Chapter 1

INTRODUCTION

Cloud computing has revolutionized the way we store, process, and access data, but with the increasing adoption of cloud services, ensuring the security and integrity of data stored in the cloud has become a pressing concern, and one of the critical challenges in cloud computing is maintaining the privacy and security of data. While allowing for efficient auditing and verification mechanisms, which is where the concept of secure embedded cloud servers comes into play, offering a promising solution for secure data storage and processing. Particularly in applications where security and privacy are paramount, and with the proliferation of Internet of Things (IoT) devices, there's been an explosion in the amount of data being generated, processed, and stored, and traditional cloud computing architectures may not be suitable for IoT applications due to concerns about data security, latency, and bandwidth.

Whereas embedded cloud servers offer a decentralized approach to data processing and storage, where data is processed and stored at the edge of the network, closer to the source of the data, and Raspberry Pi, a low-cost, credit-card-sized computer, has gained popularity in recent years due to its versatility and ease of use, making it an attractive option for building secure embedded cloud servers, and with its small form factor, low power consumption, and affordability.

Raspberry Pi can be used to create a secure embedded cloud server that can store and process sensitive data, providing a high level of security and privacy, and a secure embedded cloud server built using Raspberry Pi can incorporate various security features to ensure the confidentiality, integrity, and availability of data, including encryption, access control, secure communication protocols, and intrusion detection and prevention systems, and encryption is a critical security feature that ensures data is protected from unauthorized access, and encryption algorithms can be used to encrypt data both in transit and at rest. While access control mechanisms can be implemented to restrict access to authorized users and devices, and secure communication protocols like Secure Sockets Layer/Transport Layer Security (SSL/TLS) can be used to secure data in transit, and intrusion detection and prevention systems can be used to detect and prevent potential security threats, and a secure embedded cloud server built using Raspberry Pi offers several benefits, including low cost, flexibility, security, and ease of use, making it suitable for various applications, from IoT to industrial automation, and while Raspberry Pi has some limitations, like performance and storage, it can still provide a high level of security and privacy and with proper configuration and security measures.

Raspberry Pi can be used to build a secure embedded cloud server that meets the needs of various applications, and it's essential to consider the challenges and limitations associated with Raspberry Pi and ensure that the device is properly configured and secured to prevent potential security risks, and by doing so, we can harness the power of Raspberry Pi to create secure and efficient embedded cloud servers that can store and process sensitive data, providing a high level of security and privacy, and meeting the needs of 1 Cloud server using Raspberry Pi various applications, and as technology continues to evolve, it's likely that we'll see even more innovative uses of Raspberry Pi and other embedded devices in the field of cloud computing and IoT, and by leveraging these technologies, we can create more secure, efficient, and scalable solutions for data storage and processing, and ultimately drive innovation and progress in various industries and fields.

1.1 Present Systems

- **Box (for Business/Enterprise):** Noted for its strong enterprise-grade security, compliance protocols, and integrations.
- **Microsoft OneDrive/SharePoint**: Seamlessly integrates with the Microsoft 365 ecosystem, offering features like the "Personal Vault" for highly sensitive documents.
- **Google Drive (for Business):** Offers excellent collaboration features and integrates well with Google Workspace apps.
- **File Cloud:** Offers strong security with options for self-hosting (on-premise) or public cloud storage with data residency controls.
- **Dropbox:** Is a cloud storage service that saves your files online, allowing you to access and sync them across different devices via the

1.2 Problem Definition

- **High cost**: Most of the cloud server offer limited free storage. If user need more space, they should pay monthly or yearly the cost is high.
- **Privacy concerns:** If in case data is stored in third party cloud, user may loss the data or data may corrupted. Users lose full control over it.

- **Security issues**: Even commercial cloud companies invest heavily in security; no system is 100% safe. They have all the access so it can be hacked or leak of important information.
- **Limited customization**: Users cannot change or customize how these cloud services work. they must use the features and interface provided by the company this can be a problem for advanced users or developers who want more control.

1.3 Proposed System

- A Secure Embedded Cloud System is proposed using Raspberry Pi.
- The storage server is fully owned and managed by the user, not by third-party providers.
- The system uses Next cloud, an open-source self-hosted cloud platform.
- It is deployed on the Raspberry Pi, making it a low-cost and energy-efficient solution.
- The system allows storing a large amount of data securely.
- Users can create multiple user accounts.
- It supports different access permissions based on user roles.
- Data is stored locally and transmitted through encrypted channels to ensure security.
- Ensures high levels of security, confidentiality, and privacy.
- The setup is cost-effective, scalable, and secure.
- Provides a viable alternative to commercial cloud storage services.

1.4 Objectives

1. To create a low cost and energy-efficient cloud server using Raspberry Pi.
2. To create safe and secure cloud server for personal use and fully controlled by users.
3. To enable remote access and file management.
4. To create open-source cloud server.
5. To handle real-time data storage.

Chapter 2

LITERATURE REVIEW

According to S Prasath kumar1, P Rayavel , N Anbarasi , B Renukadevi and D Maalini, in the growing digital world, Cloud Storage plays a major role. Cloud Storage is preferred for storing enormous amounts of data that our system cannot hold. But the cost of using private Cloud Storage services will be costly and our data is under the control of others. The main objective of our idea is that our data will be within ourselves rather than sending the data to unknown Cloud Storage providers and which is also cost effective. Here they are using the hard disks as our own Cloud Storage which will be controlled by raspberry pi and it can be accessed from anywhere. It is also more secure of using the material cloud with ourselves rather than depending on a private Cloud Storage service which may or may not be secure but here our data is only with ourselves as we will implement our suitable security measures. The design of our project is also much compact and portable as it gives a major advantage, we can carry it anywhere. The main motto of this project is to give a user personal Cloud Storage of his own in a cost-efficient manner which will be much more reliable than any other material cloud services [1].

According to S Emima Princy, Mr K Gerard Joe Nigel, Private cloud server can be set up in a Raspberry Pi which could be used as a storage device for applications involving real time signals. Raspberry Pi is a cheaper microprocessor in which cloud computing infrastructure can be obtained using cloud platforms provided by specific cloud vendors. Real time signals acquired by any sensor that measures environmental factors are analog in nature. Using microcontrollers like Arduino or any analog to digital converters these analog signals can be discretized and transmitted serially to the Raspberry Pi. Hence Raspberry Pi can be used as a cloud server which serves as a storage device for real time application. This can be extended to critical applications for example in the medical field where data about a patient in the form of real time signals can be recorded and stored in the cloud server and could be accessed by the doctor from a different part altogether. Raspberry Pi consist of OwnCloud is a free and open-source software. It operated similar to that of the Drop-box or google drive that is widely used for storing files for personal use syncing with the own files on a own server or website [2].

According to S Naga Jyothi, K Vijaya Vardhan, this paper presents an IoT-based real-time security surveillance system that uses motion detection algorithms on a Raspberry Pi and

Pi camera to live stream video, detect moving objects, send alerts, and store footage locally or in the cloud, with remote camera control via an IoT webpage. Key features include motion detection it alerts and records footage only when motion is detected. In cloud storage Footage is sent to the cloud, with local storage as a backup. Remote camera control user can control camera movement via IoT webpage. Cost-effective it uses low-processing power chip and reduces storage needs. This system aims to address limitations of traditional video surveillance systems, such as high storage costs and poor stability [3].

According to Mr T J Salma, in this we investigate the problem of data security in cloud data storage, which is essentially a distributed storage system. To achieve the assurances of cloud data integrity and availability and enforce the quality of dependable cloud storage service for users, we propose an effective and flexible distributed scheme with explicit dynamic data support, including block update, delete, and append. We rely on erasure correcting code in the file distribution preparation to provide redundancy parity vectors and guarantee the data dependability. Here they also provide the extension of the proposed main scheme to support third-party auditing, where users can safely delegate the integrity checking tasks to third-party auditors and be worry-free to use the cloud storage services. Through detailed security and extensive experiment results, we show that our scheme is highly efficient and resilient to Byzantine failure, malicious data modification attack, and even server colluding attacks. It has advantages like relief of the burden for storage management, universal data access with independent geographical locations, and avoidance of expenditure on hardware, software, and personnel maintenances. TPA performs efficiently the multiple auditing process in a batch manner, increases the opaqueness of the user data from the auditor, support scalable and efficient public auditing in the cloud computing, provides a privacy-preserving auditing process [4].

According to Somchart fugkew, Narun throne Chinvorant, Traditional steganography is generally based on the way of hiding payloads into a cover image. In this way, the quality of the visibility is subject to the size of cover image and the payload size. Existing works provided the attempts to manage multiple payloads to multiple cover images simultaneously. Steganography is an information hiding method that focuses on embedding information into the medium. This method can provide data confidentiality for sending sensitive data. However, if the attacker knows the pattern of the information hiding scheme, the data confidentiality would not be preserved unlike encryption because the attacker can extract the data from the medium file which is usually presented in the plaintext form. As the demand for big data sharing is highly demanded, the efficient and secure way of sharing stego images to multiple users is challenging. Here, storage as a service is a promising means of providing the efficient and unlimited data storage and high accessibility to the customers. we proposed Crypt Steg which is a crypto steganography system for securing multiple files with fine grained data sharing. This system enables the encrypted payloads to be embedded in multiple images dynamically and they are shared on the cloud with secure access control secrecy. We have proposed a Crypt Steg scheme to deliver a secure and fine-grained data hidden in the image files shared on cloud [5].

According to Lu Liu, Richard Hill, Zhizhun Ding, this paper explores securing Business Intelligence (BI) systems on cloud platforms, addressing significant security challenges due to sensitive business data. Two simulated models are unified Threat Management (UTM) and distributed security controls. Results show UTM causes more overheads, while distributed security poses administrative challenges. A hybrid approach combining both models is recommended for optimal BI security on cloud. It solves the problems like BI security challenges by protecting sensitive business data from unauthorized access, It includes two security models first one is UTM and distributed security controls and Hybrid approach combining UTM for network security and distributed controls for application layer security. A security architect for BI may be challenged to implement all possible technical controls at various stages of the BI process. A BI framework will need access controls at the hardware systems, at network systems, at the instances and objects of data marts and data warehouses, at the metadata repositories, at OLAP servers, at the data view systems, at the data presentation layer, at the application services layer, and all layers of authentication cloud computing offers significant computing power and capacity [6].

