**IPFS Based File Storage Access Control and Authentication Model for Secure Data Transfer using Blockchain Technique**

|  |  |
| --- | --- |
| **CONTENT** | **PAGE NO** |
| **Abstract** |  |
| **1.INTRODUCTION** |  |
| 1.1 Motivation |  |
| 1.2 Problem Statement |  |
| 1.3 Objective of the Project |  |
| 1.4 Scope |  |
| 1.5 Project Introduction |  |
| **2.LTERATURE SURVEY** |  |
| 2.1 Related Work |  |
| **3. SYSTEM ANALYSIS** |  |
| 3.1 Existing System |  |
| 3.2 Disadvantages |  |
| 3.3 Proposed System |  |
| 3.4 Advantages |  |
| 3.5 work Flow of Proposed system |  |
| **4. REQUIREMENT ANALYSIS** |  |
| 4.1Function and non-functional requirements |  |
| 4.2 Hardware Requirements |  |
| 4.3 Software Requirements |  |
| 4.4 Architecture |  |
| **5. SYSTEM DESIGN** |  |
| 5.1 Introduction of Input design |  |
| 5.2 UML Diagram(class, use case, sequence, collaborative, deployment, activity, ER diagram and Component diagram) |  |
| 5.3 Data Flow Diagram |  |
| **6. IMPLEMENTATION AND RESULTS** |  |
| 6.1 Modules |  |
| 6.2 Output Screens |  |
| 1. **SYSTEM STUDY AND TESTING** |  |
| 7.1 Feasibility study |  |
| 7.2 Types of test & Test Cases |  |
| **8. CONCLUSION** |  |
| **9. FUTURE ENHANCEMENT** |  |
| **10. REFERENCES** |  |

**ABSTRACT:**

Large files cannot be efficiently stored on blockchains. On one hand side, the blockchain becomes bloated with data that has to be propagated within the blockchain network. On the other hand, since the blockchain is replicated on many nodes, a lot of storage space is required without serving an immediate purpose, especially if the node operator does not need to view every file that is stored on the blockchain. It furthermore leads to an increase in the price of operating blockchain nodes because more data needs to be processed, transferred and stored. IPFS is a file sharing system that can be leveraged to more efficiently store and share large files. It relies on cryptographic hashes that can easily be stored on a blockchain. Nonetheless, IPFS does not permit users to share files with selected parties. This is necessary, if sensitive or personal data needs to be shared. Therefore, this paper presents a modified version of the Interplanetary Filesystem (IPFS) that leverages Ethereum smart contracts to provide access-controlled file sharing. The smart contract is used to maintain the access control list, while the modified IPFS software enforces it. For this, it interacts with the smart contract whenever a file is uploaded, downloaded or transferred. Using an experimental setup, the impact of the access controlled IPFS is analyzed and discussed.

**Keywords:** 1. Blockchain efficiency,2. IPFS file sharing, 3. Cryptographic hashes,4. Access-controlled sharing, ,6. Data security.

1. **INTRODUCTION**

**1.1 Motivation:**

The motivation behind exploring blockchain technology in agriculture product supply chains lies in the need for greater transparency, efficiency, and security within this vital industry. Traditional supply chain processes often involve numerous stakeholders, complex logistics, and potential risks related to food safety and authenticity. Blockchain technology has the potential to revolutionize how agricultural products are tracked, verified, and authenticated throughout their journey from farm to table. By harnessing the decentralized and tamper-resistant nature of blockchain, we can address critical issues and ensure the integrity and safety of agricultural products for both producers and consumers.

**1.2 Problem Statement:**

The integration of blockchain with file storage, particularly in large files, presents a critical challenge. Storing substantial data on blockchains results in network bloat and increased operational costs for mining nodes. Alternatively, using smart contracts for file storage incurs high transaction costs and strains network bandwidth. Additionally, public accessibility to files on platforms like IPFS poses privacy concerns, especially for sensitive data. To address these issues, this paper focuses on developing acl-IPFS, a modified InterPlanetary File System (IPFS) leveraging the Ethereum blockchain. The objective is to establish access control for IPFS files through an Ethereum smart contract, ensuring secure and permission-controlled file sharing.

**1.3 Objective of the project:**

This project aims to address the inefficiencies and limitations of utilizing blockchain for large file storage by leveraging the Ethereum blockchain to enhance the InterPlanetary File System (IPFS). The objective is to develop and implement an access-controlled IPFS solution, named acl-IPFS, using Ethereum smart contracts. By dynamically managing access control lists on the blockchain, acl-IPFS enables secure and permission-based file sharing. The project seeks to optimize file storage and retrieval, ensuring confidentiality for sensitive data, and establishing a robust linkage between IPFS nodes and Ethereum accounts for permission enforcement.

**1.4 Scope:**

This project aims to address the inefficiencies and challenges associated with storing large files on traditional blockchains, specifically focusing on the InterPlanetary File System (IPFS). Leveraging the Ethereum blockchain, the project introduces an access-controlled IPFS system (acl-IPFS) through a smart contract. The scope includes the design and implementation of acl-IPFS, enabling users to register, access, and manage large files securely. The system allows dynamic modification of access control lists, granting and revoking permissions via Ethereum transactions. By establishing a secure and permissioned layer over IPFS, the project enhances the viability of blockchain applications dealing with sensitive or personal data in large files.

**1.5 Project Introduction:**

Blockchain applications interact either directly with blockchains or with smart contracts in order to achieve consensus on transactions, data or code execution. A network of heterogeneous nodes stores the blockchain, processes transactions and, if necessary, executes smart contracts. This leads to the following issue when working with large data files. Because the files are usually not required for the blockchain nodes to function, the blockchain becomes bloated, resulting in data being replicated on a large amount of nodes. On one hand, storing large files on the blockchain is inefficient. Limitations on the block size require files to be split and reassembled off-blockchain. Additional data relevant to reassembling files would also have to be stored, requiring either even more space or a distinct system that provides the reassembly information. If smart contracts are leveraged to directly store file parts, the data can more easily be accessed and the reassembly information could be stored as well. However, sending and storing large files, even partially, using smart contracts is expensive (for example regarding gas costs) and needs to be executed at every mining or verifying node. On the other hand, operating the mining nodes becomes more expensive. More data needs to be propagated through the network, processed and stored by the node. Mining nodes would thus require connections with higher bandwidths and more storage space to store the blockchain, even partially, thus leading to increased costs. It concludes that blockchains are not the right platform to share and store large files. Fortunately, file sharing platforms can be leveraged to support applications while keeping the blockchain small in size. Users can efficiently share large files and still benefit from the blockchain. Cryptographic hashes that serve to securely identify a file’s content, can be send to the latter, thus proving that the file was available to someone at a certain time. One particularly interesting file sharing platform for this purpose, combining file sharing and the mentioned hashes, is the InterPlanetary File System (IPFS) [1]. IPFS identifies, verifies and transfers files relying on the cryptographic hashes of their contents.

Similar to public blockchains, files stored on ipfs can be requested and viewed by anyone who can connect to or deploy an ipfs node. This is an issue for blockchain applications working with large files that contain sensitive or personal data. Therefore, this paper leverages the Ethereum blockchain [2] to provide an access controlled IPFS. An Ethereum smart contract stores and allows dynamic modification of the access control list. The modified IPFS software, hereinafter named acl-IPFS, whose design and implementation is discussed in the following, can then connect to the smart contract and enforce the permissions given by the access control list. AclIPFS allows users to register new files, and grant and revoke permissions by forming and sending transactions to the smart contract. With every request for a file, acl-IPFS nodes provide the public key and sign the message using a linked Ethereum account. This creates a relation between the nodes and the account, thus allowing the nodes to rely on the smart contract to request permissions and enforce them.

1. **LITERATURE SURVEY**

**2.1 Related Work:**

**[1] G. Zyskind, O. Nathan, and A. Pentland, “Decentralizing privacy: Using blockchain to protect personal data,” in 2015 IEEE Security and Privacy Workshops, May 2015, pp. 180–184.**

The primary objective of this research is to explore and implement a decentralized privacy protection system utilizing blockchain technology. Drawing inspiration from the work of G. Zyskind, O. Nathan, and A. Pentland in "Decentralizing privacy: Using blockchain to protect personal data," the aim is to address the challenges associated with safeguarding personal data in a distributed and secure manner. The focus is on developing an innovative solution that leverages the inherent strengths of blockchain while mitigating the limitations associated with centralized storage of sensitive information.

The proposed system builds upon the concept of decentralizing privacy using blockchain, aiming to enhance the protection of personal data. We propose the integration of Ethereum smart contracts with a modified version of the Interplanetary File System (IPFS). This hybrid system ensures secure and access-controlled file sharing by maintaining an access control list within the smart contracts. The modified IPFS software enforces these access controls, interacting with the smart contract during file upload, download, and transfer operations.

The cryptographic hashes generated by IPFS provide an efficient means of linking decentralized file storage with the secure access control infrastructure facilitated by Ethereum smart contracts. By combining the strengths of both technologies, our system strives to offer a practical solution for sharing sensitive or personal data with selected parties while benefiting from the decentralized and tamper-resistant nature of blockchain.

In the context of an increasingly data-centric world, privacy concerns have become paramount. The research presented herein builds on the pioneering work of Zyskind, Nathan, and Pentland, aiming to extend the decentralization of privacy through the integration of blockchain and IPFS. The inadequacies of traditional blockchain systems in handling large files are acknowledged, leading to the adoption of IPFS for efficient file storage. However, the inability of IPFS to support selective file sharing necessitates the development of a modified version. This paper introduces a novel approach, utilizing Ethereum smart contracts to enforce access controls within IPFS, thereby addressing the critical need for secure and controlled sharing of sensitive information.

