MOBILE PHONE CLIENT APP FOR DECENTRALIZED STORAGE USING SWARM

SYSTEM DESIGN DOCUMENT

<PROJECT NAME>

COMPANY NAME
STREET ADDRESS
CITY, STATE ZIP CODE

DATE

TABLE OF CONTENTS

INTRODUCTION 3
PURPOSE 3
SYSTEM OVERVIEW 3
DESIGN CONSTRAINTS 4
ROLES AND RESPONSIBILITIES 5
PROJECT REFERENCES 5
SYSTEM ARCHITECTURE 6
DATABASE DESIGN 7
HARDWARE AND SOFTWARE DETAILED DESIGN 8
SYSTEM SECURITY AND INTEGRITY CONTROLS 9

INTRODUCTION

This document outlines the design and implementation of a decentralized Distributed Storage System accessible via a mobile application, leveraging the Swarm network for robust, distributed file storage. This system aims to provide a secure, reliable, and user-friendly alternative to traditional cloud storage solutions such as Google Cloud, Dropbox, and Amazon S3.
The core architecture of the system involves the use of multiple "Bee Nodes" to distribute and store data, ensuring high availability and fault tolerance. When a user uploads a file, it is partitioned into multiple segments, each of which is stored redundantly across several Bee Nodes. This approach enhances data security and integrity, making the system resilient to node failures and malicious attacks.

The project builds on the existing Swarm backend infrastructure and the Swarm Desktop Client, extending their functionality to mobile platforms. The primary objective is to develop a smartphone app that enables users to easily upload, manage, and access their files within the Swarm network, ensuring a seamless and intuitive user experience on both Android and iOS devices.

This document will cover the system architecture, component interactions, data flow, security considerations, and the user interface design, providing a comprehensive guide to the development and deployment of the decentralized storage system.

PURPOSE

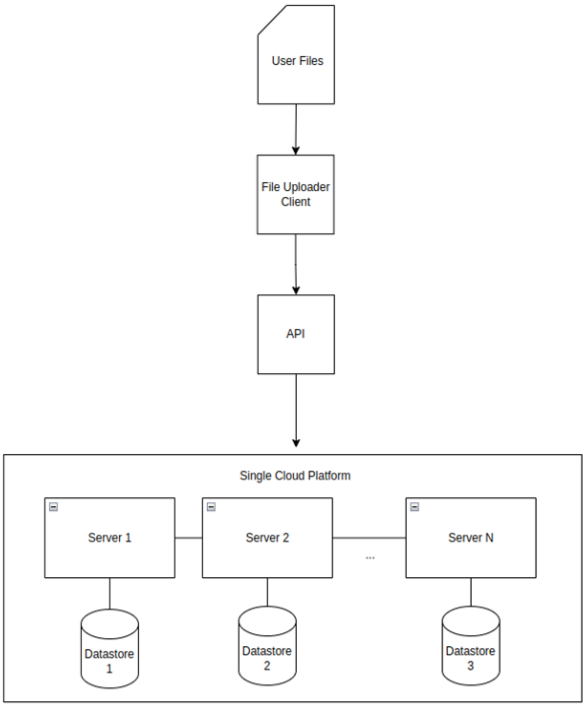
The proposed system is a decentralized Distributed Storage System accessible to users via a smartphone app that leverages the Swarm network for file storage. This architecture enables data to be distributed across numerous physical servers, typically spanning multiple data centers, providing enhanced reliability and redundancy. Unlike traditional distributed storage systems such as Google Cloud, Dropbox, and Amazon S3, this system stores data across multiple "Bee Nodes" instead of centralized servers.
When a file is uploaded to the decentralized network, it is partitioned into multiple segments, with each segment being distributed and stored across various Bee Nodes. This ensures that the data is resilient to failures and tampering, as the integrity and availability of the file are maintained by the distributed nature of the network. The Swarm network already provides the backend infrastructure for this process, along with an existing client system for uploading files through the Swarm Desktop Client.

