INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 3, March 2026 | ISSN: 2456-4184 | IJNRD.ORG

To Design and Develop a Cloud-Based File Management System

Prof. Bhushan Deshpande, Professor, Department of Computer Technology, Kavikulguru Institute of Technology and Science, Ramtek
Ms. Twinkle Paunikar, Student, Department of Computer Technology, Kavikulguru Institute of Technology and Science, Ramtek
Mr. Tanish Makde, Student, Department of Computer Technology, Kavikulguru Institute of Technology and Science, Ramtek
Mr. Saurabh Nanhe, Student, Department of Computer Technology, Kavikulguru Institute of Technology and Science, Ramtek
Ms. Shrushti Kamble, Student, Department of Computer Technology, Kavikulguru Institute of Technology and Science, Ramtek
Ms. Namrata Pahune, Student, Department of Computer Technology, Kavikulguru Institute of Technology and Science, Ramtek

Abstract:

Cloud computing has grown in popularity as a method for storing and managing data online, enabling users to protect their files and access them from any internet-connected device. Cloud-based file management systems are becoming more popular because they lessen the requirement for physical storage and provide more freedom and efficiency in data management. Nevertheless, the majority of currently available platforms are geared toward commercial users or are too difficult for personal usage, typically offering only basic features and few customization possibilities, necessitating a simple and user-friendly cloud file manager. The goal of the project is to create a cloud-based file management system with a front-end interface that prioritizes dynamic user interaction and optimal performance for safe file operations like uploading, renaming, deleting, downloading, previewing, and sharing files online.

The system provides secure user registration and authentication, as well as access to tailored cloud workspaces where users may manage files using an interactive dashboard, while backend APIs manage requests and communicate with cloud storage services. User metadata and file data are kept safely in the cloud, with live synchronization between sessions and devices. The project used Tailwind CSS for simple, flexible layouts and React. js to create a dynamic user interface, while Node. js with Express. js offered a lightweight server environment for file activities and API requests. User-specific metadata and file data were managed in a scalable database by MongoDB Atlas, secure user logins and registrations were made possible by Firebase Authentication, and dependable cloud storage was provided by AWS S3. The successful implementation illustrates the potential for developing user-friendly, secure, and scalable personal data management solutions that bridge the gap between advanced cloud storage capabilities and accessible user experiences, all while preserving professional-grade security and performance.

KEYWORDS: Cloud Storage, Virtual File Management System, File Security, User Authentication, Amazon S3

Introduction:

The way that people and organizations handle data has undergone a transformation as a result of the quick rise in digital information. Local devices like external drives or personal computers are heavily relied upon by conventional file storage systems. Users of these systems are restricted in their access since files are kept on the same device that they must use. Furthermore, they may experience data loss from hardware failures, storage capacity restrictions, and a lack of device synchronization. In order to address these restrictions, cloud computing has become a viable option. Users may upload and retrieve files from any location using the internet thanks to cloud storage systems. Users gain from centralized data administration, increased availability, and greater scalability by switching storage infrastructure from local hardware to distant cloud servers. In computer environments, file management systems are essential. Users may use them to arrange, save, access, and change

files with ease. Traditional file systems, on the other hand, are device-dependent and lack support for real-time synchronization or remote access. Users anticipate unrestricted access to their files across several devices in today's computing environment, which makes this restriction problematic.

The purpose of this study is to create and deploy a cloud-based file management system that supports secure authentication and enables users to carry out typical file activities like uploading, previewing, renaming, downloading, and deleting. The suggested system makes use of contemporary web technologies to create a scalable and user-friendly solution. The frontend user interface is created using Tailwind CSS and React. js. The backend services are implemented using Express. js and Node. js. Amazon S3 handles binary file storage, and MongoDB Atlas stores file metadata. This design promotes effective storage administration and optimal system performance. Additionally, the system includes security features like JWT-based authentication, password hashing using bcrypt, secure communication, and restricted cloud resource access. The architecture is based on a three-tier model made up of a presentation layer, an application layer, and a storage layer. The increasing need for safe and accessible cloud-based file management systems is addressed by the creation of this platform, which also illustrates how cloud technologies may be successfully integrated with contemporary web frameworks.

