

KIGALI INDEPENDENT UNIVERSITY ULK
SCHOOL OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE
P.O. Box: 2280 KIGALI

**EFFICIENT DATA MANAGEMENT IN CLOUD
STORAGE SYSTEMS
CASE STUDY: ULK**

By

ORI Richman Uzochukwu-Theophilus

Roll No. 202110089

Supervisor: DR. MUSABE J. BOSCO

A Dissertation Submitted to School of Science and Technology, Department of computer science in Partial Fulfilment of the Requirements for the Award of Bachelor's Degree in Computer Science

September 2024

DECLARATION

I, ORI Richman hereby declare that this work entitled “**Efficient Data Management in Cloud Storage case study: ULK**” is our original work and has never been presented elsewhere for any academic qualifications in any other university or any other award, except were stated by reference or acknowledgment.

ORI RICHMAN UZOCHUKWU-THEOPHILUS

Date: / /

Signature:

APPROVAL

I, DR. MUSABE J. BOSCO hereby certify that the dissertation titled “**Efficient Data Management in Cloud Storage case study: ULK**” was done and submitted by ORI Richman Uzochukwu-Theophilus under my supervision.

DR. MUSABE J. BOSCO

Date: / /

Signature:

DEDICATION

This research is dedicated firstly to the Almighty God who granted us life and the opportunity to be at the finishing line of our Undergraduate study, then to our lovely parents who constantly supported us financially, emotionally, and spiritually during this journey, and then to our supervisor and head of the department who worked tirelessly to ensure that this research was done accurately and supported us whenever we needed.

ACKNOWLEDGEMENT

We would like to openly thank the Almighty God who granted us this great opportunity to be concluding our study in this prestigious institution. Our heartfelt gratitude goes to the fonder of ULK PROF.DR. RWIGAMBA BALINDA and the school that granted us the opportunity to study in a conducive environment and learn from learned and experienced professors. And finally, we would like to thank our supervisor and dean of science and technology DR. MUSABE J. BOSCO who contributed a lot of support to ensure this research was successful.

Name: ORI Richman Uzochukwu-Theophilus

TABLE OF CONTENTS

DECLARATION	ii
APPROVAL	iii
DEDICATION	iv
ACKNOWLEDGEMENT.....	v
TABLE OF CONTENTS	vi
LIST OF FIGURES	x
LIST OF TABLES	xi
ABBREVIATIONS AND ACRONYMS.....	xii
ABSTRACT.....	xiii
CHAPTER ONE: GENERAL INTRODUCTION	1
1.1. Introduction.....	1
1.2. Background to the study	2
1.3. Problem statement	4
1.4. Research objectives.....	4
1.4.1. General Objective:	4
1.4.2. Specific Objectives:	4
1.5. Research question	5
1.6. Scope of the study	5
1.6.1. Content Scope:	5
1.6.2. Geographical Scope	5
1.6.3. Time Scope	5

1.7. Significance of the study	6
1.7.1. Personnel Interest.....	6
1.7.2. Institutional Interest	6
1.7.3. Public Interest	6
1.8. Project methodology	6
1.9. Limitations	7
1.10. Organization of the project	7
CHAPTER 2: LITERATURE REVIEW	8
2.1. Introduction	8
2.2. Definition of Concepts	8
2.2.1. Data Management	8
2.2.2. Cloud Storage.....	8
2.2.3. Cloud Storage Systems	8
2.2.4. Efficient Data Management	9
2.2.5. Management System.....	9
2.3. Review of Related Literature	9
2.3.1. Building a Cloud Storage Service System	10
2.3.1.1. Introduction	10
2.3.1.2. System Design	11
2.3.2. Data Management Solutions in the Cloud Accessing, sharing, and processing data that's new to Google Cloud Platform (GCP).....	15

2.3.2.1. Introduction.....	15
2.3.3. Google Drive: An Efficient and Collaborative Cloud-Based Data Management Solution.....	18
2.4. Summary.....	19
CHAPTER 3: SYSTEM ANALYSIS AND DESIGN.....	20
3.1. Introduction.....	20
3.2. Analysis of the current system.....	20
3.2.1. Introduction.....	20
3.2.2. Problem of the current system.....	20
3.3. Analysis of the new system.....	21
3.3.1. Introduction.....	21
3.3.2. System requirements.....	21
3.3.2.1. Functional requirements.....	21
3.3.2.2. Non-Functional Requirements.....	22
3.3.4. Methodological Approach.....	25
3.3.4.1. Data Collection Techniques.....	25
3.3.4.2. Software Development Methodology.....	26
CHAPTER 4: SYSTEM IMPLEMENTATION AND TESTING.....	31
4.1. Implementation and Coding.....	31
4.1.1. Introduction.....	31
4.1.2. Description of Implementation Tools and Technology.....	31
4.1.3. Screenshots and source codes.....	32

4.2. Testing	34
4.2.1. Introduction.....	34
4.2.2. Unit testing Outputs	34
4.2.3. Validation testing Outputs	35
4.2.4. Integration testing Outputs.....	36
4.2.5. Functional testing Outputs	36
4.2.6. Acceptance testing Report	36
CONCLUSION AND RECOMMENDATIONS	37
Conclusion	37
Recommendations	37
REFERENCES	39
Appendix	A
Interview Questions	A
Interview Answers	A
Source code.....	B

LIST OF FIGURES

Fig 1: Architecture of a cloud storage system ESIAT (2019)	12
Fig 2: Transformation Processing Layer ESIAT (2019).....	13
Fig 3: Prototype System ZDNet (2023).....	15
Fig 4: Google Space GCP(2023)	16
Fig 5: Data Lifecycle GCP(2023).....	16
Fig 6: Google Cloud Services	17
Fig 7: Google Drive Landing Page Google (2024).....	18
Fig 8: Google Drive Files	19
Fig 9: Functional Diagram	23
Fig 10: Agile Development Model Agile Model (2024)	27
Fig 11: User registration activity	28
Fig 12: Level 0 Data Flow Diagram	29
Fig 13: Entity Relationship Diagram (ERD)	30
Fig 14: Web Application Homepage	32
Fig 15: User Interface Overview.....	32
Fig 16: User account settings page	33
Fig 17: Admin Dashboard.....	33
Fig 18: Registered users.....	34
Fig 19: Part Source code for Homepage	B
Fig 20: Part source code for user interface overview	C
Fig 21: Part source code for user account settings.....	C
Fig 22: Part source code for admin dashboard.....	D
Fig 23: Part source code for registered users	D

LIST OF TABLES

Table 1: Functional and Non-Functional Requirements.....	22
Table 2: Data Dictionary.....	23
Table 3:Users Data Dictionary.....	24
Table 4:Admin Data Dictionary.....	25
Table 5: Unit testing Output	35
Table 6: Validation Testing Output	35
Table 7: Admin functionalities testing output	36

ABBREVIATIONS AND ACRONYMS

API	Application Programming Interface
AWS	Amazon Web Services
CPU	Central Processing Unit
CSS	Cascading Style Sheet
CS3	Cloud Storage Services Conference
DFD	Data Flow Diagram
ERD	Entity Relationship Diagram
ESIAT	Environmental Science and information Application Technology
FC12	Fedora Core 12
GCP	Google Cloud Platform
GDPR	General Data Protection Regulation
HDD	Hard Disk Drive
HIPAA	Health Insurance Portability and Accountability Act
HMAC	Hash-based Message Authentication Code
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
ISP	Internet Service Provider
IT	Information Technology
MIS	Management Information System
PaaS	Platform as a Service
PHP	Hypertext Processor
RAM	Random Access Memory
REST	Representational State Transfer
SAD	System Analysis and Design
SaaS	Software as a Service
SDLC	Software Development Life Cycle
SQL	Structure Query Language
SSADM	Structured System Analysis and Design Method
SSD	Solid State Drive
SOAP	Simple Object Access Protocol

ABSTRACT

In the digital era, effective data management is crucial for organizations that rely on cloud storage solutions to handle vast amounts of data securely and efficiently. This research explores the creation of a cloud storage management system aimed at making data more accessible, secure, and easy to manage in a cloud environment. The system is designed with a user-friendly interface that allows people to effortlessly upload, download, share, and organize their files, all while keeping their data safe with strong security features like user authentication and access controls. Built using React, Node.js, and MYSQL, the system ensures that users can manage their data smoothly across different devices. The project shows that this system not only solves current challenges in cloud storage but is also flexible enough to adapt to future needs, making data management more efficient and reliable.

Key words:

Cloud storage systems

Data management

Efficient storage solutions

Cloud computing

Data scalability

Data redundancy

Cloud storage optimization

Distributed storage

Data replication

Storage performance

Cloud data security

Data retrieval efficiency

Storage cost optimization

Data availability

Big data storage

CHAPTER ONE: GENERAL INTRODUCTION

1.1. Introduction

In today's age of data growth and increasing storage needs Cloud Storage Systems have become essential for data management solutions. Offering scalability, flexibility and cost effectiveness cloud storage systems have transformed how organizations store, retrieve and handle their data resources. However, as data volume and complexity continue to rise challenges related to data management in cloud storage systems are becoming more pronounced TechTarget (2023).

In addition to the benefits of scalability and flexibility, cloud storage systems also bring new considerations in terms of data security, privacy, and compliance. As organizations increasingly rely on cloud-based solutions, they must ensure that sensitive information is protected from breaches and unauthorized access. This has led to a growing demand for encryption technologies, access control mechanisms, and compliance with international data protection regulations such as GDPR and HIPAA. Moreover, efficient data retrieval and reducing latency in data access are key areas where cloud storage providers continue to innovate, offering tiered storage solutions to balance cost and performance Forbes (2024).

This research proposal aims to explore ways to enhance the efficiency of managing data in cloud storage systems. By addressing issues such as minimizing data optimizing latency allocating resources efficiently and ensuring security measures are robust this proposal looks to investigate innovative strategies and technologies that can streamline data management processes in the cloud setting.

The significance of this research lies in its potential to open up possibilities for improving how data is stored and retrieved efficiently. This can lead to system performance overall reduced expenses and heightened data security measures. Through a review of existing literature, empirical analysis work conducted on real world scenarios or experiments) and the development of approaches or methodologies this research aims to make valuable contributions, to the fields of cloud computing and data management.

This proposal seeks to enhance cloud storage systems by encouraging partnerships, between institutions and industry players. The goal is to connect research with real world applications leading to improvements, in efficiency, reliability and scalability.

1.2. Background to the study

The history of cloud storage traces back to the early 2000s when advancements in internet connectivity and virtualization technologies laid the groundwork for a shift in data storage and management. Before the introduction of cloud storage, organizations relied mainly on on-premises/physical storage solutions, which often posed limitations in terms of scalability, accessibility, and cost-effectiveness Doe, A. (2023).

The concept of cloud storage was created as a response to these challenges, offering a distributed and scalable model for storing and accessing data over the internet. One of the pioneering services in this space was Amazon Web Services (AWS), launched by Amazon in 2006, which gave developers a reliable and scalable online storage capabilities TechCrunch (2023)

Following AWS, other tech giants such as Google and Microsoft introduced their own cloud storage options, including Google Cloud Storage and Microsoft Azure Blob Storage, further improving the growth of cloud storage adoption. These platforms offered not only storage but also additional services and functionalities, such as data analytics, machine learning, and database management. This provided a comprehensive solution for various business needs.

The worldwide adoption of mobile devices, social media, and IoT (Internet of Things) contribute to the increased demand for cloud storage, as users and organizations looked for more efficient ways to store, share, and analyze large amounts of data generated from various sources ZDNet (2023).

Moreover, advancements in cloud technology, such as server virtualization, containerization, and software-defined storage, have continued to drive innovation in cloud storage systems, enabling enhanced performance, reliability, and security. With the advent of edge computing and multi-

cloud architectures, the landscape of cloud storage is continuously evolving, offering even greater opportunities for organizations to leverage distributed storage resources closer to the point of data generation.

In recent years, the increase in remote work has highlighted how essential cloud storage is. It serves as a hub for teamwork, making it easy for everyone to access and collaborate on projects whenever and wherever they are Forbes (2023).