The primary objective of this project is to develop a mobile version of the Swarm Desktop Client, making the decentralized storage system accessible via a smartphone app. This app will be available on both Android and iOS devices, offering users the convenience of managing and accessing their files on the go. The mobile app will mirror the functionality of the desktop client, ensuring a seamless user experience across different platforms. This approach not only democratizes data storage by utilizing decentralized technology but also enhances user control and data privacy.

SYSTEM OVERVIEW

The proposed system is a decentralized Distributed Storage System that is accessible to users via a smartphone app that uses the Swarm network for storing files. The distributed storage system is a type of architecture that allows data to be shared among numerous physical servers, typically across multiple data centers. While this app will function in the same way as other distributed storage systems, such as Google Cloud, Dropbox and Amazon S3, the difference is that the data data will be stored over multiple "Bee Nodes" instead of servers.

When a user uploads a file to the system, the file is divided into multiple segments. Each segment is encrypted and then distributed across multiple Bee Nodes within the Swarm network. This segmentation and distribution ensure that even if some nodes fail or are compromised, the data can still be reconstructed and accessed securely. The main components of the system include the mobile application, the Swarm network, and the Bee Nodes.



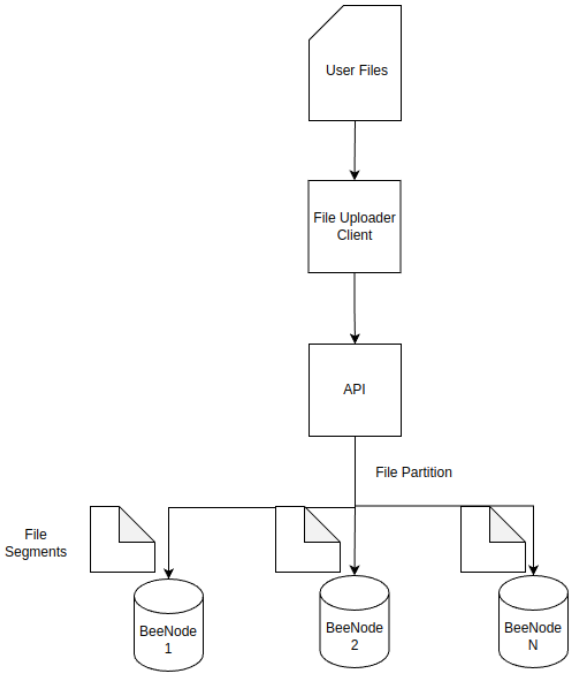


Figure 1: Typical structure of file upload system *Figure 2: Proposed structure of Swarm Upload app*

From the two figures above, we see a simplified representation of a typical cloud based storage service and the new "decentralized" Swarm file upload app. Although they appear to be very similar, the two main differences are:

1. There is no centralized authority of the servers with associated data stores,
2. In a decentralized service, the stored in many partitions across multiple BeeNodes as opposed to a single unit on a single datastore.

The potential benefits of this system include enhanced security, high scalability, cost-effective storage solutions, and user-friendly service comparable to traditional cloud storage options. This system instills greater confidence in data security due to its decentralized architecture and advanced encryption techniques. It is designed to scale efficiently, accommodating increasing amounts of data without a loss in performance. Additionally, as storage needs grow, the cost per unit of storage is likely to decrease, offering a more economical solution over time. Users will enjoy a seamless and intuitive experience, matching the ease of use and reliability of popular cloud storage services.

How it will work

When a user uploads a file to the system, the file is divided into multiple segments. Each segment is encrypted and then distributed across multiple Bee Nodes within the Swarm network. This segmentation and distribution ensure that even if some nodes fail or are compromised, the data can still be reconstructed and accessed securely. The main components of the system include the mobile application, the Swarm network, and the Bee Nodes.

