Encrypted And Distributed File System

Submitted By:
Aaditya Anand
Apoorv Bansal
Mahesh Reddy Changal
Ramesh Reddy Changal
Mehagana Kasana

Abstract

The project aims to create a reliable and safe peer-to-peer (P2P) file storage system using a decentralized file management and sharing network. This is accomplished using a central tracking server and guaranteeing data integrity, security, and user-friendliness. Moreover, the system's extensive feature set includes access control, data encryption, and auditing.

Introduction

In this contemporary digital landscape, concerns surrounding data privacy, security, and centralized storage vulnerabilities have driven the need for decentralized solutions. The proposed system leverages a P2P architecture, ensuring enhanced security through encryption and increased resilience through distributed storage.

The system utilizes end-to-end encryption for data transmission and storage. The system utilizes cryptography to protect user data from unauthorized access, providing users with a secure platform for storing and sharing files. Furthermore, the decentralized nature of the system minimized the risk of single points of failure, enhancing overall system security.

The report delves into the technical aspects of the project, discussing the architecture, encryption protocols, and the mechanisms employed for file distribution and retrieval. Furthermore, to evaluate the performance and effectiveness of the proposed system, a series of comprehensive tests were conducted, and the results demonstrate that the system effectively addresses the identified concerns.

The code can be accessed from the following GitHub repository: https://github.com/maheshdrago/PCS-Project-TEAM-5.git

Working and Architecture

The P2P file storage system implemented by us operates through a complex mechanism that integrates data integrity, security, and decentralization to give users a reliable platform for managing and storing their files.

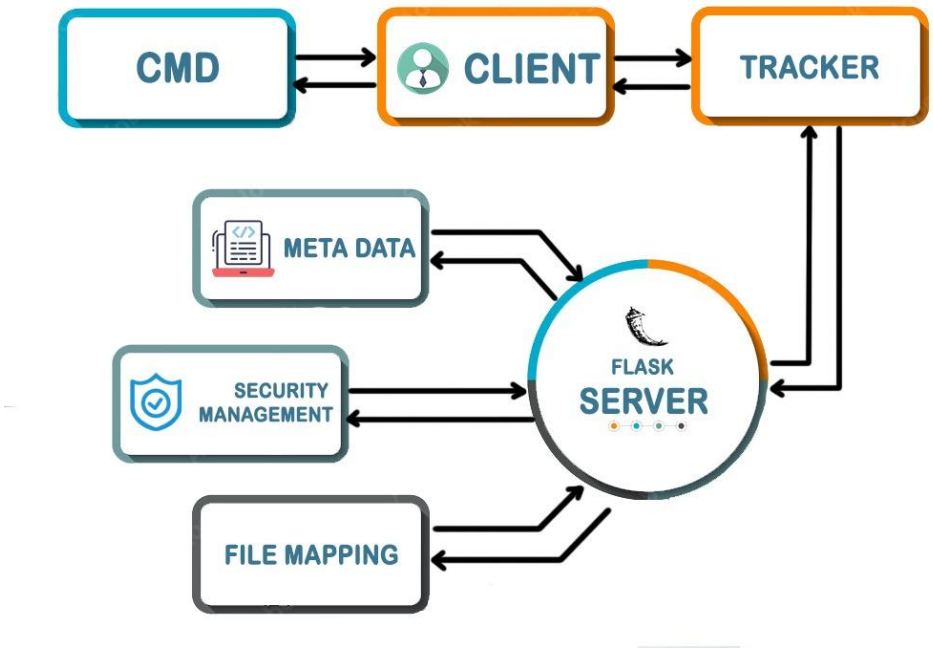


Fig. 1: Architecture Diagram

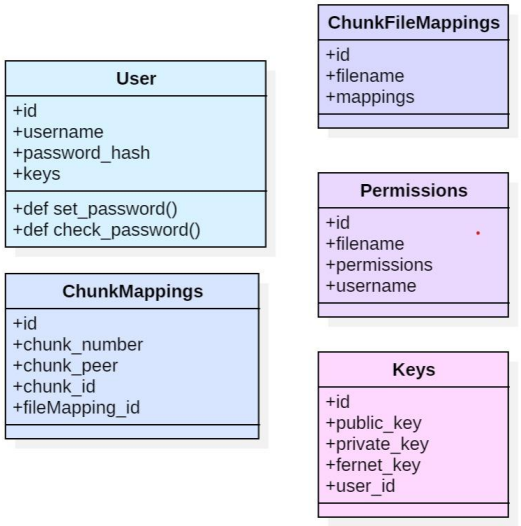


Fig. 2: Class Diagram

1. **Initialization and Node Joining:**

   When a user joins the network, they become part of the central tracking server, which serves as the foundation of the system.