Overall, the story of cloud storage is like a fascinating adventure. It started with old-school ways of storing data in physical locations, but now it has evolved into something dynamic and flexible in the cloud. This shift has completely changed how we handle data in today's digital world. And it doesn't stop there. Cloud storage is set to become even more important, driving exciting new innovations and services in various industries.

The story of cloud storage in Africa is like a journey filled with ups and downs, but ultimately one of progress and hope. At first, we faced challenges like limited internet access and concerns about privacy. But as time went on, things started to change. With more people getting online and using smartphones, the potential of cloud storage became clearer TechCabal (2023).

We've seen local and international companies stepping in to offer cloud services tailored to our needs. And with efforts to teach people about digital technology and support small businesses, awareness about the benefits of cloud storage has grown.

Nowadays, you'll find cloud storage being used in all sorts of places across Africa. From helping farmers manage their crops more efficiently to giving students access to educational resources, it's making a real difference in people's lives. And it's not just about big businesses – startups and everyday folks are getting in on the action too, finding new ways to use cloud storage to make their lives easier African Business (2023).

In Rwanda, the journey of cloud storage is a tale of resilience and progress. Despite initial challenges like limited internet access, the country invested in ICT infrastructure and initiatives

to drive digital transformation. Today, cloud storage empowers businesses, government services, and individuals alike, fostering innovation and inclusivity. With a focus on education and entrepreneurship, Rwanda's future in cloud storage shines brightly as it continues to lead the charge towards a digital future.

Cloud storage continues to grow in Rwanda; hence it requires more efficient systems. With the volume of data being processed and uploaded daily, it is proposed to have a faster and more reliable cloud storage system.

1.3. Problem statement

Efficient data management in cloud storage is crucial for ensuring the integrity and accessibility of important documents and transactions. At Kigali Independent University (ULK), the loss of files and documents belonging to staff and students, including critical records such as payment receipts and academic transcripts, poses significant challenges. Such losses can disrupt administrative operations, hinder academic progress, and compromise financial accountability. The current cloud storage solutions lack the robustness and efficiency required to securely manage and store these vital documents, leading to inefficiencies and potential data loss. Therefore, it is essential to develop a more effective data management system that addresses these issues by providing reliable, secure, and efficient cloud storage to safeguard important records and enhance operational efficacy.

1.4. Research objectives

1.4.1. General Objective:

The main objective of this project is to design and implement an efficient cloud storage system that will provide and improve the quality of online storage in ULK.

1.4.2. Specific Objectives:

- i. To develop a straightforward and easy-to-use upload interface that lets staff and students easily add files.
- ii. To build a user-friendly interface that's intuitive and easy to navigate.

- iii. To implement robust security measures like encryption and secure access controls to protect sensitive files.
- iv. To develop an automation report for the ULK time management for decision marking.

1.5. Research question

- i. How to develop a straightforward and easy-to-use upload interface that lets, staff and students easily add files?
- ii. How to build a user-friendly interface that is intuitive and easy to navigate?
- iii. How to implement robust security measures like encryption and secure access controls to protect sensitive files?
- iv. How to develop an automation report for the ULK lime management for decision marking?

1.6. Scope of the study

1.6.1. Content Scope:

In this research, the researcher focused only on the design and implementation of a cloud storage management system. This is the application of the web app that will be used by the school and individuals to enhance the storage and security of data.

1.6.2. Geographical Scope

The physical area where this study is being carried out is in Rwanda, Kigali Province, Gasabo District, and Gisozi Sector, Kigali Independent University ULK. due to the high volume of data processed by the institution daily.

The implementation of this project will be accessible to this institution to enhance the quality, affordability, and security of cloud storage.

1.6.3. Time Scope

The study examines and uses data generated over three years: 2019 - 2021. The year 2019 corresponds to the pre-pandemic and the year 2020 when the pandemic caused a few social

restrictions, which included distancing from people. Within this time, the use of cloud storage grew immensely as most people were confined to their homes, thereby skyrocketing the employment of online services.

1.7. Significance of the study

1.7.1. Personnel Interest

The fulfillment of the requirement for the award of a Bachelor of Science and Technology degree in the Department of Computer Science. To practice the acquired knowledge gained during the academic journey in the fields of computer programming, database management, and software engineering. To get the experience of how to solve real-world problems as is the case for this current study.

1.7.2. Institutional Interest

This project is vital for Kigali Independent University as it enhances the Computer Science curriculum by incorporating practical experience in cloud storage systems. It will also strengthen the university's research capabilities in data management, providing a platform for innovation and collaboration with industry partners.

1.7.3. Public Interest

For the public, this project addresses the growing need for secure and efficient data management in cloud storage, benefiting sectors like healthcare and finance. The outcomes could guide public policy and raise awareness about the importance of data security in Rwanda and beyond.

1.8. Project methodology

In conducting this research, the information-gathering process was very vital for decision-making. The information gathered differs from one person to another and this helped in understanding the problems faced by different groups and individuals while making use of the current system. In this project, will use data collection techniques observation, and Interviews.

Additionally, will system development life cycle such as the Agile Model, and some advanced tools for designing systems.

1.9. Limitations

There are a few limitations that came up in the implementation of this system. The major problem and limitation of this project is the fact that users need an internet connection in order to use the system to upload and access uploaded data. Without an internet connection, they can't use this system. Furthermore, the provided storage is limited, thereby allowing the upload of specific kinds of data.

1.10. Organization of the project

Chapter 1: General Introduction

This chapter contains the introduction and background of the research. It introduces the purpose of the study, including the statement of the problem, objectives of the study, research questions, scope, and limitations of the study.

Chapter 2: Literature Review

The main purpose of this second chapter is to define the key terms used in this study, review existing related systems, and what previous research relating to this topic has already established.

Chapter 3: System analysis and Design

This chapter will present the system analysis and design which will describe the methods and tools used to analyse and design our new system.

Chapter 4: System Implementation and Testing

This chapter will show the implementation of the system and also have attached images that will show the functionality of the new system.

Conclusion and Recommendation

This last chapter of the research will end with the general conclusion and recommendation.

CHAPTER 2: LITERATURE REVIEW

2.1. Introduction

This chapter reviews existing literature on data management systems, focusing on cloud storage solutions and their applications in educational institutions. The review highlights key concepts, related works, and the advantages of cloud-based systems.

2.2. Definition of Concepts

This project is titled ‘Efficient Data Management in Cloud Storage Systems,’ and the different concepts deduced from the project are the following:

2.2.1. Data Management

Data management refers to the process of collecting, storing, organizing, and maintaining the data created and collected by an organization. It ensures data accuracy, accessibility, and reliability for decision-making and operational purposes. Data management involves practices and procedures that help to manage the data lifecycle needs of an enterprise in an effective manner Kate Harrison (2018).

2.2.2. Cloud Storage

Cloud storage is a service model in which data is maintained, managed, and backed up remotely and made available to users over a network, typically the internet. Cloud storage allows organizations to store data on remote servers, accessed via the internet, maintained by a cloud storage service provider. This method offers scalable, flexible, and cost-effective storage solutions Ramaswamy Chandramouli (2020).

2.2.3. Cloud Storage Systems

A cloud storage system refers to the infrastructure, software, and services used to store, manage, and access data in the cloud. Cloud storage systems provide a way to store and access data over

the internet, leveraging distributed servers to offer redundancy, reliability, and remote access capabilities. These systems are essential for modern data management, offering scalable storage solutions that can grow with an organization's needs Avinash Bandaru (2020).

2.2.4. Efficient Data Management

Efficient data management involves using best practices and technologies to ensure data is handled in the most effective and efficient way possible, maximizing its value while minimizing costs and risks. Efficient data management is the process of organizing and maintaining data processes, policies, and technologies to ensure the highest levels of accuracy, accessibility, and reliability while reducing time, effort, and costs associated with data handling Kate Harrison (2019)

2.2.5. Management System

Management Systems are systematic frameworks designed to manage an organization's policies, procedures, and processes and promote continual improvement within. A management system is a set of policies, processes, and procedures used by an organization to ensure that it can fulfill the tasks required to achieve its objectives. These objectives cover many aspects of the organization's operations, including financial success, safe operation, product quality, client relationships, legislative and regulatory conformance, and worker management Daniela Patricia (2019).

2.3. Review of Related Literature

The literature review will cover studies on cloud storage implementations in educational institutions, highlighting their benefits and challenges.

2.3.1. Building a Cloud Storage Service System

2.3.1.1. Introduction

Cloud computing regards infrastructure, platform, and software as services, which are made available as order-based services in a pay-as-you-go model to users. In industry, these services are respectively referred to as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). The importance of these services is highlighted in a recent report from Berkeley as: "Cloud computing, the long-held dream of computing as a utility, has the potential to transform a large part of the IT industry, making software even more attractive as a service". According to the current researching achievements about cloud computing, several research fields such as web computing, grid computing, distributing computing and virtual computing highly contributed to today's cloud computing Chengzhang Peng, Zejun Jiang (2019).

It is sure that cloud computing must go with cloud storage, that is why cloud storage are increasingly noticed. Although sometimes cloud storage's success is considered not technology-driven but economical, various research and commercial cloud storage systems presented in Tim's thesis told us there are many interesting areas. Cloud storage service is an emerging infrastructure that offers Platforms as a Service (PaaS). In industry, Amazon Simple Storage Service (Amazon S3) is regarded as the best reference of cloud storage service. Amazon S3 provides virtually elastic and unlimited storage space, we can access the internet-based storage service through using a set of simple interfaces. Users pay for the service of the amount of storage, Get Request, Put Request, data transfer in and data transfer out each month for one year. If reading Amazon S3 API, we will summarize several features of Amazon S3 as follows: First, users create buckets that can contain the arbitrary objects up to 5 terabytes in size, each accompanied by up to 2 kilobytes of meta-data, generally, and we refer to buckets as containers and objects as basic unit of user data. Second, users upload or download own entire objects, each up to 5G in one operation, and objects are identified within each bucket by a unique, user-assigned key through REST and SOAP interfaces Chengzhang Peng, Zejun Jiang (2019)

To access objects, HTTP is the primary protocol. Third, accessing Amazon S3 buckets or objects' requests use a customized HTTP scheme based on a Hash Message Authentication

CODE (HMAC). We will get both Access Key ID and Secret Access Key when registering an account, and therefore, every request at least includes your Access Key ID, selected elements and its signature which is the output of using your Secret Access Key to calculate the HMAC of selected elements. Upon receiving an authenticated request, Amazon S3 service gets a Secret Access Key that you claim to have, and then compute a “signature” for the message it received in the same way. It then compares the signature it calculated with the signature presented by the requester. If both signatures match, the system thinks that the requester must have access to the Secret Access Key, and so that has the authority of the principal to whom the key was issued. The request is given up and the system responds with an error message if both do not match. Fourth, Amazon S3 holds eventual consistency, therefore, users may not get their own newest content of an object identifier very soon after it has been overwritten. In open-source, Eucalyptus is an infrastructure for cloud computing. It is built in the Amazon EC2, in which users configure virtual machines in the cloud and run tasks on them. Eucalyptus mainly includes four components. The bottommost level component is node controller, One Eucalyptus may have one or more nodes, a node controller executes on every node. The node controller queries and controls the host operating system and the hypervisor on its node. The middle-level component is cluster controller, One Eucalyptus system may have several clusters, every cluster needs a cluster controller to schedule resource in its own cluster and controller network connecting to both the nodes running node controller and the machine running. The two top-level components are node controller and storage controller (Walrus); the former offers a web interface for cloud management and EC2-compatible SOAP, and performs high-level resource scheduling and system accounting; the latter offers S3-compatible cloud storage service. In cloud storage system Dynamo, a highly available key/value store in cluster environment developed by Amazon, chooses to provide “always writable” data availability with the trade-off eventual consistency. Cassandra is a distributed storage system for managing very large amounts of structured data spread out across many commodity servers. Cassandra provides highly available service with no single point of failure with the use of peer-to-peer model Chengzhang Peng, Zejun Jiang (2019).