1. File Upload Process

- **File Segmentation:** The file is split into smaller chunks to enable distributed storage.
- **Encryption:** Each chunk is encrypted to ensure data security and privacy.
- **Distribution:** The encrypted chunks are distributed across various Bee Nodes in the Swarm network.

2. File Retrieval Process

- **Request:** When a user wants to retrieve a file, a request is sent via the mobile app.
- **Discovery:** The system locates the necessary chunks across the Bee Nodes.
- **Decryption and Assembly:** The chunks are retrieved, decrypted, and reassembled to reconstruct the original file.

Operational Environment

The system is designed to operate on mobile platforms, specifically targeting Android and iOS devices. This ensures that users can access and manage their files on-the-go, providing the same functionality and ease of use as traditional cloud storage solutions. The Swarm network and Bee Nodes provide the backend infrastructure necessary for the decentralized storage system.

1. Mobile Platforms

- **Android Devices:** The app will support a wide range of Android versions, ensuring compatibility with various devices.
- **iOS Devices:** The app will also be compatible with iOS, covering different versions of the operating system and devices.

2. Swarm Network

- **Bee Nodes:** These are the individual nodes in the Swarm network that store the encrypted file chunks.
- **Network Protocols:** The system will utilize Swarm's native protocols for data distribution, storage, and retrieval.

The system architecture is designed to ensure scalability, reliability, and security. It comprises several layers, each responsible for different aspects of the system's functionality.

1. Client Layer

This layer includes the mobile application that users interact with. It provides the user interface (UI) and handles user input and output.

- **User Interface:** The UI is designed to be intuitive and user-friendly, allowing users to easily upload, manage, and retrieve files.
- **Mobile Application:** The app will be developed using platform-specific frameworks like Swift for iOS and Kotlin for Android, ensuring optimal performance and compatibility.

2. Application Layer

This layer manages the core functionalities of the system, including file segmentation, encryption, and communication with the Swarm network.

- **File Management:** This module handles file segmentation, encryption, and assembly.

- **Network Communication:** This module manages the interaction with the Swarm network, including uploading and retrieving file chunks.

3. Swarm Network Layer

This layer consists of the Swarm network and the Bee Nodes, which handle the storage and distribution of the file chunks.

- **Bee Nodes:** These nodes store the encrypted file chunks.
- **Swarm Protocols:** These protocols handle the distribution, storage, and retrieval of data across the network.

4. Data Storage Layer

This layer deals with the physical storage of data on the Bee Nodes.

- **Distributed Storage:** Data is stored across multiple nodes to ensure redundancy and fault tolerance.
- **Data Encryption:** All data stored on the Bee Nodes is encrypted to maintain security and privacy

Data is stored on the Swarm Network using the following process:

- The User must define a Gnosis Blockchain endpoint – either default or custom, which the UI must assist the user with and which must be stored in a configuration setting for the user. This is required for sending and receiving data,
- For reading content, a Swarm hash must find the data and read it on the client, which is required for authentication,
- For uploading content, a user must purchase “postage stamps” where it is proposed that the system will provide an “allowance” that will enable to upload and store data for free but will then be expected to purchase postage stamps to upload more data and purchase “bee node devices” tto access further storage.

Authentication and Configuration

Ensuring secure and seamless access to the system is critical. Authentication and configuration are key components that ensure the system's integrity and usability.

1. Authentication

The system will use robust authentication mechanisms to verify user identity and secure access to the mobile application.

- **User Authentication:** Users will authenticate using a combination of passwords, biometric verification (fingerprint or facial recognition), and two-factor authentication (2FA).
- **Node Authentication:** Bee Nodes will authenticate with each other using cryptographic keys to ensure secure communication and data transfer.

2. Configuration

The configuration settings allow users to customize the application according to their preferences and needs.

- **User Preferences:** Users can configure settings such as file synchronization frequency, storage limits, and notification preferences.
- **System Configuration:** Administrators can manage system-wide settings, including node management, encryption standards, and network protocols.

