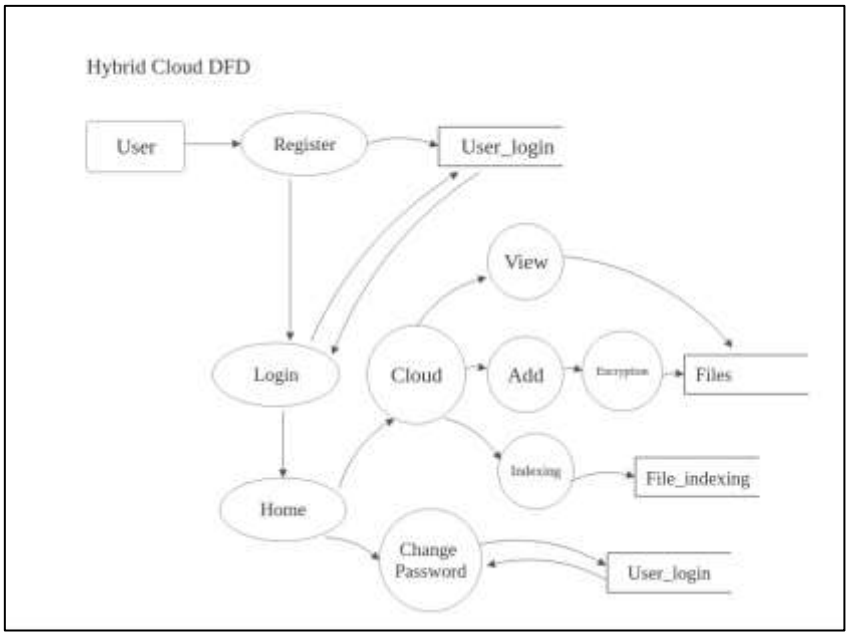

Figure 6.1 User DFD

User DFD Steps:

1. User Registration:
    - The user registers an account.
    - Their credentials are stored under `User_login`.
2. User Login:
    - The user logs in with their credentials.
    - If authentication is successful, access is granted to the system.
3. Navigating to Home:
    - Once logged in, the user reaches the home screen.

- The user can then proceed to different functionalities.

4. Cloud Storage Interaction:
    - The user can add files to the cloud.
    - Files undergo encryption before being stored.
    - Files are indexed for easier retrieval.
5. File Viewing:
    - Users can view stored files.
6. Change Password:
    - The user can update their password.
    - Updated credentials are stored in `User_login`.

Admin DFD

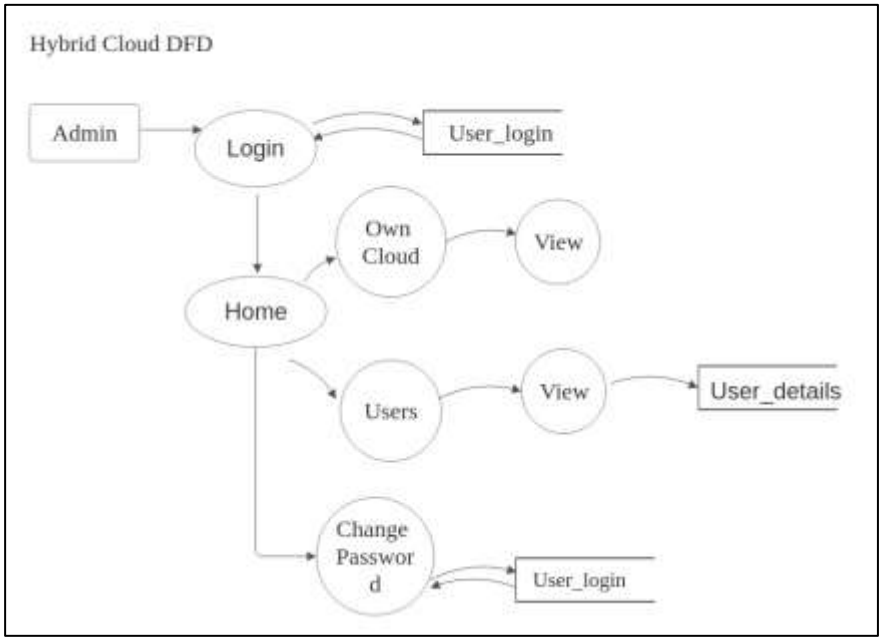

Figure 6.2 Admin DFD

Admin DFD Steps:

1. Admin Login:
    - The admin logs in using their credentials.
    - Authentication is checked via `User_login`.
2. Navigating to Home:
    - Upon successful login, the admin reaches the home interface.