2.3.1.2. System Design

In this section, we propose a general architecture of a cloud storage service system-CS3. The architecture of the CS3 is simple, flexible and modular with a hierarchical design reflecting

common resource environments found in many academic settings. By and large, the system allows users to access their own data using SOAP and REST interfaces. Of course, for the system's better compatibility, we will implement an emulation of Amazon S3's SOAP and REST interfaces in the near future. In addition to the programmatic interfaces, CS3 also offers a Web interface for users. Using a Web browser, users can upload, download, and look through their own data in CS3, which is similar to on-line disk you used. To offer the service of the above described CS3, Figure 1 depicts the proposed architecture of CS3. This system consists of three main stratum: A Data Storage layer, a Transformation Processing layer, and a Web system layer. The key challenge is how to design techniques for each layer component and make these components work together in a coherent system Chengzhang Peng, Zejun Jiang (2019).

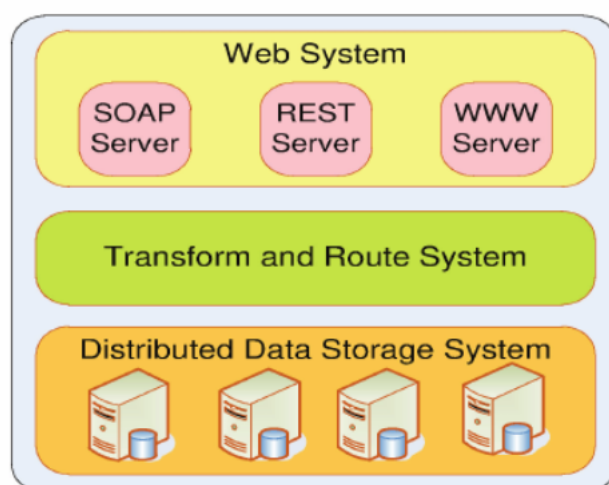


Fig 1: Architecture of a cloud storage system ESIAT (2019)

In storage layer, the distributed database (e.g., Cassandra) distributed file system (e.g., HDFS) or parallel file system (e.g., PVFS) were successfully used in some famous projects. Whatever scheme is chosen, the final goal is that the storage system must be highly scalable and highly available. In our project, we use distributed database as the storage component. Transformation Processing Layer mainly changes different type of Web requests (e.g., SOAP or REST) into uniform bottom-level data storage request. As more and more users want to share the cloud storage service, several different interfaces must be provided, so that three primary types of Web service interfaces in Web System Layer will be discussed Procedia Chengzhang Peng, Zejun Jiang (2019).

Data Storing Layer:

According to the rules of pay-by-use model, and cloud computing services being organized as a utility, our data storing layer must be able to meet following requirements, the most important feature is dynamic scalability being achieved whenever storage nodes could be easily added into or removed from the system; the second feature is consistency, and so forth. From Point of providing one kind of cloud storage service (e.g., Amazon S3 mainly being used to store videos, photos, audio), we can choose one in Cassandra, CouchDB, HBase, Redis, Scalaris, Project-Voldemort, and so on.

Transformation Processing Layer:

In this layer, the received SOAP, REST, or Web-based request located in the front server such as Tomcat will be transformed into the above-described database request. One major challenge for implementing this layer is that user should can choose his/her favorite one of popular programming languages, and one of three kinds of internet interfaces to connect to the front server, but the database request only can be generally implemented in one programming language (e.g., Java or Erlang), and the getting/putting data request generally also should be relatively single.

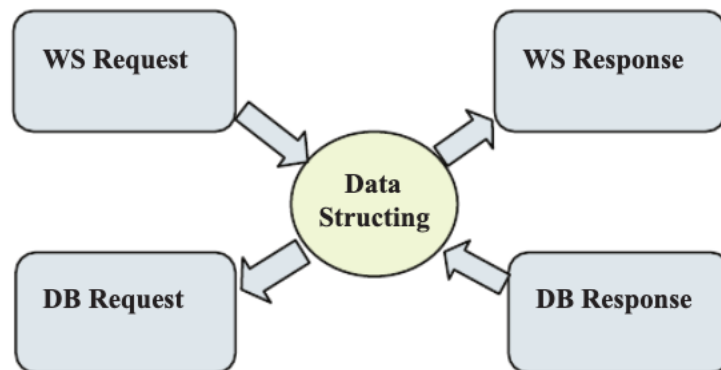


Fig 2: Transformation Processing Layer ESIAT (2019)

DB Request/Response In this component, we will encapsulate the data from Data structuring component into the distributed database client request, or send the response from the database to the Data Structuring component.

Web System Layer:

In general, this layer is located in the front server with Tomcat or other Web application server software. It includes SOAP, REST and Web-based server software, which are usually deployed in Tomcat or other Web application server software. When the Web application server receives a request, it will activate the corresponding server (e.g., SOAP server) to process this request. Implementing an available Web System being able to response SOAP, REST and Web-based request is not difficult thing, because there are many open-source components such as Axis, Tomcat and so on.

Prototype System:

We build our prototype system CS3 according to the above proposed architecture. CS3 contains 3 Cassandra nodes, which are located on 3 independent FC12 installed on VMware whose host machine is HP ProLiant ML150 G6 Server, a Web server with Apache Tomcat equipped with Intel(R) Core (TM)2 Duo CPU E7500 @ 2.93GHz processor, 3 GB of memory, 300GB SATA disks. We used Cassandra as the distributed database layer. Then, we develop a transformation processing module, it can transform a SOAP request into Cassandra client request, and transform a Cassandra response into SOAP response parameters. In Web system layer, we deploy SOAP services on the Tomcat server. According to data model features of the Cassandra, we design our own data model as shown in Figure 3. We use WSDL to describe SOAP service similar to Amazon's S3. In this prototype system we define three major APIs for this service system: get (filename) get one file from CS3 according to the provided file's name. put (file Attributes, file Contents) upload a file and its attributes. delete (filename) delete a file according to the provided file's name. Of course, we can use Amazon's WSDL file for compatible with S3 if necessary. In addition, we develop a terminal user application with SWT for validating our cloud storage service system *Procedia Environmental Sciences* 10 (2020) 691 – 696.

key-space				
ColumnFamily: FileStoring				
Key	Columns			
File	SuperColumns			
	key	Columns		
	profile	name	value	timestamp
		filename	profile	127069400469900
		Filesize	2K	12706940469900
		Filetype	Bash	12706940469900
		fileconts	byteArray	12706940469900
	dhblock.C	name	value	timestamp
		filename	dhblock.C	12706940469900
	
⋮	⋮			

Fig 3: Prototype System ZDNet (2023)

2.3.2. Data Management Solutions in the Cloud Accessing, sharing, and processing data that's new to Google Cloud Platform (GCP).

2.3.2.1. Introduction

The National Institutes of Health (NIH) established The Science and Technology Research Infrastructure for Discovery, Experimentation, and Sustainability (STRIDES) initiative to provide biomedical researchers with access to advanced, cost-effective, cloud-based computational infrastructure, tools, and services. Through STRIDES, researchers can take advantage of emerging data management methodologies, technological expertise, computational platforms, and tools to support cutting-edge experimentation and innovation. NIH has partnered with Google Cloud to support the STRIDES initiative through cloud services. In support of STRIDES, we've developed sets of playbooks to help enable researchers to build healthcare and life sciences solutions on Google Cloud Platform (GCP). The goal of this playbook is to aid researchers as they transition historically on-premise datasets, workloads and pipelines to GCP.

This playbook will provide researchers with methods for managing data lifecycles, accessing public datasets, leveraging cloud APIs and API gateways, managing data costs, and sharing and visualizing data. Additionally, this playbook will outline training and digital resources to help upskill and enable researchers to build on Google Cloud, while highlighting the appropriate products and services to use when architecting on GCP Google Cloud Platform (2023).

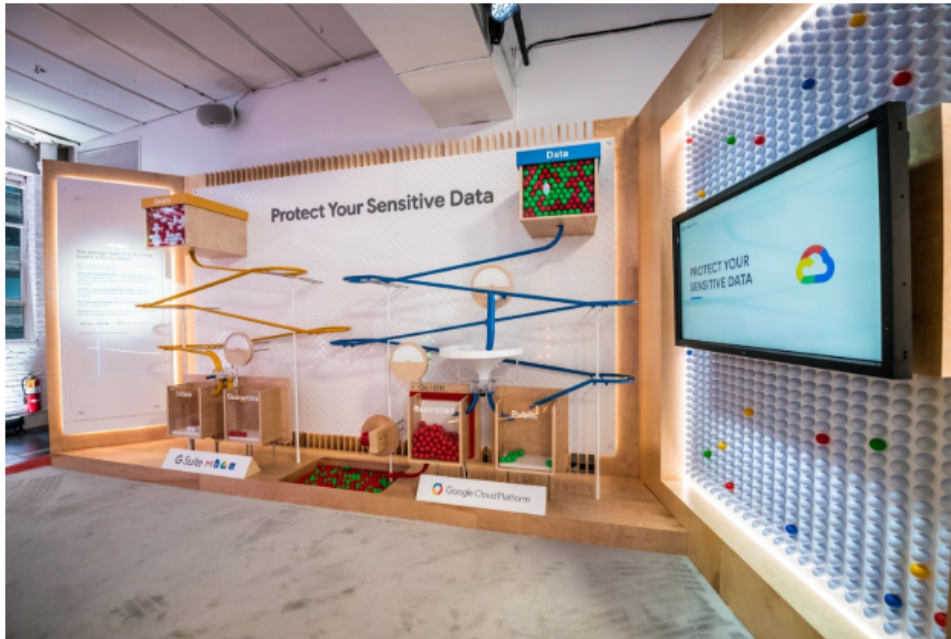


Fig 4: Google Space GCP(2023)

This image describes google space and components that make up GCP platform and how they interact.

Data Lifecycle Management

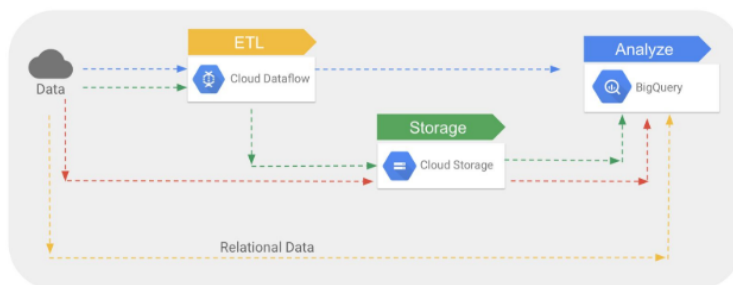


Fig 5: Data Lifecycle GCP(2023)

This image describes the lifecycle of data in google cloud platform.

With Google Cloud services, you can manage data throughout its lifecycle, from ingestion and storage to processing and visualization.

In Google Cloud Platform (GCP), data can be ingested through services like Google App Engine, Compute Engine, Kubernetes Engine, Cloud Pub/Sub, and Transfer Services. Once ingested, it can be stored using solutions such as Google Cloud Storage, Cloud SQL, Cloud Datastore, Bigtable, Firestore, and Spanner. For processing and analysis, tools like Cloud Dataflow, Dataproc, Machine Learning APIs (such as Vision, Speech, and NLP), and Cloud Dataprep are utilized. Archived data can be stored using low-cost, durable options like Nearline, Coldline, and Archive. For visualization, GCP offers tools like Cloud Datalab, Data Studio, and Google Sheets.