**[2] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, “A taxonomy of blockchain-based systems for architecture design,” in 2017 IEEE International Conference on Software Architecture (ICSA), April 2017, pp. 243–252.**

The objective of this study is to provide a comprehensive understanding of blockchain-based systems within the context of architecture design. By building upon the work of X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, as presented in their paper "A taxonomy of blockchain-based systems for architecture design," our aim is to delve deeper into the intricacies of these systems. Through an extensive analysis, we seek to identify key concepts, applications, and challenges associated with blockchain technology in architectural design. The objective is to contribute to the existing body of knowledge, offering insights that can inform practitioners, researchers, and decision-makers in the field.

This paper proposes an in-depth exploration of the taxonomy of blockchain-based systems for architecture design, building upon the foundational work of Xu et al. (2017). We intend to elucidate the various dimensions of blockchain integration in architectural contexts, examining its implications on system design, security, and scalability. Furthermore, we aim to identify emergent patterns and trends within the field, bridging theoretical insights with practical applications. The proposed study aspires to offer a nuanced perspective on how blockchain technology is shaping the landscape of architectural design, fostering a deeper understanding of its potential benefits and challenges.

In recent years, the integration of blockchain technology in architecture design has emerged as a topic of significant interest and relevance. The seminal work by Xu et al. (2017) provides a taxonomy that categorizes blockchain-based systems, laying the groundwork for our exploration. This paper extends their efforts by delving into the intricate relationship between blockchain and architecture, aiming to uncover novel perspectives and insights. As industries increasingly turn to blockchain for its decentralized and secure features, understanding its specific applications in architecture becomes imperative. This introduction sets the stage for a comprehensive examination, offering a roadmap for readers to navigate the evolving intersection of blockchain technology and architectural design.

**[3] I. Baumgart and S. Mies, “S/kademlia: A practicable approach towards secure key-based routing,” in Proceedings of the 13th International Conference on Parallel and Distributed Systems - Volume 02, ser. ICPADS ’07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 1–8. [Online]. Available: http://dx.doi.org/10.1109/ICPADS.2007. 4447808.**

The primary objective of this study is to explore and advance the field of secure key-based routing, leveraging the S/kademlia approach introduced by Baumgart and Mies. In the realm of distributed systems, efficient and secure routing is paramount, and the S/kademlia protocol offers a practicable solution. This research aims to delve deeper into the mechanisms and principles behind S/kademlia, seeking to enhance its security and overall performance in key-based routing scenarios. By building upon the foundation laid out in the seminal work by Baumgart and Mies, the goal is to propose modifications and improvements that contribute to the robustness and reliability of S/kademlia, addressing potential vulnerabilities and refining its application in parallel and distributed systems.

Our proposed approach involves a meticulous examination of the S/kademlia protocol, identifying areas where security can be fortified and routing efficiency optimized. Through a combination of theoretical analysis and practical experimentation, we intend to refine the protocol's design to better meet the demands of contemporary distributed systems. This may include novel encryption techniques, adaptive routing strategies, or other innovative enhancements aimed at elevating the protocol's overall performance. Furthermore, we aspire to validate the proposed modifications through simulations and, where applicable, real-world implementations, providing empirical evidence of the efficacy of our contributions.

In the landscape of parallel and distributed systems, secure key-based routing plays a pivotal role in facilitating efficient communication. The seminal work of Baumgart and Mies introduced the S/kademlia protocol as a practical solution to the challenges associated with this critical aspect of distributed computing. This paper builds upon their foundation, aiming to extend the protocol's capabilities and fortify its security measures. By undertaking a comprehensive analysis and proposing novel enhancements, we seek to contribute to the evolution of S/kademlia, making it even more adept at handling the intricacies of contemporary distributed environments. Through this research, we envision a refined protocol that not only meets the standards set by its predecessors but also adapts to the evolving landscape of parallel and distributed systems, ensuring secure and efficient key-based routing.

**[4] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan, “Smart contract-based access control for the internet of things,” CoRR, vol. abs/1802.04410, 2018. [Online]. Available:** [**http://arxiv.org/abs/1802. 04410**](http://arxiv.org/abs/1802.%2004410)**.**

The objective of this research is to address the burgeoning security challenges in the Internet of Things (IoT) ecosystem through the implementation of smart contract-based access control mechanisms. With the proliferation of interconnected devices in IoT, ensuring secure and authorized access to sensitive data has become a paramount concern. The study aims to leverage the capabilities of smart contracts to enhance access control protocols, providing a robust and decentralized framework for managing permissions and safeguarding IoT devices and data.

The proposed approach revolves around the integration of smart contracts into the existing IoT architecture to establish a resilient access control system. Smart contracts, executed on blockchain platforms, offer transparency, immutability, and decentralization, making them an ideal solution for enhancing security in IoT environments. By employing these self-executing contracts, the research seeks to streamline and automate access control processes, minimizing vulnerabilities associated with centralized authority and reducing the risk of unauthorized access.

The advent of the Internet of Things has ushered in unprecedented connectivity, enabling seamless communication and data exchange among a myriad of devices. However, this interconnectedness exposes IoT ecosystems to security threats, necessitating innovative solutions to fortify access control mechanisms. This paper introduces a novel approach based on smart contracts, drawing upon the study conducted by Zhang et al. [7]. The integration of smart contracts into the fabric of IoT holds the promise of revolutionizing access control, ensuring the confidentiality and integrity of data exchanged within these networks. Through a comprehensive exploration of this paradigm, the research aims to contribute to the development of secure and scalable solutions for the evolving landscape of the Internet of Things.

**[5] C. K. and T. M. V., Distributed Access Control in Cloud Computing Systems. Wiley-Blackwell, 2016, ch. 35, pp. 417– 432. [Online].**

This study aims to investigate and propose advancements in distributed access control mechanisms within the context of cloud computing systems. With the increasing reliance on cloud services for data storage, processing, and application deployment, the need for robust security measures has become paramount. The objective is to address the challenges associated with access control in distributed cloud environments, where data is dispersed across multiple servers and locations.

The research proposes an in-depth analysis of existing distributed access control models, with a focus on understanding their strengths and weaknesses. Leveraging insights from the work of C. K. and T. M. V. as presented in their 2016 publication, the study will explore innovative approaches to enhance the efficiency and effectiveness of access control in cloud systems. This may involve the integration of cutting-edge technologies such as blockchain or the development of adaptive access control strategies to accommodate the dynamic nature of cloud infrastructures.

In the era of cloud computing, where data is often distributed across a multitude of servers and locations, ensuring secure access to sensitive information is a critical concern. This study delves into the realm of distributed access control, drawing inspiration from the work of C. K. and T. M. V. as outlined in their 2016 publication. As cloud technologies evolve, so too must the strategies employed to safeguard data from unauthorized access. This research seeks to contribute to the ongoing discourse by proposing novel approaches to enhance access control mechanisms in cloud environments, ultimately fortifying the security posture of these systems.

**3. SYSTEM ANALYSIS**

**3.1 Existing System**

The existing system involves blockchain applications interacting with smart contracts for achieving consensus on transactions, data, or code execution. However, the system faces challenges when dealing with large data files, as storing them directly on the blockchain leads to inefficiency and bloating. Smart contracts can be used to store file parts, but this incurs high costs in terms of gas expenses and increased operational expenses for mining nodes. To address this, the paper proposes leveraging the InterPlanetary File System (IPFS) for efficient file sharing while keeping the blockchain size manageable. Additionally, the paper introduces acl-IPFS, a modified IPFS software integrated with an Ethereum smart contract for access-controlled IPFS.

**3.2** **Disadvantages**

**1. Scalability Concerns:**

As the number of files and users increases, the scalability of the IPFS network may become a challenge. The decentralized nature of IPFS relies on nodes to store and retrieve files, and the system may experience bottlenecks as the demand for storage and access control grows.

**2. Dependency on Ethereum Blockchain:**

The proposed model relies on the Ethereum blockchain for access control, introducing a dependency on the Ethereum network's performance and scalability. Fluctuations in Ethereum transaction costs and network congestion could impact the efficiency and cost-effectiveness of the file storage and access control system.

**3. Latency in File Retrieval:**

Access control enforcement through smart contracts on the Ethereum blockchain may introduce latency in retrieving files from IPFS. The need to interact with the blockchain for every access request can result in delays, especially during periods of high network activity or congestion.

**4. Cost Implications:**

Leveraging the Ethereum blockchain for access control involves transaction costs (gas fees). As the number of transactions and data interactions increases, the associated costs for users and nodes may become a significant factor, potentially limiting the practicality of the system for large-scale or frequent file transfers.

**5. Limited Privacy:**

While cryptographic hashes secure the content of files on IPFS, the inherent transparency of blockchain technology may pose privacy concerns. The details of file access and transactions are recorded on the public Ethereum blockchain, potentially exposing sensitive information to unintended parties.

**6. Complexity in Smart Contract Management:**

Managing and updating access control lists through Ethereum smart contracts can be complex. Changes in permissions or user access require on-chain transactions, and errors in smart contract execution or misconfigurations may lead to unintended consequences in the access control mechanism.