According to Dr Balajee Maram, Dr Manikanta Srinivas Sesha Sai et.al, the safety in the transmission of medical images cannot be compromised in the digital age, as the information contained is very sensitive. This paper presents a new approach to providing security for medical image transmission using the Color Secret Sharing protocol. This is ensured by the proposed approach of dividing the medical image into many shares, each of which is encoded with independent color information, to ensure that the restoration of the original image can be done only if all the shares cloud server using Raspberry Piare combined. Thus, the risk related to unauthorized access and data breaching is much reduced during the transmission process. The proposed approach for securing medical image transmission employs a Color Secret Sharing (CSS) protocol to enhance data confidentiality during digital transfers. In this method, the original medical image is divided into multiple shares, each encoded with distinct color information [7].

According to Aiman Sultan, Mehmood Hassan et.al, Symmetric Cryptography Based Authentication Protocol for Efficient Smart Object Tracking. Supply chain management systems (SCM) are the most intensive and statistical RFID application for object tracking. A lot of research has been carried out to overcome security issues in the field of online offline object tracking as well as authentication protocols involving RFID technology. The security of the proposed protocol is formally analyzed using the attack model of the automated security testing tool. The proposed scheme outperforms competing schemes based on security. Global Positioning System (GPS) and Wi-Fi are among the best location and object tracking systems used nowadays. Compared to other navigation systems, the GPS is less expensive but it fails to pin down the precise location or position specially if the target is placed in a multi-store building. Lot of research has been carried out to overcome security issues in the field of online offline object tracking as well as authentication protocols involving RFID technology. In this section, some latest and state of the art research is presented covering aforementioned fields[8].

Chapter 3

SYSTEM DESIGN

The system design describes how the Raspberry Pi is used as the main controller to create a secure embedded cloud server. It explains the connection between hardware components, network setup, and web interface that allows users to remotely access, store, and manage data through connected storage devices. Referring to the figure 3.1 Raspberry Pi-based network-attached storage (NAS) system that allows remote access to hard drives via a webpage. Power supply supplies power to the Raspberry Pi, which acts as the central processing unit of the system. Raspberry Pi the core of the system manages communication between connected hard drives and the network. Hosts server software for file sharing and web interface. Hard disk 1 & Hard disk 2 external storage devices connected to the Raspberry Pi used to store and retrieve files (e.g., documents, media). Ethernet connects the Raspberry Pi to the local network (LAN) or internet ensures stable and faster communication for data transfer. Port forwarding configured on the router to allow external devices (outside the LAN) to access the Raspberry Pi's server forwards incoming traffic on specific ports (e.g., HTTP/HTTPS) to the Raspberry Pi. Webpage represents the web-based interface or portal that users can access from anywhere using a browser users can log in to view/download/upload files from the connected hard drives.

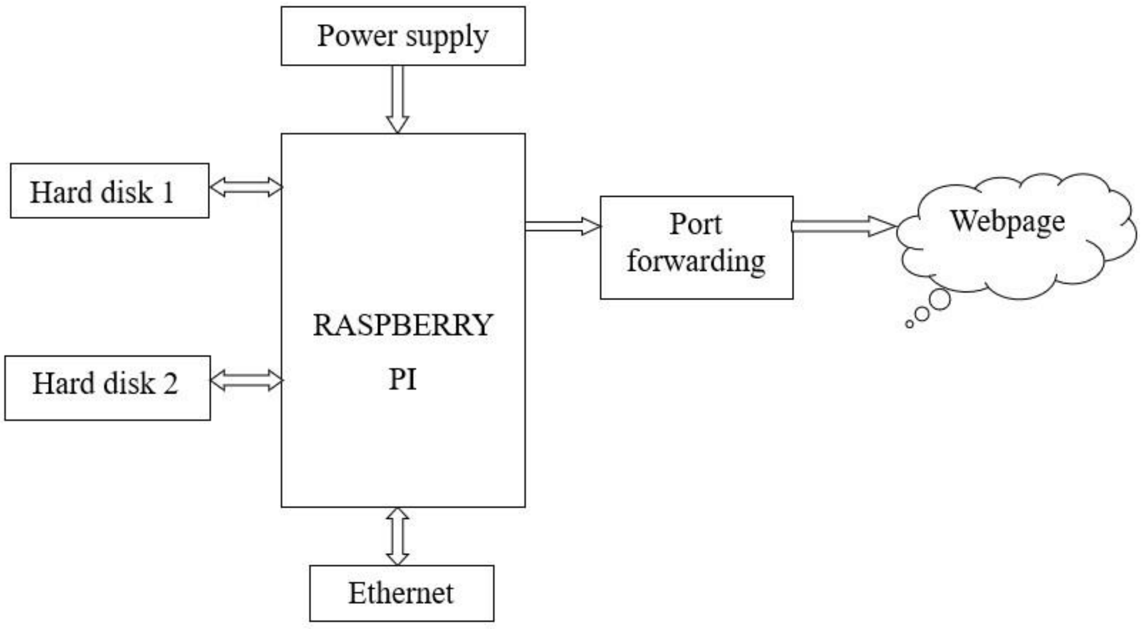


Fig. 3.1: Block diagram of Embedded Cloud Server

3.1 Preliminary System Preparation

Preliminary System Preparation refers to the initial set of activities required before installing and configuring the Embedded Cloud environment. This phase ensures that all hardware, software, network components, and security settings are properly set up to support the smooth deployment and functioning of the cloud server.During this stage, the essential resources are gathered, verified, and configured to create a stable base for the system. Proper preparation reduces installation errors, improves performance, and ensures reliable operation.

- **Requirements Analysis**

The first step in developing an embedded cloud server using a Raspberry Pi was to define the scope and purpose of the project. The aim was to create a self-hosted cloud platform that provides secure file storage, remote access, and data synchronization features like commercial cloud services, but with full control over data privacy. The primary goals included low-cost deployment, ease of maintenance, and energy efficiency, making Raspberry Pi an ideal choice due to its compact size and low power consumption.

- **Hardware Setup**

The selected hardware for this project included a Raspberry Pi 4 with sufficient RAM (preferably 4GB or 8GB), an official power supply to ensure stable performance, and a high-speed microSD card to install the operating system. Additionally, an external USB 3.0 hard drive was used for data storage, allowing for large-capacity and persistent file management. The Raspberry Pi was connected to the internet via Ethernet for improved stability, though Wi-Fi could also be used. For better thermal performance during continuous operation, heat sinks or a cooling fan were installed to regulate temperature.

- **Software Stack Installation**

To manage the cloud services efficiently, a container-based software approach was chosen. The Raspberry Pi operating system was installed in a lightweight version suitable for headless operation. Then, Docker and Docker Compose were installed to manage different software components in isolated containers. This approach ensured modularity and easier updates. The primary cloud service platform selected was EmbeddedCloud, which offers user friendly web access to cloud storage, file sharing, and synchronization. EmbeddedCloud was paired with a database service and other supporting components within Docker, ensuring seamless communication between containers and simplified system maintenance.

3.2 Embedded Cloud Design

The embedded cloud system lets users connect through mobile, web, or desktop apps using a secure HTTPS connection. The web server handles user requests and works with a database and Redis cache to manage data quickly. All files are stored in a storage backend like a local disk external HDD, or S3/MinIO. A backup and monitoring system keeps track of logs, alerts, and snapshots to ensure the system runs safely and smoothly.