3. Managing Cloud Storage:
    - The admin can view the "Own Cloud" data.
4. User Management:
    - The admin can view user details and manage users.
5. Change Password:
    - The admin can update their password.
    - Updated credentials are stored in `User_login`.

6.2 Proposed System Flow Diagram

The system design shows how the deduplication process works.

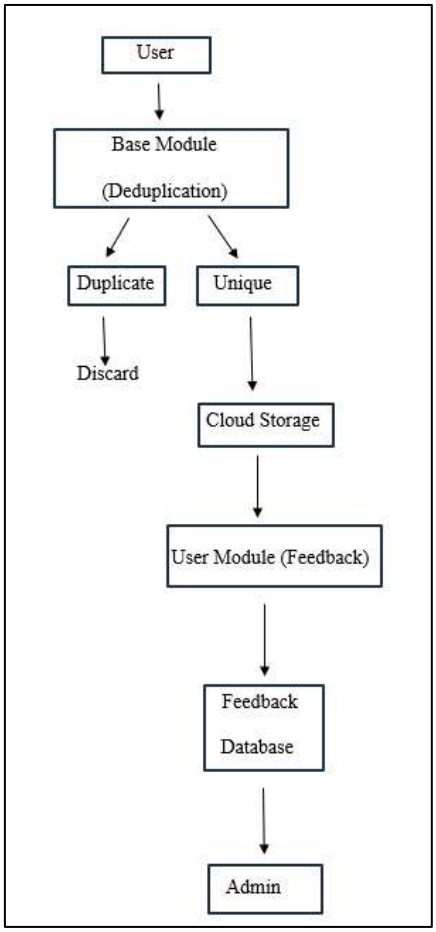

Figure 6.3 Proposed system flow diagram

Steps of the system

1. User Input: The process starts with the user uploading a file to the system.

2. Base Module (Deduplication): The uploaded file is sent to the base module, where the deduplication process is initiated. The system checks whether the file is a duplicate or unique by generating a hash and comparing it with existing hashes.

3. Duplicate Check: If the file is identified as a Duplicate, it is immediately Discarded, preventing redundant storage.

4. Unique File Handling: If the file is identified as Unique, it is forwarded to Cloud Storage for preservation.

5. User Module (Feedback): After successful storage, the user has the option to provide feedback on the system's performance or experience.

6. Feedback Database: The feedback from users is collected and stored in the database for analysis and improvement.

7. Admin Panel: The administrator can access the feedback database through the admin panel to monitor user feedback, manage users, and view system performance.

6.3 Hybrid Cloud

The system architecture is based on a hybrid cloud approach, designed to optimize storage efficiency while maintaining data security. The cloud infrastructure is divided into two parts:

1. Private Cloud:
    - This section of the cloud stores sensitive data that requires higher security and privacy.
    - The primary deduplication process is performed here to ensure data integrity and security before any storage operation.
    - It acts as the main control point for data handling policies and user management.
2. Public Cloud:
    - Handles non-sensitive data, primarily aimed at enhancing storage optimization by storing deduplicated data chunks.

- Provides additional storage space and serves as a backup for non-sensitive files.

6.4 User Authentication and Token Management:

To maintain data privacy and integrity, each user is assigned a unique token. This token-based system helps authenticate and authorize data uploads and retrievals, ensuring that the deduplication process adheres to user privileges and predefined security policies. Tokens are periodically refreshed to maintain security.

