A PRELIMINARY MINI PROJECT REPORT ON

"CloudDocs: Cloud-Based Document Management System with AWS S3 Integration"

SUBMITTED TOWARDS THE PARTIAL FULFILLMENT OF THE REQUIREMENTS OF

BACHELOR OF ENGINEERING (Final Year B. Tech.)

Academic Year: 2024-25

By:

Sr. No.	Name of Student	PRN Number
01.	Atharva Dilip Lende	121B1B144
02.	Sakshi Sharad Kulkarni	122B2B289
03.	Janhavi Vijay Mali	122B2B292
04.	Vinayak Madan Shete	122B2B303

Under The Guidance of

Prof. Bodireddy Mahalakshmi

DEPARTMENT OF COMPUTER ENGINEERING,
PIMPRI CHINCHWAD COLLEGE OF ENGINEERING SECTOR 26, NIGDI, PRADHIKARAN

PIMPRI CHINCHWAD COLLEGE OF ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certify that, the project entitled
"CloudDocs: Cloud-Based Document Management System with AWS S3 Integration"
is successfully carried out as a mini project and successfully submitted by following students if "PCET's Pimpri Chinchwad College of Engineering, Nigdi, Pune-44".

Under the guidance of Prof. Bodireddy Mahalakshmi
In the partial fulfillment of the requirements for the Final Year B. Tech. (Computer Engineering)

Sr. No.	Name of Student	PRN Number
01.	Atharva Dilip Lende	121B1B144
02.	Sakshi Sharad Kulkarni	122B2B289
03.	Janhavi Vijay Mali	122B2B292
04.	Vinayak Madan Shete	122B2B303

Prof. Bodireddy Mahalakshmi
Project Guide

ABSTRACT

The **CloudDocs : Cloud-Based Document Management System with AWS S3 Integration** is designed to address the increasing need for secure, scalable, and efficient document management in today's digital-first landscape. Traditional file storage solutions, often limited by scalability, accessibility, and fault tolerance, can no longer meet the demands of growing organizations. This project leverages the power of AWS services—primarily S3, EC2, and NGINX—to create a cloud-native system that allows users to upload, manage, download, and delete documents securely and efficiently.

The system's core is built on **AWS S3**, which ensures durable and highly available storage for various document types such as PDFs, images, and Word files. **MongoDB** is utilized for storing document metadata, ensuring high availability and fault tolerance via a cluster setup that spans multiple regions. By hosting the application on **AWS EC2** and incorporating **NGINX** for load balancing and caching, the system can handle large volumes of user requests without performance degradation.

Key features include:

- **Document Management**: Users can upload, view, download, and delete documents through an intuitive, web-based interface built using **React.js**. Files are stored in AWS S3, and metadata, such as file type, upload date, and user information, is managed using MongoDB.
- **High Availability**: The system ensures continuous availability of documents and metadata by using AWS S3's fault-tolerant design and MongoDB's multi-region cluster setup, ensuring no single point of failure.
- **Security and Permissions**: With AWS Identity and Access Management (IAM), the system implements role-based access control, ensuring that only authorized users can perform actions such as uploading, downloading, or deleting files.
- **Scalability and Cost Efficiency**: AWS's pay-as-you-go model allows the system to scale effortlessly, meeting the growing needs of users and organizations without significant upfront costs.
- **Disaster Recovery**: The system supports disaster recovery mechanisms to ensure that documents remain accessible even in the event of system or region-wide failures. The use of AWS S3's multi-region replication ensures documents are secure, while MongoDB clusters handle metadata redundancy.

The system has applications across a wide range of industries, including education, healthcare, and corporate environments, offering a reliable and modern approach to document management. By combining robust cloud infrastructure with user-friendly features, this project presents an efficient solution to the challenges organizations face in managing their documents securely and efficiently. Future improvements can include advanced features such as document versioning, full-text search, and enhanced security measures, making it even more adaptable to diverse organizational needs

INDEX

List of Figures and Tables

Chapter 01 – Introduction

a) Problem Statement

Project Title : CloudDocs: Cloud-Based Document Management System with AWS S3 Integration

In today's digital landscape, organizations rely heavily on documents and files to manage their operations, collaboration, and communication. These documents must be stored securely, accessed quickly, and managed efficiently. Traditional file storage solutions like local servers or basic cloud storage may not always offer the scalability, accessibility, and fault tolerance necessary to handle growing document volumes. Additionally, file access and management across multiple devices and geographic locations become challenging, especially when considering security, availability, and ease of use.

The problem is that many existing document management solutions are either costly or inefficient in scaling with an organization's growth. Furthermore, they may lack integration with modern cloud-based technologies that provide high availability and failover mechanisms. The challenge is to create a cloud-based document management system that ensures:

1. **Scalability**: The system should handle a growing number of users and documents without performance degradation.
2. **Security**: Documents must be securely uploaded, stored, and accessed with proper permissions in place.
3. **Availability and Fault Tolerance**: Downtime of the document storage system should be minimized with redundant systems to ensure continuous access.
4. **Cost-Effectiveness**: A cloud-based solution should be able to scale and grow as needed without excessive upfront costs.

The project aims to solve these issues by leveraging AWS S3 for document storage, which provides high durability and availability. The system will be hosted on AWS EC2, ensuring scalability and reliability, with NGINX serving as a load balancer and frontend cache. Additionally, a robust MongoDB cluster will be set up to ensure database availability with multiple nodes across different regions, ensuring that if one node fails, others continue to serve data.

b) Project Idea

The project idea revolves around building a **cloud-based document management system** that enables users to **upload, view, download, and manage documents** through a web-based interface. The core of the system will use **AWS S3** for securely storing documents, while other AWS services, such as **EC2** for hosting the application and **NGINX** for load balancing, will be utilized to ensure scalability, fault tolerance, and high availability.

Key Features:

1. **Document Upload and Storage**: Users will be able to upload files (documents, PDFs,

images, etc.) that will be stored in AWS S3, ensuring durability and accessibility from anywhere.