**7. Incompatibility with Certain Use Cases:**

The proposed model may not be suitable for scenarios where real-time access to files or low-latency interactions is crucial. The reliance on blockchain transactions and decentralized storage may introduce delays that are not acceptable for certain applications, such as multimedia streaming or interactive content delivery.

**8. Regulatory Challenges:**

The decentralized and often anonymous nature of blockchain systems may pose challenges in complying with data protection regulations and legal requirements. Navigating the regulatory landscape and ensuring the model's adherence to various jurisdictions can be complex and may hinder widespread adoption.

**3.3 Proposed System**

This research introduces a novel system, acl-IPFS, designed to address the challenges of securely sharing large files on the Interplanetary File System (IPFS) while maintaining access control. Leveraging Ethereum blockchain technology, acl-IPFS employs a smart contract to store and dynamically manage an access control list. Users can register, grant, and revoke permissions for files, ensuring secure and controlled access. Each file request involves the acl-IPFS nodes providing a public key and signing the message with a linked Ethereum account, establishing a robust connection between nodes and the smart contract to enforce permissions. This innovative approach enhances file sharing security on IPFS within a decentralized and permissioned framework.

**3.4 Advantages**

**1. Enhanced Security:** The integration of a blockchain-based access control and authentication model adds an extra layer of security to IPFS file storage. Blockchain's inherent tamper-resistant nature and cryptographic principles contribute to robust protection against unauthorized access and data tampering.

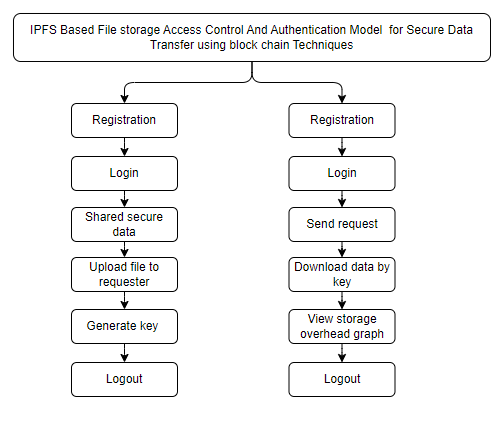
**2. Decentralization**: Leveraging IPFS for file storage, coupled with blockchain technology, ensures a decentralized and distributed network. This mitigates the risk of a single point of failure, enhances reliability, and reduces the susceptibility to cyber attacks.

**3. Granular Access Control:** The proposed model allows for dynamic modification of access control lists through smart contracts. This granular control empowers users to manage permissions efficiently, granting or revoking access as needed, providing flexibility and adaptability in data sharing scenarios.

**4. Immutable Records:** Utilizing blockchain ensures the immutability of access control records. Once permissions are set in the smart contract, they cannot be altered retroactively, ensuring the integrity and auditability of access control decisions over time.

**5. Cost-Efficiency:** By offloading the storage of large files from the blockchain to IPFS, the proposed model reduces the cost associated with storing voluminous data directly on the blockchain. This optimizes resource usage, making the system more economically viable.

**3.5 work Flow of Proposed system**

****

**4. REQUIREMENT ANALYSIS**

**4.1 Functional and non-functional requirements**

Requirement’s analysis is very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: Functional and non-functional requirements.

**Functional Requirements**: These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

Examples of functional requirements:

1. Authentication of user whenever he/she logs into the system
2. System shutdown in case of a cyber-attack
3. A verification email is sent to user whenever he/she register for the first time on some software system.

**Non-functional requirements**: These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements.  
They basically deal with issues like:

* Portability
* Security
* Maintainability
* Reliability
* Scalability
* Performance
* Reusability
* Flexibility

Examples of non-functional requirements:

1. Emails should be sent with a latency of no greater than 12 hours from such an activity.
2. The processing of each request should be done within 10 seconds
3. The site should load in 3 seconds whenever of simultaneous users are > 10000
   1. **Hardware Requirements**

# Processor - I3/Intel Processor

Hard Disk - 160GB

Key Board - Standard Windows Keyboard

Mouse - Two or Three Button Mouse

Monitor - SVGA

RAM - 8GB

* 1. **Software Requirements:**

Operating System : Windows 7/8/10

Server side Script : HTML, CSS, Bootstrap & JS

Programming Language : Python

Libraries : Flask, Pandas, Mysql.connector, Os, Smtplib, Numpy

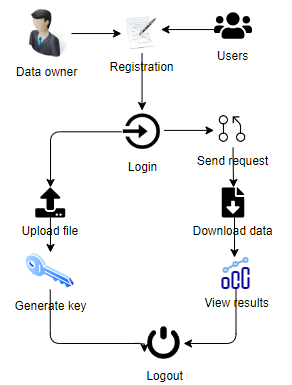
IDE/Workbench : PyCharm

Technology : Python 3.6+

Server Deployment : Xampp Server

Database : MySQL

* 1. **Architecture:**

****

**5. SYSTEM DESIGN**

**5.1 Introduction of Input Design:**

In an information system, input is the raw data that is processed to produce output. During the input design, the developers must consider the input devices such as PC, MICR, OMR, etc.

Therefore, the quality of system input determines the quality of system output. Well-designed input forms and screens have following properties −

* It should serve specific purpose effectively such as storing, recording, and retrieving the information.
* It ensures proper completion with accuracy.
* It should be easy to fill and straightforward.
* It should focus on user’s attention, consistency, and simplicity.
* All these objectives are obtained using the knowledge of basic design principles regarding −
  + What are the inputs needed for the system?
  + How end users respond to different elements of forms and screens.

### **Objectives for Input Design:**

The objectives of input design are −

* To design data entry and input procedures
* To reduce input volume
* To design source documents for data capture or devise other data capture methods
* To design input data records, data entry screens, user interface screens, etc.
* To use validation checks and develop effective input controls.

**Output Design:**

The design of output is the most important task of any system. During output design, developers identify the type of outputs needed, and consider the necessary output controls and prototype report layouts.

### Objectives of Output Design:

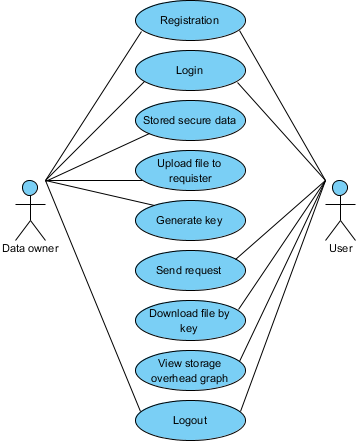
The objectives of input design are:

* To develop output design that serves the intended purpose and eliminates the production of unwanted output.
* To develop the output design that meets the end user’s requirements.
* To deliver the appropriate quantity of output.
* To form the output in appropriate format and direct it to the right person.
* To make the output available on time for making good decisions.

**5.2 UML Diagrams:**

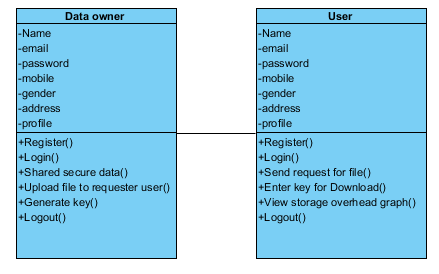
**5.2.1 Use Case Diagram:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



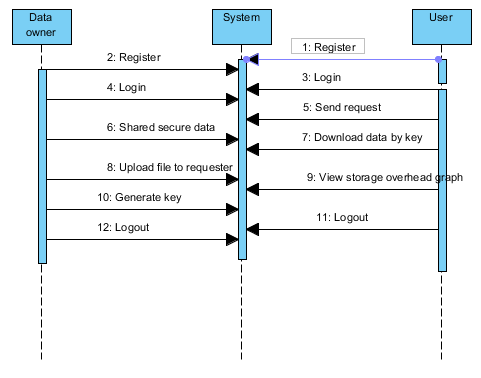
**5.2.2 Class Diagram:**

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



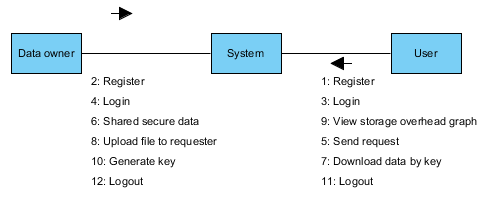
**5.2.3 Sequence Diagram:**

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



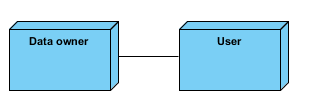
**5.2.4 Collaboration Diagram:**

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization.



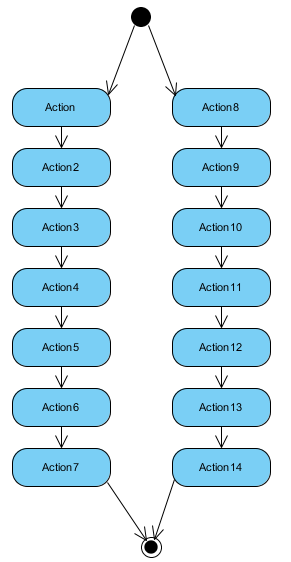
**5.2.5 Deployment Diagram**

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware’s used to deploy the application.