   The tracking server assigns a unique user ID to each user. If the user id is the same it won't register.

2. **Decentralized file storage**

   Users can upload files to the system. When a user uploads a file, it is divided into smaller chunks or blocks.

   Each chunk is associated with a unique key derived from its content using cryptographic hashing.

   These chunks are distributed across the network and stored on appropriate nodes based on their keys.

3. **Access Control and Permissions**

   Users can set access control and permissions on files and directories. These permissions define who can read, write, delete, or download files.

   Access control lists are maintained for each file or directory, ensuring that only authorized users can access or modify them.

4. **Security and Data Privacy**

   The system takes data privacy and security seriously.

   File contents and metadata are encrypted, ensuring only authorized users can access and understand the data.

   The system uses symmetric encryption for chunk encryption, and asymmetric encryption for establishing a secure P2P connection.

   Secure communication protocols protect data in transit, and user authentication mechanisms prevent unauthorized access.

   Auditing and logging track user activities, enhancing security and accountability.

5. **Concurrent Write and Read Management**

   To handle concurrent write and read operations, the system implements mechanisms such as file locking and conflict resolution.

   When multiple users attempt to modify the same file simultaneously, the system identifies conflicts and resolves them using predefined strategies.

6. **File Retrieval**

   When a user requests a specific file, the system uses the Flask Server (MetaData Server) to locate the file's chunks based on their keys.

   The system retrieves the chunks from their respective nodes, reassembles the file, and decrypts if it is necessary.