Data Integrity Considerations

Data integrity is a critical aspect of the decentralized storage system. The system employs several mechanisms to ensure that data remains accurate, consistent, and secure.

1. Data Encryption

All data is encrypted before being uploaded to the Swarm network. This ensures that only authorized users can access and decrypt the data.

- **Encryption Algorithms:** The system uses strong encryption algorithms (such as AES-256) to protect data.
- **Key Management:** Encryption keys are securely managed and stored, ensuring that they are only accessible to authorized users.

2. Redundancy

Data redundancy is achieved by storing multiple copies of each file chunk across different Bee Nodes. This ensures that data can be recovered even if some nodes fail.

- **Replication:** Each file chunk is replicated across several Bee Nodes to ensure high availability.
- **Fault Tolerance:** The system is designed to tolerate node failures without data loss.

3. Data Integrity Checks

The system performs regular integrity checks to ensure that data has not been tampered with or corrupted.

- **Hashing:** Each file chunk is hashed, and the hash values are stored to verify data integrity.
- **Integrity Verification:** During file retrieval, the system verifies the hash values to ensure that the data has not been altered.

4. Secure Data Transfer

Data transfer between the mobile app and the Swarm network is secured using encryption protocols.

- **TLS/SSL:** Secure protocols such as TLS/SSL are used to encrypt data during transmission.

- **Secure APIs:** The mobile app communicates with the Swarm network using secure APIs to prevent unauthorized access and data breaches.

DESIGN CONSTRAINTS

Design constraints play a crucial role in shaping the development and functionality of the decentralized Distributed Storage System. These constraints encompass various aspects, including technical, operational, security, and user-centric considerations. Below is a detailed summary of the design constraints for the system.

Technical Constraints

1. **Decentralized Architecture**

    - The system must adhere to a decentralized architecture, eliminating any single point of failure. This necessitates the use of multiple Bee Nodes for data storage and retrieval.
    - Decentralization requires robust algorithms for data partitioning, encryption, and distribution across nodes.
2. **Platform Compatibility**

    - The mobile application must be compatible with both Android and iOS devices, supporting a wide range of versions to ensure accessibility for all users.
    - Development frameworks such as Swift for iOS and Kotlin for Android must be used to achieve optimal performance and compatibility.
3. **Network Latency and Bandwidth**

    - The system must account for varying network conditions, ensuring efficient data transfer even with high latency or limited bandwidth.
    - Adaptive data transfer protocols and compression techniques may be necessary to optimize performance.
4. **Resource Constraints**

    - Mobile devices have limited processing power, memory, and storage compared to desktop systems. The application must be optimized for efficient resource usage.
    - The app should minimize battery consumption, ensuring it does not drain the device's power quickly.

Operational Constraints

1. **Scalability**

    - The system must be highly scalable, capable of handling increasing amounts of data without degradation in performance.
    - This involves efficient node management and load balancing across the Swarm network.

2. **Availability**

    - The system must ensure high availability of data, even in the presence of node failures or network partitions.
    - Redundancy and replication mechanisms must be implemented to guarantee data availability.
3. **Interoperability**

    - The system should be interoperable with existing Swarm network protocols and standards, ensuring seamless integration and communication with other components of the Swarm ecosystem.
4. **User Experience**

    - The application must provide a user-friendly interface that is intuitive and easy to navigate.
    - Consistent performance and quick response times are essential to maintain a positive user experience.