6.5 Deduplication Process:

The deduplication process is crucial for optimizing cloud storage. It is divided into two main types:

- Content-Level Deduplication:
  - The content of each file is hashed using a cryptographic hashing algorithm (e.g., SHA-256).
  - The generated hash value is compared with existing hashes stored in the metadata database.
  - If a match is found, the duplicate data is not uploaded, resulting in reduced storage consumption.
- File-Level Deduplication:
  - This process checks if the entire file (by name and size) exists in the cloud.
  - If a file with the same name and size is found, the file is not uploaded again, preventing unnecessary storage of duplicate files.

6.6 Front-End and Back-End:

The system consists of a front-end interface and a back-end database management system:

- Front-End:
  - Developed using Python and Django, providing a user-friendly web application.

- Users can upload their files, view uploaded files, and manage their storage.
- The front-end includes an admin panel that allows administrators to monitor system performance, manage users, and view deduplication statistics.

- Back-End:
  - Managed using SQLite to store metadata such as file hashes, user details, and tokens.
  - It supports efficient querying for deduplication processes and user authentication.

Chapter 7

MODULE DESCRIPTION

7.1 MODULE 1 - ADMIN MODULE

The admin module is responsible for overseeing the entire system. It includes the following functionalities:

- Login: The admin can log in to the system using their credentials.
- Manage Users: The admin has the authority to remove users.
- View Users: The admin can access and monitor the list of registered users.
- View Feedback: The admin can review feedback submitted by users regarding the system.

7.2 MODULE 2 - USER MODULE

The User module allows users to interact with the system and perform essential tasks. It includes:

- Register: New users can create an account.
- Login: Registered users can log in to access the system.
- Upload Files: Users can upload files for storing it in the cloud.
- Deduplication: The system performs deduplication checks by computing and comparing file hashes to identify duplicates before storage.
- Add Feedback: Users can provide feedback on their experience with the system.

7.3 MODULE 3 - BASE MODULE

The Base Module is the core of the deduplication system, handling file storage, deduplication, and cloud integration to optimize storage efficiency. Key Responsibilities:

- File Management: Manages file uploads and stores metadata (file names, sizes, hashes, locations).
- Deduplication Process: Uses SHA-256 hashing to detect duplicates; stores only unique files, linking duplicates to existing ones.
- Cloud Storage Integration: Supports hybrid cloud storage, keeping sensitive data in a private cloud and non-sensitive data in a public cloud.
- Storage Optimization: Reduces storage costs by eliminating redundant files and using compression techniques.
- Performance Monitoring & Logs: Maintains deduplication logs and generates reports for system performance analysis.

7.4 MODULE 4 - FEEDBACK MODULE

The Feedback Module collects and manages user feedback to improve system functionality and user experience.

Key Responsibilities:

- Feedback Collection: Allows users to submit feedback through a form interface regarding system performance, issues, or suggestions.
- Storage & Review: Stores feedback in a database (e.g., MySQL or PostgreSQL) and displays it in the admin panel for review.
- Admin Actions: Admins can analyze feedback, implement improvements, and address reported issues to enhance system efficiency.

Chapter -8

IMPLEMENTATION

8.1 Front-End Development:

The front-end of the system is developed using **Python-Django**, a high-level framework that simplifies the process of building web applications by providing tools for handling URL routing, template rendering, and user authentication. Django's **MTV (Model-Template-View)** architecture is used to separate concerns effectively.

The user interface allows users to:

- **Upload Files:** Users can upload files through a simple web interface with file input fields.
- **View Existing Files:** A dashboard displays a list of files previously uploaded by the user, including file names, upload dates, and sizes.
- **Check for Duplicates:** The interface provides visual feedback to inform the user whether a file being uploaded is a duplicate or not.

Django's **template system** and **static files management** are used to render the HTML, CSS, and JavaScript for the user interface. Forms are managed using Django's **Form Class**, providing robust validation and error handling.

8.2 Back-End Development:

The back-end of the system is powered by **Django's ORM (Object-Relational Mapping)** along with **SQLite** as the database. SQLite is chosen due to its simplicity, ease of setup, and efficiency for handling moderate-sized datasets.

The back-end handles:

- **User Authentication & Management:** Implemented using Django's built-in authentication system.
- **Database Management:** SQLite stores user information, file metadata (such as file name, size, upload date), and file hashes for deduplication purposes.