Ingest	Store	Process & Analyze	Explore & Visualize
App Engine	Cloud Storage	Cloud Dataflow	Cloud Datalab
Compute Engine	Cloud SQL	Cloud Dataproc	Google Data Studio
Kubernetes Engine	Cloud Datastore	BigQuery	Google Sheets
Cloud Pub/Sub	Cloud Bigtable	Cloud ML	
Stackdriver Logging	BigQuery	Cloud Vision API	
Cloud Transfer Service	Cloud Storage for Firebase	Cloud Speech API	
Transfer Appliance	Cloud Firestore	Translate API	
	Cloud Spanner	Cloud Natural Language API	
		Cloud Dataprep	
		Cloud Video Intelligence API	

Fig 6: Google Cloud Services

The above images display all google cloud services and all their components

2.3.3. Google Drive: An Efficient and Collaborative Cloud-Based Data Management Solution.

Google Drive stands out as a highly efficient data management system due to its seamless integration of storage, accessibility, and collaborative functionalities. With a user-friendly interface and robust cloud infrastructure, Google Drive allows users to store and organize documents, spreadsheets, and other files effortlessly. Its real-time collaboration features enable multiple users to work on the same document simultaneously, enhancing productivity and teamwork. Moreover, Google Drive offers powerful search capabilities and integration with other Google services, ensuring that users can quickly retrieve and manage their data from any device with internet access. The system's advanced security measures, including encryption and access controls, provide a secure environment for storing sensitive information, making Google Drive a reliable choice for personal and professional data management needs Google(2024)

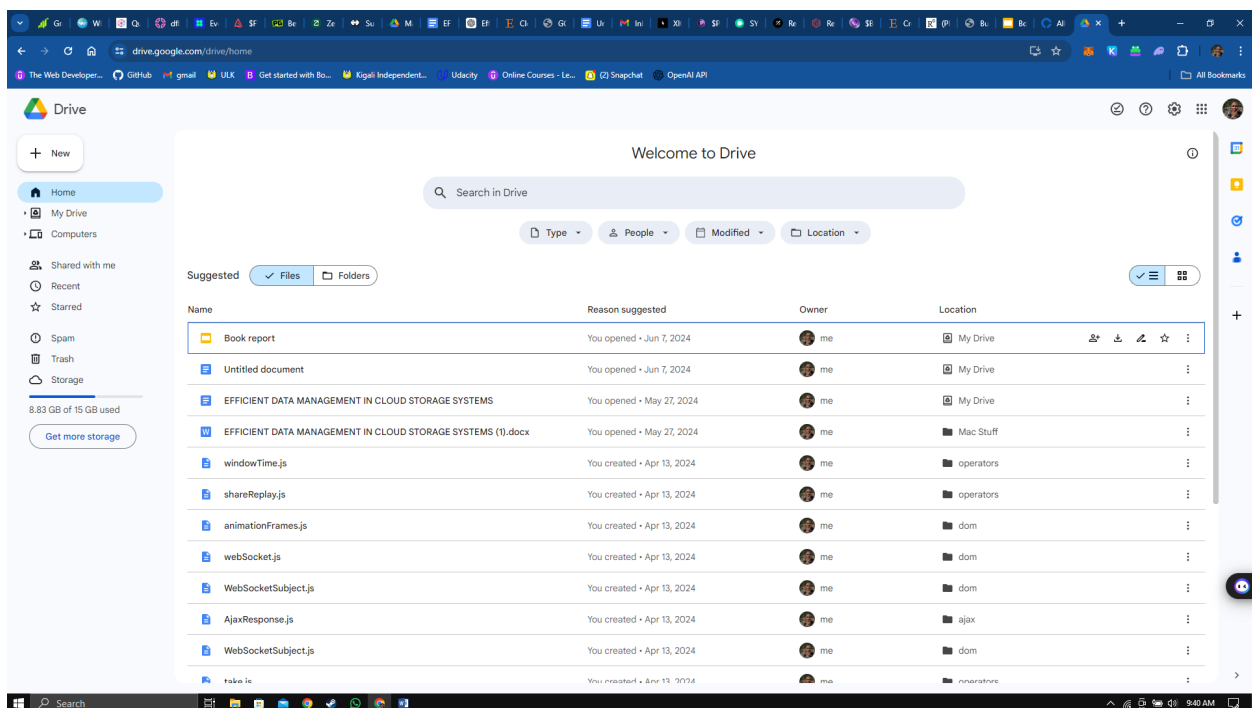


Fig 7: Google Drive Landing Page Google (2024)

This is a screenshot of the home page or landing page of Google Drive.

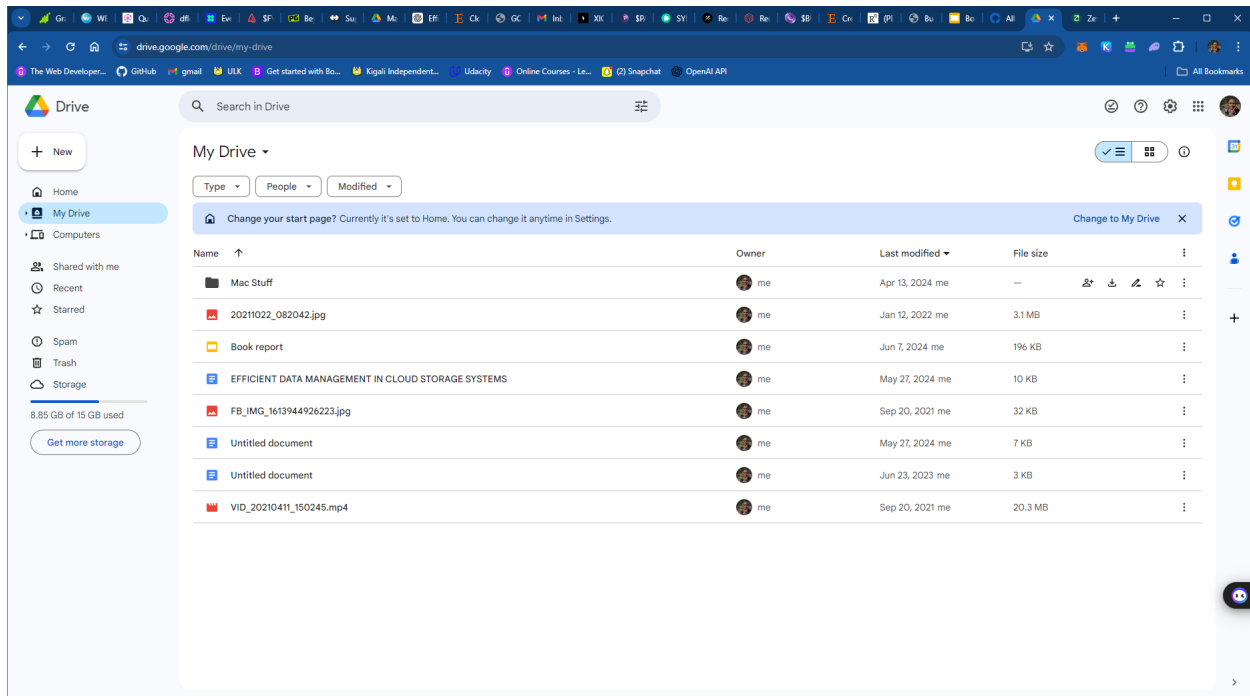


Fig 8: Google Drive Files

This is a screenshot of already uploaded and saved files in Google Drive.

2.4. Summary

Efficient data management in cloud storage systems is crucial for organizations, especially educational institutions, seeking to enhance their data handling capabilities. Cloud storage offers scalable and cost-effective solutions, allowing for remote access and backup of data. The literature review highlights various cloud storage implementations, such as Amazon S3 and Eucalyptus, emphasizing their features like elastic storage, accessibility, and eventual consistency. It also discusses the architecture of a cloud storage service system (CS3), detailing its components and the challenges involved in its design. Additionally, the review touches upon data lifecycle management on Google Cloud Platform (GCP) and the efficiency of Google Drive as a collaborative data management solution. Overall, the review underscores the importance of adopting efficient data management practices in cloud storage systems to optimize data handling processes and improve organizational efficiency.

CHAPTER 3: SYSTEM ANALYSIS AND DESIGN

3.1. Introduction

Systems Analysis and Design (SAD) is a broad term for describing methodologies for developing high-quality Information System which combines Information Technology, people, and Data to support business requirements. System Analysis is there to check the system that has been developed, how it's going to work and try to make some changes if needed.

In this chapter, we are going to present the design and analysis process of our project both in its logical and structural form.

3.2. Analysis of the current system

3.2.1. Introduction

This section presents a clear and concise analysis of the current system and mentions the problems which are being addressed by the study and development of the current system. It contains the full description of the current or old system and states the problems being faced.

3.2.2. Problem of the current system

The current cloud storage system struggles with scalability, leading to slow performance and limited storage capacity. Data retrieval delays are common, especially during peak times, affecting productivity. Security vulnerabilities expose sensitive information to unauthorized access. Manual backup and recovery processes increase the risk of data loss and downtime. Inadequate user management complicates access control and activity tracking. Poor integration with data analytics tools hinders real-time analysis. High operational costs result from inefficient resource use and frequent hardware upgrades. Limited accessibility and inconsistent data formats further complicate data management, highlighting the need for a more robust and automated system.

3.3. Analysis of the new system

3.3.1. Introduction

The new cloud storage system effectively addresses the previous system's challenges, ensuring efficient data management. It features enhanced scalability to handle increasing data volumes without performance issues, utilizing advanced load balancing and distributed storage techniques. Optimized data retrieval ensures quick access even during peak times, boosting productivity. Automated backup and recovery processes minimize data loss risk and downtime. Improved user management provides granular access control and detailed activity tracking. Seamless integration with data analytics tools enables real-time analysis and valuable insights. The system is free and more efficient, utilizing cost-effective resource management to reduce operational costs and eliminate the need for frequent hardware upgrades. Enhanced accessibility allows data access from various locations and devices, while support for diverse data formats ensures efficient processing of structured and unstructured data. Automation of routine tasks decreases errors and frees up time for other important activities.

3.3.2. System requirements

3.3.2.1. Functional requirements

Data Storage and Retrieval: The system should provide reliable and efficient mechanisms for storing and retrieving data.

Scalability: The system must handle increasing amounts of data seamlessly.

Data Security: Implement strong security measures to protect data from unauthorized access.

User Management: Facilitate user authentication and authorization.

Data Backup and Recovery: Ensure data backup and provide recovery solutions in case of data loss.

Data Analytics: Provide tools for analyzing stored data.

3.3.2.2. Non-Functional Requirements

Performance: The system should offer high performance in data processing and retrieval.

Reliability: Ensure the system is reliable and has high availability.

Usability: The user interface should be intuitive and easy to use.

Maintainability: The system should be maintainable and allow for easy updates and scaling.

Table 1: Functional and Non-Functional Requirements

Req. No.	Description	Type
R-1	A user should be able to create an account in the system	Functional
R-2	A registered user should be able to log in to their account on the system	Functional
R-3	A user should be able to upload and download files	Functional
R-4	A user should be able to log out of the system	Functional
R-5	A user should be able to access their data from any device and location	Functional
R-6	A user should be able to manage their files (organize, rename, delete)	Functional
R-7	The admin should be able to log in to the system	Functional
R-8	The admin should be able to manage user accounts and permissions	Functional
R-9	The system should support automated backup and recovery processes	Functional
R-10	The system should integrate with existing data analytics tools	Functional
R-11	The admin should be able to view and generate usage and performance reports	Functional
R-12	The system should be able to handle both structured and unstructured data efficiently	Functional
R-13	All user activities and changes should be logged and stored in the database	Functional
R-14	User and admin accounts require passwords to access them	Security
R-15	The system should be trusted and reliable	Reliability

R-16	The system should not be difficult to use	Usability
R-17	The system should run on any hardware and on any kind of browser	Compatibility
R-18	The system should be quick and responsive	Performance
R-19	The system should be accessed from any location	Accessibility
R-20	The system is scalable and fully responsive to any platform it runs on	Flexibility

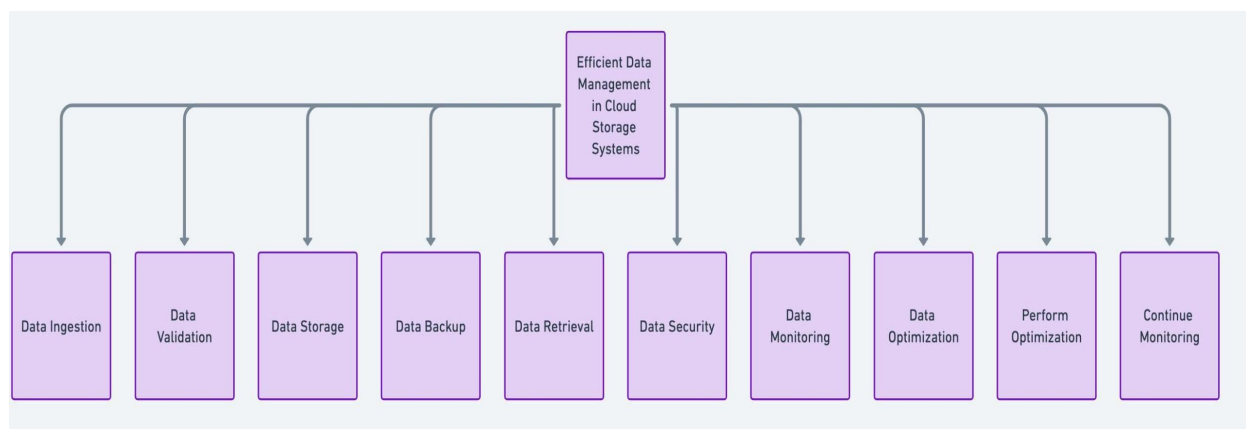


Fig 9: Functional Diagram

This diagram describes the functional elements of the system and their relationship to the system.