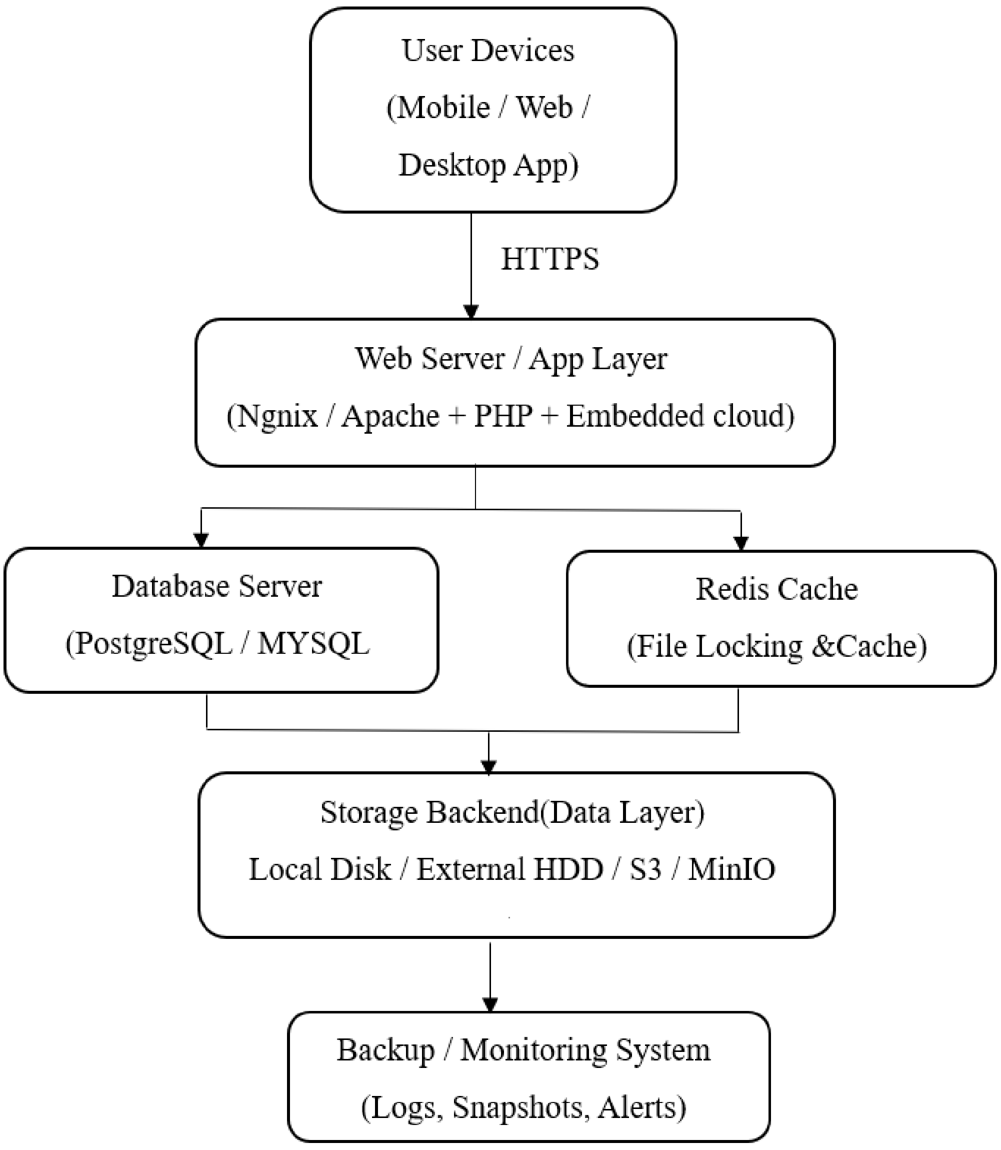

User Devices
(Mobile / Web / Desktop App)
HTTPS
Web Server / App Layer
(Ngnix / Apache + PHP + Embedded cloud)
Database Server
(PostgreSQL / MYSQL
Redis Cache
(File Locking &Cache)
Storage Backend(Data Layer)
Local Disk / External HDD / S3 / MinIO
Backup / Monitoring System
(Logs, Snapshots, Alerts)

Fig.3.2: System Architecture

3.2.1 Architecture Overview

The system architecture of the Secure Embedded Cloud Server is designed to provide a reliable, secure, and efficient platform for storing and accessing data through a Raspberry Pi. The overall workflow begins with user devices such as mobile phones, web browsers, and desktop applications, which communicate with the server over a secure HTTPS channel. All incoming requests are handled by the web server and application layer, where services like Nginx or Apache process HTTP traffic, and backend scripts (PHP/embedded cloud application) manage user authentication, file operations, and application logic. The processed data is then passed to the database server, which stores user credentials and metadata using PostgreSQL or MySQL. To enhance performance, a Redis cache is used for fast access to frequently requested data and file-locking operations. The actual user files are stored in the storage backend, which may include a local disk, external HDD, or network-based storage like S3/MinIO. The entire system is supported by a backup and monitoring module that continuously records logs, generates snapshots, and sends alerts to maintain security, integrity, and system stability.

3.2.2 Component Description

The architecture of the secure embedded cloud server consists of several interconnected components, each performing a specific role to ensure efficient, secure, and reliable operation. Understanding these components is essential to see how user requests flow through the system and how different backend modules work together to provide services such as authentication, file storage, caching, and monitoring. The following section describes each component in detail, explaining its purpose, functionality, and contribution to the overall cloud server workflow.

1. User Devices (Mobile / Web / Desktop App)

As shown in Fig. 3.2, the architecture begins with user devices such as mobile applications, web browsers, and desktop clients. User devices are the entry point for interacting with the cloud platform. These devices may include smartphones, desktop computers, laptops, or tablets. Users access the platform through a web browser, mobile app, or a desktop application, depending on convenience and the features they require. Each device communicates with the cloud system using secure protocols to ensure data privacy and reliability.

On mobile and desktop apps, the experience is optimized for features like automatic file sync, offline access, and background uploads. Meanwhile, browsers offer instant access without installation, making it easy for users who want quick access from anywhere. Regardless of the type of device, all communication with the server happens using HTTPS or WebDAV protocols to maintain end-to-end security.

User devices essentially serve as the interactive layer of the entire system. They generate requests such as uploading files, viewing folders, changing settings, or downloading documents. These requests travel through the network to the web server, which processes them and sends back the appropriate response. Without the user device layer, there would be no interface for real-world interaction with the cloud system.

2. HTTPS / WebDAV API

In Fig. 3.2, all communication between the user devices and the server occurs through HTTPS. HTTPS ensures that all data exchanged between the device and the server is encrypted, protecting sensitive information such as passwords, file contents, and personal details. This encryption prevents unauthorized access even if a hacker intercepts the network traffic.

WebDAV, on the other hand, is a protocol specifically designed for file operations such as uploading, editing, deleting, and syncing files. It allows desktop clients and mobile applications to treat the cloud server like a network drive, enabling faster and more efficient file management. Many cloud platforms use WebDAV because it offers strong compatibility and reliable performance.

Together, HTTPS and WebDAV form a secure communication layer that ensures safe data transfer and proper functionality across all devices. This layer acts as a bridge between user actions and server-side processing, making sure the system runs securely and smoothly.

3. Web Server / App Layer (Nginx / Apache + PHP + Next cloud)

As illustrated in Fig. 3.2, the web server and application layer consist of servers such as Nginx or Apache, integrated with PHP. The web server and application layer form the central processing unit of the cloud platform. The web server typically Apache or Nginx receives incoming requests from users and directs them to the appropriate backend scripts. It handles routing, SSL encryption, static file delivery, and load balancing. Without this component, user requests would not be properly organized or processed.

Inside this layer, PHP executes the main application logic. This includes authentication, permissions, file indexing, share management, and user interface generation. PHP interacts with the database, cache system, and storage backend to produce final output for the user. EmbeddedCloud (or a similar cloud platform) is built on top of PHP and extends these capabilities with additional features like calendar, contacts, sharing links, and collaboration tools.

The web server and app layer essentially act as the "brain" and "controller" of the entire architecture. They determine what needs to be done when a user performs any action and coordinate with other components to complete the operation. They ensure the platform runs consistently, securely, and efficiently.

4. Database Server (PostgreSQL / MySQL)

According to Fig. 3.2, the database server stores essential backend information required by the application. The database server stores all structured, non-file information needed by the system. This includes user accounts, login details, folder structures, file metadata, sharing settings, activity logs, and app configurations. Without a database, the platform would not be able to keep track of user data or manage permissions.

Databases like PostgreSQL and MySQL are used because they are reliable, fast, and widely supported. They offer strong data integrity features, meaning information is stored safely even during unexpected shutdowns or high user activity. These databases can handle complex queries and large volumes of data, making them ideal for cloud storage platforms.

In addition to basic storage, the database server helps maintain consistency in multi-user environments. It ensures that when many users perform operations simultaneously, the data remains accurate and synchronized. This makes the database a crucial backbone of the system.

5. Redis Cache (File Locking & Cache)

In Fig. 3.2, the Redis cache is used to enhance the system’s performance Redis acts as a high-speed in-memory data store that improves overall performance. It stores frequently accessed information temporarily so that the system can retrieve data faster without querying the database repeatedly. This significantly reduces load on the backend and improves response times for users.

Another important function of Redis is file locking. When multiple users try to access or edit the same file, Redis prevents conflicts by locking the file. This ensures that only one person modifies a file at a time, avoiding data corruption or accidental overwrites.

Redis plays a vital role in scaling the cloud system. In environments where hundreds or thousands of users are active, Redis ensures smooth operation by handling sessions, metadata caching, and concurrency control. This makes it essential for maintaining speed and reliability.

6. Storage Backend (Data Layer)

As shown in Fig. 3.2, the storage backend forms the main data layer of the architecture. The storage backend is where actual files documents, photos, videos, backups are stored. While the database stores metadata, the storage backend holds the real content uploaded by users. This storage can be local, such as a hard disk or SSD, or remote, such as Amazon S3, MinIO, or external HDDs.

Local storage is usually simple to set up and cost-effective, making it ideal for small deployments. Remote or cloud-based storage, however, provides scalability and redundancy. Services like S3 and MinIO offer highly durable, distributed storage that protects data even if hardware fails.

The storage layer is designed to be flexible so administrators can expand it as needed. Whether the system needs more space or more reliability, the storage layer can be upgraded independently. This modularity makes the cloud platform scalable and future-proof.

7. Backup / Monitoring System (Logs, Snapshots, Alerts)

Finally, Fig. 3.2 includes a backup and monitoring block responsible for system health and data protection. A backup system ensures that user data is safe even if there is a system failure, accidental deletion, or hardware malfunction. Backups can include file snapshots, user data dumps, or complete system images. These backups protect against data loss and are essential for recovery in emergency situations.

Monitoring systems, such as log analysers or alert tools, continuously track system performance. They help detect issues early, such as high CPU usage, low disk space, failed login attempts, or abnormal network activity. When problems occur, the system can send alerts to administrators for quick intervention.

Together, the backup and monitoring systems enhance reliability and security. They ensure the cloud platform remains stable, secure, and fully operational even under unexpected conditions. Without proper monitoring and backup, the entire system would be vulnerable to failures and data loss.

Table 3.1 Technologies Used in Embedded Cloud Server

Component	**Technology**	**Purpose**
Backend	PHP (main language)	Core application logic
Web server	Apache / Nginx	HTTP(S) request handling
Database	MariaDB / MySQL / PostgreSQL	Stores metadata, users, permissions
Frontend	HTML, CSS, JavaScript, Vue.js	Web interface for users
API/ Layer	WebDAV, REST APIs	Communication between clients and server
Cache/Locking	Redis or Memcached	Speeds up performance and prevents file conflicts
Storage Layer	Local or remote (SSD, MinIO, HDD, etc.)	Stores actual files

Table 3.1 summarizes the key technologies used in the embedded cloud server architecture. Each component plays a specific role in ensuring performance, security, and smooth functioning. A detailed explanation of each component is provided in the subsequent paragraphs.