- **Hash Generation & Storage:** When a file is uploaded, a unique hash is generated using a secure hashing algorithm (**SHA-256**) to ensure that even minor changes in the file result in a completely different hash.

The following tables are created in the SQLite database:

1. **Users:** Stores user credentials and related information.
2. **Files:** Stores file metadata including file name, hash, size, upload date, and user reference.

8.3 Deduplication Logic:

The deduplication mechanism works by:

1. **File Upload Process:**
    - When a user uploads a file, the system reads the file in binary mode and computes a hash value using the SHA-256 algorithm.
    - The generated hash is then compared against existing hashes stored in the database.
2. **Duplicate Checking:**
    - If the computed hash matches a hash already present in the database, the system recognizes it as a duplicate and notifies the user.
    - If the hash does not exist in the database, the file is uploaded successfully, stored in the cloud, and the hash is saved in the database for future reference.
3. **Cloud Storage Integration:**
    - Files that are not duplicates are stored in a designated cloud storage system (e.g., AWS S3, Google Cloud Storage) for better scalability and availability.
4. **Efficient Retrieval:**
    - The system allows users to view their uploaded files with metadata details and download them if needed.

The deduplication logic ensures that redundant data storage is minimized, improving storage efficiency and reducing operational costs. The use of SHA-256 hashing provides a high degree of accuracy in identifying duplicates even if minor modifications are made to the file content.

Chapter -9

RESULT AND ANALYSIS

The system was rigorously tested with multiple file uploads and deduplication processes to evaluate its performance, efficiency, and security. The results of these tests are summarized below:

1: Initial File Upload

Objective: To test the upload functionality and initial hash generation.

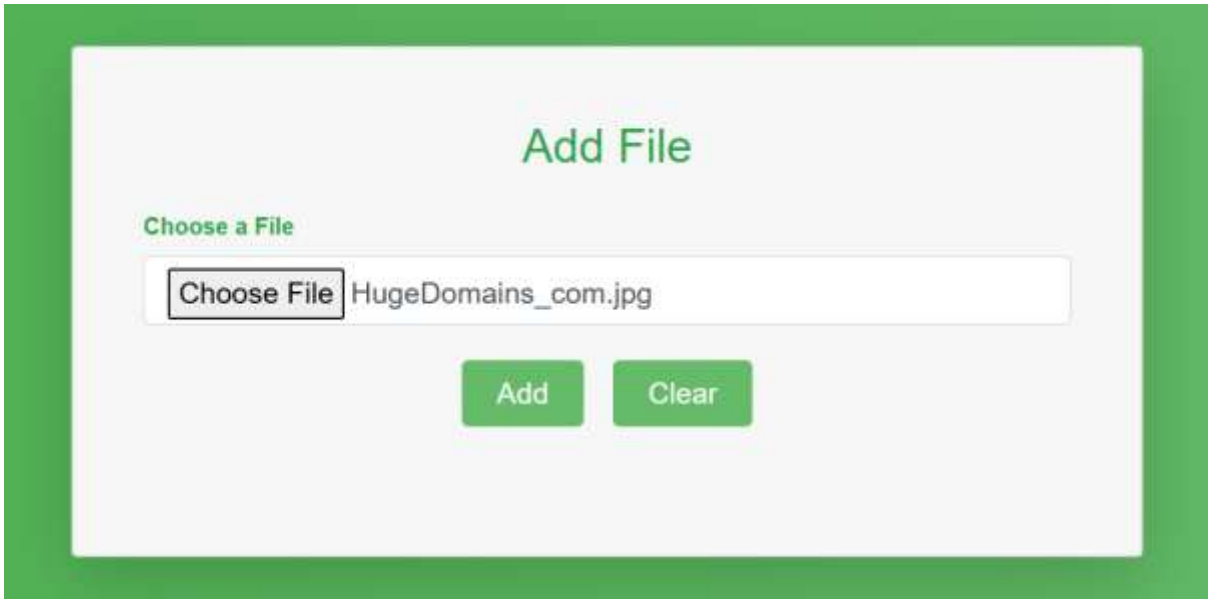

figure 9.1 Uploading file

figure 9.2 uploaded file within the own cloud

2: Duplicate File Detection

Objective: To test the efficiency of the deduplication logic.