2. **Document Listing**: A clean, user-friendly interface will allow users to view a list of uploaded documents, with options to sort or filter by date, file type, or other metadata.
3. **Download and Delete Files**: Users can easily download documents from AWS S3 or delete them if they no longer need them, using secure APIs integrated with the system.
4. **User Permissions**: Basic user management and permissions will be implemented to control who can access, upload, or delete files, ensuring secure document handling.
5. **High Availability and Fault Tolerance**: The system will use **MongoDB clusters** for storing document metadata (file names, upload dates, user info), ensuring that the database remains highly available. AWS S3's inherent fault tolerance will ensure documents are safe even during system failures.
6. **Scalable and Cost-Efficient**: By using AWS's pay-as-you-go model, the system will scale easily as the number of users or documents grows, keeping costs low without compromising performance.

This project will provide a highly reliable and secure solution for managing documents, with the potential to be used in various industries such as education, healthcare, or corporate environments.

c) Motivation

The motivation behind creating a **Cloud-Based Document Management System with AWS S3 Integration** stems from several key challenges and opportunities in document management that businesses, organizations, and individuals face today. Traditional methods of document storage, whether on physical drives or local servers, come with significant limitations regarding **scalability, accessibility, and security**. As cloud technology continues to evolve, there is a growing need for a reliable, scalable, and secure solution for document management, which is where this project aims to bridge the gap.

1. Accessibility and Convenience

One of the primary motivations for this system is to provide users with **anytime, anywhere access** to their documents. In the modern world, where remote work and distributed teams are becoming the norm, it is crucial that documents can be easily uploaded, accessed, and shared from any location. Cloud storage solutions like **AWS S3** allow for this level of accessibility, ensuring users can interact with their files without worrying about physical storage limitations.

2. Scalability for Growing Data Needs

As organizations expand, so do their data and document storage requirements. Traditional storage solutions may require constant upgrades to hardware and infrastructure, which can be both costly and time-consuming. A **cloud-based system**, like the one proposed in this project, can **automatically scale** to accommodate increasing numbers of documents and users. **AWS S3** offers unlimited storage, meaning there's no need for organizations to worry about capacity planning or expensive upgrades.

3. Enhanced Security and Data Integrity

Security is a major concern when it comes to document management, especially in industries dealing with sensitive information like healthcare, finance, or legal sectors. AWS provides robust **encryption mechanisms**, access control policies, and audit trails, ensuring that only authorized users can access or modify documents. With **MongoDB clusters** for metadata storage, the system guarantees high availability and consistency even if there are failures. AWS S3 also provides **11 9's of durability**, ensuring that documents are securely stored and protected against loss or corruption.

4. Cost Efficiency

Cloud-based document management systems, particularly those using services like AWS, are inherently more cost-effective compared to on-premises infrastructure. By adopting a **pay-as-you-go model**, this system reduces the costs associated with maintaining local servers, purchasing physical storage devices, and managing data centers. Users and organizations only pay for the resources they consume, making it a financially sustainable option, especially for small to medium-sized businesses.

5. Simplifying Document Management

Many existing document management systems are complex and come with steep learning curves. The motivation for this project is to create a **simple, intuitive interface** that can be used by non-technical users while maintaining powerful features in the backend. By offering easy upload, download, and delete functionalities, alongside basic permission management, this system aims to streamline document management and reduce the overhead associated with complicated software systems.

6. Disaster Recovery and Business Continuity

In case of system failures, natural disasters, or other catastrophic events, **disaster recovery** becomes a critical aspect of document management. With traditional on-premises storage, recovering lost data can be difficult or even impossible. This project leverages AWS's disaster recovery capabilities, ensuring that documents stored in S3 are always available, even if the primary system goes down. By using **multi-cluster MongoDB setups**, the system also ensures that metadata is consistently replicated across multiple databases, reducing the risk of data loss.

7. Support for Remote and Distributed Teams

As businesses increasingly adopt remote work models, it becomes more challenging to manage and share documents across various locations. A cloud-based system allows for **centralized document storage** with decentralized access, meaning teams from different geographies can collaborate seamlessly, without the need for cumbersome file transfer methods.

8. Environmentally Friendly Solution

With the rising importance of sustainability and environmental responsibility, cloud-based solutions help reduce the **carbon footprint** associated with maintaining physical servers and data centers. By utilizing AWS's infrastructure, which is optimized for energy efficiency and minimal waste, this project aligns with the goal of creating eco-friendly technology solutions.

d) Scope

This project primarily focuses on building a **scalable**, **secure**, and **reliable** document management system using AWS S3 for storage. The key features of the project are:

- **Uploading and Managing Documents**: Users will be able to upload, list, download, and delete documents using a web-based UI.
- **Cloud-Based Infrastructure**: Leveraging AWS services such as S3, EC2, and IAM for secure and scalable storage and processing.
- **Database Scalability**: Using a MongoDB cluster for document metadata management, ensuring data availability across different nodes in case of failure.
- **Load Balancing and Frontend Caching**: NGINX will be used to handle load balancing and cache frontend requests, improving performance and ensuring fault tolerance.
- **Security**: Implementing IAM roles and access policies to control who can upload, view, or delete documents.
- **Testing and Monitoring**: The system will include continuous monitoring and periodic testing to ensure reliability.

The system can be expanded in the future to support more advanced features such as user authentication, version control for documents, and integration with other AWS services like DynamoDB for enhanced metadata storage.

e) Literature Survey/Requirement Analysis

For the development of a **Cloud-Based Document Management System with AWS S3 Integration**, several research papers and studies have been explored to understand the best practices, design considerations, and technological advancements in cloud computing, document management, and data storage. The literature survey focuses on academic research related to **cloud storage systems**, **AWS services**, **document management systems (DMS)**, and **scalable architecture** for web applications.