- **Backend – PHP (Main Application Language)**

According to Table 3.1 the backend of the Embedded cloud server uses PHP as the main programming language. The backend is the heart of the cloud system, and PHP is the primary programming language used to implement its functionality. PHP processes all internal logic such as handling user login requests, storing settings, processing file uploads, and controlling permissions for sharing. Whenever a user interacts with the system—whether renaming a file, creating a folder, or changing a password—the PHP backend interprets the action and performs the necessary operations.

Another important role of PHP is maintaining security and data consistency. It ensures that files are accessed only by authorized users, checks whether the correct permissions are applied, and validates every operation before executing it. Additionally, PHP works closely with the database and storage layer, making sure that every file or user-related action is properly recorded and managed. In simple terms, PHP functions as the "brain" of the entire cloud platform.

- **Web Server – Apache / Nginx**

As shown in the Table 3.1, the webserver commonly uses Apache or Ngnix. The web server acts as the gateway between the user and the backend system. When someone accesses the cloud platform through a browser or mobile device, the web server is the first component that handles the incoming HTTP or HTTPS request. It evaluates the request, determines what type of operation is needed, and then forwards the instructions to the PHP backend for further processing. Apache and Nginx are the two most popular choices due to their performance and reliability.

Beyond routing requests, the web server also plays a major role in security and optimization. It manages SSL certificates to enable secure HTTPS connections, protects the system against unauthorized access attempts, and optimizes delivery of static content like images, CSS, and JavaScript files. In high-traffic environments, Nginx is often preferred because of its ability to handle thousands of simultaneous connections efficiently. Overall, the web server ensures smooth, secure, and fast communication between users and the backend.

- **Database – MariaDB / MySQL / PostgreSQL**

Based on Table 3.1 the database layer relies on MariaDB, MySQL or PostgreSQL. The database stores all structural information required for the cloud server to function, except for the actual file data. This includes user accounts, file metadata, activity logs, link-sharing details, access permissions, and folder structures. When a user uploads a file, the file itself goes to the storage layer, while its name, size, and ownership details are stored in the database. This separation allows the system to function efficiently and remain organized.

Databases like MariaDB, MySQL, and PostgreSQL are chosen because they offer high speed, reliability, and support for complex queries. They ensure that user information is stored securely and can be retrieved quickly when required. The database also helps maintain consistency, even when many users are performing operations at the same time.

By indexing data and optimizing queries, these relational databases contribute significantly to the overall performance of the cloud platform.

- **Frontend – HTML, CSS, JavaScript, Vue.js**

In Table 3.1, the frontend technologies include HTML, CSS, JavaScript, and Vue.js. The frontend is the part of the system that users directly see and interact with. It is built using HTML for structure, CSS for styling, and JavaScript for interactive features. Modern frameworks like Vue.js are used to make the interface dynamic, allowing actions such as instant file preview, drag-and-drop uploads, and real-time notifications. This combination creates a smooth and user-friendly experience.

In addition to appearance, the frontend communicates constantly with the backend using APIs. When a user clicks a button or performs an action, the frontend sends requests to the backend and updates the interface based on the response. Vue.js helps create a responsive and fast UI that adjusts instantly without reloading the entire page. This improves performance and provides users with a modern, app-like experience inside their browser.

- **API Layer – WebDAV and REST APIs**

As detailed in Table 3.1, the API layer uses WebDAV and REST APIs to facilitate communication between the client devices and the server.The API layer acts as a communication channel between the cloud server and external clients such as desktop sync apps, mobile apps, or third-party software. WebDAV is specifically used for file operations—like uploading, downloading, renaming, or deleting files—while REST APIs handle more general tasks such as fetching user data or managing settings. These APIs ensure that the cloud platform is not limited to just a web interface.

Because of these APIs, the cloud platform can integrate with many different systems. For example, a user can sync files across multiple devices, or organizations can connect external applications like document editors, backup tools, or authentication systems. The API layer ensures reliability and consistency across all devices, making the cloud server a versatile and expandable platform.

- **Cache / Locking – Redis or Memcached**

According to Table 3.1, caching and locking are handled using Redis or Memcached. Caching tools like Redis and Memcached store frequently accessed data temporarily in memory, speeding up responses and reducing the load on the database. For example, when many users repeatedly access the same file information, the system can quickly retrieve it from the cache instead of querying the database multiple times. This improves overall performance, especially in large deployments with many users.

Redis also handles file locking, which prevents multiple users from editing the same document at the same time. Without locking, two people saving changes simultaneously could corrupt the file or cause data loss. By managing locks efficiently, Redis ensures safe collaboration and prevents conflicts. As a result, caching and locking improve both the speed and reliability of the cloud platform.

- **Storage Layer – Local or Remote (SDD, MinIO, HDD, etc.)**

As listed in Table 3.1, the storage layer uses local or remote devices such as SSDs, HDDs, or MinIO.The storage layer is where the actual files documents, photos, videos, and backups are saved. Unlike the database, which stores metadata, this layer handles file content. Storage can be configured in many ways: a local hard disk, SSD, USB drive, or remote cloud storage such as Amazon S3 or MinIO. This flexibility allows the system to scale based on user needs.

Using external object storage services like S3 or MinIO provides high durability and almost unlimited storage capacity. On the other hand, local storage is cheaper and ideal for small personal or office setups. The system is designed in such a way that administrators can change or expand the storage without affecting other components. This modular design makes the cloud platform adaptable and future-proof.

3.3 User interaction and Embedded Cloud System Workflow

Figure 3.3 shows the overall workflow of how users interact with the secure embedded cloud server and how different backend components support these operations. Users access the system through client devices such as mobile phones, web applications, or desktop clients to upload or synchronize files, view or download stored data, and manage sharing permissions.

These user actions are processed by the embedded cloud running on the Raspberry Pi, which integrates the application logic, database handling, and storage management. To support efficient and secure operations, the server communicates with external components such as a database for storing user and authentication details, a storage backend for maintaining actual user files on local disks or S3-compatible storage, and a Redis cache system for fast data access and file-locking. Together, these components ensure a smooth, reliable, and secure cloud service workflow.

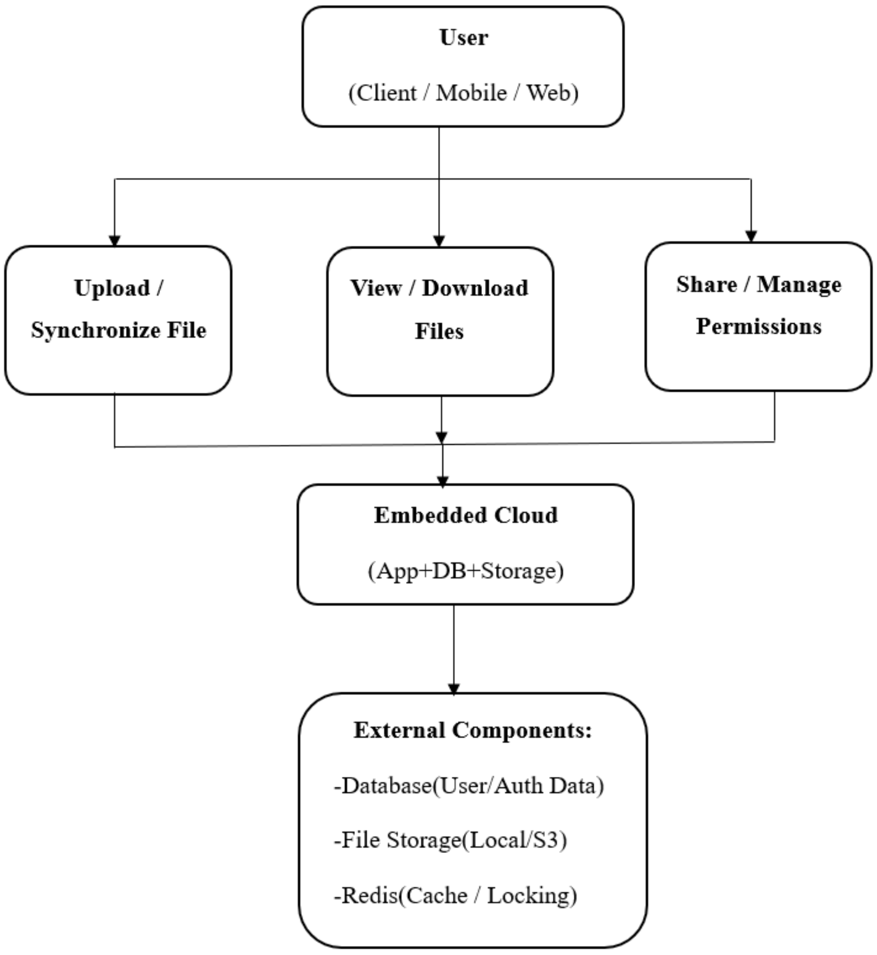


Fig 3.3: Use case Diagram

To understand the functioning of the secure embedded cloud server more clearly, each block shown in Figure 3.3 is described in detail below. The following section explains the role and purpose of every component in the architecture, outlining how user devices, the embedded cloud system, and external modules work together to deliver secure file storage, synchronization, and access service.

- **User Layer (Client / Mobile / Web)**

The top layer of the architecture represents the User, who interacts with the system through different interfaces such as a web browser, mobile application, or desktop client. This layer is responsible for initiating all operations like uploading, viewing, or sharing files. In a cloud-based system like EmbeddedCloud, users may belong to different roles (admin, normal user, guest user), but all interactions start from this layer. The user devices send requests over the network to the server, and based on authentication and permissions, the system provides access to stored files and services. This ensures a seamless and platform-independent experience.