Results

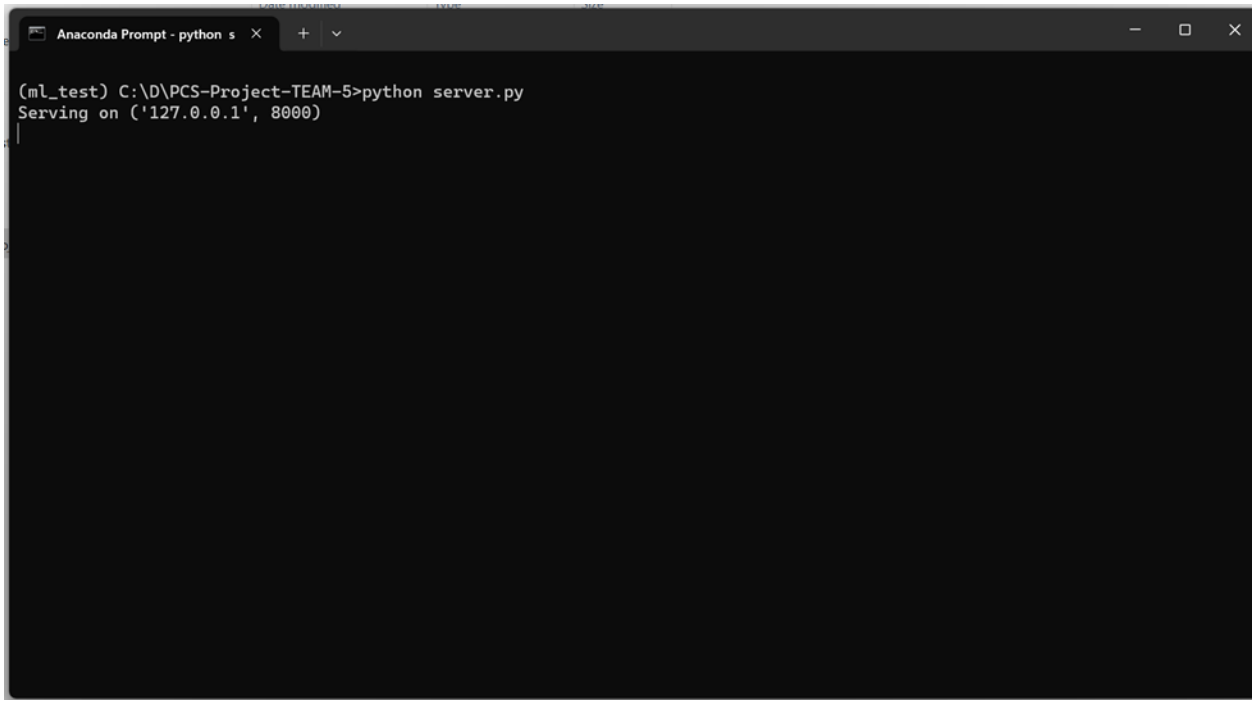


Fig. 3: Server Starting

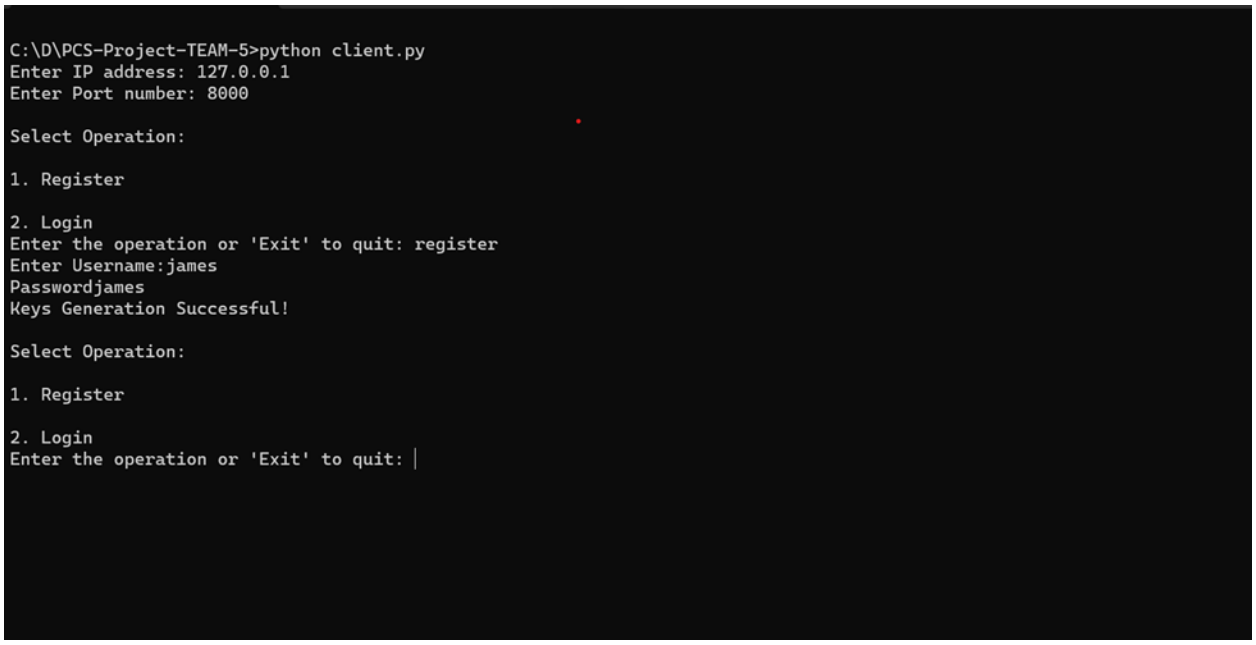


Fig. 4: Register Operation

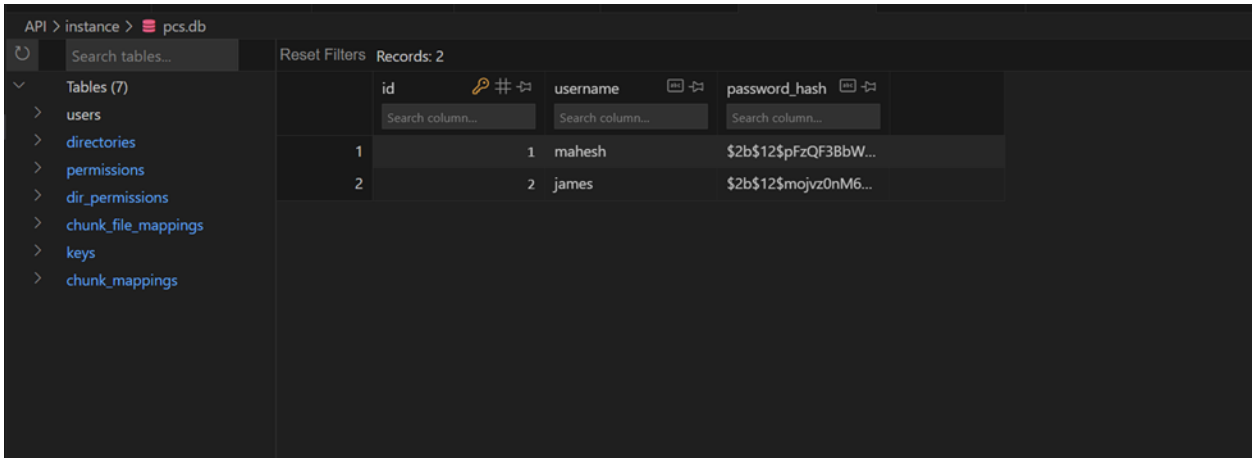


Fig 5: Client Machine, connection to the server, and user registration

```
Select Operation:

1. Register

2. Login
Enter the operation or 'Exit' to quit: login
Enter Username:james
Passwordjames
Sucess