Literature Review:

1. Cloud Based File Storage System:

The study "Cloud Based File Storage System" by Shubham Patil and Soham Bharat Patil describes a cloud storage system built using Firebase for storage and authentication, and React and Next. js for the frontend. The system aims to offer a safe and engaging environment for users to upload, manage, and remove files inside a single cloud platform. Real-time synchronization, where any modifications to files are immediately replicated throughout the program, is one of the system's main features. Next. js enhances rendering performance and responsiveness, whereas React makes component-based interface design possible. By integrating authentication and storage services, Firebase makes backend infrastructure simpler. However, the system relies solely on Firebase for authentication and storage, which could limit scalability in big installations. Furthermore, it lacks sophisticated security measures, such as robust encryption methods and role-based access management. This idea is expanded upon in the current project, which integrates MongoDB Atlas, AWS S3, and Node. js, resulting in improved scalability and separation between metadata and file storage.

2. Efficient File Sharing System Utilizing MongoDB and Node.js:

Vinaykumar D and Venkat Pranav's study, "Efficient File Sharing System Using MongoDB and Node. js," suggests a flexible file-sharing infrastructure that makes use of MongoDB for metadata storage and Node. js for backend processing. Through Node. js's asynchronous architecture, the system effectively manages several file requests while enabling real-time file uploading, downloading, and editing. The peer-to-peer sharing mechanism is a significant characteristic that enhances file transfer efficiency while lessening reliance on central servers. With its versatile document-based storage, MongoDB may readily adjust to shifting file properties. In spite of these benefits, the peer-to-peer architecture of the system raises the possibility of security breaches. Token-based authentication, encryption techniques, and conflict resolution during concurrent modifications are not thoroughly covered in the book. By incorporating JWT authentication and AWS S3 for safe object storage, the suggested initiative enhances reliability, scalability, and security while also extending this methodology.

3. Toward a Cloud Operating System:

Fabio Pianese and Marcus Brunner's paper, "Toward a Cloud Operating System," presents the notion of integrating computing resources, such as storage, networking, and memory, within a single cloud environment. The study examines the use of load balancing, virtualization, and automation in cloud infrastructure management. By treating cloud infrastructure like an operating system, the suggested architecture seeks to enhance scalability and streamline resource management. However, the study focuses primarily on infrastructure-level management as opposed to file operations at the user level. The research doesn't cover

features like file upload, rename, preview, or secure user authentication. While using the idea of centralized cloud management, the current project focuses on cloud-based file management systems in particular. By integrating React, Node. js, MongoDB Atlas, and AWS S3, the system offers a useful and intuitive platform for managing files in academic and personal settings.

Research Design:

The purpose of the Personalized Virtual Cloud-Based File Management System is to serve as a cloud-native platform for secure and scalable file storage and management through internet-based services. The suggested system, unlike conventional file systems that depend on local storage, runs completely in the cloud, giving users the ability to manage their files from anywhere with an internet connection. The general structure of the system adheres to a layered design methodology that divides the user interface, application processing, and storage administration. This structural segregation safeguards sensitive configurations within server-side components and increases maintainability, scalability, and overall system dependability. Figure 1, which shows the high-level workflow of the cloud file management system, illustrates how user requests go through validation, authentication, and processing phases before accessing stored data.