- **Upload / Synchronize File**

This block represents the user's ability to upload new files to the cloud or synchronize already existing files across multiple devices. When a user uploads a file, the system stores it in the backend storage and updates the database with metadata such as file name, size, path, and ownership. Synchronization ensures that if a user changes a file on one device, the same version updates automatically across all connected devices. This feature is essential for maintaining consistency and eliminating version conflicts.

- **View / Download Files**

This block shows the ability of users to view or download files stored in their cloud account. When a user sends a request to view or download a file, the system first verifies permissions and then fetches the file from the storage. The file can be displayed in the user interface (for images, documents, PDFs, etc.) or downloaded directly to the user's device. This functionality forms the core of cloud-based file accessibility and ensures that users can reach their data anytime, anywhere.

- **Share / Manage Permissions**

This function enables collaboration by allowing users to share files or folders with other users. Here, users can set permissions such as read-only, edit, or full access. They can also share files via links with optional password protection, expiry dates, or access restrictions. Permission management ensures secure and controlled file sharing within an organization or among specific individuals. It also prevents unauthorized access by enforcing strict authentication policies.

- **EmbeddedCloud Core (App + DB + Storage)**

The central block in the architecture is the EmbeddedCloud Core, which acts as the brain of the entire system. This layer consists of three major components:

Application Layer: Handles user requests, authentication, interface communication, file operations, sharing features, and overall system logic.

Database Layer: Stores information about users, permissions, file metadata, logs, session details, and configuration settings.

Storage Layer: Manages actual file data stored either on local disks or external storage systems.

The EmbeddedCloud Core ensures proper communication between user actions and backend services. It validates operations, enforces security policies, processes data, interacts with the database, and connects to external components. This block ensures the entire platform functions efficiently and securely.

➢ **External Components**

This is the final block that represents various services supporting the core EmbeddedCloud environment. These components are essential for improving performance, reliability, and scalability.

- Database (User/Auth Data)

  The database stores all crucial user-related information such as login credentials, passwords (hashed), authentication tokens, and session details. It also maintains records of file metadata like file names, directories, timestamps, and ownership information. Secure database handling ensures correct authentication and authorization for every user request.
- File Storage (Local / S3

  This component holds the actual files uploaded by users. The storage can be either local storage (Raspberry Pi / server hard disk), or Cloud object storage like Amazon S3 or other compatible platforms. EmbeddedCloud

  supports scalable storage solutions to handle large amounts of data, ensuring high availability and durability.
- Redis (Cache / Locking)

  Redis is used to improve the performance of the EmbeddedCloud system. It acts as a caching layer to store frequently accessed information, reducing load on the main database. Redis is also used for file locking which prevents conflicts when multiple users edit a file at the same time. This ensures smooth file operations and prevents data corruption.
- Backup / Monitor Services

  These services ensure that the entire system remains secure, stable, and reliable. Backup services help in taking regular copies of user data, preventing data loss in case of system failure. Monitoring tools track system health, CPU usage, storage capacity, network load, and any unusual activity. These external services help maintain the long-term performance and security of the cloud platform.

Chapter 4

System Implementation

The system implementation phase describes the practical steps taken to build and deploy the secure embedded cloud server on the Raspberry Pi platform. This stage focuses on converting the planned design into a working system by configuring the hardware, installing the necessary software stack, integrating the application components, and establishing secure communication between all modules. It involves setting up the operating system, preparing the web server environment, configuring the database and storage backend, and implementing the security measures required for encrypted data transfer and safe user authentication. Overall, this chapter explains how each component was practically developed, connected, and executed to create a fully functional embedded cloud server.

1.1 Installation of Supporting Software Packages

This subsection outlines the step-by-step installation of all necessary software packages required for the functioning of the embedded cloud server, including the web server, application runtime, database, caching system, and security tools.

Step 1 : Prepare the Raspberry Pi

Flash OS (Ubuntu Server 22.04 LTS ARM64 or Raspberry Pi OS Lite) onto SD card / SSD. Boot up Raspberry Pi and connect via SSH or directly with monitor & keyboard. Update and upgrade system:

```
sudo apt update && sudo apt upgrade -y
```

Step 2: Install Essential Tools

- These are required for system management and compilation:

```
sudo apt install -y build-essential git curl wget unzip software-properties-
common apt-transport-https ca-certificates lsb-release gnupg
```

Step 3: Install Web Server

```
sudo apt install -y build-essential git curl wget unzip software-properties-common
apt-transport-https ca-certificates lsb-release gnupg
```

- Apache2

```
sudo apt install -y apache2
```

Step 4: Install Database Server

- MariaDB (lightweight, good for Raspberry Pi):

```
sudo apt install -y mariadb-server mariadb-client
```

- Secure database setup

```
sudo mysql_secure_installation
```

Step 5: Install PHP and Extensions

- Nextcloud (and most cloud frameworks) require PHP + extensions:

```
sudo apt install -y php php-cli php-fpm php-mysql php-gd php-curl php-xml php-
mbstring php-zip php-intl php-bcmath php-gmp
```

Step 6: Install Redis (for caching)

- Improves performance of embedded cloud apps:

```
sudo apt install -y redis-server php-redis
```

Step 7: Install Supporting Tools for Nextcloud

- ZIP/Unzip and Image libraries

```
sudo apt install -y unzip bzip2 rsync imagemagick ffmpeg
```

Step 8: Set Permissions & Users

- Create a dedicated user (e.g., Embedded Cloud) to manage the server.
- Ensure webserver user (www-data) has correct access to /var/www/nextcloud.

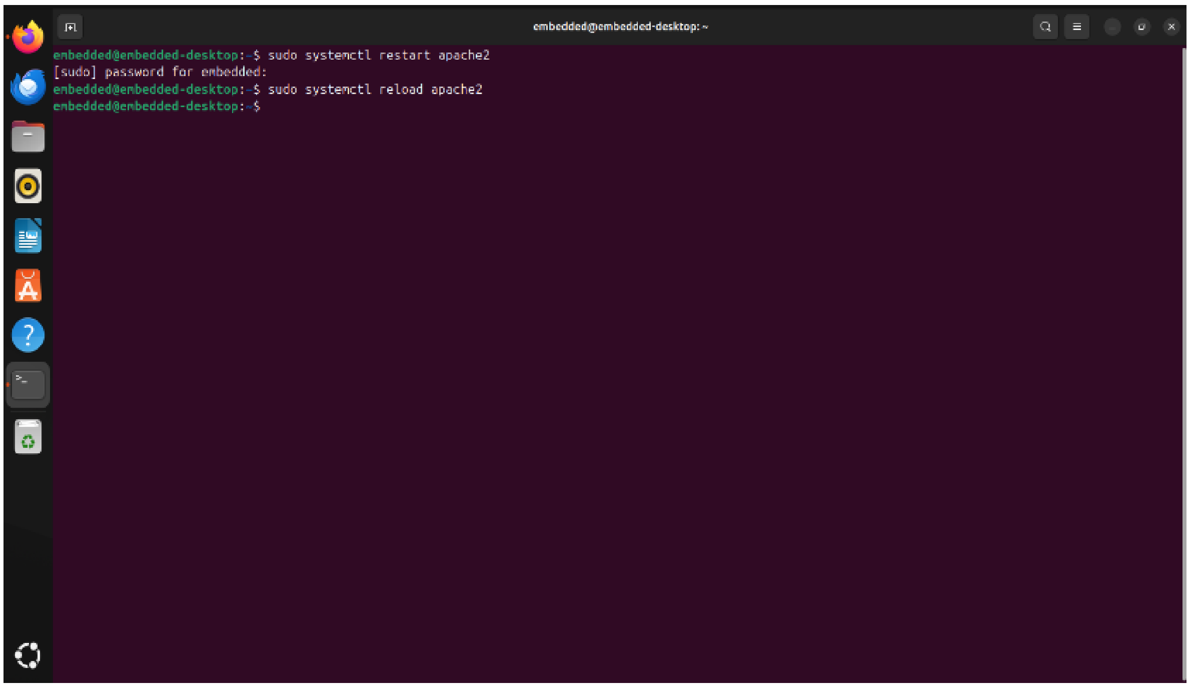
Step 9: Download and Deploy Cloud Software

```
cd /var/www/
sudo wget https://download.nextcloud.com/server/releases/latest.zip
sudo unzip latest.zip
sudo chown -R www-data: www-data nextcloud
```

Chapter 5

TEST AND DESIGN

The design and testing phases are essential stages in the development of the embedded cloud-based user management system. The design phase involves planning and structuring the system to define how different components will function and interact. This includes designing the user interface, specifying input fields, organizing user groups such as Faculty, Students, HoD, and Admin, setting storage quotas, and implementing role-based access control. The design provides a clear blueprint that ensures usability, security, and scalability before the system is developed. Once the system design is implemented, the testing phase is conducted to verify and validate that all components work correctly according to the requirements..



```
embedded@embedded-desktop:~$ sudo systemctl restart apache2
[sudo] password for embedded:
embedded@embedded-desktop:~$ sudo systemctl reload apache2
embedded@embedded-desktop:~$
```

Fig.5.1:Before login page

Various test cases are executed such as creating new user accounts, assigning groups, validating login functions, checking password handling, and verifying admin privileges. Testing ensures that all features are functioning reliably, free from errors, and ready for real-world deployment. Through systematic design and thorough testing, the system achieves accuracy, stability, and smooth user experience.

Log in to Embedded Cloud:

The image shows a custom login interface for a system named "Embedded Cloud – Personal Embedded Storage." This appears to be an institution-hosted cloud platform, likely used by students or staff for internal file storage, access, and sharing.