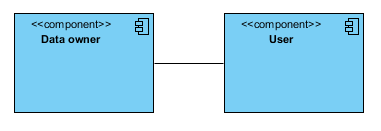
**5.2.6 Activity Diagram:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



**5.2.7 Component Diagram**: 128080

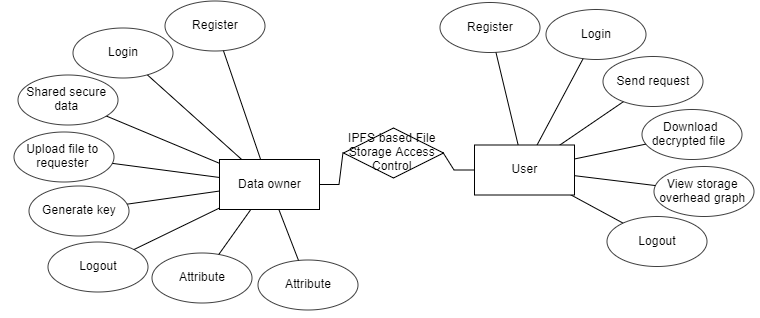
A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical **c**omponents in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development.



**5.2.8 ER Diagram:**

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

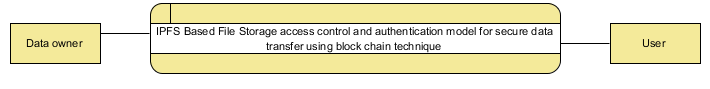
An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let’s have a look at a simple ER diagram to understand this concept.

****

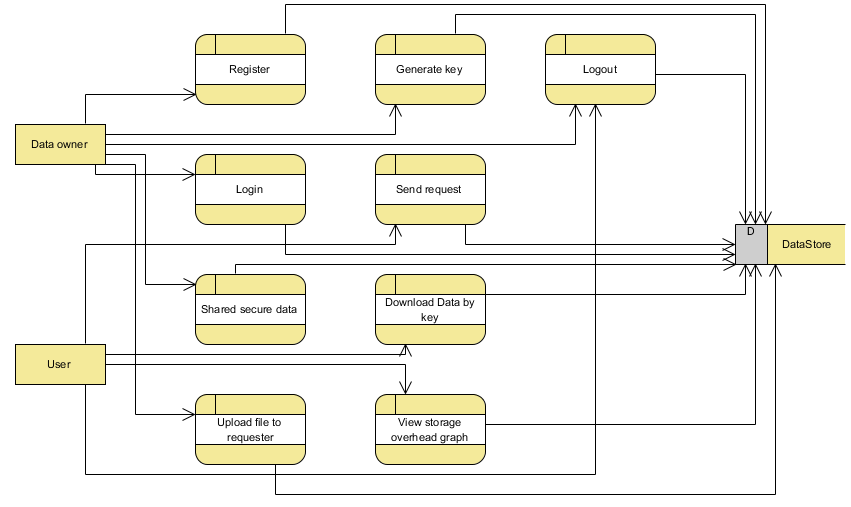
**5.3 DFD Diagram:**

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.

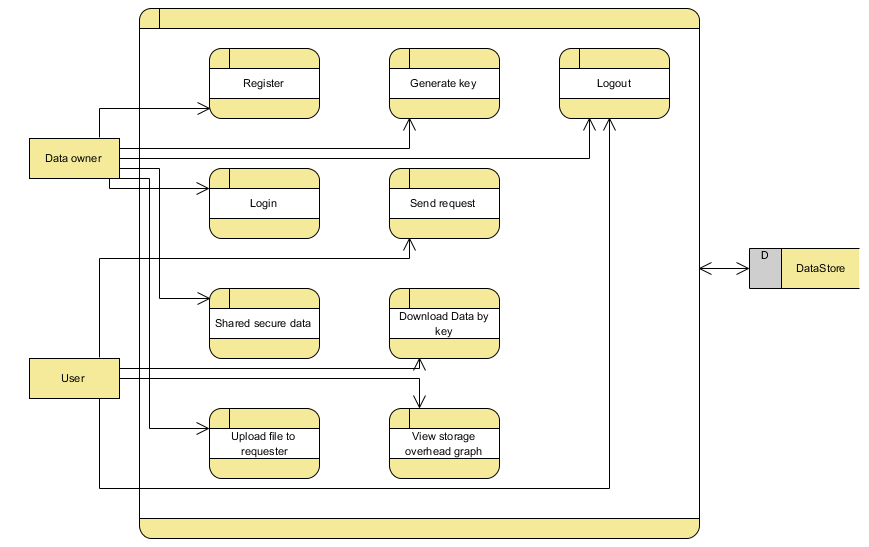
**Level 0 diagram:**



**Level 1 Diagram:**

****

**Level 2 Diagram:**

****

**6. IMPLEMENTATION AND RESULTS**

**6.1 Modules:**

**DATA OWNER:**

**Register:** The data owner will register with name, email, password, mobile number, address and profile.

**Login:** After successful login the data owner can login with valid credentials, like email and password.

**Upload file:** The data owner can share file to particular file requested user.

**Generate key:** For security purpose the data owner will share key to user.

**Logout:** Data owner should be logout.

**USER MODULE:**

**Register:** The data user will register with name, email, password, mobile number, address and profile.

**Login:** After successful login the data user can login with valid credentials, like email and password.

**Send request file:** The data user needs to send a request to data owner for data.

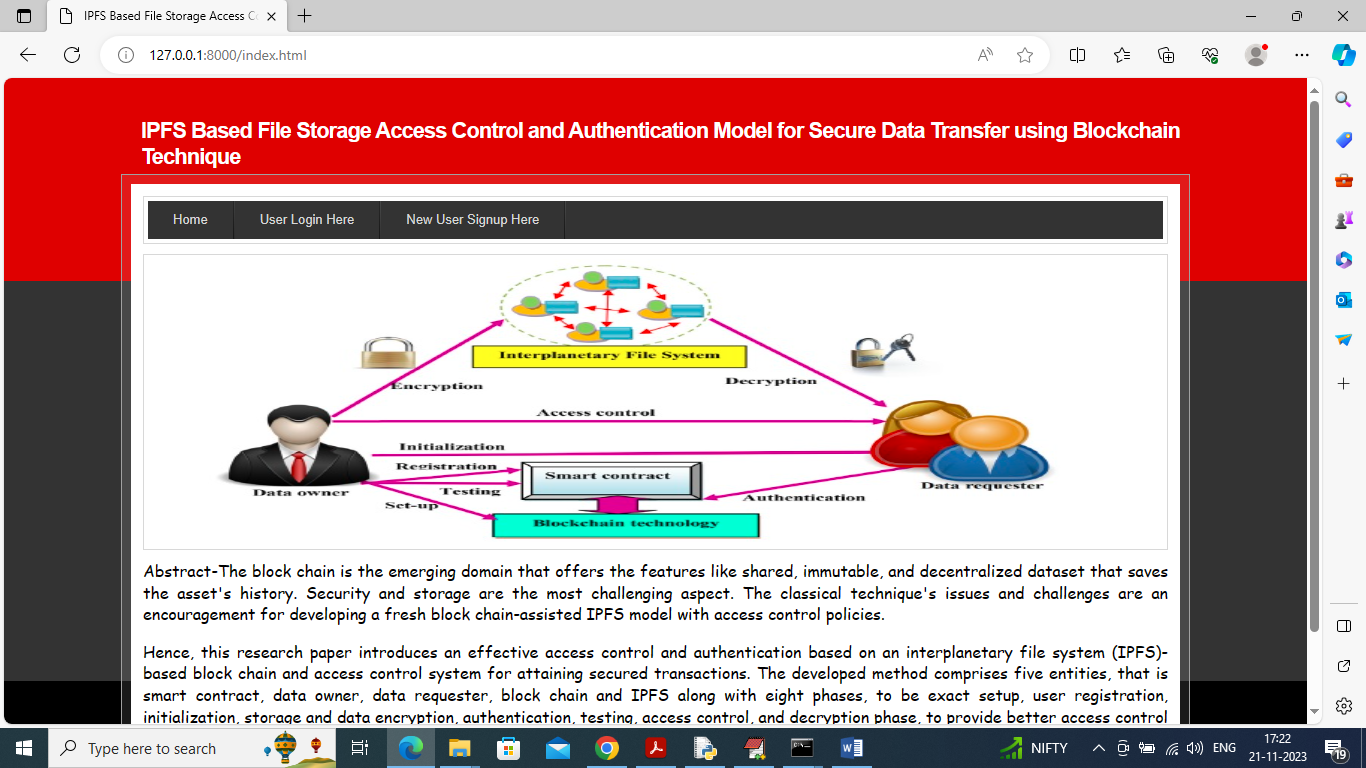
**Enter key to download:** The user needs to enter key sent by data owners to download file.