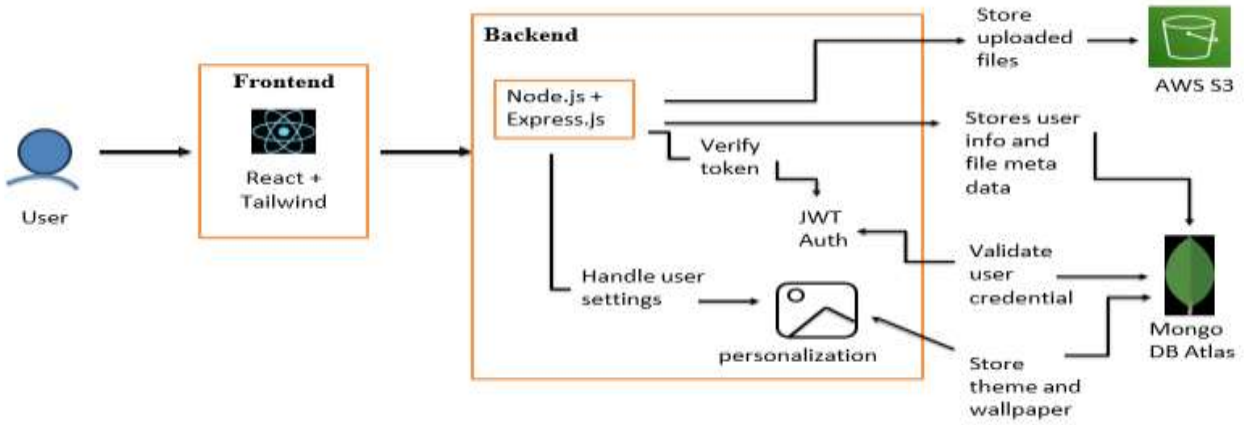


Fig 1 : high-level workflow of the cloud file management system

When a user engages with the web interface and makes a request, such as registration, login, or file operations, the operational workflow begins. These requests are sent via safe HTTP communication to the application server, where they are checked and handled. JSON Web Tokens (JWT) are used for authentication, making sure that only verified users have access to restricted content. bcrypt hashing is used to protect passwords, which prevents the storage of plaintext credentials. Upon successful authentication, users are permitted to carry out file management tasks such as uploading, listing, renaming, and deleting files. File uploads are made directly to cloud object storage using pre-signed access URLs to provide safe transient access without disclosing storage credentials, which enhances performance and lowers server load.

The real files are stored in a distributed object storage environment, while the cloud-hosted database service houses structured data such as user information, file characteristics, timestamps, and access references. By preventing massive binary files from overwhelming the database system, this division between file objects and metadata enhances efficiency. The system synchronizes metadata records with cloud storage objects whenever users carry out operations like deleting or renaming files in order to preserve consistency and avoid orphaned storage references. The list of files linked to each verified user is also shown by metadata retrieval.

To safeguard user data and system resources, security concerns are taken into account at every stage of system design. Token-based authentication, encrypted communication, middleware request validation, and regulated access rules ensure that users can only access their own files. Environment variables, as opposed to direct storage in application code, are used to manage sensitive configuration data like database connection strings, access keys, and authentication secrets. By lowering the danger of illegal access, these actions enhance the platform's overall security profile.

To guarantee global accessibility, scalability, and reliability, the whole system is hosted in the cloud. The application interface is hosted on a cloud platform, whereas the server-side processing, database services, and object storage are housed in managed cloud infrastructure. Users may use any internet-connected device to log in, upload files, rename documents, delete stored data, and change settings thanks to this deployment approach. The suggested system offers a practical and scalable solution for customized virtual file management that does not rely on local storage infrastructure by integrating secure authentication, cloud storage, and distributed data management.

Research Methodology:

The research methodology used to create the Customized Virtual Cloud-Based File Management System is systematic and implementation-oriented, with the goal of creating, developing, and testing a secure cloud-based file management environment. The research aims at integrating modern cloud technologies to allow users to store, access, and manage their files via an internet-based platform, as opposed to depending on conventional local storage solutions. To make sure that the system being developed satisfies functional and security requirements, the methodology includes need analysis, system design, implementation, testing, and deployment phases.

A review of available cloud storage platforms and prior publications first examined the needs of a cloud-based file management system. This study aided in pinpointing frequent shortcomings like subpar security protections in file management and user authentication, little integration of cloud services, and few customization options. In order to maintain scalability and reliability, the suggested system was created in light of these findings to enable real-time file operations, effective file storage, metadata management, and safe user authentication.

The deployment phase entailed creating a web-based program that manages file-related activities while integrating cloud storage services, database administration, and user authentication. Users can sign up, log in, submit files, view stored files, rename files, and remove unwanted files using the system's web interface. Only authorized users are able to access system resources thanks to secure authentication mechanisms. Cloud object storage is used for file uploading and storage, while a cloud-hosted database environment is used to store metadata like file names, timestamps, and user references. By separating data management and system performance, this separation enhances system efficiency.

The system's functionality and reliability were assessed by running a number of operational scenarios, such as user registration, authentication, file uploading, file listing, renaming, and deleting. The purpose of these tests was to ensure data consistency, secure access control, and successful synchronization between cloud file objects and stored metadata. To guarantee that system operations would remain stable under various circumstances and that unauthorized access attempts would be blocked, security methods and error management were also investigated.

Lastly, the system was set up in the cloud in order to guarantee availability and scalability. Users can interact with the system via a web browser without the need for local installation or specific hardware resources when it is deployed in the cloud. Using contemporary cloud technologies, this method approach demonstrates the feasibility of creating a secure and scalable customized file management system while maintaining data security, operational efficiency, and user accessibility.