Security Constraints

1. **Data Encryption**

    - All data must be encrypted before being stored on the Bee Nodes to ensure confidentiality and prevent unauthorized access.
    - Strong encryption algorithms (such as AES-256) and secure key management practices must be employed.
2. **Authentication**

    - Robust user and node authentication mechanisms must be implemented to prevent unauthorized access.
    - The system should support multiple authentication methods, including passwords, biometric verification, and two-factor authentication (2FA).
3. **Data Integrity**

    - Mechanisms must be in place to ensure data integrity, preventing tampering and corruption.
    - Regular integrity checks, hashing, and verification processes are required to maintain data accuracy and consistency.
4. **Secure Data Transfer**

    - Data transfer between the mobile application and the Swarm network must be secured using encryption protocols like TLS/SSL.
    - Secure APIs should be used for communication to protect against data breaches and unauthorized access.

User-Centric Constraints

1. **Ease of Use**

    - The application must be designed to be user-friendly, with a simple and intuitive interface that requires minimal technical knowledge to operate.
    - Features such as guided setup, clear instructions, and helpful prompts should be included to assist users.
2. **Customization**

    - Users should have the ability to customize their application settings, such as file synchronization frequency, storage limits, and notification preferences.
    - The app must provide options for configuring Gnosis Blockchain endpoints and managing postage stamps for data uploads.
3. **Cost Considerations**

    - The system should offer cost-effective storage solutions, with a pricing model that becomes more economical as storage needs increase.
    - Users should have access to an initial free storage allowance, with options to purchase additional storage and postage stamps as needed.

Regulatory and Compliance Constraints

1. **Data Privacy**

    - The system must comply with data privacy regulations such as GDPR, ensuring that user data is protected and handled appropriately.
    - Transparent privacy policies and user consent mechanisms should be implemented.
2. **Legal Compliance**

    - The system must adhere to relevant legal and regulatory requirements in all jurisdictions where it operates.
    - This includes compliance with laws regarding data storage, encryption, and user authentication.
3. **Audit and Logging**

    - The system should maintain detailed logs of all transactions and activities for audit purposes.
    - Secure logging mechanisms must be in place to track data access, modifications, and transfers.

Environmental Constraints

1. **Network Infrastructure**

- The system must operate efficiently over various network infrastructures, including 4G/5G mobile networks, Wi-Fi, and wired connections.
- Network reliability and speed can vary significantly, and the system must adapt to these variations.

2. **Device Variability**

    - Mobile devices vary widely in terms of hardware capabilities, screen sizes, and operating system versions.
    - The application must be designed to function smoothly across a broad spectrum of devices, providing a consistent user experience.

3. **Geographical Distribution**

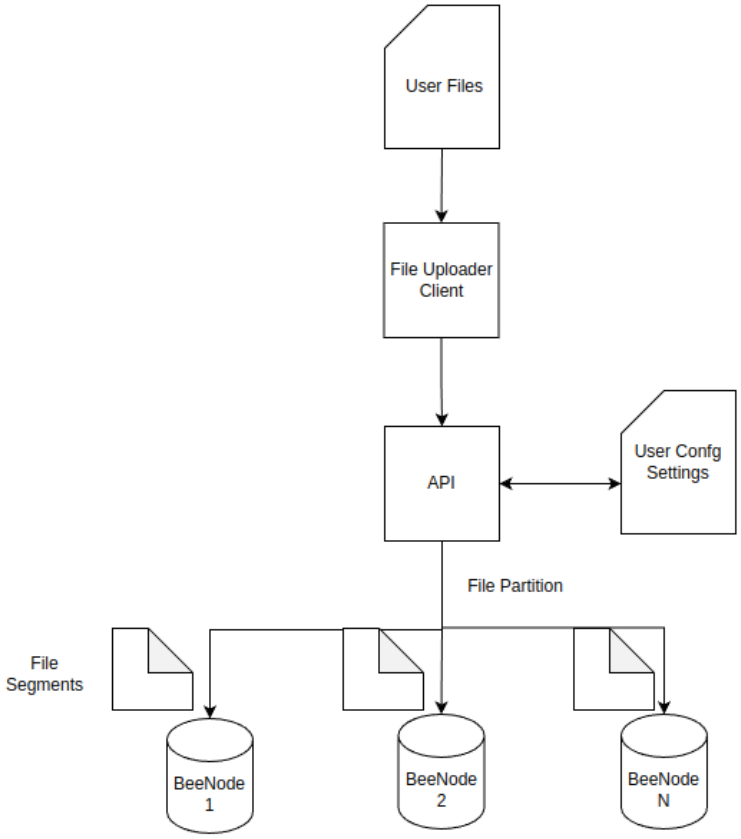
    - The decentralized nature of the system means that Bee Nodes may be located in different geographical regions, each with its own regulatory and environmental challenges.
    - The system must account for regional differences in network performance, data privacy laws, and user behavior.