Table 2: Data Dictionary

Name	Datatype	Example
UserID	Integer	1001
UserName	String	JaneDoe
Email	Varchar	jane.doe@example.com
IngestionTimestamp	DateTime	2024-08-03 14:35:00
DataSource	String	SensorDevice01
DataType	String	JSON
ValidationStatus	Boolean	True
StorageLocation	String	s3://bucketname/path
StorageSize	Float	15.75
BackupStatus	Boolean	True

BackupTimestamp	dateTime	2024-08-03 15:00:00
RetrievalMethod	String	API
SecurityLevel	String	High
MonitoringStatus	Boolean	True
OptimizationType	String	Compression
OptimizationStatus	Boolean	True
OptimizationTimestamp	DateTime	2024-08-03 16:00:00
DataUsageFrequency	Interger	5
AccessPermissions	String	Read, Write
DataRetentionPeriod	Integer	365

Table 3:Users Data Dictionary

Name	Datatype	Example	Description
UserID	Integer	1001	Unique identifier for the user
UserName	String	JaneDoe	Username chosen by the user
Email	Varchar	jane.doe@example.com	User's email address
IngestionTimestamp	DateTime	2024-08-03 14:35:00	Timestamp when data is uploaded
DataSource	String	SensorDevice01	Source from which data is collected
DataType	String	JSON	Type of data (e.g., JSON, XML, etc.)
ValidationStatus	Boolean	True	Status if the data has been validated
StorageLocation	String	s3://bucketname/path	Cloud storage location for user data
StorageSize	Float	15.75	Size of the data stored in the cloud
BackupStatus	Boolean	True	Status if the data has been backed up
BackupTimestamp	dateTime	2024-08-03 15:00:00	Timestamp when the backup was done
RetrievalMethod	String	API	Method used to retrieve user data
DataUsageFrequency	Interger	5	Frequency of data usage or access
AccessPermissions	String	Read, Write	User's access permissions
DataRetentionPeriod	Integer	365	Period in days for retaining the data

Table 4:Admin Data Dictionary

Name	Datatype	Example	Description
AdminID	Integer	5001	Unique identifier for the admin
AdminName	String	Admin001	Username for the admin
Email	Varchar	admin@example.com	Admin's email address
SecurityLevel	String	High	Security clearance level for the admin
MonitoringStatus	Boolean	True	Status indicating whether monitoring is active
OptimizationType	String	Compression	Type of optimization applied to the data
OptimizationStatus	Boolean	True	Status of data optimization
OptimizationTimestamp	DateTime	2024-08-03 16:00:00	Timestamp when the optimization occurred
DataRetentionPeriod	Integer	365	Number of days the data is retained

3.3.4. Methodological Approach

3.3.4.1. Data Collection Techniques

Observation:

This technique was carried out in three ways, personal observation, and direct and indirect observation.

Personal Observation - accessing and analyzing the entire data management systems was carried out by personally observing the current system. Trying out all the functionalities of the system and determining what segments needed improvements.

Direct Observation - This second method of observation was carried out by studying the behaviors of people that make use of the current system. In this method, the users were aware of the observation process so their behaviors are affected by their knowledge of the ongoing data collection process.

Indirect Observation - In this method, the presence of the observatory was unknown and this proved to be more effective as it helped to collect the actual behaviors of people while using the

current system. In this method, since the users are not aware of the ongoing data collection, their behavior is more neutral and the data collected is more accurate.

Interview:

This is one of the most common techniques used in research and it consists of an interviewer, the one who conducts the process and asks the questions, and an interviewee, who responds to the questions.

Information was gathered about the current system by asking questions to the system users.

The answers provided by the interviewees helped in gathering a lot more information to proceed with the study.

3.3.4.2. Software Development Methodology

The software development model used by the researcher is the 'Agile Model'.

The meaning of Agile is swift or versatile. "Agile process model" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations or parts that do not directly involve long-term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration, and the scope of each iteration are clearly defined in advance.

Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client. (<https://www.javatpoint.com/software-engineering-agile-model>).

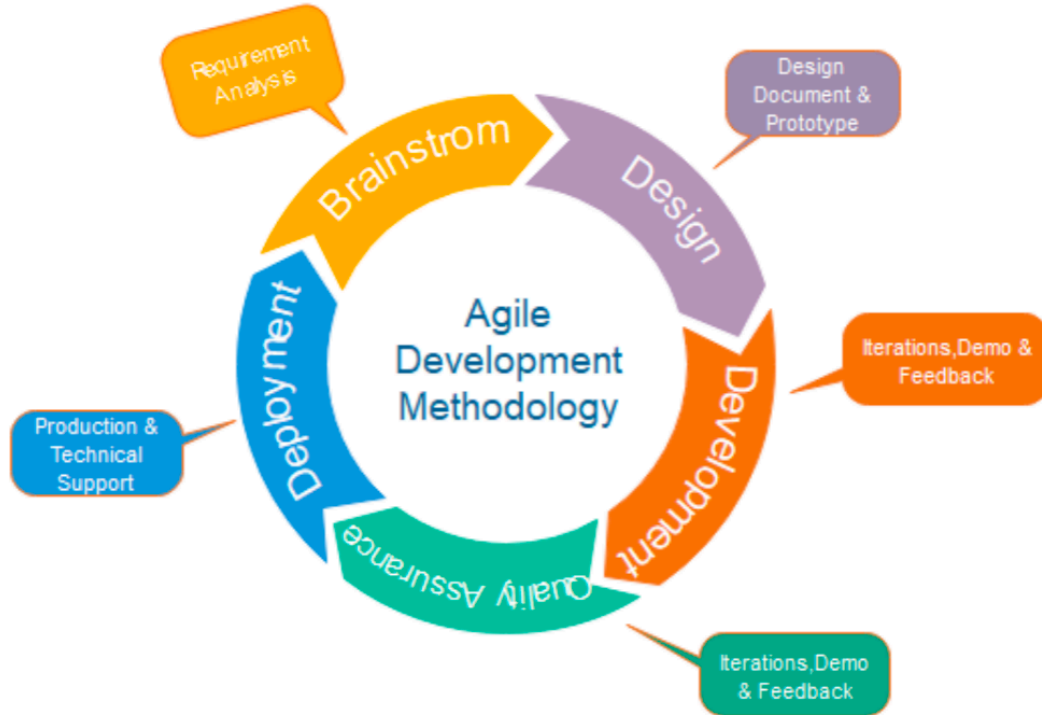


Fig 10: Agile Development Model Agile Model (2024)

The Agile model is an iterative approach to software development focused on flexibility, collaboration, and continuous customer feedback. Projects are divided into short sprints, with each delivering a working product increment. Agile allows for ongoing adjustments based on changing requirements and priorities.

Phases

Requirements gathering: In this phase, the requirements are considered and the time is planned, and each activity is allocated a determined time for implementation.

Designing the requirements: Here, the user flow diagram is designed. The following are sample images from the requirements design phase.

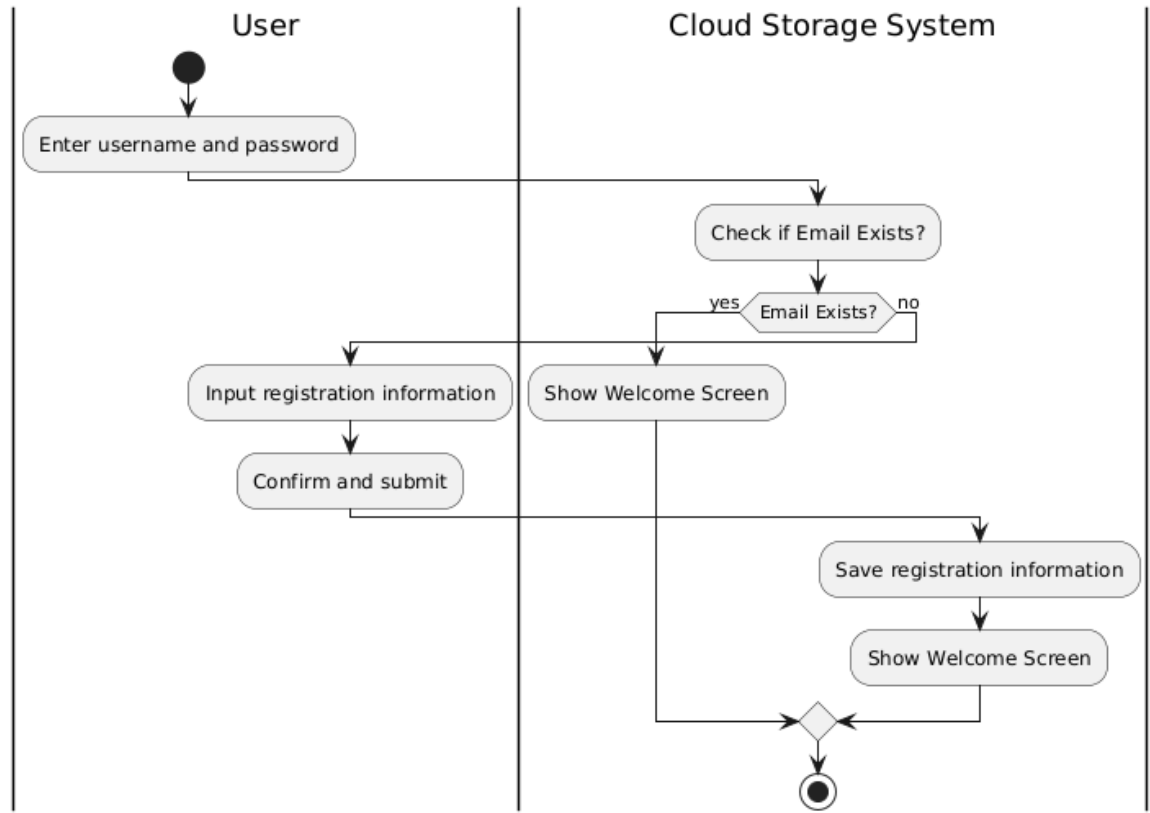


Fig 11: User registration activity

The above diagram shows the process which a customer registers his account. And as shown, the customer starts by inputting information which is collected by the system and then checks if the email already exists in the system before proceeding to confirmation and creating the user account.

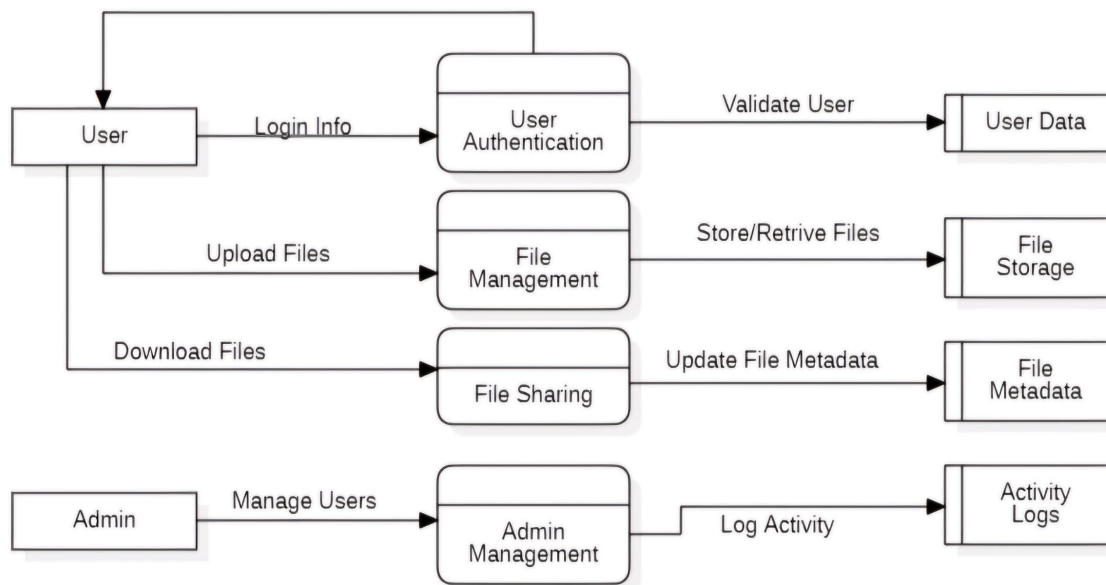


Fig 12: Level 0 Data Flow Diagram

The above diagram shows level 0 data flow diagram which shows how the systems works. The general activity of the system.