Result and Discussion:

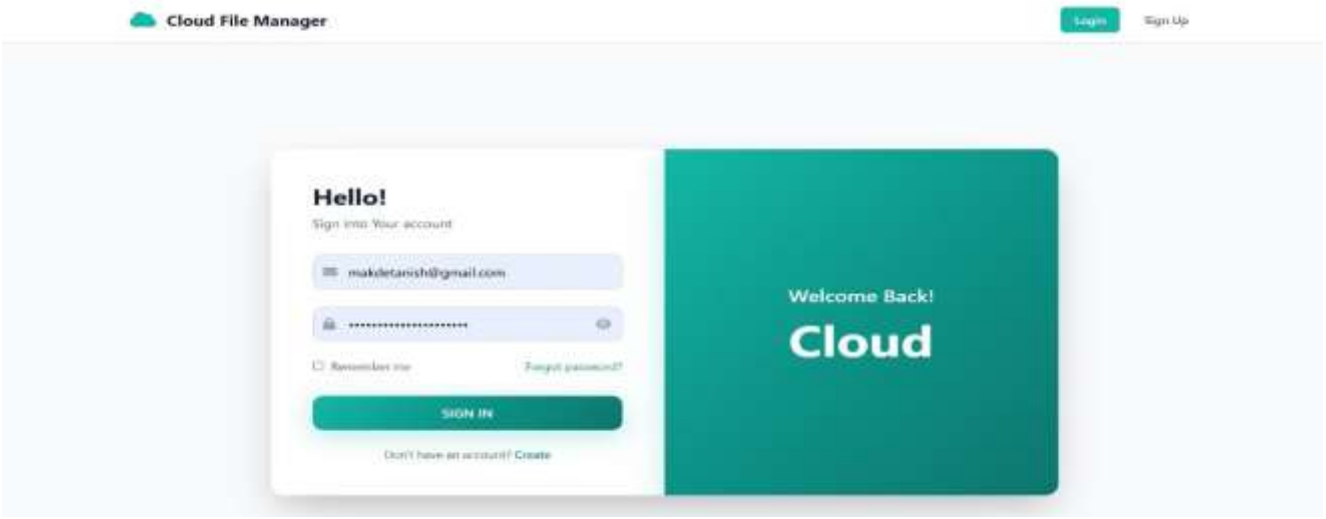
A working prototype that combines user customization capabilities, cloud storage, real-time synchronization, and secure authentication was the result of the cloud-based virtual file management system's successful implementation. The created software was tested in a variety of situations, including user verification, file operations, metadata management, and customization. The findings show that the system offers a user-friendly, safe, and quick interface for cloud file management. Every module, from logging in to keeping an eye on storage, collaborated to provide a seamless user experience.

1. Authentication Pages:

- There are two primary screens in the authentication module: the Login Page and the Sign-Up Page.

1.1. Login Page:

- The Login Page, which allows registered users to authenticate using an email and password combination, is shown in Figure 2. It has other choices, such as "Remember me," for session persistence and



"Forgot password? " for account recovery. To guarantee proper formatting prior to submitting requests to the server, input fields are validated on both the frontend and backend. The input boxes, buttons, and layouts were styled using Tailwind utilities, giving the page a polished and professional vibe.

Fig 2 : Login Page Interface

1.2. Sign up page:

- New users may establish an account through the Sign-Up Page. As seen in Figure 3, it has fields for email, password, and username. Prior to submission, email formats and password complexity are verified by input validation to make sure they satisfy the minimum standards. In addition to a terms and conditions checkbox for legal compliance in user agreements, the site also includes a terms and conditions checkbox. The system keeps the hashed credentials after a successful registration.