figure 9.3 Uploading Two Duplicate Files to the OwnCloud Storage System

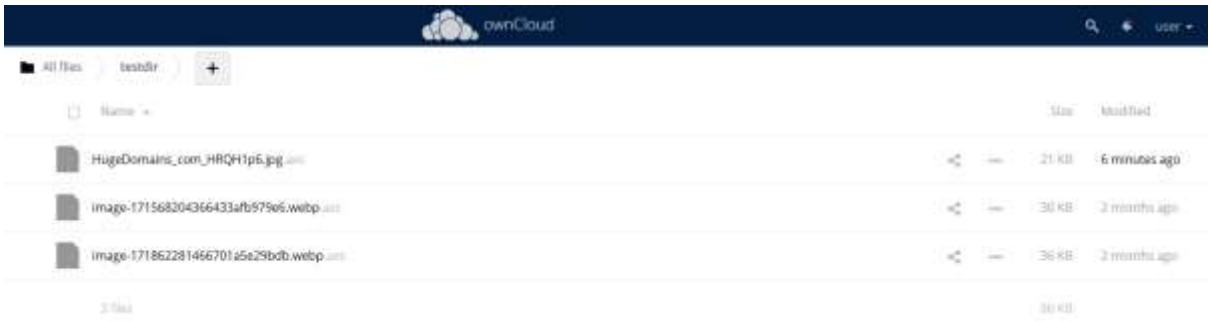

figure 9.4 Only the one original file is stored in the own cloud

3: File Encryption During Upload

Objective: To enhance data security by encrypting files before uploading them to the cloud, ensuring that stored files are protected.

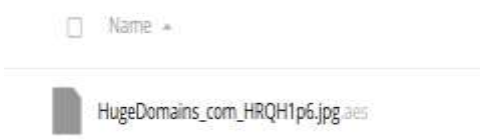

figure 9.5 AES encryption is applied to the files stored in OwnCloud

Chapter 10

CONCLUSION

The project “Data Deduplication Using OwnCloud For Efficient Cloud Storage Management” effectively optimizes storage by eliminating redundant data while maintaining security, efficiency, and scalability. By leveraging file-level and content-level deduplication within a hybrid cloud architecture, the system ensures that sensitive data remains protected in a private cloud, while non-sensitive data is efficiently stored in a public cloud, reducing storage costs and bandwidth usage. The implementation of inline deduplication allows real-time redundancy elimination, improving performance and reducing computational overhead. User-specific tokens and encryption mechanisms enhance data security, ensuring that only authorized users can access or modify stored files. The system integrates secure hashing (SHA-256), adaptive block sizing, and AI-driven deduplication techniques, further improving accuracy and efficiency. With applications in enterprise storage, cloud backup solutions, disaster recovery, and hybrid cloud management, this deduplication system provides a scalable and cost-effective solution for modern data storage challenges. Built using Python, Django, SQLite, and cloud infrastructure, the framework can be enhanced with advanced AI models, blockchain-based integrity verification, and real-time analytics to further improve performance. As data storage demands continue to rise, this system offers a future-proof, high-performance, and secure solution for cloud-based deduplication.

REFERENCES

[1]Paulo, J., & Pereira, J. (2014). A survey and classification of storage deduplication systems. *ACM Computing Surveys*, *47*(1), 1–30. https://doi.org/10.1145/2611778

[2]Concepts, strategies, and challenges of data deduplication. (n.d.). https://www.sciencedirect.com/science/article/pii/B9780128233955000136

[3] A Review on Secure Data Deduplication: Cloud Storage Security Issue. (n.d.). https://www.sciencedirect.com/science/article/pii/S1319157820305140

[4] G, S. P., K, N. R., Menon, V. G., P, V., Abbasi, M., & Khosravi, M. R. (2020). A secure data deduplication system for integrated cloud-edge networks. Journal of Cloud Computing Advances Systems and Applications, 9(1). https://doi.org/10.1186/s13677-020-00214-6

[5] A Review on Secure Data Deduplication: Cloud Storage Security Issue. (n.d.). https://www.sciencedirect.com/science/article/pii/S1319157820305140