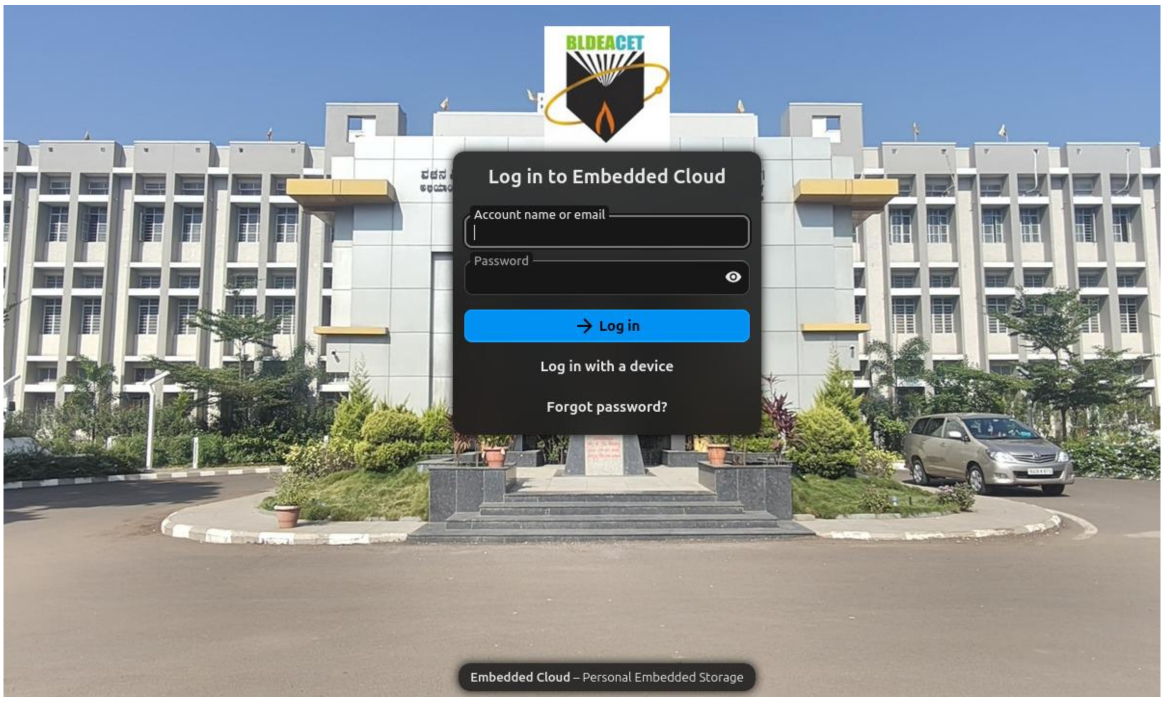


Fig 5.2: Log in to Embedded Cloud

Background Building (Institution Premises)

- The login box is placed over a background image of a institutional building. This suggests the cloud system is hosted by the institution.
- Accessible only to authorized users (students, faculty, staff).
- At the top center of the page, the BLDEA CET logo is displayed, indicating the platform belongs to BLDE Association's College of Engineering & Technology (BLDEACET).

Login Interface (Main Black Box in Center)

This is the main component where the user enters credentials.

Header

It clearly indicates the purpose of the page.

Account Field

Account name or email

1. The user can log in using either a username or an email address.
2. Standard text field.

Password Field

1. Users enter their password.
2. There is an **eye icon** to toggle visibility of password.

Log In Button (Blue Button)

- A rectangular button labelled → **Log in**
- Clicking this verifies credentials and logs the user into their personal cloud dashboard.
- There are two extra options:

  Log in with a device – for alternative login methods.

  Forgot password? – to reset your password.

➢ At the bottom, it says "Embedded Cloud – Personal Embedded Storage", which means this platform is your college's private storage system where students or staff can save files, notes, and documents online.

Dashboard of Embedded Cloud

The figure shows the main dashboard page of a cloud-based file management system. When a user logs in through a web browser, they are redirected to this dashboard, which provides a quick overview of their files and recent activities. The interface is designed to help users access their stored data easily and view personalized information at a glance.

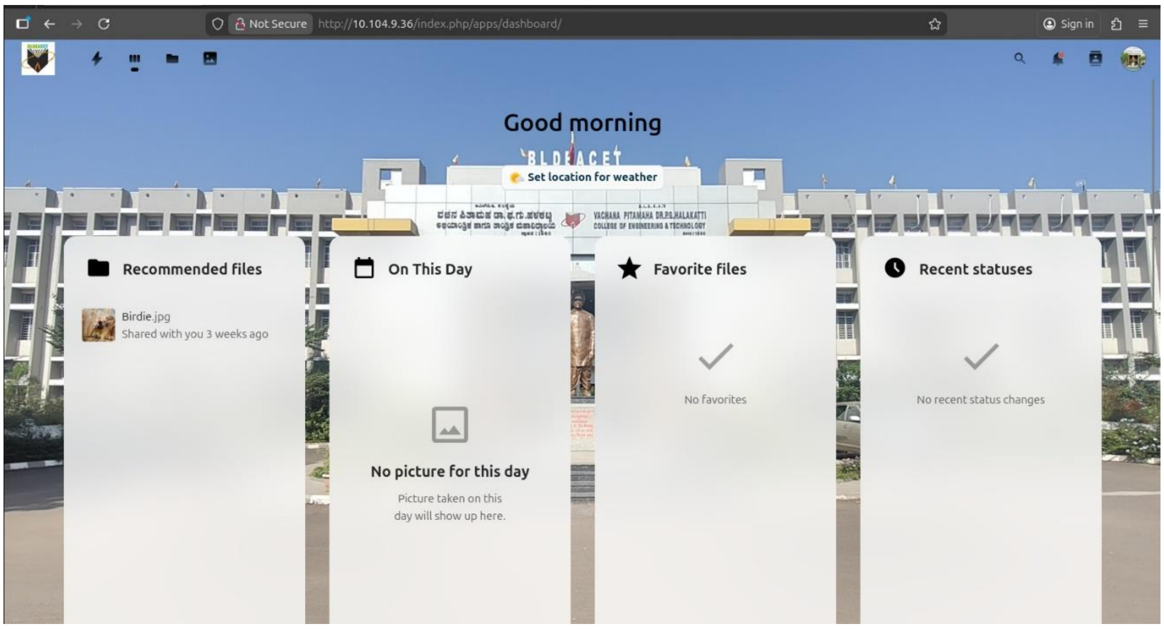


Fig 5.3: Dashboard of Embedded Cloud

At the top of the page, the system displays a greeting message such as **"Good morning"**, along with an option to set the user's location for weather updates. This indicates that the dashboard is personalized based on the user's profile and preferences. The background image appears to be of an institutional building, which suggests that the cloud system might be deployed in a college or organizational environment.

Below the header, the dashboard is divided into four main sections:

- **Recommended files**

This section shows files automatically suggested for the user based on recent activity or shared content. For example, in the image, a file named *"Birdie.jpg"* appears, which was shared three weeks ago. This helps users quickly access important or frequently used files.

- **On this day**

This section highlights photos or files uploaded on the same date in previous years. If there are no files from this day, the system displays a message like "No picture for this day." This feature is useful for reminding users of old photos or documents.

- **Favorite files**

Users can mark important documents as "favorites." Those selected files will appear in this section for fast and easy access. In the current dashboard, no files have been marked as favorite, so it shows "No favorites."

- **Recent statuses**

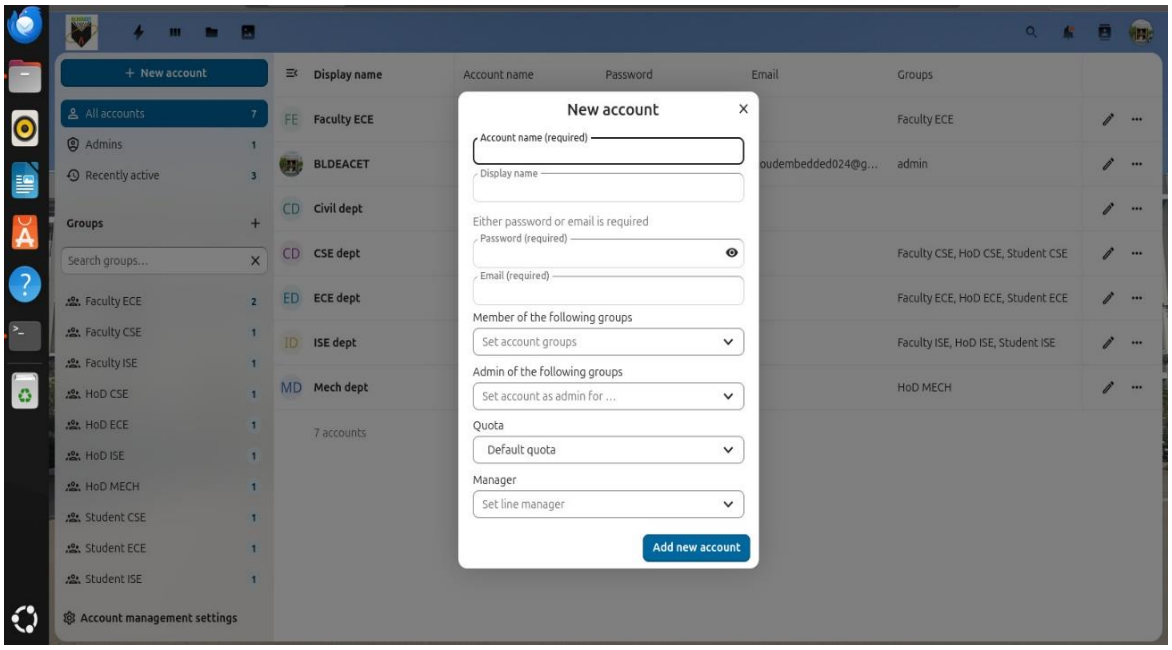
This panel shows recent changes in the user's storage space, such as uploaded files, shared documents, or updates made by other users. Since there are no recent activities, it displays "No recent status changes."

At the top-right corner of the interface, there are options for **search**, **notifications**, **user settings**, and **profile management**. On the left side, icons are present for navigating to different modules like files, photos, settings, or other applications within the cloud platform.

User Account Creation

Figure 5.4 shows the User Account Management interface of the EmbeddedCloud cloud server, which is used by the system administrator to create and manage user accounts.