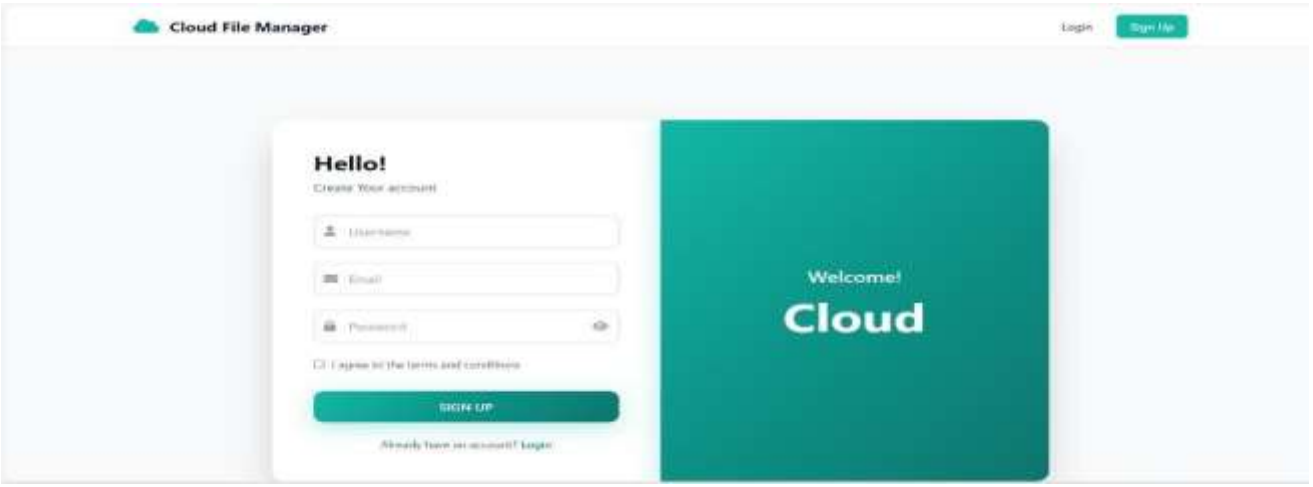


Fig 3 : Sign up Page Interface

2. Dashboard and Home Interface:

● Once authenticated, users are directed to the Home page and File Manager section illustrated in Figure 4. This dashboard acts as the central hub for file operations and provides easy navigation to different sections of the application through a sidebar menu. The navigation is organized into categories such as File Manager, Documents, Personal, Work and Settings. This structured layout improves usability and allows users to quickly switch between modules without confusion.

2.1. File Manager:

● The main component in charge of managing user files is the File Manager. It offers options for uploading, viewing, previewing, renaming, downloading, and deleting files all in one interface. Users may easily upload files to the system using either drag-and-drop or manual file selection. The backend creates a temporary, pre-signed URL that enables the file to be uploaded directly to cloud storage when a file upload is started. By lowering server memory consumption and guaranteeing safe file transfer, this method boosts efficiency. The database stores metadata, including the file name, size, upload date, and ownership information, for future retrieval after a successful upload.

● The Active Files section lists the files that have been uploaded, along with pertinent information and action buttons for previewing, downloading, renaming, or deleting the file. The preview feature lets users see supported file types right inside the interface without having to download them. The system updates both the storage location and the relevant metadata record whenever a file is renamed or removed to keep the cloud storage and database in sync. To ensure safety, deleted files are first transferred to a recovery area before being permanently deleted. This design helps to maintain system integrity and avoid unintentional data loss.

Fig 4: File Manager Page

3. Settings and Personalization:

● The Settings module provides users with a range of customization and management options. It is subdivided into three distinct sections: profile settings, themes and appearance and system settings. Each subsection enhances user control and personalization, while ensuring that security and transparency are maintained.

3.1. Profile Settings:

- Users may change account-related information, such as their name, email address, and profile photo, on the Profile Settings page. When necessary, users may also update their account password. To keep user credentials safe, password changes are handled using secure encryption methods.

Fig 5 : Profile Settings

3.2. Themes and Appearance:

- Users can customize their experience via the Themes and Appearance page, which is shown in Figure 6. Users may select a background wallpaper and switch between light and dark modes. For users who prefer particular color schemes for extended periods, this enhances personal comfort and accessibility.

- The MongoDB Atlas database stores all user choices, making sure that theme and appearance settings are maintained across sessions and devices. These preferences are retrieved by the system and applied automatically when the user logs in again.