Sending JSON: {"message": "REGISTER_IP", "username": "james"}
Peer server listening on ('127.0.0.1', 57314)

Select Operation:

1. Create

2. Delete

3. Restore

4. Exit

5. List_Peers

6. Download

7. Write

8. List_Files
Enter the operation or 'Exit' to quit:
```

Fig 6: User Login

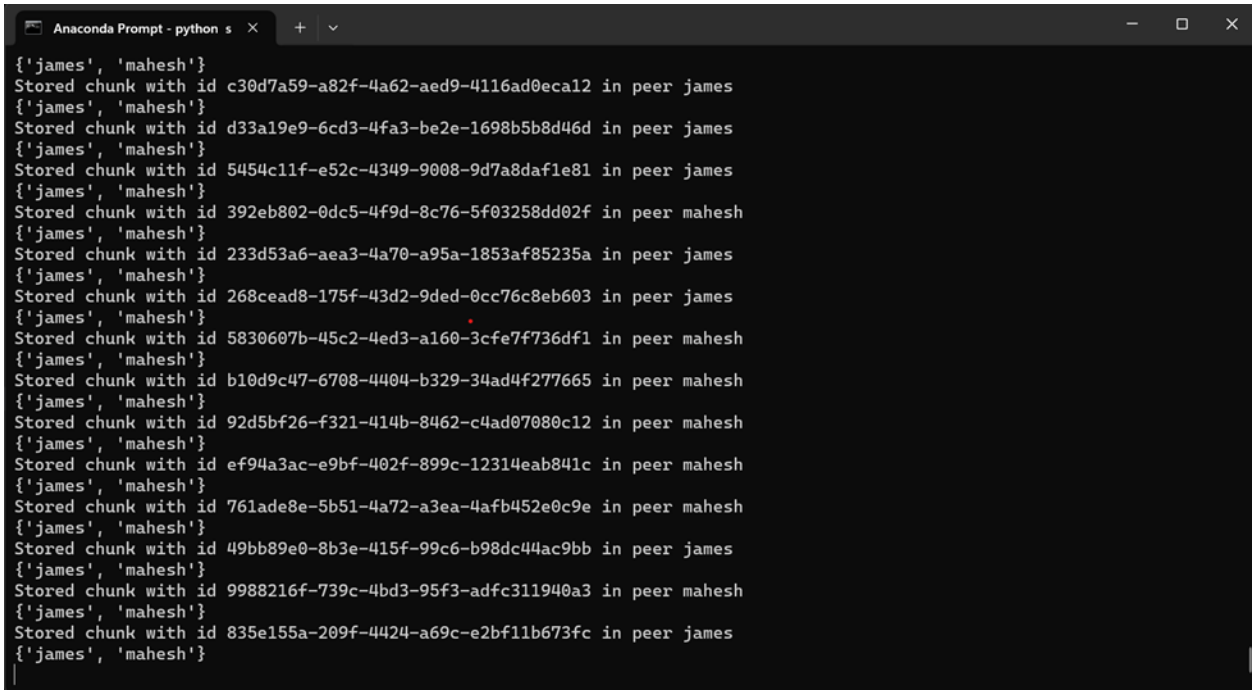


Fig. 7: File divided into chunks and stored in live peers.