**View storage overhead graph:**  The data user can view the output by graphical representation.

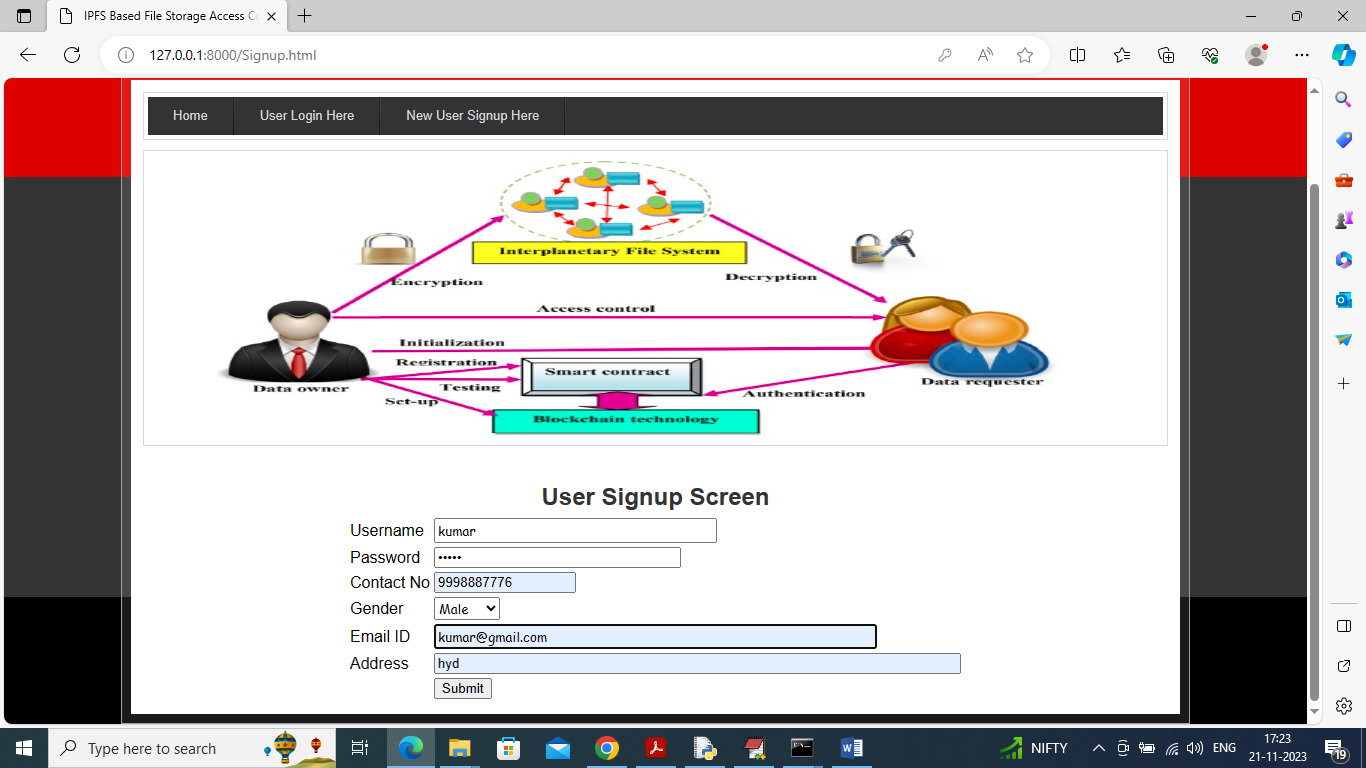
**Logout:** Data user should be logout.

**6.2 Results:**

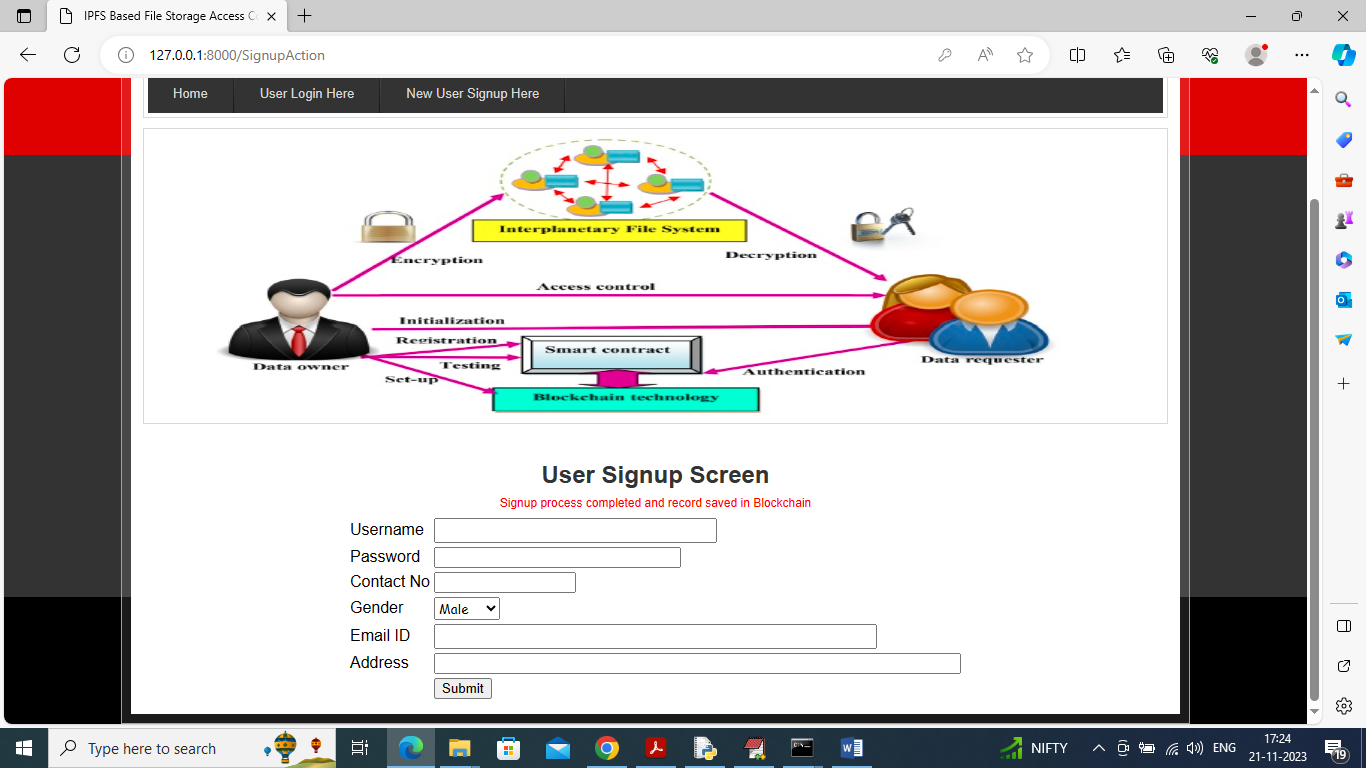
**New registration:** In above screen click on ‘New User Signup Here’ link and then add 3 different users so we can provide access between them



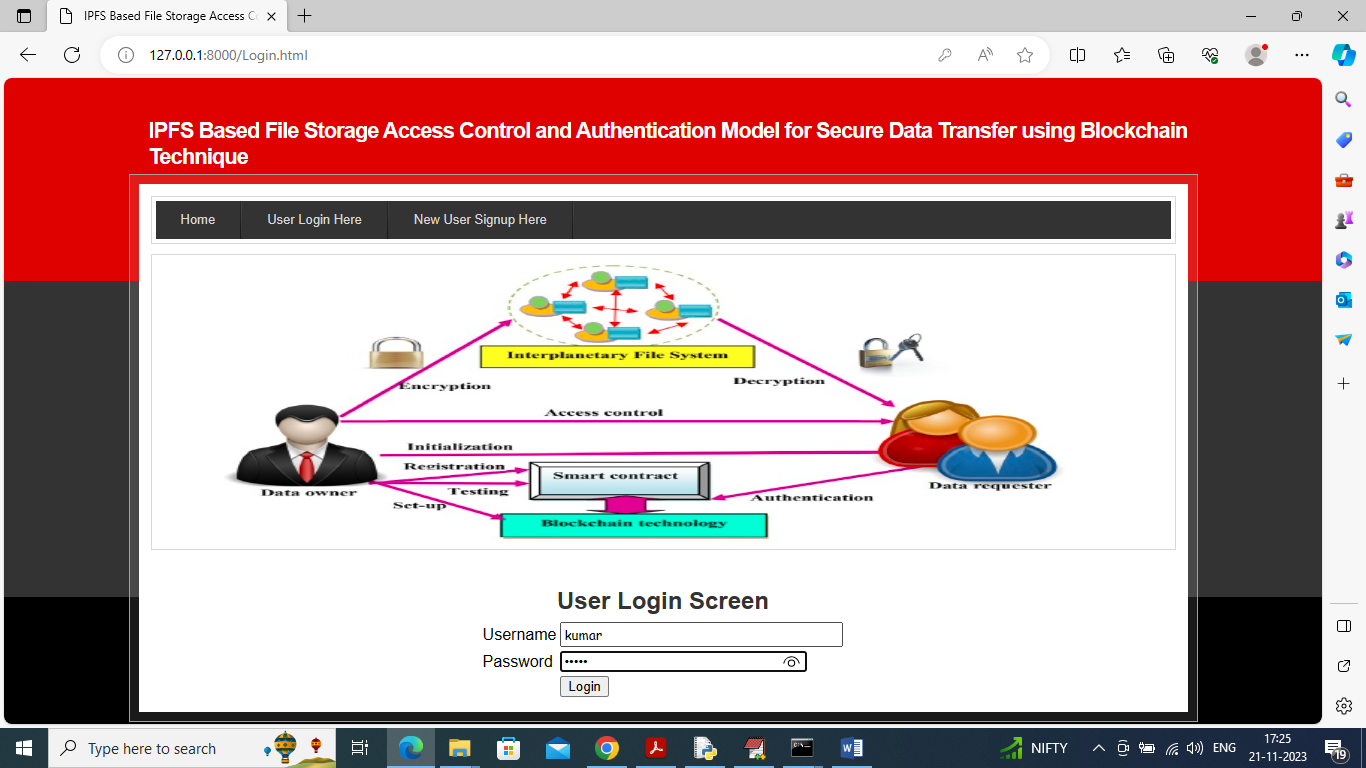
**Registration Page:** In below screen adding user details and any user can upload data and become “data owner” and any user can get access from data owner and then click on ‘Submit’ button to store data in Blockchain and get below output