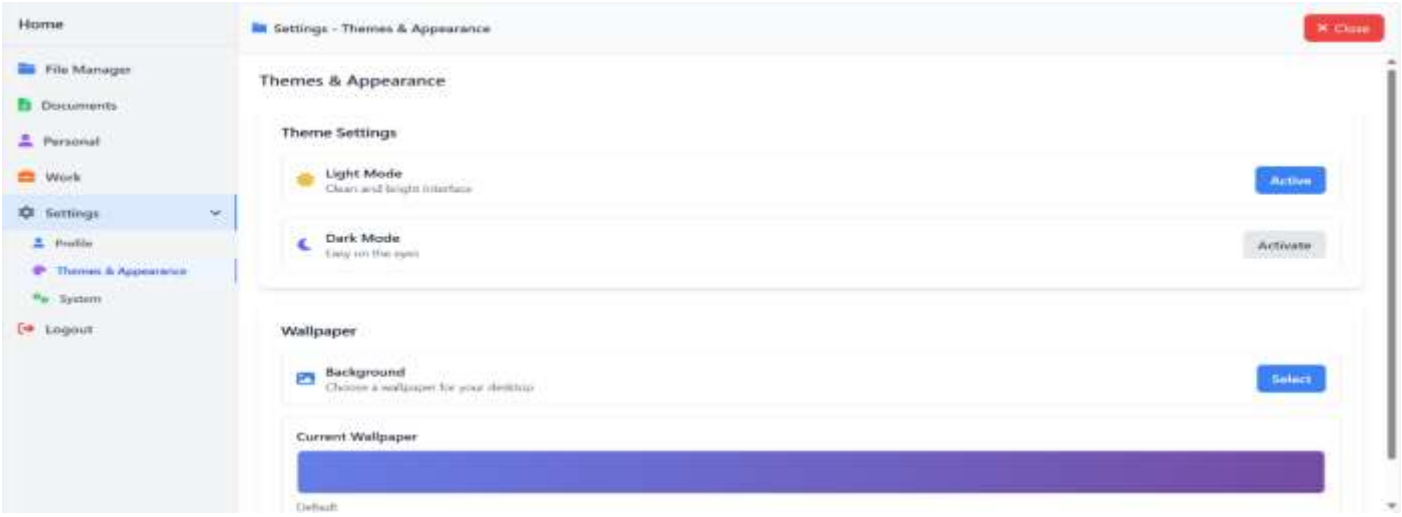


Fig 6 : Settings- Themes and Apperances

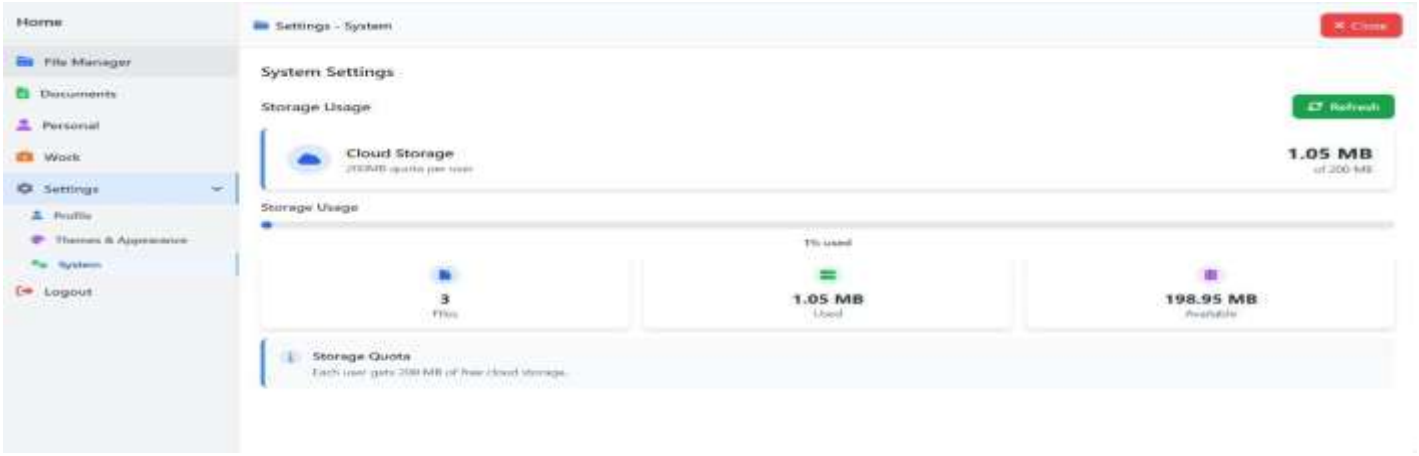
3.3. System Settings:

- Figure 7 shows System Settings page which provides transparency regarding storage allocation and usage. Each user has a quota of 200 MB and the interface displays current usage with both numeric values and a progress bar. Additional details such as the total number of files uploaded, used space and available space are provided for clarity. The refresh option queries the backend for the most up-to-date values from MongoDB

Atlas and AWS S3, ensuring accuracy. This transparency empowers users to monitor their resource usage and manage files accordingly.