This interface allows the admin to add new users by specifying details such as account name, display name, password or email, group membership, and assigned storage quota.



All accounts	7
Admins	1
Recently active	3
Faculty ECE	2
Faculty CSE	1
Faculty ISE	1
HoD CSE	1
HoD ECE	1
HoD ISE	1
HoD MECH	1
Student CSE	1
Student ECE	1
Student ISE	1

Display name	Account name	Password	Email	Groups
FE Faculty ECE				Faculty ECE
BLDEACET			oudembedded024@g...	admin
CD Civil dept				
CD CSE dept				Faculty CSE, HoD CSE, Student CSE
ED ECE dept				Faculty ECE, HoD ECE, Student ECE
ID ISE dept				Faculty ISE, HoD ISE, Student ISE
MD Mech dept				HoD MECH

7 accounts

New account

Account name (required)

Display name

Either password or email is required

Password (required)

Email (required)

Member of the following groups: Set account groups

Admin of the following groups: Set account as admin for ...

Quota: Default quota

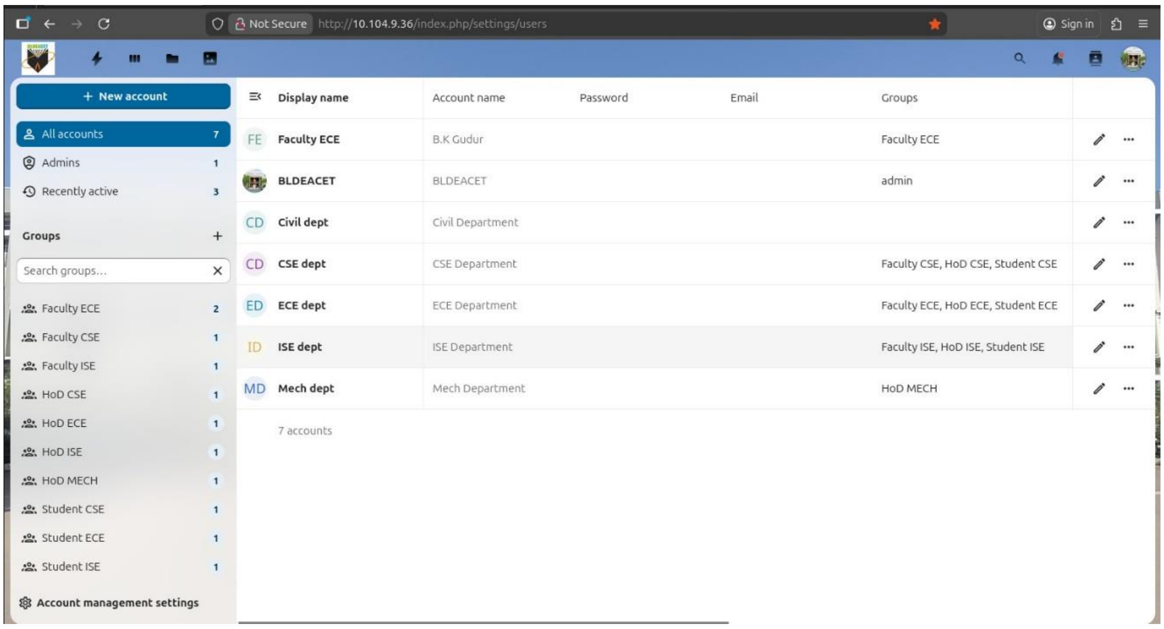
Manager: Set line manager

Fig 5.4: Account Creation of user.

Users can be organized into different groups such as Faculty, Students, and Heads of Departments (HoD), which helps in managing access permissions and resource allocation efficiently. The left panel displays all available groups and their members, while the right panel shows the list of existing accounts along with their associated groups and management options. The pop-up window at the center provides the form for creating a new account, where the administrator can also assign group admin roles and set a manager if required. This interface enables structured and secure management of user roles in the embedded cloud environment, making it suitable for institutional cloud deployments.

User Management Page

The fig 5.5 shows the User Management page of a cloud platform interface. This page is used by an administrator to manage all user accounts, their details, and the groups they belong to. The web page is being accessed through a local network IP (10.104.9.36), meaning the cloud system is hosted internally, not on a public server. In the center of the screen, there is a list of accounts. Each account has details such as the display name, account name, email, and the groups it belongs to. These accounts mostly represent departments like ECE, CSE, Mechanical, Civil, and so on. There are also admin accounts. The system shows each account neatly in a table format.



Display name	Account name	Password	Email	Groups
FE Faculty ECE	B.K Gudur			Faculty ECE
BLDEACET	BLDEACET			admin
CD Civil dept	Civil Department			
CD CSE dept	CSE Department			Faculty CSE, HoD CSE, Student CSE
ED ECE dept	ECE Department			Faculty ECE, HoD ECE, Student ECE
ID ISE dept	ISE Department			Faculty ISE, HoD ISE, Student ISE
MD Mech dept	Mech Department			HoD MECH

7 accounts

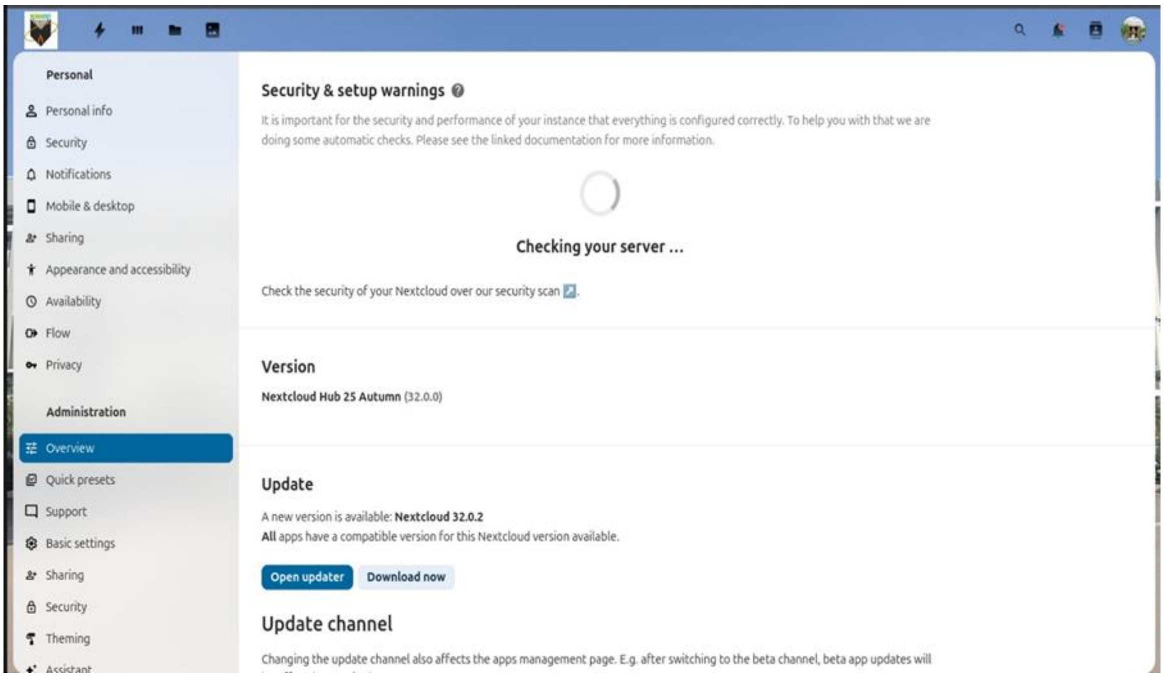
Fig.5.5: User Management Page.

On the left side, there is a section called "Groups." These groups help to categorize users based on their role, such as Faculty, HoD (Head of Department), and Students from different branches. Each group also shows how many users belong to it, making it easier to manage the organization structure.

There is also a "+ New account" button at the top left, which lets the admin create new users. On the right side of each account row, there are icons for editing or opening more options. These controls help the admin update or change account details whenever needed. Overall, the page helps the administrator maintain order in the system by managing accounts and arranging them into the correct departments or roles. It ensures that every user has the right permissions and is part of the correct group.

Security And Setup Warning

The Embedded Cloud system includes several layers of security to protect user data and prevent unauthorized access as shown in figure 5.6. The platform uses user authentication, where every user must log in with a valid username and password before accessing files or services. Passwords are hashed in the database, which means they are stored in an encrypted format to prevent misuse even if the database is compromised.



Security & setup warnings

It is important for the security and performance of your instance that everything is configured correctly. To help you with that we are doing some automatic checks. Please see the linked documentation for more information.

Checking your server ...

Check the security of your Nextcloud over our security scan.

Version

Nextcloud Hub 25 Autumn (32.0.0)

Update

A new version is available: Nextcloud 32.0.2
All apps have a compatible version for this Nextcloud version available.

Update channel

Changing the update channel also affects the apps management page. E.g. after switching to the beta channel, beta app updates will

Fig 5.6: Security and Setup Warning

The system also uses role-based access control, ensuring that normal users cannot perform administrative operations such as creating databases, modifying system settings, or accessing other users' storage spaces. In addition, secure protocols such as HTTPS are used to encrypt all communication between the client and the server, preventing data interception or man-in-the-middle attacks. File permissions and directory-level security are enforced on the storage backend so that each user's files remain isolated and accessible only to authorized accounts. The cloud also implements session management, where Redis is used to store and validate session tokens, reducing the risk of session hijacking. Audit logs maintain a record of all critical activities, enabling administrators to trace suspicious actions and ensure accountability.

File uploading

The fig 5.6 shows the main file management dashboard of a cloud storage system. The user is currently inside the "All files" section. When the user clicks on the "New" button at the top, a dropdown menu appears displaying several options related to file upload and creation.

1. **Navigation Panel (Left Side):**

- Options such as All files, Personal files, Recent, Favorites, and Shares.
- Additional sections like Deleted files, File requests, Pending shares, and Tags.
- This allows users to quickly move between different storage categories.