The design constraints outlined above are critical for the successful development and deployment of the decentralized Distributed Storage System. By addressing these constraints, the system can ensure enhanced security, high scalability, cost-effectiveness, and a user-friendly experience, providing a robust and reliable alternative to traditional cloud storage solutions. Through careful consideration of technical, operational, security, user-centric, regulatory, and environmental factors, the system can achieve its goal of delivering secure, efficient, and accessible decentralized storage for mobile users.

PROJECT REFERENCES

- https://docs.ethswarm.org/docs/desktop/introduction
- https://www.ethswarm.org/swarm-whitepaper.pdf
- https://fairdatasociety.org/
- https://waku.org/

SYSTEM ARCHITECTURE AND REQUIREMENTS



This section will explain the functions of the User Interface (File Uploader Client) and the API of this system, which is the part that we propose to implement, with the rest already being implemented by Swarm.

FILE UPLOADER CLIENT (UI)

The user interface (UI) of the decentralized Distributed Storage System mobile app is designed to provide an intuitive and seamless experience for users to manage their files. Below are the core functions of the UI, detailed to explain how each part works to ensure users can effectively upload, manage, and retrieve their files within the Swarm network.

1. User Authentication Interface

Upon launching the app, users are greeted with a secure login screen.

- **Login/Sign-Up:** Users can log in using their credentials or sign up for a new account. The sign-up process includes creating a username and password, and optionally setting up biometric authentication (fingerprint or facial recognition) and two-factor authentication (2FA).

- **Password Recovery:** A straightforward password recovery option is available, allowing users to reset their password via email or SMS.

2. Home Dashboard

After successful authentication, users are directed to the home dashboard, which serves as the central hub for navigating the app.

- **File Upload:** A prominent upload button allows users to easily select and upload files from their device. The UI guides the user through selecting files and initiating the upload process.
- **Recent Activity:** A section displaying recent uploads, downloads, and other activities, providing users with an overview of their recent interactions with the app.

3. File Management

The file management interface is where users can view and manage their stored files.

- **File List:** Displays a list of all uploaded files, organized by categories such as documents, photos, videos, and others. Users can sort and filter files based on different criteria like date, size, and file type.
- **Search Functionality:** An integrated search bar allows users to quickly find specific files by name or metadata.
- **File Details:** Tapping on a file opens a details page, showing metadata such as upload date, size, type, and encryption status.

4. File Upload Process

The file upload process is designed to be straightforward and secure.

- **File Selection:** Users can select files from their device storage or cloud services (e.g., Google Drive, iCloud).
- **Segmentation and Encryption:** The app segments and encrypts the selected file(s) in the background. Users can see a progress indicator showing the status of these operations.
- **Confirmation:** Once the file is segmented, encrypted, and distributed across Bee Nodes, users receive a confirmation message with details of the upload.

5. File Retrieval Process

Retrieving files is made easy with a few simple steps.

- **File Selection:** Users navigate to the file they wish to retrieve from the file list or search results.
- **Request and Discovery:** Upon selecting a file, the app sends a request to the Swarm network to locate and gather the necessary chunks from the Bee Nodes.

- **Decryption and Assembly:** The file chunks are retrieved, decrypted, and reassembled in the background. Users see a progress indicator and receive a notification once the file is ready for viewing or download.