Fig 7 : Settings- System Settings

4. Performance Evaluation:



- The performance of the created system was assessed using a variety of operational scenarios, such as file uploads, downloads, authentication procedures, and synchronization operations. According to the findings, file uploads and downloads were executed effectively, with network circumstances having the greatest impact on response times. Data may be transferred safely and quickly between the client interface and cloud storage using pre-signed access links.

- The system additionally showed that the application interface, database data, and cloud storage items could all be synchronized in real time. Changes made to files as they were uploaded, renamed, or deleted were instantly visible on the user interface. During testing, security measures like token-based authentication, encrypted passwords, and secure communication protocols effectively blocked unauthorized access. The system demonstrated overall consistent performance, trustworthy data synchronization, and high security, demonstrating that it may be used as a scalable cloud-based file management system.

Conclusion:

The Personalised Virtual Cloud-Based File Management System exemplifies the effective implementation of a comprehensive, secure, and cloud-hosted file management solution. The initiative aims to overcome the drawbacks of conventional local storage systems, which frequently have restricted scalability, remote accessibility, centralized management, and organized authentication. The system successfully accomplishes its goals of delivering a dependable and efficient cloud-based file management system via meticulous architectural design, modular implementation, and cloud deployment.

With JSON Web Token (JWT) authorization and password hashing, the created platform provides secure user authentication, guaranteeing that only verified users have access to protected file operations. This strategy gets rid of the need for session storage on the server side and improves the system's security. Modern web technologies are integrated to create a responsive and interactive user experience as well as facilitate effective communication with backend services.

For effective data management, the system integrates cloud technology. For scalability and increased performance, binary files are stored in AWS S3 object storage, and metadata is stored in MongoDB Atlas. The division of metadata and file storage results in effective file management and lower database usage. The fundamental functions of file uploading, listing, renaming, deleting, and user personalization were effectively implemented and validated. Pre-signed URLs provide a safe way to transfer files and safeguard storage credentials.

Each piece in the entire system carries out a distinct task, adhering to a modular and manageable structure. By enabling users to access the platform from any internet-connected device, cloud deployment helps realize the objective of a completely cloud-based system as opposed to a locally dependent application. Tests of performance revealed consistent file operations, quick metadata retrieval, and secure authentication procedures.

References:

1. "Virtual File System for Cloud-Based Shared Content" by United States Patent and Trademark Office (U.S. Patent No. 10,114,835 B2, 2018) – This patent describes a virtual file system architecture designed to manage and share cloud-based content efficiently.
2. "Cloud Operating System and Method" by United States Patent and Trademark Office (U.S. Patent No. 10,362,109 B2, 2019) – This patent proposes a cloud operating system framework that enables efficient management of distributed cloud resources and services.
3. "Cloud File System" by United States Patent and Trademark Office (U.S. Patent Application No. US2019/0044706A1, 2019) – This application presents a file system architecture specifically designed for storing and accessing files within cloud environments.

4. "Cloud File Storage System" by S. B. Patil, S. Patil, et al. (2025) – This research work from Pillai HOC College of Engineering and Technology focuses on the design and implementation of a cloud-based file storage platform for secure file management.
5. "Efficient File Sharing System Using MongoDB and Node.js" by V. D. Vinaykumar, V. Pranav, et al. (2024) – Published in the IJCLI Journal, this study proposes a scalable file-sharing system using Node.js and MongoDB to manage file storage and transfer operations.
6. "Toward a Cloud Operating System" by F. Pianese, M. Brunner, et al. (2010) – Presented at an IEEE conference, this research explores the concept of managing cloud resources through an operating system–like architecture.
7. "The NIST Definition of Cloud Computing" by P. Mell and T. Grance (2011) – This NIST publication provides the widely accepted definition and core characteristics of cloud computing technologies.

8. "Cloud Computing: Implementation, Management, and Security" by J. Rittinghouse and J. F. Ransome (2016) – This book discusses practical approaches to implementing and managing cloud computing systems while addressing security challenges.
9. "Mastering Cloud Computing: Foundations and Applications Programming" by R. Buyya, C. Vecchiola, and S. T. Selvi (2013) – This book provides comprehensive insights into the foundations, architecture, and programming models of cloud computing systems.
10. "A View of Cloud Computing" by M. Armbrust, A. Fox, R. Griffith, et al. (2010) – Published in Communications of the ACM, this influential paper presents an overview of cloud computing concepts, challenges, and future research directions.