1. Cloud Storage and Security Considerations

One of the most critical components of this project is secure and reliable cloud storage, and AWS S3 is a prominent choice due to its scalability and durability. A paper by **Zhang et al. (2010)**, titled *Cloud Storage: The Future of Data Backup and Recovery*, discusses how cloud storage solutions provide cost-effective alternatives to traditional storage systems by eliminating the need for physical hardware, while also offering extensive durability and reliability. The paper highlights the **redundancy and data integrity** mechanisms used by AWS S3, where data is replicated across multiple availability zones to prevent loss due to hardware failures.

Another relevant study by **Ali et al. (2015)**, *Cloud Computing Security Issues and Challenges*, emphasizes the importance of **data security** and access management in cloud storage environments. It addresses challenges like secure data transmission, encryption, and access control—key aspects considered in this project. The research suggests that robust **encryption**

mechanisms and **role-based access control** (RBAC) are essential in cloud storage to protect sensitive documents.

2. Document Management Systems (DMS) in Cloud Environments

The deployment of document management systems in the cloud has been well-documented in research. **Singh et al. (2012)**, in their paper *Cloud-Based Document Management System and Security Framework*, discuss how cloud-based DMS improves accessibility and reduces the complexity of traditional DMS architectures. They argue that using cloud platforms, such as AWS, provides the flexibility and scalability needed for managing large volumes of documents. The paper also suggests the use of **AWS IAM (Identity and Access Management)** for controlling access to documents, which has been integrated into this project for managing user permissions.

Similarly, **Gupta et al. (2016)**, in *A Comparative Analysis of Document Management Systems*, evaluate different cloud-based DMS solutions, comparing their features in terms of **cost efficiency**, **scalability**, and **ease of use**. The study concludes that using a cloud platform significantly reduces costs associated with traditional hardware-based document storage solutions while also improving the efficiency of document retrieval, management, and sharing. These findings were instrumental in shaping the design and functionality of this project.

3. AWS Services for Scalable and Fault-Tolerant Systems

Varia et al. (2013), in the book *Architecting for the Cloud: AWS Best Practices*, provide a comprehensive overview of how to build scalable and fault-tolerant applications using AWS services. Their work highlights the advantages of **S3 for durable object storage**, **EC2 for scalable compute resources**, and **NGINX for load balancing**. The discussion on AWS's **auto-scaling** and **pay-as-you-go** pricing model provided key insights into making the system scalable without incurring high costs.

Moreover, **MongoDB for scalable metadata storage** is discussed in **Chodorow (2013)**, *MongoDB: The Definitive Guide*. This book offers detailed explanations of how to use MongoDB's clustering and replication features to ensure high availability, even in the case of hardware or network failures. It provides the foundation for implementing MongoDB clusters to store metadata and ensure no single point of failure in the system.

4. High Availability and Disaster Recovery

High availability and disaster recovery are crucial components of any cloud-based system. **Jain et al. (2014)**, in *Disaster Recovery in Cloud Computing: A Survey*, highlight how cloud providers like AWS offer built-in disaster recovery mechanisms. They discuss how **AWS S3's multi-region replication** ensures that data is backed up and available, even in the event of region-wide failures. The use of **MongoDB clusters** for fault-tolerant metadata storage, as described in this project, aligns with the best practices recommended by this research.

Another important study by **Chen et al. (2010)**, *Ensuring Data Integrity in Cloud Storage*, introduces techniques for maintaining data integrity in cloud storage, particularly in distributed systems. They explore how **erasure coding** and replication can prevent data loss during server outages, which informs the project's use of S3's **built-in redundancy** and MongoDB's multi-node architecture for fault tolerance.

5. Document Metadata Management in Distributed Systems

Managing document metadata efficiently in distributed systems is critical for this project. **Dede et al. (2013)**, in *Designing and Implementing Metadata Management Systems for Large-Scale Document Repositories*, explore the use of NoSQL databases like MongoDB for managing large volumes of metadata. They provide insights into how to structure metadata storage in distributed environments to enable **quick retrieval**, **efficient indexing**, and **data consistency** across nodes. This research supports the project's choice of MongoDB clusters for managing metadata, ensuring that users can interact with their documents efficiently, even under heavy system loads.

6.Cloud-based Document Management: A Review

This paper offers a comprehensive overview of cloud-based document management systems, discussing various architectures, functionalities, and the evolution of such systems. It highlights the advantages of cloud solutions, including scalability, cost-effectiveness, and accessibility, while also addressing key challenges such as data security, compliance, and integration with existing systems. The authors provide insights into emerging trends and propose future research directions to enhance the effectiveness of cloud-based document management.

7.Challenges in Scalable Document Management Systems

This study delves into the specific challenges faced by scalable document management systems, including issues related to performance, data consistency, and user management. The authors analyze various approaches to scalability, emphasizing the importance of adaptive architectures that can respond to changing workloads and user demands. Additionally, the paper explores how emerging technologies like artificial intelligence and machine learning can be leveraged to address these challenges.

8.Security Concerns in Cloud Storage

This paper focuses on the security challenges associated with cloud storage, a critical component of cloud-based document management systems. The authors discuss common vulnerabilities and threats, such as unauthorized access, data breaches, and loss of data integrity. They also provide an overview of best practices and strategies for enhancing security in cloud environments, including encryption techniques and access control mechanisms.

Chapter 02 – Project Design

a) Hardware and Software Requirements

1. Frontend: React.js for Building the User Interface

- **React.js**: React is a popular JavaScript library used for building dynamic and responsive user interfaces. It provides components that enable the modular design of the application, making it efficient to render and update based on user interactions.
- **Why React?** React's component-based architecture makes it ideal for building scalable, maintainable user interfaces. Its virtual DOM optimizes the performance, especially in a document management system where changes like uploading and viewing files are frequent.