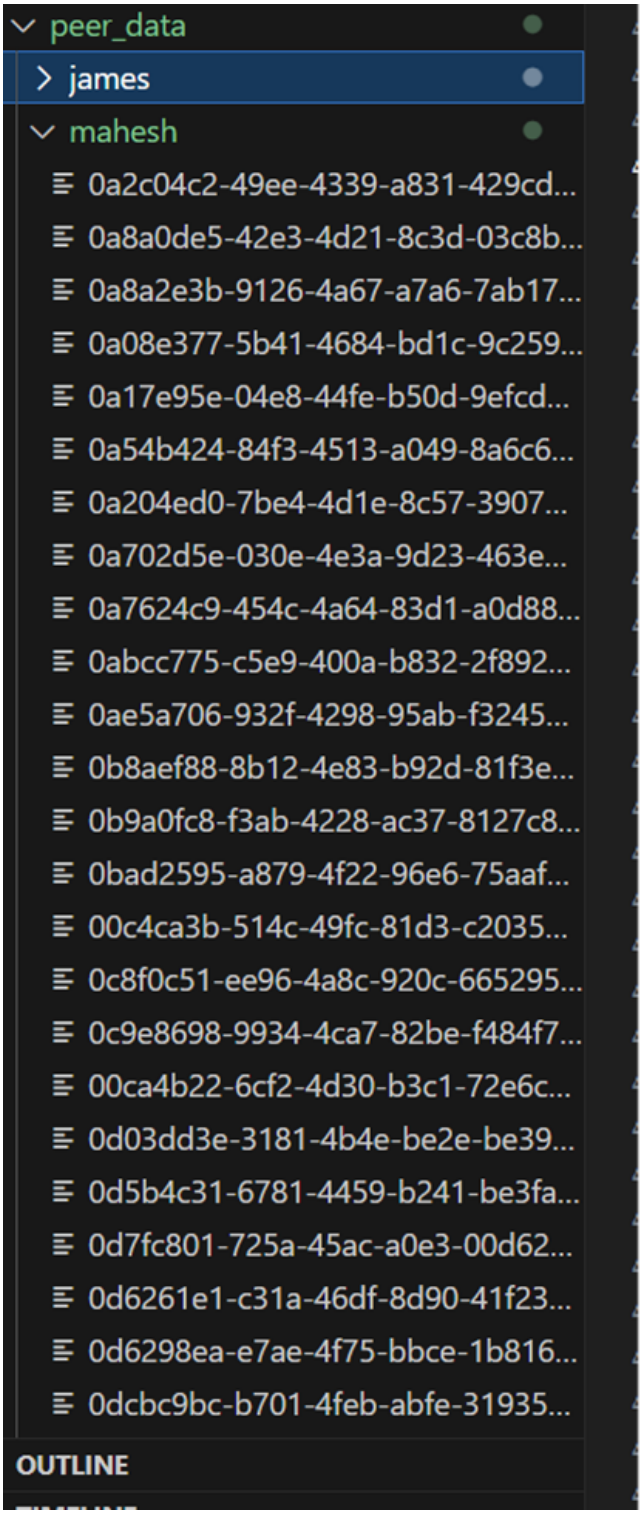


Fig. 8: Chunks stored in respective peers

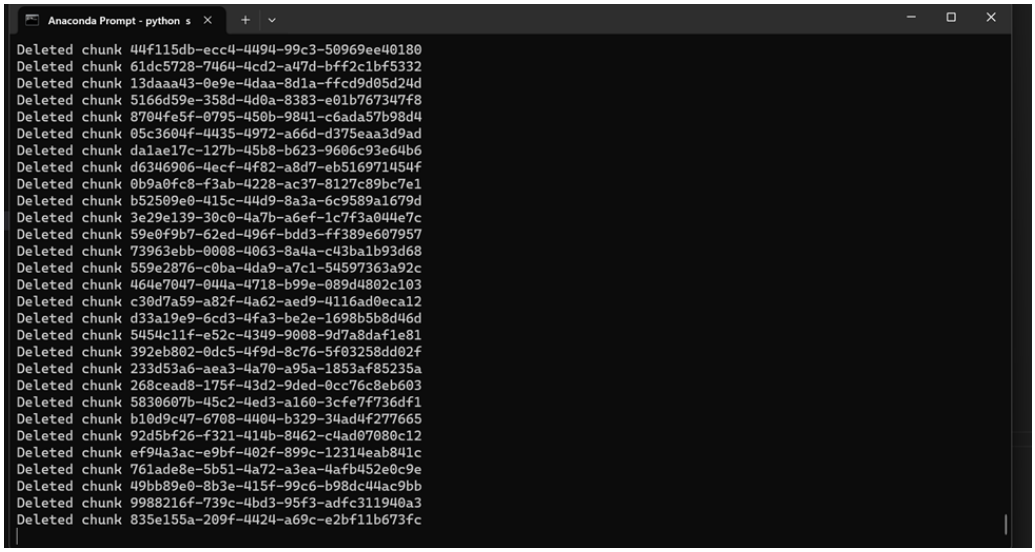


Fig. 9: Deletion of the file, check for relevant