6. Settings and Configuration

The settings interface allows users to customize their app experience.

- **User Preferences:** Options to adjust file synchronization frequency, set storage limits, enable/disable notifications, and manage biometric and 2FA settings.
- **System Configuration:** Allows users to configure their Gnosis Blockchain endpoint, manage postage stamps, and set up custom endpoints for advanced users.
- **Profile Management:** Users can update their personal information, change passwords, and view their account status and storage usage.

7. Notifications and Alerts

The app includes a notification system to keep users informed.

- **Upload/Download Status:** Real-time notifications about the status of file uploads and downloads.
- **Security Alerts:** Alerts for any suspicious activity, unauthorized access attempts, or changes in account settings.
- **Storage Warnings:** Notifications when approaching storage limits or when additional postage stamps need to be purchased.

8. Help and Support

A dedicated section for user assistance.

- **Help Center:** Provides FAQs, user guides, and troubleshooting tips.
- **Customer Support:** In-app chat support or contact options (email, phone) for direct assistance.
- **Feedback:** Users can submit feedback or report bugs to help improve the app.

The UI of the decentralized Distributed Storage System mobile app is designed to be user-friendly and intuitive, ensuring that users can easily manage their files within the decentralized network. By providing straightforward processes for file upload, retrieval, and management, alongside robust settings and support features, the app delivers a seamless experience comparable to traditional cloud storage solutions while leveraging the security and scalability of the Swarm network.

API FUNCTIONALITY

The API for the decentralized Distributed Storage System serves as the communication bridge between the client app (on Android and iOS devices) and the Swarm network. It facilitates essential operations such as file upload, file viewing, file editing, and deletion. Additionally, it manages interactions with the

Gnosis Blockchain for transaction management and the handling of postage stamps required for data uploads to Bee Nodes. Below is a detailed explanation of how the API will work to support these functionalities.

1. File Upload Process

The file upload process involves several steps that ensure secure, efficient, and reliable storage of data across Bee Nodes.

Steps:

1. **File Selection and Segmentation:**
    - The client app allows users to select files for upload.
    - The file is segmented into smaller chunks for distributed storage.
2. **Encryption:**
    - Each file chunk is encrypted locally on the client device.
3. **Postage Stamp Management:**
    - The app checks the user's postage stamp balance.
    - If necessary, the app prompts the user to purchase additional postage stamps via a transaction on the Gnosis Blockchain.
4. **Transaction Management:**
    - The app initiates a transaction to purchase postage stamps using the Gnosis Blockchain endpoint.
    - A transaction management strategy such as “optimistic concurrency” ensures the transaction is processed without delays or conflicts.
5. **Chunk Distribution:**
    - The encrypted chunks are uploaded to various Bee Nodes using the Swarm API.
    - The Swarm API handles the distribution of these chunks, ensuring redundancy and fault tolerance.

API Endpoints:

- POST /upload: Initiates the file upload process.
- POST /segment: Handles file segmentation.
- POST /encrypt: Encrypts file chunks.
- POST /purchase-stamps: Manages the purchase of postage stamps.
- POST /upload-chunks: Distributes encrypted chunks to Bee Nodes.

Error Handling Strategies:

- **Network Errors:** Implement retries with exponential backoff.
- **Insufficient Stamps:** Notify the user and prompt for additional postage stamp purchase.
- **Transaction Failures:** Roll back transactions and provide detailed error messages.

2. File Viewing Process

The file viewing process involves retrieving and decrypting file chunks from the Swarm network.

Steps:

1. **File Request:**
    - The user selects a file to view.
    - The app sends a request to the Swarm network to locate the file chunks.
2. **Chunk Retrieval:**
    - The Swarm API locates and retrieves the necessary encrypted chunks from the Bee Nodes.
3. **Decryption:**
    - The retrieved chunks are decrypted locally on the client device.
4. **File Assembly:**
    - The decrypted chunks are reassembled into the original file for viewing.