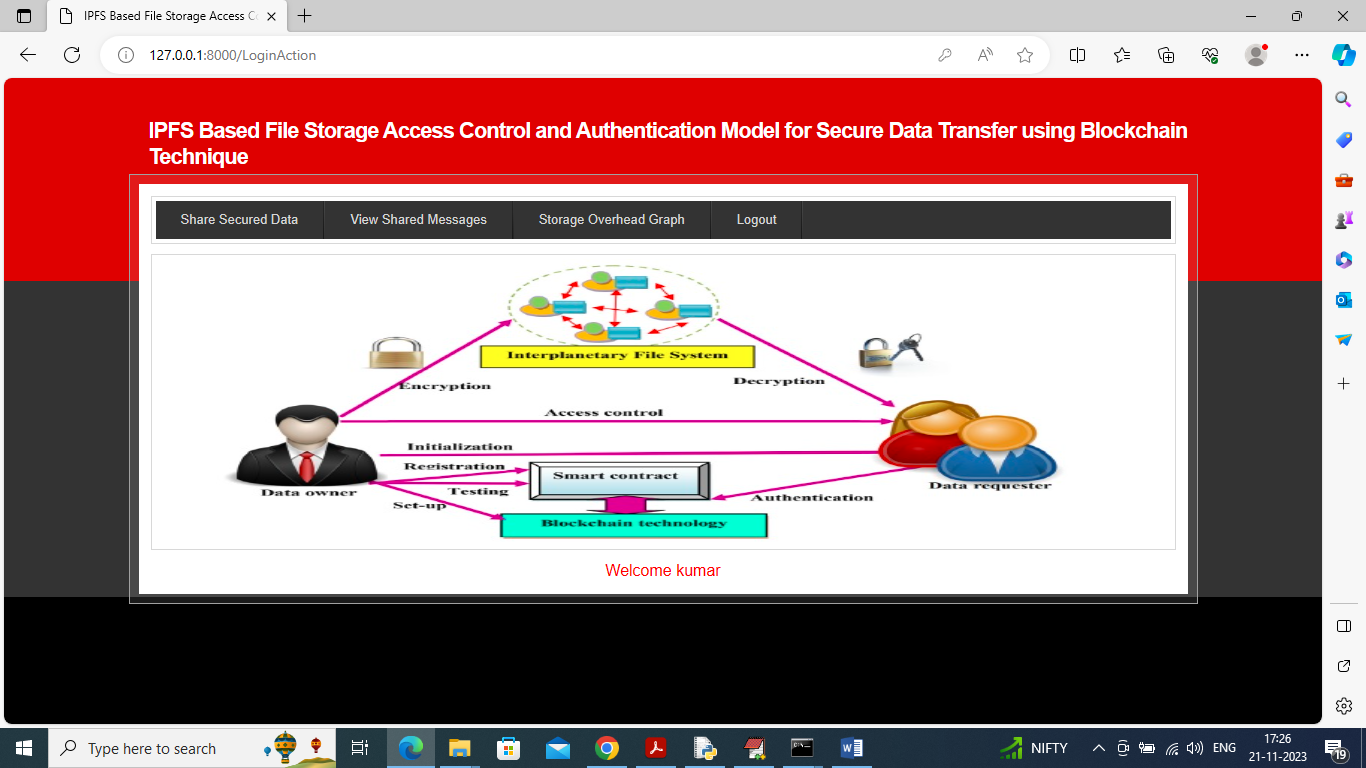
**Registration success:** In below screen signup task completed and similarly you can add many data owners and now click on ‘User Login’ link to get below page



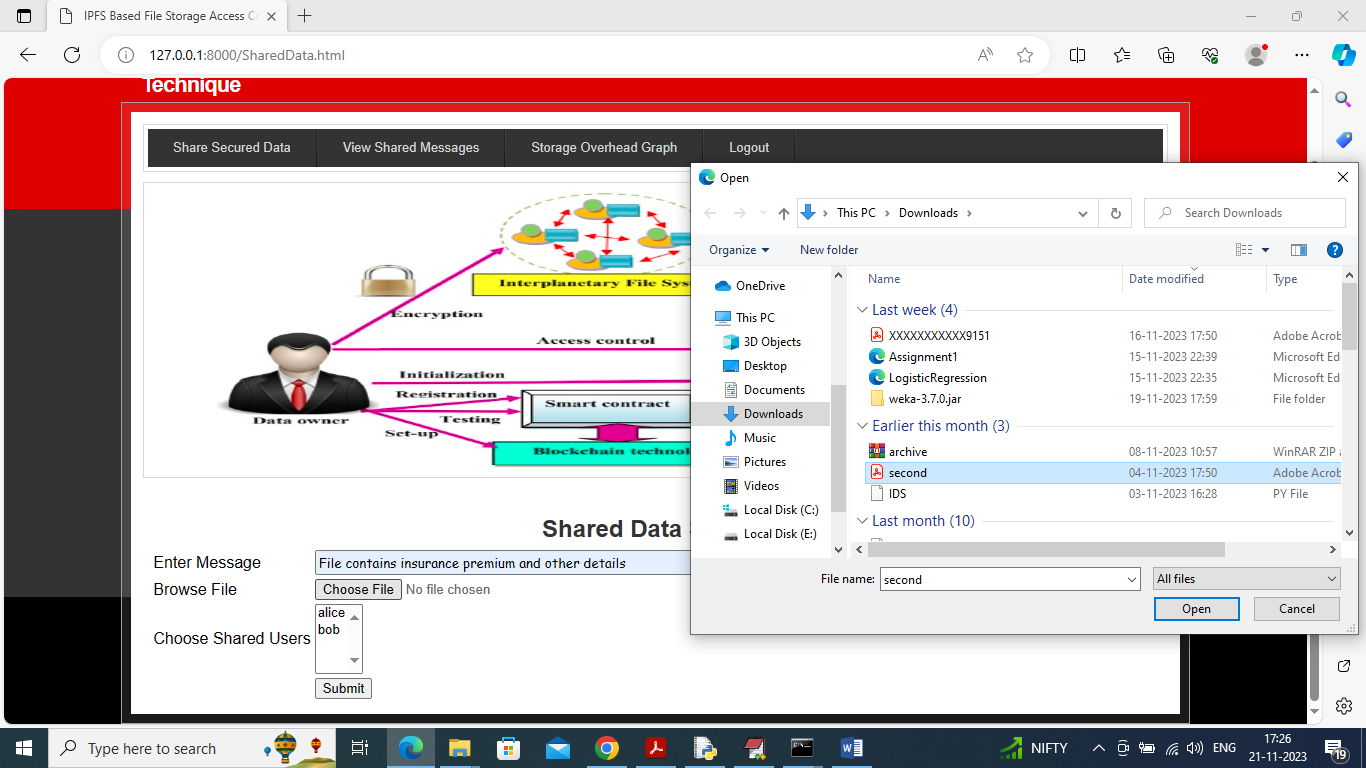
**Login Data owner:** In below screen owner is login and after login will get below page



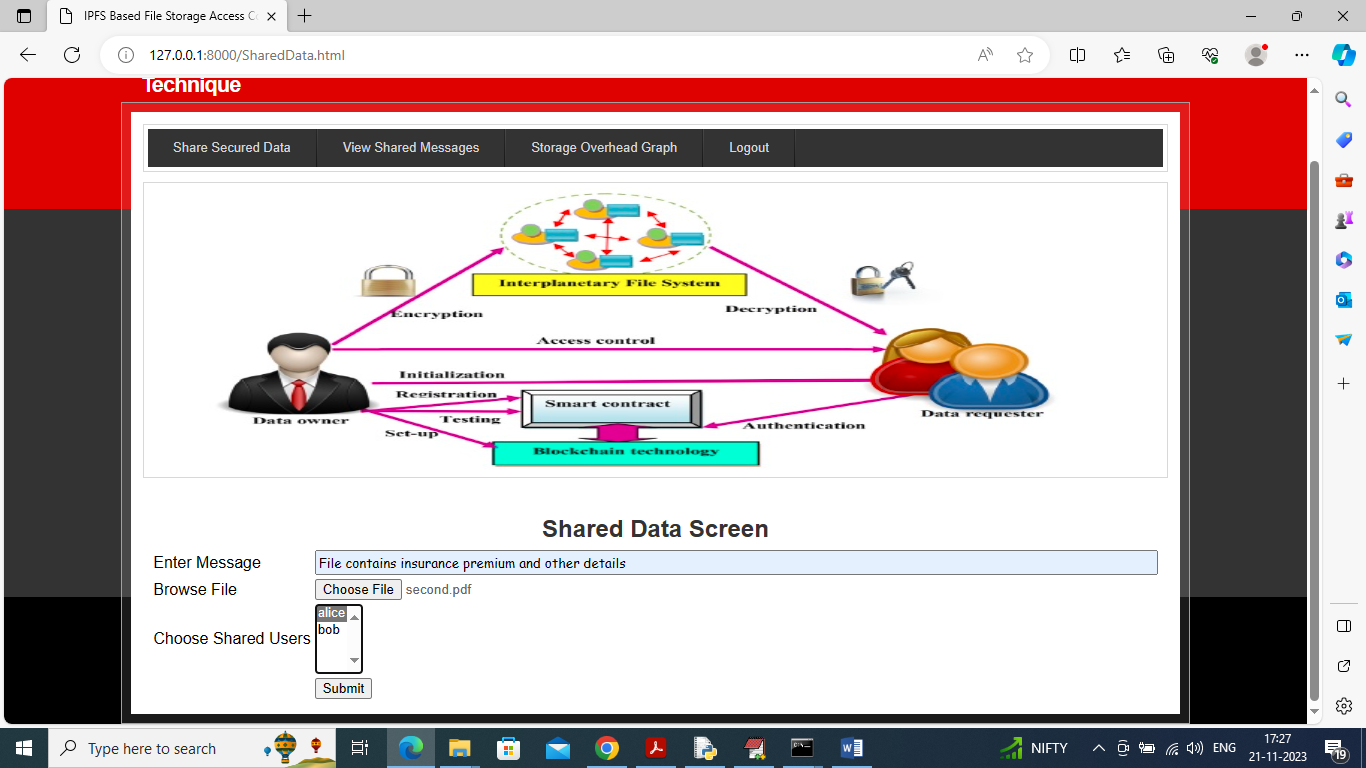
**Share secured data:** In below screen user can click on ‘Shared Secured Data’ link to get below page