permissions, and perform an operation.

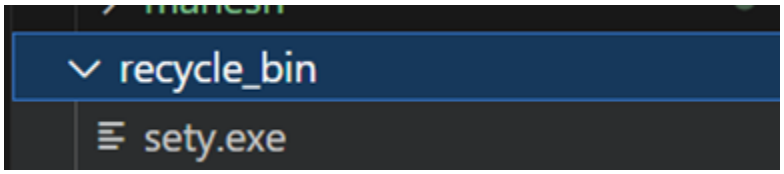
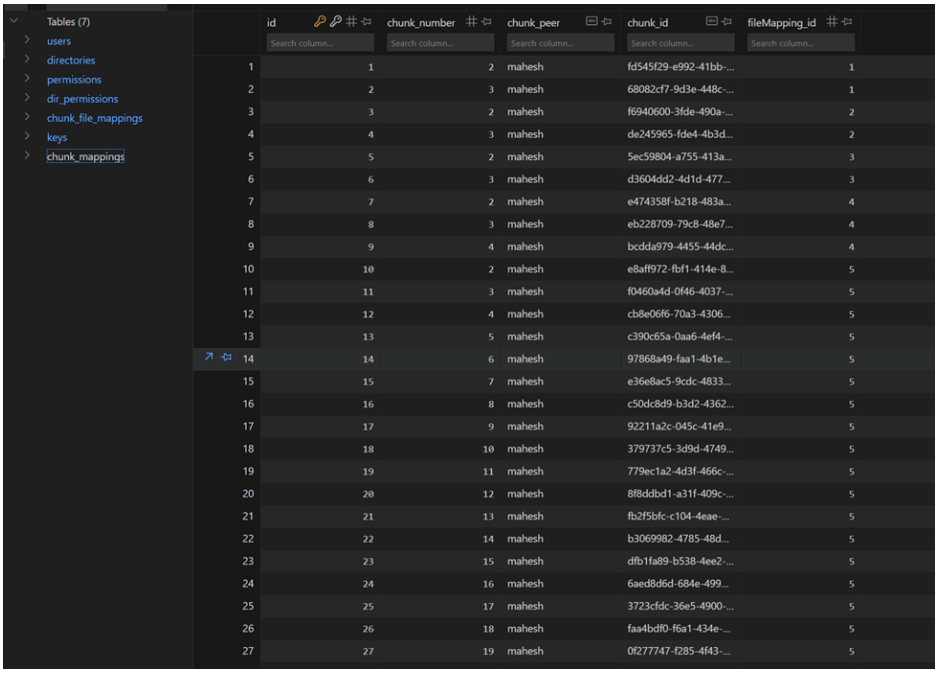


Fig. 10: File stored in servers recycle bin for future restoration.