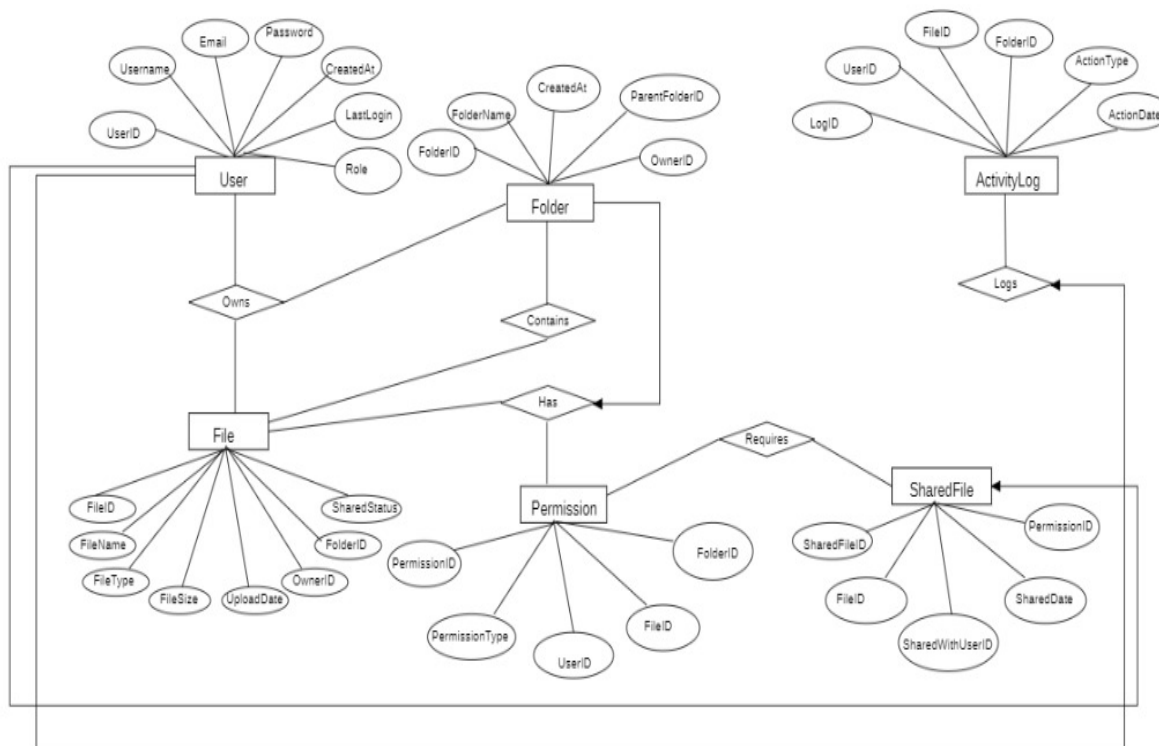


Fig 13: Entity Relationship Diagram (ERD)

The above diagram shows the entity relationship which shows the connection with different components and units of the system. The diagram describes how they interact with one another.

CHAPTER 4: SYSTEM IMPLEMENTATION AND TESTING

4.1. Implementation and Coding

4.1.1. Introduction

This chapter covers the implementation and testing of data management in cloud storage systems and discusses the technologies and tools that were employed while creating/implementing the system.

4.1.2. Description of Implementation Tools and Technology

The instruments used throughout the project's implementation are listed below:

- A MacBook with an Apple MacBook intel core i7, 16GB RAM, and 256GB SSD.
- Visual Studio Code 2024 (v1.92.2) as IDE
- HOSTINGER

Technologies used are:

- Bootstrap CSS

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains HTML, CSS, and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

- React JS

ReactJS is a JavaScript library for building dynamic and efficient user interfaces, focusing on reusable components and a virtual DOM for faster updates. It simplifies the development of complex, single-page applications by using a component-based architecture.

- JavaScript

JavaScript, often abbreviated JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS.

- PHP

PHP is a general-purpose scripting language geared toward web development.

4.1.3. Screenshots and source codes

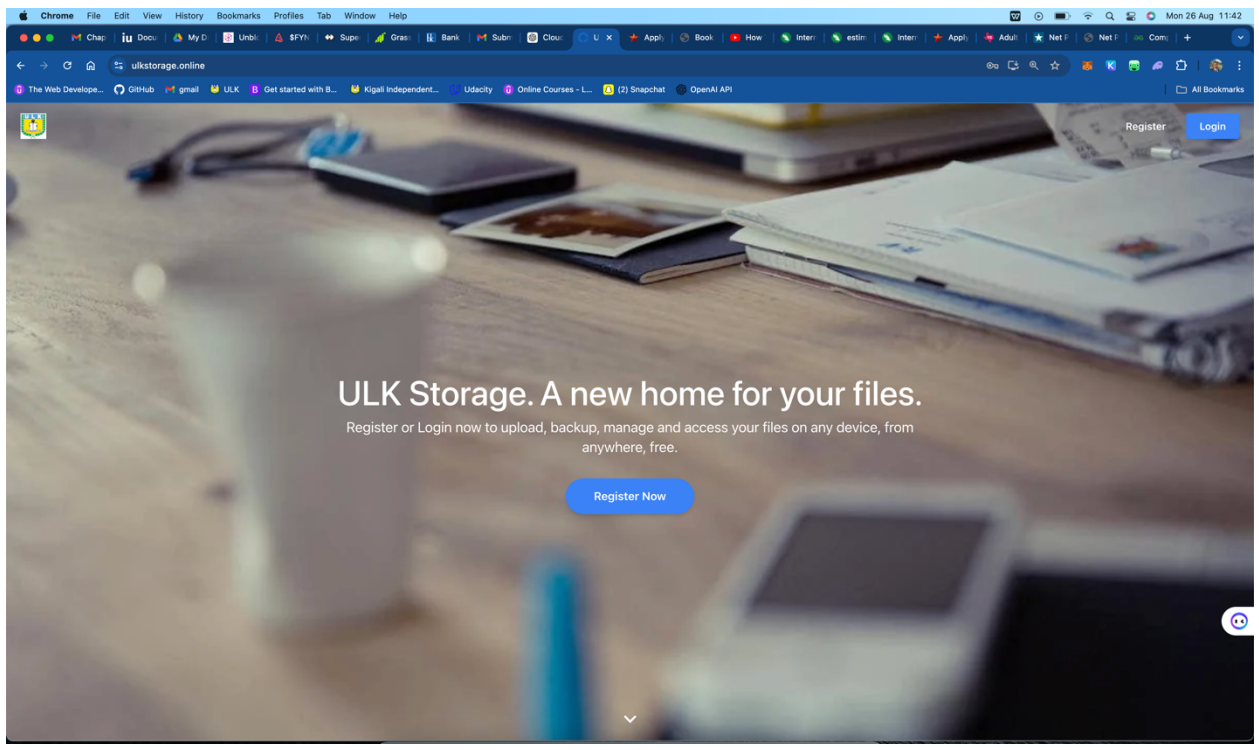


Fig 14: Web Application Homepage

This is the homepage/landing page of the current system for ULK Storage.

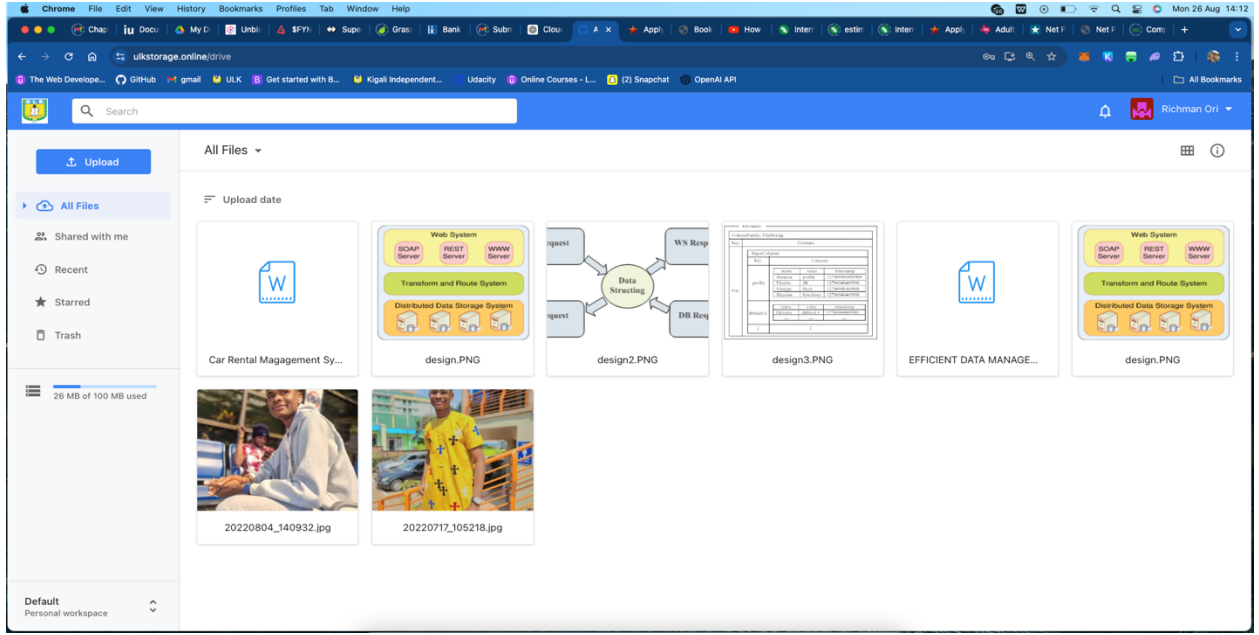


Fig 15: User Interface Overview

This is the page where users can view their uploaded files and interact with the system generally.

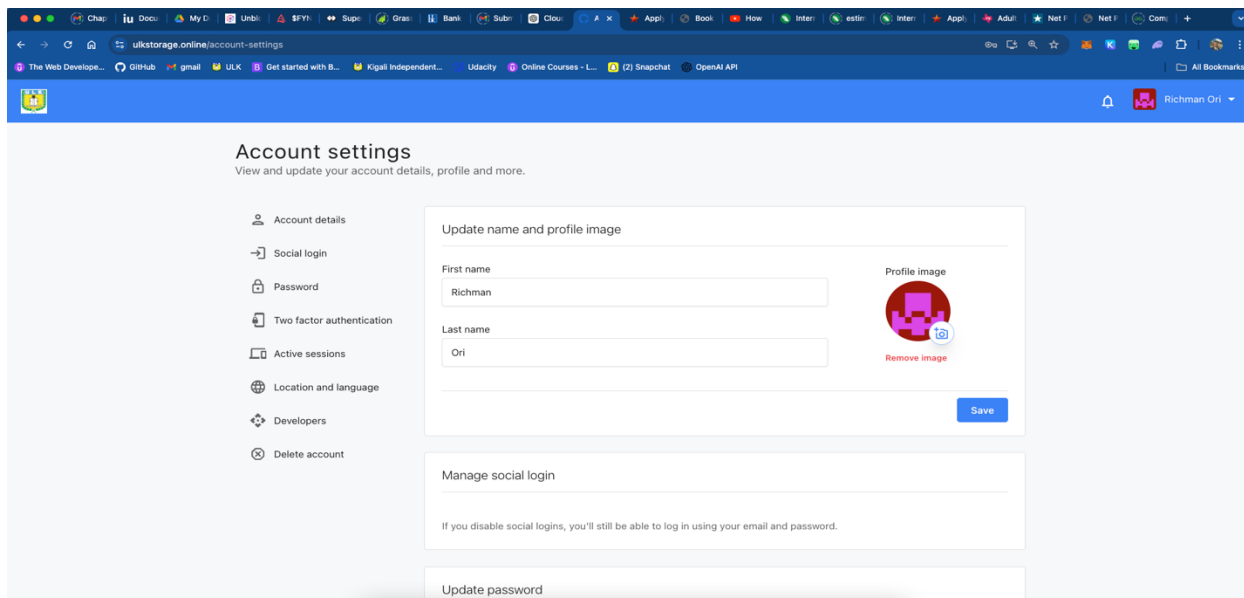


Fig 16: User account settings page

This is the page where users view and edit their account to their satisfaction.