2. Backend: Node.js and Express.js for Server-Side Logic

- **Node.js**: This is a JavaScript runtime that allows for running server-side applications. Its non-blocking, event-driven architecture is ideal for I/O-heavy operations like file uploads.
- **Express.js**: A minimalist framework for Node.js that simplifies API creation and routes handling. It helps in creating RESTful endpoints for uploading, downloading, and managing files stored in AWS S3.
- **Why Node.js?** Its scalability, performance, and ability to handle multiple concurrent requests make it perfect for a cloud-based system that needs to manage document uploads, downloads, and metadata operations.

3. Database: MongoDB with Multiple Clusters for High Availability

- **MongoDB**: A NoSQL database designed for distributed data and scalability. In this project, it's used to store metadata about documents such as file name, type, size, and upload date.
- **Multiple Clusters**: The setup involves three production clusters to ensure **high availability** and **fault tolerance**. If one cluster goes offline, the others can continue to serve requests, ensuring uninterrupted service.
- **Why MongoDB?** It offers flexible schema design and horizontal scalability, making it suitable for cloud-native applications that may require dynamic fields for metadata management.

4. Storage: AWS S3 for Document Storage

- **AWS S3**: Amazon Simple Storage Service (S3) is a cloud storage service that provides durable and scalable storage. Documents such as PDFs, Word files, and images are uploaded and stored in the S3 bucket.
- **Why S3?** AWS S3 is highly reliable, with 99.999999999% (11 9's) durability. It offers built-in redundancy and scales to any size of data, making it ideal for document storage and backup.

5. Hosting: AWS EC2 for Both Frontend and Backend Hosting

- **AWS EC2**: Amazon EC2 provides scalable compute capacity in the cloud. It hosts both the frontend (React) and backend (Node.js and Express) of the document management

system.

- **Why EC2?** EC2 allows users to scale computing resources easily, deploy both the client and server applications, and benefit from the broad array of AWS features such as monitoring, security groups, and auto-scaling.

6. Load Balancer: Nginx for Load Balancing and Frontend Caching

- **Nginx**: An open-source web server that also functions as a load balancer. It is placed in front of EC2 instances to distribute incoming traffic evenly across servers and cache static resources such as frontend files.
- **Why Nginx?** It improves the system's reliability by balancing traffic, reduces latency via frontend caching, and enhances security by controlling traffic flow.

7. AWS SDK: Used for Seamless Integration with AWS Services

- **AWS SDK**: The AWS Software Development Kit (SDK) for Node.js enables seamless integration with AWS services, including S3 for document uploads, management, and EC2 for server management.
- **Why AWS SDK?** It simplifies the development of features that interact with AWS services, such as uploading and retrieving documents from S3 buckets.

b) Dataset Design

1. S3 Bucket

- The **S3 Bucket** stores the actual document files such as PDFs, Word documents, and images. Each file is given a unique key and can be accessed via its S3 URL.
- **Structure**: Files can be organized in folders within the bucket if needed, though each file is identified by a unique key even if the same name is used.
- **Advantages**: S3 automatically manages data redundancy and durability, ensuring that files are never lost.

2. Metadata: MongoDB Stores Metadata

- **MongoDB** stores metadata about the documents uploaded to S3. This includes details like:
  - File name
  - Upload date
  - File size
  - File type (PDF, Word, Image, etc.)
- **Why Metadata?** Storing metadata in MongoDB allows the system to provide quick access to document information without querying S3 for file details, improving performance.

3. Clusters: MongoDB Clusters for High Availability

- **Production Clusters**: There are three clusters in production that ensure continuous availability of the database. In case of failure of one cluster, the others take over, ensuring no downtime.
- **Test Database**: A separate test database is used for development and testing purposes. This ensures that production data remains unaffected during updates or testing.

c) Hours Estimation

Frontend Development: 30 hours

Building the document upload interface, document list, and document management functionalities (view, delete) using React.js.

Backend Development: 35 hours

Setting up API endpoints using Express.js for file upload, download, and management. This also includes handling authentication and interaction with MongoDB and S3.

AWS S3 and EC2 Integration: 25 hours

Setting up AWS S3 buckets, IAM roles, and permissions for handling document uploads. Deploying the application on AWS EC2 and configuring the backend to interact with S3 via the AWS SDK.

Database Design and Cluster Setup: 20 hours

Designing the MongoDB database schema for storing document metadata and configuring production clusters for high availability and redundancy.

Testing and Debugging: 15 hours

Testing the entire system for bugs, handling edge cases, and ensuring smooth integration of all components (frontend, backend, database, and storage).

Total Hours: 125 hours

Chapter 03 – Module Description

a) Block Diagram

The block diagram below illustrates the components of the Cloud-Based Document Management System and how they interact with each other. Each module plays a critical role in ensuring that user requests are efficiently processed, documents are securely managed, and system performance is optimized.