API Endpoints:

- GET /files: Lists all files available for viewing.
- GET /file/:id: Requests specific file chunks.
- POST /decrypt: Decrypts retrieved file chunks.
- POST /assemble: Reassembles file chunks into the original file.

Error Handling Strategies:

- **Missing Chunks:** Implement redundancy checks and request alternative copies from different nodes.
- **Decryption Errors:** Verify encryption keys and handle corrupted chunks gracefully by notifying the user.

3. File Editing Process

Editing a file typically involves downloading, modifying, and re-uploading the file.

Steps:

1. **File Retrieval:**
    - Similar to the file viewing process, the file is retrieved and decrypted.
2. **File Editing:**
    - The user edits the file on their device.
3. **Re-upload Process:**
    - The edited file is re-segmented, encrypted, and distributed across Bee Nodes, similar to the initial upload process.

API Endpoints:

- `PUT /file/:id`: Initiates file retrieval and re-upload after editing.
- `POST /edit-chunks`: Handles the re-upload of edited file chunks.

Error Handling Strategies:

- **Concurrency Issues:** Use versioning to manage changes and prevent conflicts during simultaneous edits.
- **Edit Conflicts:** Notify users of conflicts and provide options to merge changes or overwrite.

4. File Deletion Process

The file deletion process involves removing file references and ensuring data is no longer accessible.

Steps:

1. **Deletion Request:**
    - The user selects a file to delete.
    - The app sends a delete request to the Swarm API.
2. **Chunk Removal:**
    - The Swarm API ensures the removal of file chunks from the Bee Nodes.
3. **Transaction Verification:**

- The deletion is confirmed and logged via the Gnosis Blockchain to ensure transparency and accountability.

API Endpoints:

- `DELETE /file/:id`: Handles file deletion requests.
- `POST /verify-deletion`: Confirms deletion and logs the transaction.

Error Handling Strategies:

- **Node Inaccessibility:** Retry deletions for nodes temporarily offline.
- **Transaction Errors:** Ensure rollback mechanisms are in place to maintain consistency.

Transaction Management Strategies

1. **Optimistic Concurrency Control:**
    - Assumes most transactions will not conflict and only verifies transaction integrity at the end of the process.
2. **Pessimistic Concurrency Control:**
    - Locks resources before transactions to prevent conflicts, ensuring no simultaneous access.
3. **Atomic Transactions:**
    - Ensures all-or-nothing execution, critical for operations involving multiple steps (e.g., purchasing postage stamps and uploading files).

Error Handling Strategies

1. **Retries with Exponential Backoff:**
    - For transient network errors, retries are performed with increasing intervals.
2. **Graceful Degradation:**
    - If a service is unavailable, the app should continue to function with limited features and inform the user appropriately.
3. **Detailed Error Logging and Notifications:**
    - Comprehensive error logs help diagnose issues, while user notifications keep users informed of problems and resolutions.

The API serves as the backbone of the decentralized Distributed Storage System, managing critical functions from file upload to deletion while ensuring secure and efficient communication between the

client app and the Swarm network. Effective transaction management and robust error handling strategies are integral to maintaining system reliability, security, and user satisfaction. By adhering to these principles, the system provides a resilient and user-friendly decentralized storage solution.

CONCLUSION

The decentralized Distributed Storage System is a secure, scalable, and user-friendly alternative to traditional cloud storage solutions. By leveraging the Swarm network, the system ensures data redundancy, fault tolerance, and enhanced security through encryption and decentralized storage. The integration with the Gnosis Blockchain for transaction management adds an extra layer of transparency and security, particularly in managing postage stamps for data uploads. The mobile applications offer an intuitive interface, making it easy for users to upload, manage, and retrieve their files while ensuring data integrity and privacy. Through careful consideration of technical, operational, and security aspects, the system delivers a robust and reliable storage solution tailored for modern mobile users.