2. **Main Workspace Area:**

- Shows folders such as Documents, Photos, Templates.
- Some individual image files are also displayed, each with a thumbnail preview.

3. **New Button Dropdown:** The menu provides the following actions:

- Upload files
- Upload folders
- Create new folder
- Create file request
- New text file

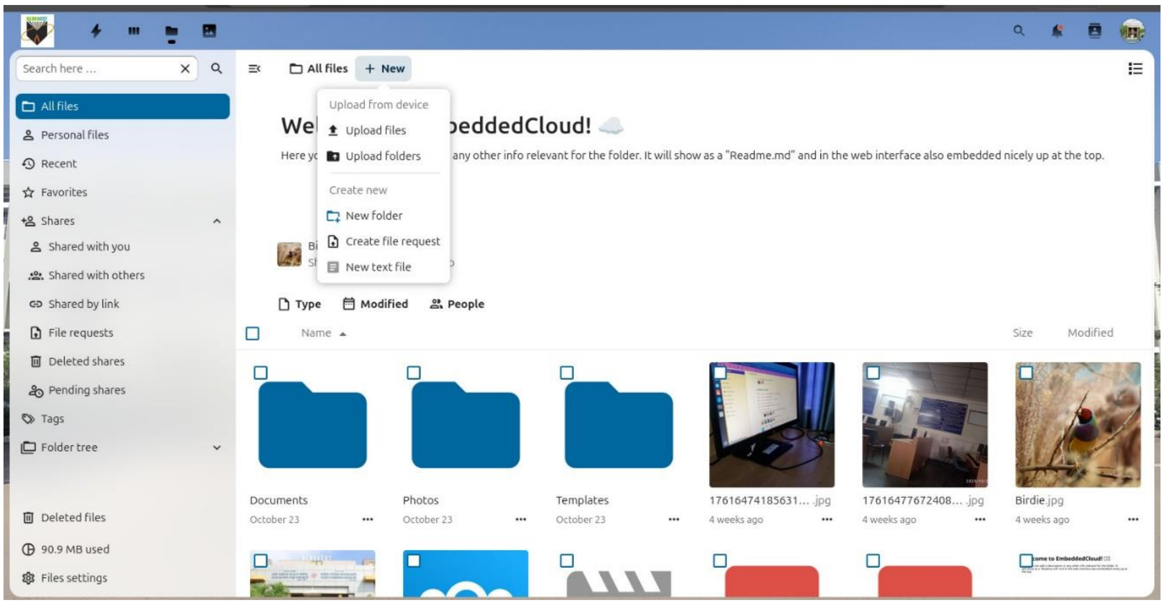


Fig 5.6: Uploading File

File Sharing Settings and Permission Configuration

The second image focuses on the file sharing panel for a selected file. The right side of the interface displays the sharing settings for an image file selected from the main workspace. This interface allows users to control who can view, edit, download, or further share the file. The screenshot demonstrates detailed permission customization in a cloud storage system, ensuring secure and role-based access.

1. Selected File Details Panel (Right Side):

- Shows file name, size, and modification date.
- Displays sharing tab with active sharing settings.

2. Sharing Options:

The user has shared the file with a group or person (e.g., “ECE dept”). The panel offers various sharing permissions:

- View only.
- Allow editing.
- Custom permissions

3. Custom Permissions Breakdown:

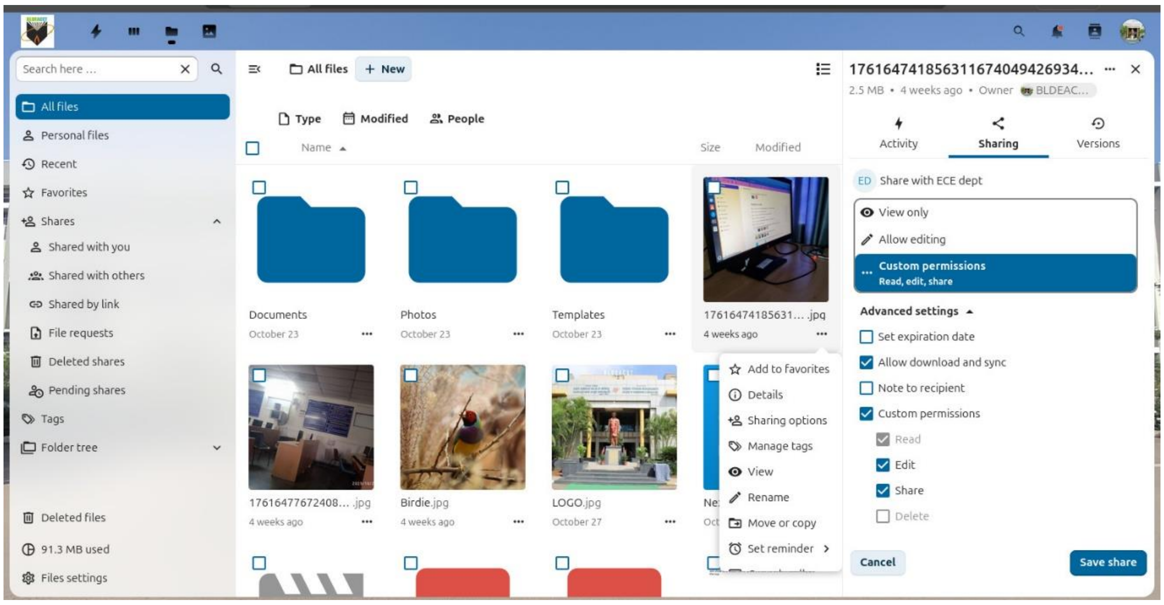
- Read
- Edit
- Share

4. Advanced Settings:

- Set expiration date.
- Allow download and sync.
- Add note to recipient.
- Enable/disable custom permissions.

5. Save Share Button:

Allows the user to finalize the changes to access rights.


176164741856311674049426934...
2.5 MB • 4 weeks ago • Owner BLDEAC...

Share with ECE dept

- View only
- Allow editing
- Custom permissions — Read, edit, share

Advanced settings

- [ ] Set expiration date
- [✓] Allow download and sync
- [ ] Note to recipient
- [✓] Custom permissions
  - [✓] Read
  - [✓] Edit
  - [✓] Share
  - [ ] Delete

Fig 5.7: File Sharing Settings and Permission Configuration

Modules and Description

1. Raspberry pi and peripherals: The Raspberry pi acts as the core embedded system depending on the use case peripherals like sensors, cameras or actuators may be connected.
2. Operating system layer lightweight OS: Here Linux based OS providing the foundation for service ensuring low overhead and good control over security configurations.
3. Networking and communication secure network stack: It handles secure data transmission and device communication supporting protocols like MQTT, HTTPS and SSH.
4. Cloud interface module: It allows a Raspberry pi to communicate with or act as a cloud server. Integration with cloud providers like AWS, Azure IoT or local private clouds(e.g., next cloud)
5. Security and access control: Implements authentication, encryption, and intrusion detection mechanisms.
6. Custom application logic: Implements the main purpose of the embedded cloud, such as home automation, environmental monitoring. written in Python/ C++/Node.js, interfaces with sensors or cloud API, runs as a background service.
7. User interface web or mobile dashboard Allows user to interact with system visually for monitoring and control. Web dashboard using flash+ bootstrap, mobile app via HTTP API or MQTT.

ADVANTAGES AND APPICATIONS

Advantages

- All the data can be stored in one place.
- It provides privacy-preserving auditing process.
- Data can be access by all, independent of location with internet.
- It saves the cost like not to buy external storage hardware and software of storage(cost-saving).
- Cloud system can handle automatic backups and disaster recovery capabilities.
- In this the data loss risk will be less.
- In this Raspberry Pi's consume low energy so, it makes server always on.

Applications:

- Personal cloud storage.
- Small business file server.
- Educational purposes.
- Home automation and IoT.
- Backup server.
- System administration and data security

CONCLUSION

The implementation of a secure embedded cloud using a Raspberry Pi demonstrates how powerful and flexible low-cost hardware can be in modern computing environments. By integrating lightweight operating systems, secure communication protocols, and encrypted data management, the system offers both reliability and security for cloud-based services at the edge. With proper configuration, a Raspberry Pi can serve as a miniature cloud server or gateway, capable of managing data locally and interacting securely with public or private cloud services. This approach is especially useful in IoT, smart homes, and remote monitoring applications where cost, size, and power efficiency are critical. Overall, the project proves that embedded cloud systems can be both secure and scalable when designed with a focus on modularity, encryption, and remote management.

REFERENCES

[1] S Prasath kumar1, P Rayavel, N Anbarasi ,B Renukadevi and D Maalini , “Raspberry pi based secured cloud data”

[2] S Emima Princy, Mr K Gerard Joe Nigel,"Implementation Of Cloud Server For Real Time Data Storage Using Raspberry Pi"

[3] S Naga Jyothi, K Vijaya Vardhan,“Design And Implementation Of Real Time Security Surveillance System Using IoT"

[4] Mr T J Salma,"A Fexible Distributed Storage Integrity Auditing Mechanism in Cloud Computing".

[5] Somchart fugkew, Narun throne Chinvorant."A Dynamic and Efficient Crypto-

Steganography System for Securing Multiple Files in Cloud.”

[6] Lu liu, Richard Hill, Zhizhun Ding, “Business Intelligence Security on The Clouds Business Intelligence Security On The Clouds”

[7] Dr Balajee Maram, Dr Manikanta Srinivas Sesha Sai, Dr Malathi Vanniappan, Dr T Adithya Sai Srinivasa, Pamarthi Nagaraju, Dr U D Prasan, “Protecting Medical Image Transmission: Color Secret Sharing Protocol For Enhanced Security”

[8] Aiman Sultan, Mehmood Hassan,Khwaza Mansoor, Syeyes Saddam

Ahmed,“Securing IoT Enabled RFID Based Object Tracking Systems ”