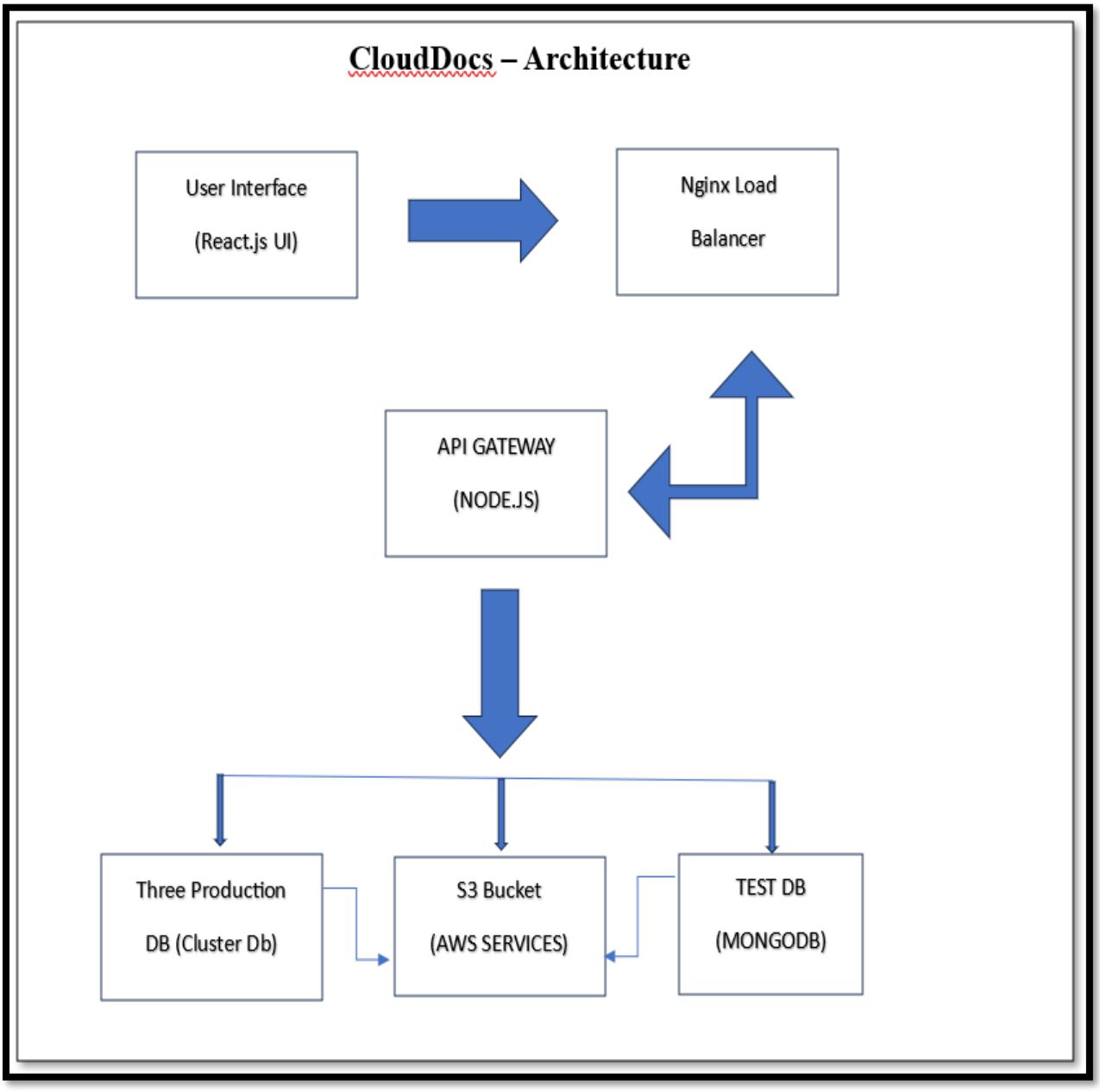

Fig. 1: Block Diagram

b) Modules of Project

1. User Interface (UI)

- **Description**: The frontend of the application is built using **React.js**, providing an interactive and user-friendly interface.
- **Functionality**: Users can upload documents, view a list of uploaded files, download them, or delete them through the UI. The UI communicates with the backend server via HTTP requests, allowing users to perform actions seamlessly.

2. API Gateway (Backend)

- **Description**: The backend is developed using **Node.js** and **Express.js**, serving as the API gateway that processes requests from the frontend.
- **Functionality**:
    - Handles document management logic, including file uploads, downloads, and deletions.
    - Interacts with **AWS S3** to perform file storage operations and with **MongoDB** to manage metadata related to documents.
    - Enforces security through access management, ensuring that only authorized users can perform specific actions.

3. AWS S3

- **Description**: **Amazon S3** is used for document storage, providing a durable and scalable solution.
- **Functionality**:
    - All document files are securely stored in S3 buckets, where each file is uniquely identified.
    - Access permissions are managed to ensure secure uploads, downloads, and deletions. This guarantees that only authorized users can access specific documents.

4. Database (MongoDB Clusters)

- **Description**: The system uses a **MongoDB database** consisting of multiple clusters to store metadata related to documents.
- **Functionality**:
    - The metadata includes information such as file names, types, and upload dates.
    - By spreading the database across three production clusters, the system ensures high availability, data redundancy, and scalability.
    - A **test database** is utilized for development purposes, allowing for testing without affecting the production data.

5. Nginx Load Balancer

- **Description**: **Nginx** acts as a load balancer, optimizing the distribution of incoming user requests across multiple backend instances.
- **Functionality**:
    - Balances the load to ensure that no single instance becomes a bottleneck, improving

system performance and responsiveness.

- Provides a reverse proxy mechanism, directing client requests to the appropriate backend services based on current load and availability.
- Implements frontend caching to enhance load times and reduce server response times for frequently accessed documents.

This architecture ensures that the Cloud-Based Document Management System is robust, scalable, and capable of efficiently handling user requests while maintaining high levels of data integrity and security.