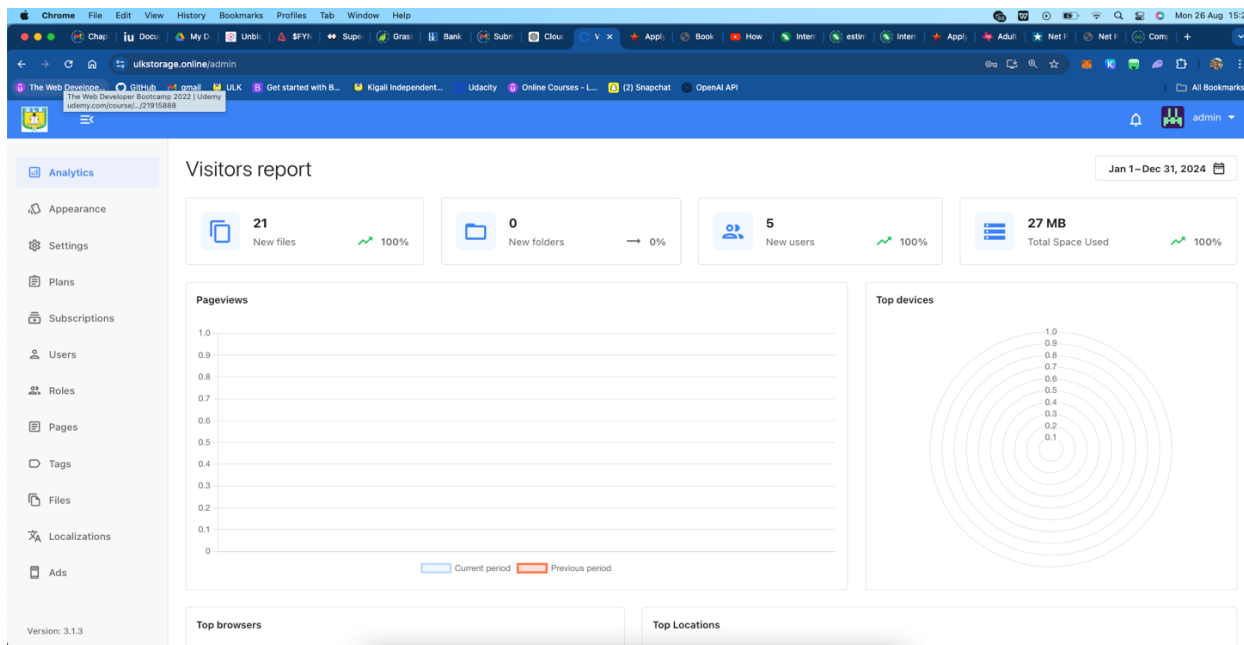


Fig 17: Admin Dashboard

This is the administrator dashboard where all information in the system is seen and controlled.

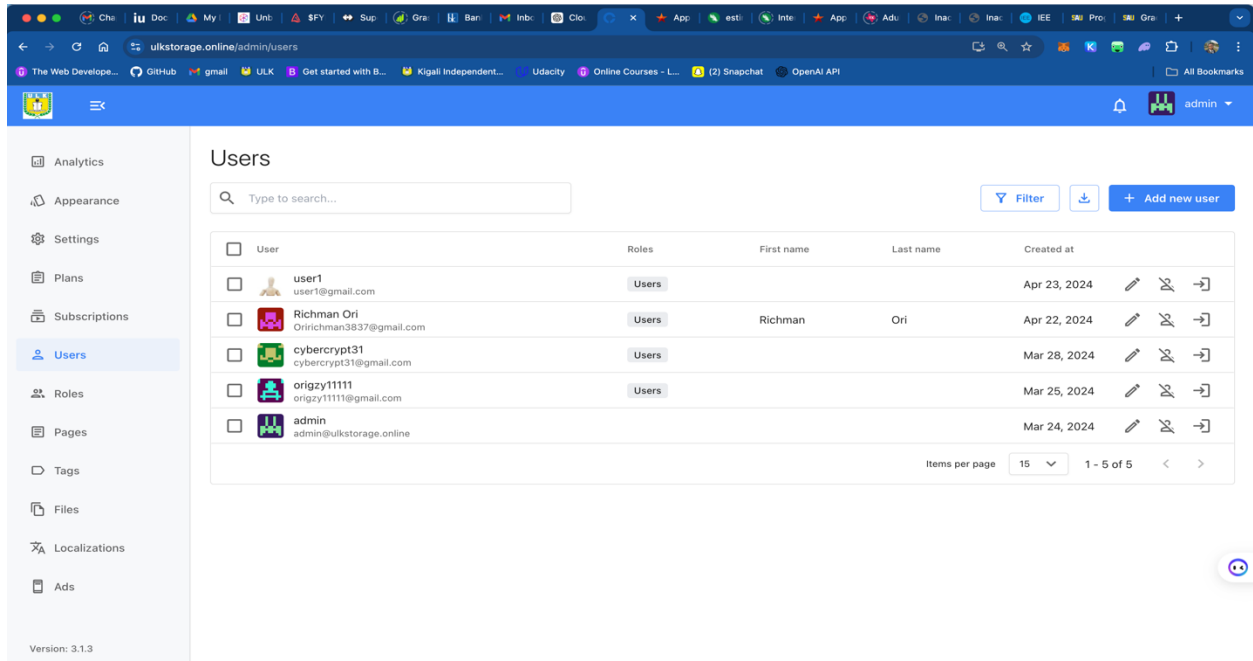


Fig 18: Registered users

This is the administrator dashboard where all the registered users are seen and manipulated.

4.2. Testing

4.2.1. Introduction

This section shows the tests carried out in order to ensure that the web application does not contain any errors. This section is divided into the following subsections: unit testing, validation testing, integration testing, functional and system testing, and acceptance testing.

4.2.2. Unit testing Outputs

Each individual unit of the system was tested separately and the result is recorded in the following table:

Table 5: Unit testing Output

Unit No.	Description	Priority	Test Result
T-1	User Authentication	HIGH	PASS
T-2	File Upload	HIGH	PASS
T-3	File Download	HIGH	PASS
T-4	File Sharing	HIGH	PASS
T-5	Folder Creation	MEDIUM	PASS
T-6	User Profile Management	MEDIUM	PASS
T-7	Access Permissions	HIGH	PASS
T-8	Activity Logging	MEDIUM	PASS
T-9	Data Backup and Recovery	HIGH	PASS

4.2.3. Validation testing Outputs

The following table contains the results of the validation testing:

Table 6: Validation Testing Output

Test No.	Description	Priority	Test Result
T-1	Email must contain a valid email address	HIGH	PASS
T-2	Email that has been previously registered will not be accepted	HIGH	PASS
T-3	Both Passwords must match during signup	HIGH	PASS
T-4	User must be registered before they can login	HIGH	PASS
T-5	Login details are case sensitive and must be corrected	HIGH	PASS
T-6	All documents must be uploaded successfully before viewing and sharing	HIGH	PASS

4.2.4. Integration testing Outputs

For integration testing, all individual units are combined and tested as a complete system. Here, the login/register unit, upload unit, download unit, sharing unit, account setting unit, report unit, manage users unit, and other units were combined and tested as a whole.

4.2.5. Functional testing Outputs

The complete system is tested to check if all functions are working as intended. The following table contains the test record and results.

Table 7: Admin functionalities testing output

Test Suite ID	Cloud Storage System 1.0
Test case summary	User should be able to interact with the system
Expected result	Users should be able to create accounts, log in and access the system functionalities
Test Status	PASS
Executed by	Richman Ori Uzochukwu-Theophilius
Date of execution	15/08/2024
Test Environment	Google Chrome

4.2.6. Acceptance testing Report

In this section, the system was presented to the stakeholders for acceptance testing to determine customer satisfaction.

At the end of the acceptance testing, the stakeholders were pleased with the outcome of the system and confirmed that all the functional requirements have been executed properly.

CONCLUSION AND RECOMMENDATIONS

Conclusion

In the course of this study, the researcher has successfully addressed the identified challenges in data management within cloud storage systems and achieved the set objectives. During the implementation of the system, a robust database was created to manage user data, files, and permissions effectively. This improvement has significantly enhanced data management, ensuring better organization and accessibility of files. Additionally, the system now offers a user-friendly interface that allows users to easily upload, download, and share files securely, while also providing comprehensive control over access permissions. The introduction of activity logging and reporting features has enabled the system administrators to monitor and generate reports accurately, improving transparency and efficiency. The new cloud storage system has played a pivotal role in enhancing data management processes, ultimately increasing the productivity and security of data management operations.

Recommendations

Expand Cloud Storage System Capabilities: It is recommended that the cloud storage system be expanded to support integration with other cloud services. This would enhance the system's scalability and flexibility, allowing users to manage and share their data across multiple platforms seamlessly.

Develop a Mobile Application: To improve accessibility and usability, a mobile application version of the cloud storage system should be developed. This would provide users with the convenience of managing their data on-the-go, directly from their smartphones or tablets.

Incorporate Automated Backup and Recovery: The system should include automated backup and recovery options to ensure data integrity and availability. This would minimize data loss in case of unexpected incidents and provide users with peace of mind knowing their files are securely backed up.

Integrate Payment and Subscription Management: For cloud storage services that offer premium features or additional storage space, a payment and subscription management system should be integrated. This would allow users to upgrade their storage plans and make payments online, enhancing the overall user experience.

REFERENCES

1. TechTarget(2023). Protecting your data at every step. <https://www.techtarget.com/about-us/>
2. Forbes. (2023). *The Future of Cloud Computing in 2023: Trends to Watch*. <https://www.forbes.com/>
3. TechCrunch. (2023). The rise of cloud storage: How AWS set the standard. *TechCrunch*. <https://www.techcrunch.com/the-rise-of-cloud-storage-aws>
4. Sun, Y., Zhang, J., Xiong, Y., & Zhu, G. (2015). Building a cloud storage service system. *Computer*,48(1),73-79. https://www.researchgate.net/publication/271617185_Building_a_Cloud_Storage_Service_System
5. Agile Model <https://www.javatpoint.com/software-engineering-agile-model>
6. DATAVERSITY. (2019). Understanding data management in cloud storage systems. *DATAVERSITY*. <https://www.dataversity.net/understanding-data-management-in-cloud-storage-systems>
7. African Business. (2023). The transformative impact of cloud storage in Africa. *African Business*. <https://www.africanbusiness.com/cloud-storage-impact-africa>
8. T. Connolly and C. Begg (2021). *Database System: A practical Approach to Design, Implementation, and Management, Third Edition*. England: Addison Wesley.
9. Physical data model definition. Accessed on 12/05/2010E.C <https://www.techopedia.com/definition/30500/physicaldata-model>
10. *What is activity diagram and its purpose*. Accessed on 30/04/2010E.C <http://www.modernanalyst.com/Careers/InterviewQuestions/tabid/117/ID/371/Wht-is-an-Activity-Diagram-and-what-is-its-purpose.aspx>
11. *What is cloud storage* https://www.geeksforgeeks.org/storage-systems-in-cloud/?ref=header_outind
12. Data Management Association. (2019). Understanding effective data management. *Data Management Association*. <https://www.dama.org/understanding-effective-data-management>
13. National Institute of Standards and Technology. (2022). Cloud storage service model: An overview. *National Institute of Standards and Technology*.