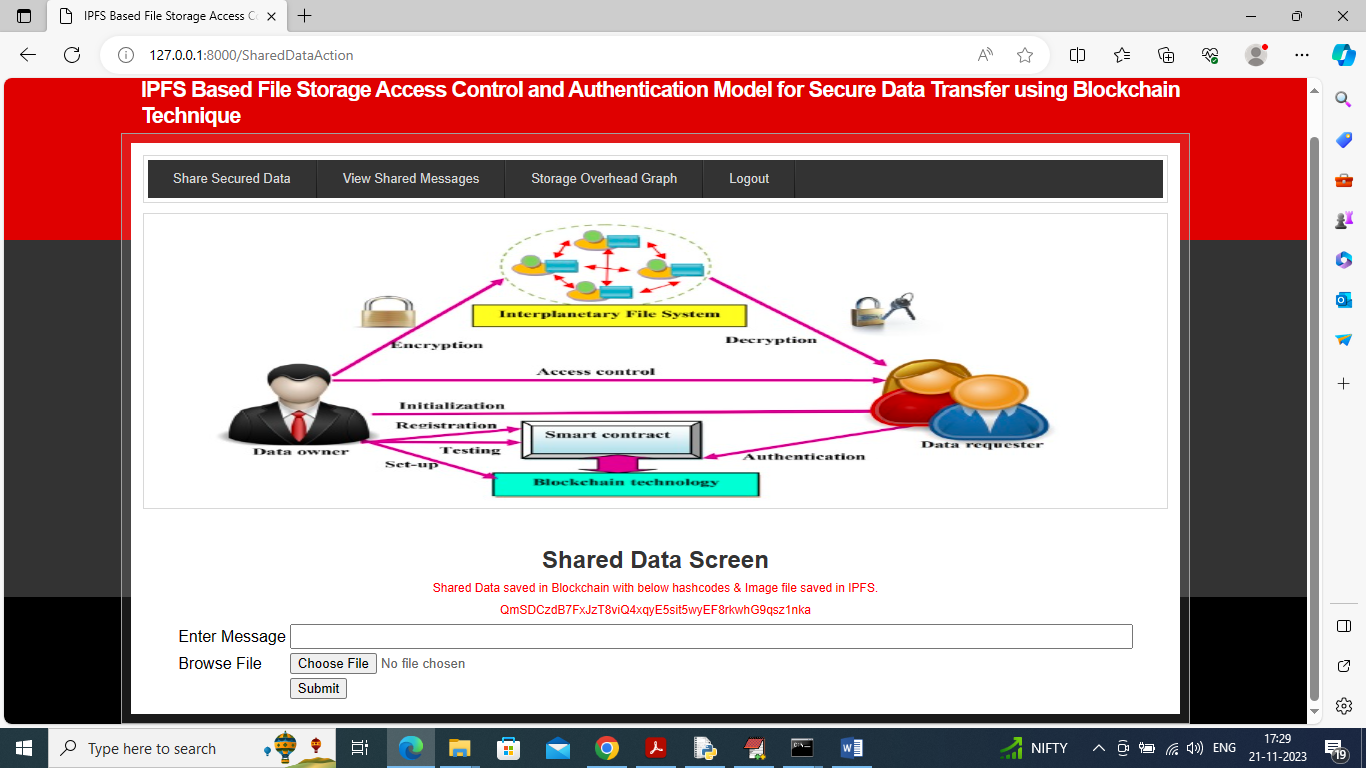
**Upload file:** In below screen data owner can write some description and then upload file which has to share and then he can see list of data requester and from above data requester list he can select desired requester and generate keys to give access to them



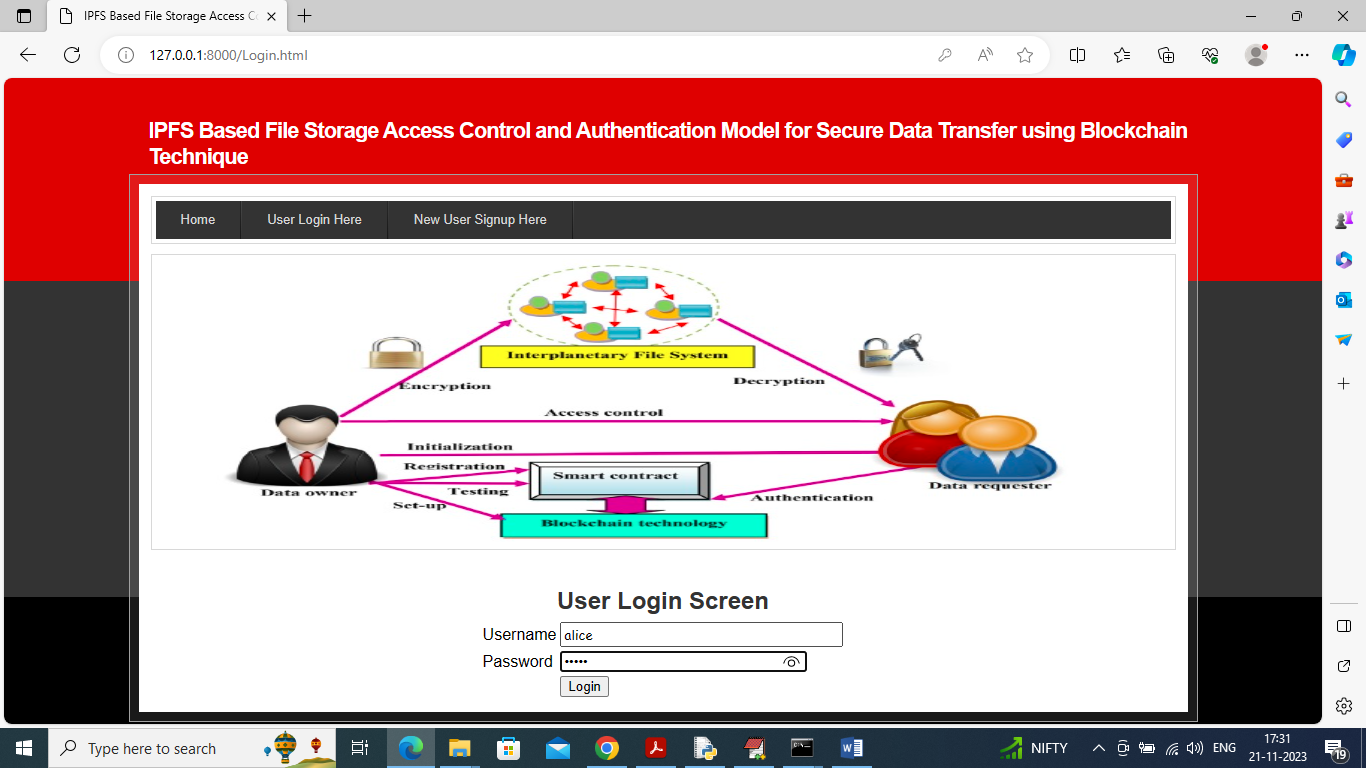
**Sending file to Alice:** In below screen I am giving access to ‘Alice’ user and if you want to give access to multiple users then hold ‘CTRL’ key and select multiple users through mouse and then press ‘Submit’ button to get below page