[6] Lee, M., & Seo, M. (2023). Secure and Efficient Deduplication for Cloud Storage with Dynamic Ownership Management. Applied Sciences, 13(24), 13270. https://doi.org/10.3390/app132413270

[7] *[PDF] Private Data Deduplication Protocols in Cloud Storage*. (n.d.). https://personal.ntu.edu.sg/ygwen/Paper/NWZ-SAC-12.pdf

[8] *Web based Feedback Supervision System - IEEE Xplore*. (n.d.). https://ieeexplore.ieee.org/document/9836001/

[9] A survey on DE – Duplication schemes in cloud servers for secured ... (n.d.). https://www.semanticscholar.org/paper/f1aa5c12066a16346a127786c6221e6b916aac32

[10] *Memory Deduplication as an Advanced Exploitation Vector*. (n.d.). https://www.semanticscholar.org/paper/Dedup-Est-Machina%3A-Memory-Deduplication-as-an-Bosman-Razavi/452a1b37bd6c7a0f26d3094fba5e9d8f9a826f8f

APPENDIX

```
import hashlib
from .own_cloud import *
from .algo import *
import pyAesCrypt
def user_file_store_add(request):
    if request.method == 'POST':
        u_file = request.FILES['document']
        fs = FileSystemStorage()
        fname = fs.get_valid_name(u_file.name)
        fpath = fs.save(u_file.name, u_file)
        ##################
        file_path = os.path.join(BASE_DIR, f'myapp\\static\\myapp\\media\\{fpath}')
     hasher = hashlib.md5()
        with open(file_path, 'rb') as afile:
            buf = afile.read()
            hasher.update(buf)
        print(hasher.hexdigest())
        fsign = hasher.hexdigest()
        ##################
        ##################
        password = "please-use-a-long-and-random-password"
        pyAesCrypt.encryptFile(file_path, file_path + ".aes", password)
```

```
##################
 fsize = fs.size(fpath)
 user_id = int(request.session['user_id'])
 dt = datetime.today().strftime('%Y-%m-%d')
 tm = datetime.today().strftime('%H:%M:%S')
 status = 'notshared'
 #fname
 file_size = fsize
 #dt
 #tm
 signature = fsign
 url = f'testdir/{fpath}.aes'
 #status
 file_obj = file_index.objects.filter(signature=signature)
 file_name = fname
 if len(file_obj) == 0:
    ############OWN Cloud Push ###################
    target = url
    src = file_path
    putfile(src=src,target=target)
    ########################################
    file_i = file_index(fname=fpath, file_size=file_size, signature=signature,
                dt=dt, tm=tm, url=url, status=status)
```

```
            file_i.save()
            file_id = file_index.objects.all().aggregate(Max('id'))['id__max']
##              del_local_file(file_path)
            print('New')
        else:
            file_id = file_obj[0].id
            print('Same')
        #user_id
        #dt
        #tm
        #status
        ufm = user_file_map(file_id=file_id, file_name=file_name, user_id=user_id,
                            dt=dt, tm=tm, status=status)
        ufm.save()
        s_d = storage_details.objects.get(user_id = user_id)
        s_d.used = str( int(s_d.used) + int(file_size) )
        s_d.save()
        #os.remove(file_path)
        context = {'msg': 'File Uploaded'}
        return render(request, './myapp/user_file_store_add.html', context)
    else:
        return render(request, './myapp/user_file_store_add.html')
def user_file_store_delete(request):
```

```
id = request.GET.get('id')

print('id = '+id)

ufm = user_file_map.objects.get(id=int(id))

fi_o = file_index.objects.get(id=ufm.file_id)

msg = 'Record Deleted'

user_id = int(request.session['user_id'])

s_d = storage_details.objects.get(user_id=user_id)

s_d.used = str(int(s_d.used) - int(fi_o.file_size))

s_d.save()

ufm.delete()

fi_list = file_index.objects.all()

ufm_l = user_file_map.objects.filter(user_id=user_id)

context = {'file_list': ufm_l, 'msg':msg, 'fi_list': fi_list}

return render(request, './myapp/user_file_store_view.html',context)
```