	id	chunk_number	chunk_peer	chunk_id	fileMapping_id
	Search column...	Search column...	Search column...	Search column...	Search column...
1	1	2	mahesh	fd545f29-e992-41bb-...	1
2	2	3	mahesh	68082cf7-9d3e-448c-...	1
3	3	2	mahesh	f6940600-3fde-490a-...	2
4	4	3	mahesh	de245965-fde4-4b3d...	2
5	5	2	mahesh	5ec59804-a755-413a...	3
6	6	3	mahesh	d3604dd2-4d1d-477...	3
7	7	2	mahesh	e474358f-b218-483a...	4
8	8	3	mahesh	eb228709-79c8-48e7...	4
9	9	4	mahesh	bcdda979-4455-44dc...	4
10	10	2	mahesh	e8aff972-fbf1-414e-8...	5
11	11	3	mahesh	f0460a4d-0f46-4037-...	5
12	12	4	mahesh	cb8e06f6-70a3-4306...	5
13	13	5	mahesh	c390c65a-0aa6-4ef4-...	5
14	14	6	mahesh	97868a49-faa1-4b1e...	5
15	15	7	mahesh	e36e8ac5-9cdc-4833...	5
16	16	8	mahesh	c50dc8d9-b3d2-4362...	5
17	17	9	mahesh	92211a2c-045c-41e9...	5
18	18	10	mahesh	379737c5-3d9d-4749...	5
19	19	11	mahesh	779ec1a2-4d3f-466c-...	5
20	20	12	mahesh	8f8ddbd1-a31f-409c-...	5
21	21	13	mahesh	fb2f5bfc-c104-4eae-...	5
22	22	14	mahesh	b3069982-4785-48d...	5
23	23	15	mahesh	dfb1fa89-b538-4ee2-...	5
24	24	16	mahesh	6aed8d6d-684e-499...	5
25	25	17	mahesh	3723cfdc-36e5-4900-...	5
26	26	18	mahesh	faa4bdf0-f6a1-434e-...	5
27	27	19	mahesh	0f277747-f285-4f43-...	5

Fig. 11 Restoring a file fetches from the server recycle bin and distributes it among the peers again.

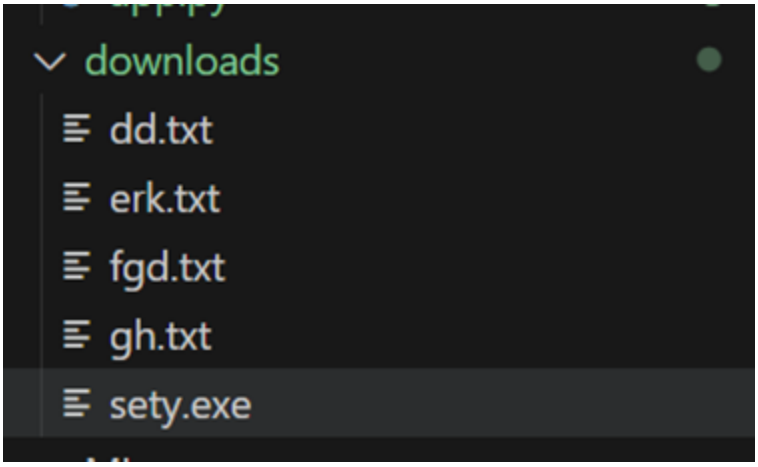


Fig. 12 Downloaded by retrieving chunks from all the peers

Conclusion

The system's architecture, incorporating robust cryptographic techniques, ensures end-to-end encryption, safeguarding user data against potential threats and unauthorized access. The modular design allows for flexibility and adaptability, making the system poised for future advancements and integrations with emerging technologies.

By contributing to the discourse on secure file systems, this project adds valuable insights to the fields of cybersecurity, distributed systems, and data privacy. The decentralized nature of the system not only mitigates single points of failure but also aligns with the ethos of user empowerment and control over personal data.

As technology continues to evolve, the Encrypted and Distributed File System serves as a foundation for future research and development in the realm of peer-to-peer technologies. The project's success highlights the potential for decentralized architectures to redefine the landscape of secure and resilient file storage, fostering a more secure digital future.

References:

[1] A Peer 2 Peer Reference Architecture

[2] File Sharing using Python in Peer-to-Peer Networks