Chapter 04 – Results and Discussions

a) Source Code

Github Repository Link : https://github.com/Atharva1213/cloud-doc-mgmt-s3

b) Screenshots

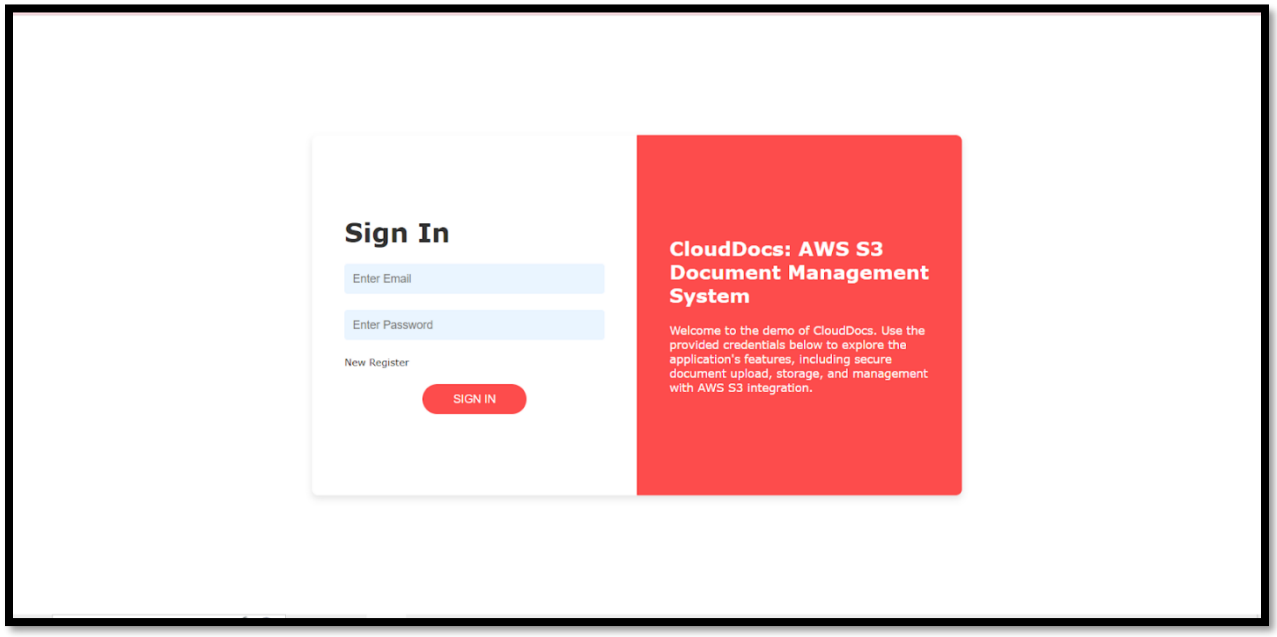

Fig. 2: Login Page

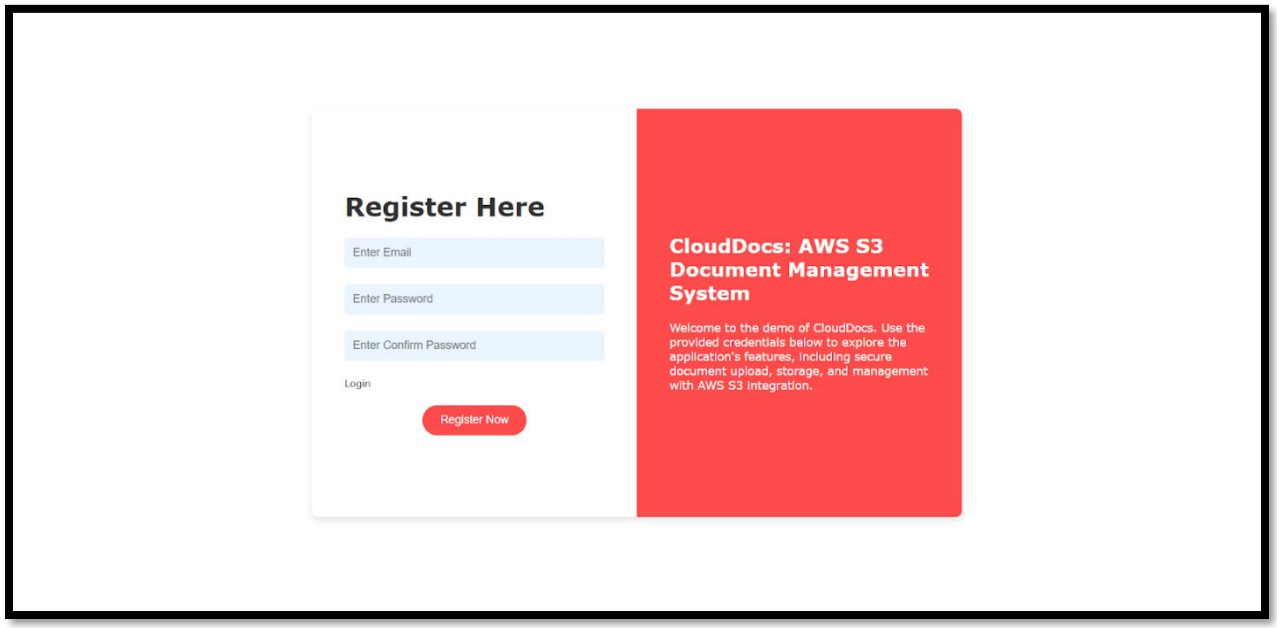

Fig. 3: Registration Page

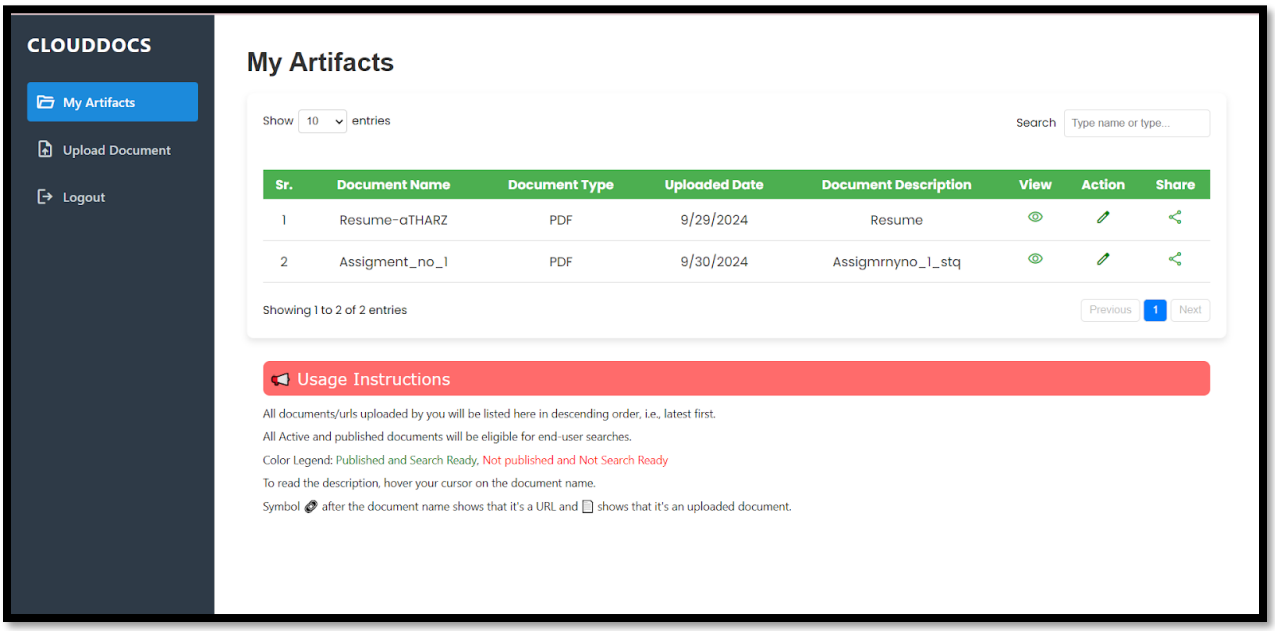

Fig. 4: Home Page (Dashboard)