14. Amazon Web Services. (2020). Overview of cloud storage systems. *Amazon Web Services*.
15. Data Management Association. (2019). Efficient data management best practices. *Data Management Association*. <https://www.dama.org/efficient-data-management-best-practices>
16. FitSM. (2020). Understanding management systems. *FitSM*. <https://www.fitsm.eu/understanding-management-systems>
17. National Institutes of Health. (n.d.). *GCP Data Management Playbook*. Retrieved from <https://cloud.nih.gov/resources/guides/science-at-cloud-providers/science-on-gcp/GCPDataManagementPlaybook.pdf>
18. University of Alberta. (2020). Improving collaboration and integration with Google Drive. *Information Services and Technology*. Retrieved from <https://www.ualberta.ca/en/information-services-and-technology/news/2020/improving-collaboration-and-integration-google-drive.html>

Appendix

Interview Questions

- How do you currently store data online?
- How do you currently manage your stored data?
- What is the cost of your current cloud storage option?
- How often do you access your stored data?
- What security challenges have you experienced?
- What quantity of data are you able to store using the current system?
- How easy is it to upload and download your data from the current systems?

Interview Answers

Currently, data is stored using a mix of cloud platforms, including Google Drive, OneDrive, and AWS. Google Drive and OneDrive are popular for everyday use and collaboration, while AWS is preferred for more substantial data storage needs.

Data is managed either manually through organizing files into folders and using labels, or through automated backup scripts. However, many still face difficulties in keeping their storage systems well-organized, especially when juggling multiple platforms.

For most users, cloud storage is free unless they exceed the available storage limits. Paid users typically spend around \$2-3 per month, while AWS costs vary depending on the amount and class of data being stored.

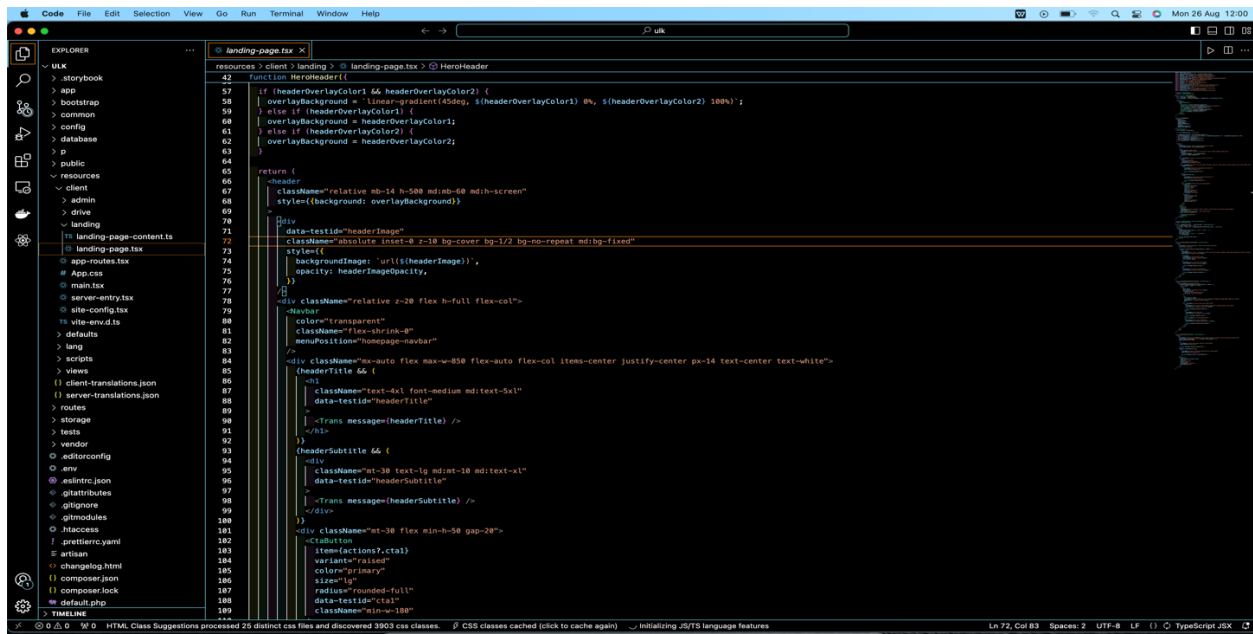
Users tend to access their stored data several times per week, especially for active projects. Archived data or rarely used files are accessed less frequently, maybe a few times a month.

Security challenges mainly involve unauthorized access to shared files and the complexity of managing permissions across platforms. Some users have expressed concerns over weak password protection and encryption for sensitive data.

In terms of storage capacity, free tiers allow for around 10-15 GB of data, with paid plans offering much more. Those needing higher capacities for larger datasets typically use AWS or upgrade to premium plans.

Uploading and downloading data is generally smooth, though some users experience slow speeds with larger files. For smaller files, the process is quick and efficient on most platforms.

Source code



```

42 function HeroHeader({
43   headerOverlayColor1, headerOverlayColor2,
44   overlayBackground = 'linear-gradient(45deg, $headerOverlayColor1 0%, $headerOverlayColor2 100%)',
45   headerImage,
46   headerImageOpacity,
47 }) {
48   return (
49     <div
50       className="relative mb-14 h-500 md:mb-60 md:h-screen"
51       style={{background: overlayBackground}}
52     >
53       <div
54         data-testid="headerImage"
55         className="absolute inset-0 z-10 bg-cover bg-1/2 bg-no-repeat md:bg-fixed"
56         style={{
57           backgroundImage: `url(${headerImage})`,
58           opacity: headerImageOpacity,
59         }}
60       />
61       <div
62         className="relative z-20 flex h-full flex-col"
63       >
64         <div
65           className="transparent"
66           className="flex-shrink-0"
67           nonVisible={true}
68         />
69         <div
70           className="flex-auto flex max-w-650 flex-auto flex-col items-center justify-center px-14 text-center text-white"
71         >
72           <h1
73             data-testid="headerTitle"
74             className="text-4xl font-medium md:text-5xl"
75           />
76           <p
77             data-testid="headerSubtitle"
78             className="text-3xl text-lg md:text-4xl"
79           />
80           <div
81             data-testid="cta"
82             className="mt-10 flex min-h-50 gap-20"
83           >
84             <button
85               type="button"
86               className="text-white"
87               href="#"
88             />
89             <button
90               type="button"
91               className="text-white"
92               href="#"
93             />
94           </div>
95         </div>
96       </div>
97     </div>
98   );
99 }
100
101 export default HeroHeader;

```

Fig 19: Part Source code for Homepage

The above image shows a part of the source code used in creating the homepage.

```

resources > client > @ main.tsx >
import { useState } from '@common/core/root-ec';
import { getBootstrapData } from '@common/core/bootstrap-data/use-backend-bootstrap-data';
import { ignoredSentryErrors } from '@common/errors/ignored-sentry-errors';
import { BrowserRouter } from 'react-router-dom';
import { AppRoutes } from '@app/app-routes';
import { Product } from '@common/billing/product';
import { fetchShareableLinkPageResponse } from '@app/drive/shareable-link/queries/use-shareable-link-page';
import { fetchCustomPageResponse } from '@common/custom-page/use-custom-page';

declare module '@common/core/settings/settings' {
  interface Settings {
    appearance: LandingPageContent;
    types: 'LoginPage' | 'registerPage' | string;
    value?: any;
  };
  drive: {
    details_default_visibility: boolean;
    default_view: 'list' | 'grid';
    send_share_notification: boolean;
  };
  share: {
    suggest_emails: boolean;
  };
  ads?: {
    drive?: string;
    'file-preview'?: string;
    'landing-top'?: string;
    disable?: boolean;
  };
};

declare module '@common/core/bootstrap-data/bootstrap-data' {
  interface BootstrapData {
    loaders: {
      landingPage: {
        products: Product[];
      };
      customPage?: FetchCustomPageResponse;
      shareableLinkPage?: FetchShareableLinkPageResponse;
    };
  };
};

const data = getBootstrapData();
const sentryDsn = data.settings.logging.sentry_public;

```

Fig 20: Part source code for user interface overview

The above image shows the source code used for creating the page that displays the user overview.

```

class Settings
{
    public function save(array $settings): void
    {
        $settings = $this->flatten($settings);
        foreach ($settings as $key => $value) {
            $setting = Settings::firstOrNew(['name' => $key]);
            $setting->value = $value;
            $setting->save();
            $this->set($key, $setting->value);
        }

        (cache::forget('settings.public'));
    }

    /**
     * Get all settings parsed from dot notation to assoc array. Also decodes JSON values.
     */
    // 2 references | 0 overrides
    public function get($notation): array
    {
        bool $includeSecret = false,
        array $settings = null,
    } array {
        if ($settings) {
            $settings = $this->all($includeSecret);
        }

        foreach ($settings as $key => $value) {
            if (in_array($key, self::$jsonKeys) && !is_string($value)) {
                $settings[$key] = json_decode($value, true);
            }
        }

        $dot = dot($settings, true);

        return $dot->all();
    }

    /**
     * Flatten specified assoc array into dot array. (['billing.enable' => true])
     */
    // 1 reference | 0 overrides
    protected function flatten(array $settings): array
    {
        foreach ($settings as $key => $value) {
            if (in_array($key, self::$jsonKeys) && !is_string($value)) {
                $settings[$key] = json_encode($value);
            }
        }
    }
}

```

Fig 21: Part source code for user account settings

The above image shows the source code used for creating the page that displays the user account settings.


```

class AnalyticsController extends BaseController
{
    public function __construct()
    {
        protected Request $request;
        protected BuildAnalyticsReport $getdataAction;
        protected GetAnalyticsHeaderDataAction $getheaderDataAction;
    }

    public function report()
    {
        $this->authorize('index', 'ReportPolicy');

        $types = explode(',', $this->request->get('types', 'visitors,header'));
        $dateRange = $this->getDateRange();
        $cacheKey = sprintf(
            '%s-%s',
            $dateRange->getCacheKey(),
            implode(',', $types)
        );

        $response = [];
        $reportParams = ['dateRange' => $dateRange];
        if (in_array('visitors', $types)) {
            try {
                $response['visitorsReport'] = Cache::remember(
                    $cacheKey,
                    CarbonImmutable::now()->addDay(),
                    fn() => $this->getdataAction->execute($reportParams),
                );
            } catch (Exception $e) {
                $response['visitorsReport'] = app(
                    BuildAnalyticsReport::class,
                )->execute($reportParams);
            }
        }
        if (in_array('header', $types)) {
            $response['headerReport'] = Cache::remember(
                $cacheKey,
                CarbonImmutable::now()->addDay(),
                fn() => $this->getheaderDataAction->execute($reportParams),
            );
        }
    }
}

```

Fig 22: Part source code for admin dashboard

The above image shows the source code used for creating the page that displays the admin dashboard.

```

class UserController extends BaseController
{
    public function show($user)
    {
        $this->authorize('show', $user);
        return $this->success(['user' => $user]);
    }

    public function store($updateUserRequest $request)
    {
        $this->authorize('store', $user::class);
        $user = (new CreateUser())->execute($request->validated());
        return $this->success(['user' => $user, 201]);
    }

    public function update($user $user, $updateUserRequest $request)
    {
        $this->authorize('update', $user);
        $user = (new UpdateUser())->execute($user, $request->validated());
        return $this->success(['user' => $user]);
    }

    public function destroy($ids)
    {
        $userIds = explode(',', $ids);
        $shouldDeleteCurrentUser = request('deleteCurrentUser');
        $this->authorize('destroy', $user::class, $userIds);
        $users = User::whereIn('id', $userIds)->get();

        // guard against current user or admin user deletion
        foreach ($users as $user) {
            if ($shouldDeleteCurrentUser && $user->id === Auth::id()) {
                return $this->error(
                    ['Could not delete currently logged in user: :email', ['email' => $user->email]],
                );
            }
        }
    }
}

```

Fig 23: Part source code for registered users

The above image shows the source code used for creating the page that displays the registered users.