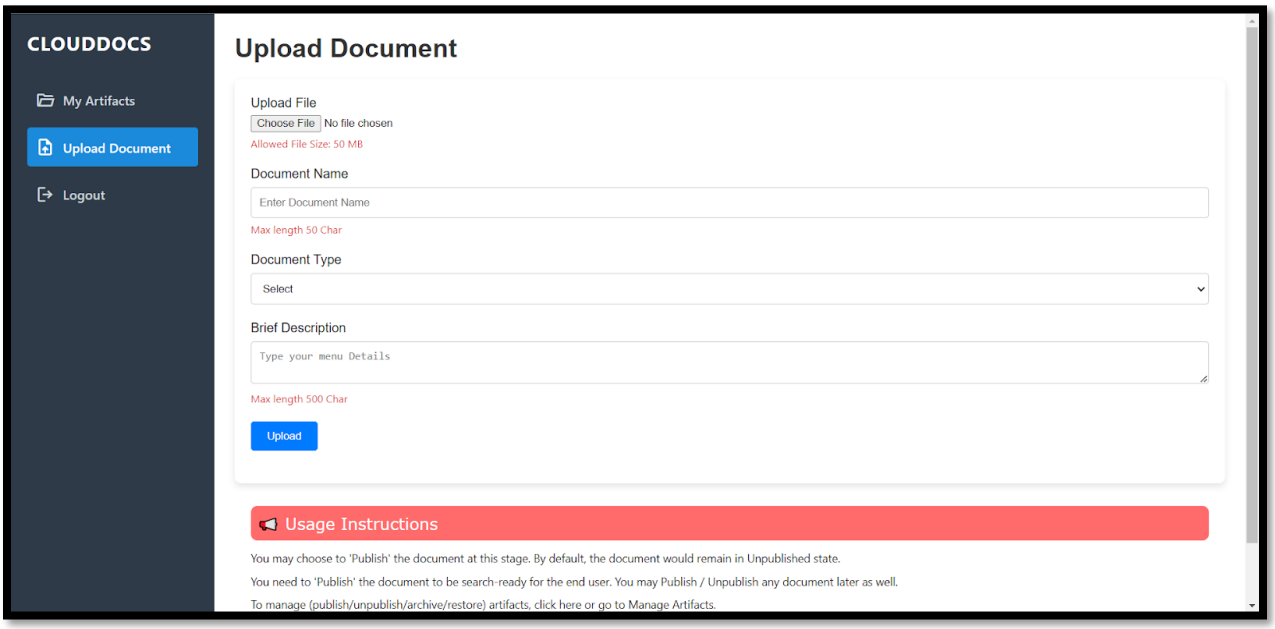

Fig. 5: Uploading Document GUI

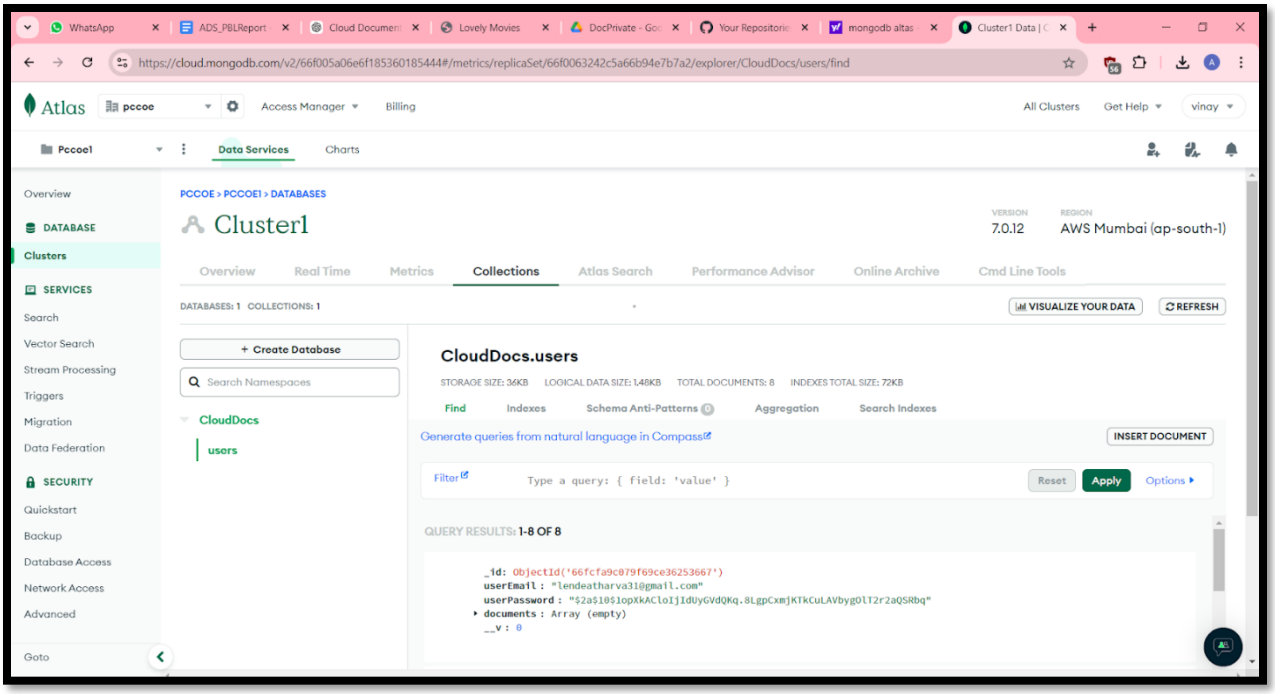

Fig. 6: MongoDB Database

c) Test Cases

Requirement ID	Requirement Specification	Test Case ID	Test Case Specification	Test Status (Pass/Fail)
R1	System must allow users to upload documents to AWS S3.	TC1.1	Upload a valid document (e.g., PDF) and verify it's stored in S3.	Pass
		TC1.2	Upload a document successfully larger than the allowed limit	Fail
		TC1.3	Upload a document without sufficient permissions.	Fail
R2	System must allow users to view a list of uploaded documents.	TC2.1	Fetch the list of documents and verify metadata (file names, dates, etc.) is displayed correctly.	Pass
		TC2.2	Verify that the list is updated after a new file is	Pass

			uploaded.	
		TC2.3	Try to access the document list without authentication and verify the access is denied.	Fail
R3	System must allow users to download documents from AWS S3.	TC3.1	Download an existing document and verify the file matches the original upload.	Pass
		TC3.2	Try downloading a non-existent file and verify that an error message is returned.	Fail
		TC3.3	Download a document with restricted access and verify unauthorized access is denied.	Fail
R4	System must allow users to delete documents from AWS S3.	TC4.1	Delete a valid document and verify it is removed from both S3 and MongoDB metadata.	Pass
		TC4.2	Try deleting a document without sufficient permissions and verify access is denied.	Fail
		TC4.3	Try deleting a document that does not exist and verify the appropriate error message is returned.	Fail
R5	System must enforce user management and permissions.	TC5.1	Assign read/write/delete permissions to users and verify correct access is given based on roles.	Pass
		TC5.2	Attempt to upload a document as a user with read-only permission and verify upload fails.	Fail
		TC5.3	Try to delete a document as a user with no delete permission and verify the	Fail

			action is denied.	
R6	System must support searching documents by file name and metadata.	TC6.1	Search for a document by file name and verify the correct result is returned.	Pass
		TC6.2	Search for a document using metadata (e.g., date, file type) and verify correct results are shown.	Pass
		TC6.3	Search for a non-existent document and verify the appropriate message is returned.	Fail
R7	System must handle concurrent uploads without performance degradation.	TC7.1	Simulate multiple users uploading documents simultaneously and verify the system handles the load.	Pass
		TC7.2	Verify the performance under heavy concurrent uploads (e.g., 50 simultaneous uploads).	Pass
		TC7.3	Simulate a server crash during uploads and verify data consistency after recovery.	Fail
R8	System must support version control for uploaded documents.	TC8.1	Upload a new version of an existing document and verify the previous version is preserved.	Pass
		TC8.2	Attempt to access an older version of a document and verify correct retrieval.	Pass
		TC8.3	Upload a new version of a document and verify an error occurs if file versioning is disabled.	Fail
R9	System must implement disaster recovery for documents in case of	TC9.1	Simulate an AWS region failure and verify that documents are still accessible via replicated	Pass

	failure.		storage.	
		TC9.2	Verify data consistency after recovery from a system crash.	Pass
		TC9.3	Simulate failure of MongoDB clusters and verify metadata is still accessible from another region's cluster.	Fail
R10	System must include auditing for document access and changes.	TC10.1	Verify that audit logs are generated when documents are uploaded, downloaded, or deleted.	Pass
		TC10.2	Verify that audit logs show who accessed or modified a document along with timestamps.	Pass
		TC10.3	Try accessing the audit logs without admin privileges and verify that access is denied.	Fail

Table 1 : Test Cases

Chapter 05 – Conclusion

The **Cloud-Based Document Management System with AWS S3 Integration** is designed to be a robust, scalable, and secure solution for organizations needing efficient document management. By leveraging AWS services like S3, EC2, and IAM, along with MongoDB for metadata, the system can provide **high availability**, **fault tolerance**, and **global accessibility**. Additionally, its professional architecture—complete with load balancing, caching, and failover mechanisms—ensures optimal performance, security, and scalability, making it suitable for both small and large enterprises.

This system can be expanded to integrate advanced features like user authentication, document versioning, and full-text search, providing a modern, cloud-native approach to document management.

Chapter 06 – References

1. Zhang, Q., Cheng, L., & Boutaba, R. (2010). *Cloud storage: The future of data backup and recovery*. ACM Computing Surveys (CSUR), 42(3), 1-17. DOI: 10.1145/1670679.1670682.
2. Ali, M., Khan, S. U., & Vasilakos, A. V. (2015). *Security in cloud computing: Opportunities and challenges*. Information Sciences, 305, 357-383. DOI: 10.1016/j.ins.2015.01.025.
3. Singh, M., Kalra, S., & Mangat, V. (2012). *Cloud-based document management system and security framework*. International Journal of Computer Applications, 44(7), 14-19. DOI: 10.5120/6232-8459.
4. Gupta, P., Gulati, N., & Kumar, A. (2016). *A comparative analysis of document management systems: A cloud perspective*. Journal of Information Technology & Software Engineering, 6(4), 1-6. DOI: 10.4172/2165-7866.1000188.
5. Varia, J., Mathew, S., & Arora, S. (2013). *Architecting for the cloud: AWS best practices*. Amazon Web Services, Inc. Retrieved from https://aws.amazon.com/whitepapers.