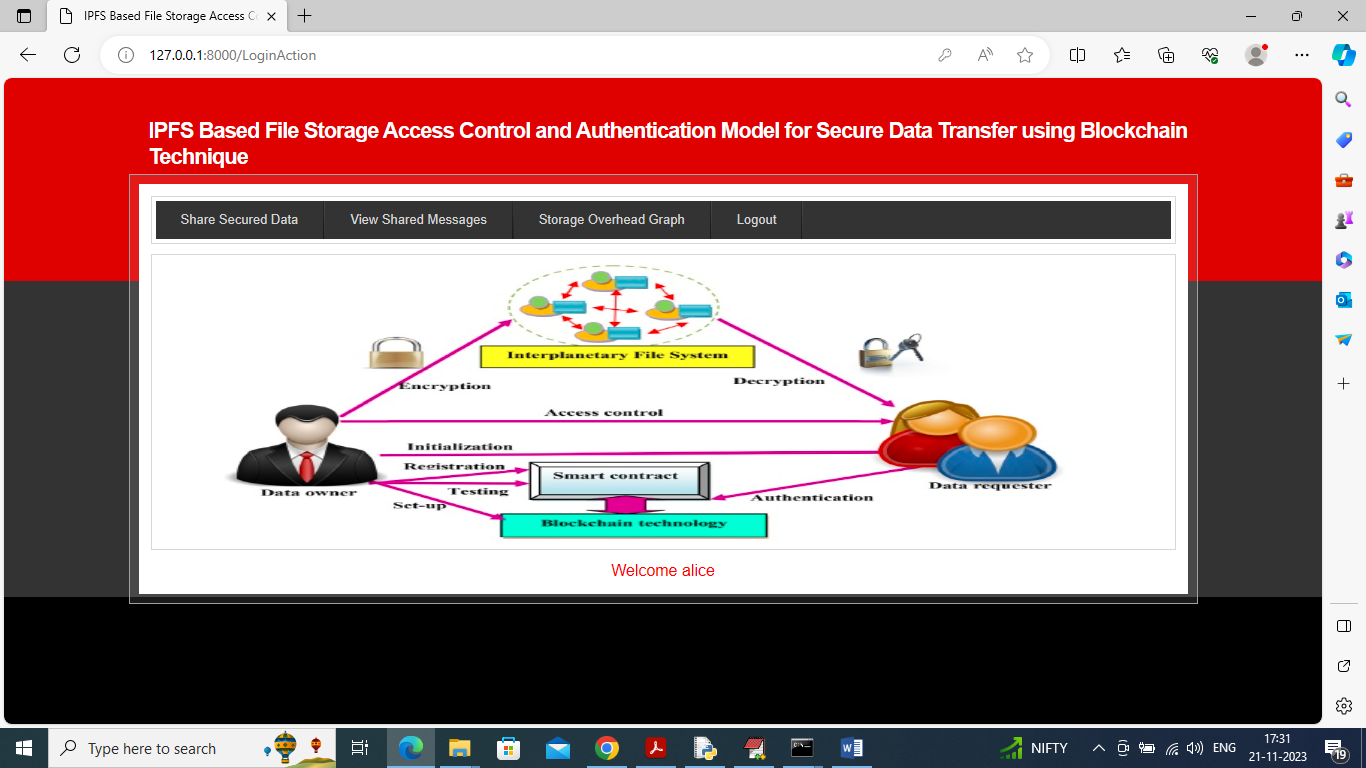
**Data saved in block chain:** In below screen in red color text can see file saved in IPFS and hash code saved in Blockchain and can see the hash code address of stored file. Similarly, by using below screen you can upload as many files as you want. Now logout and login as ‘Alice’ to access shared data



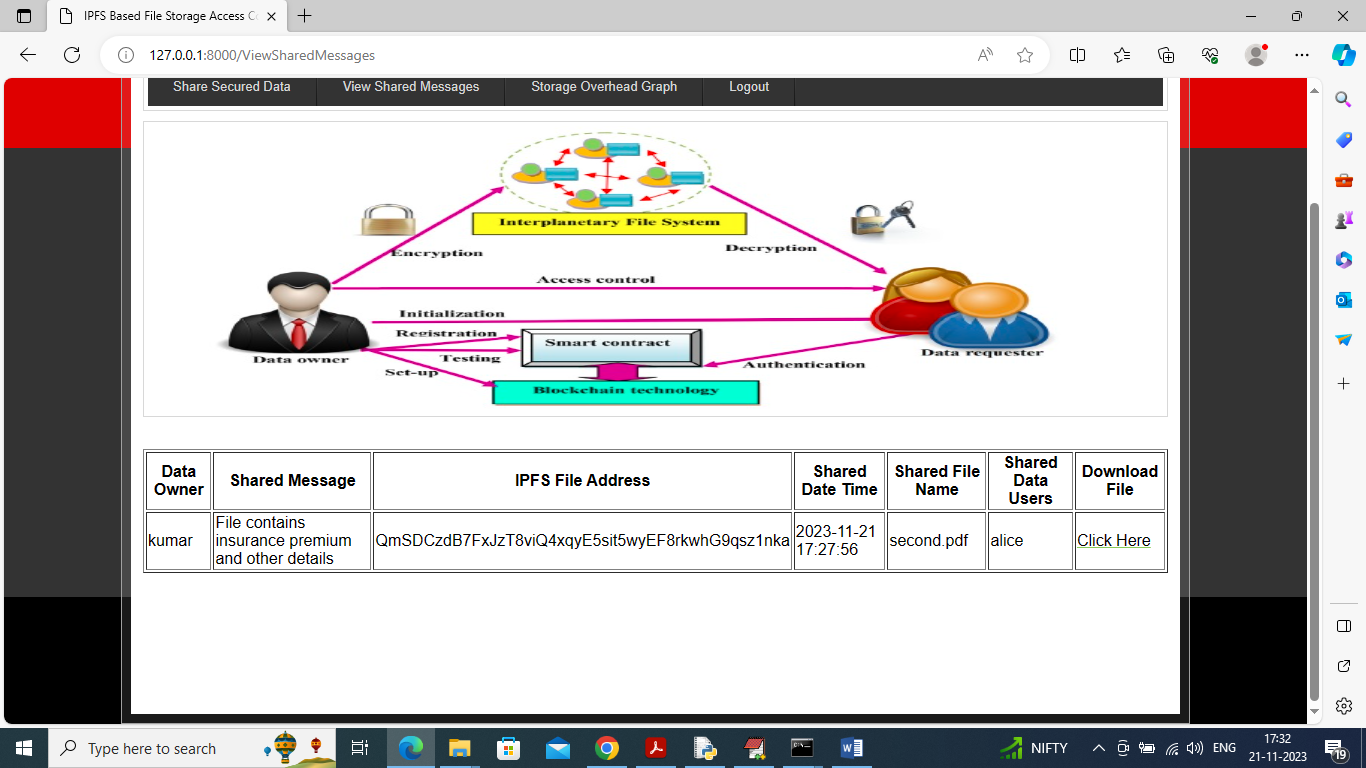
**User login:** In below screen Alice is logged in and after login will get below page



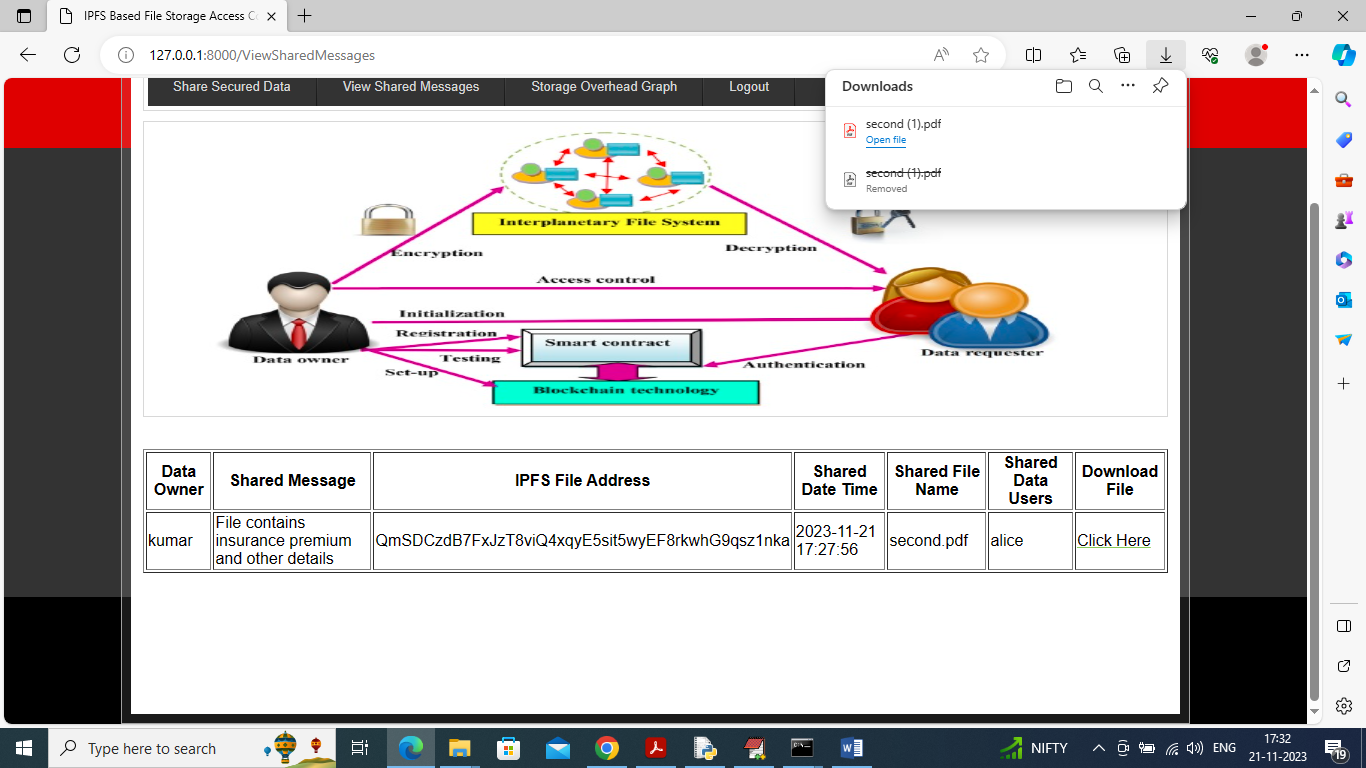
**View Shared Messaged:** In below screen click on ‘View Shared Messaged’.



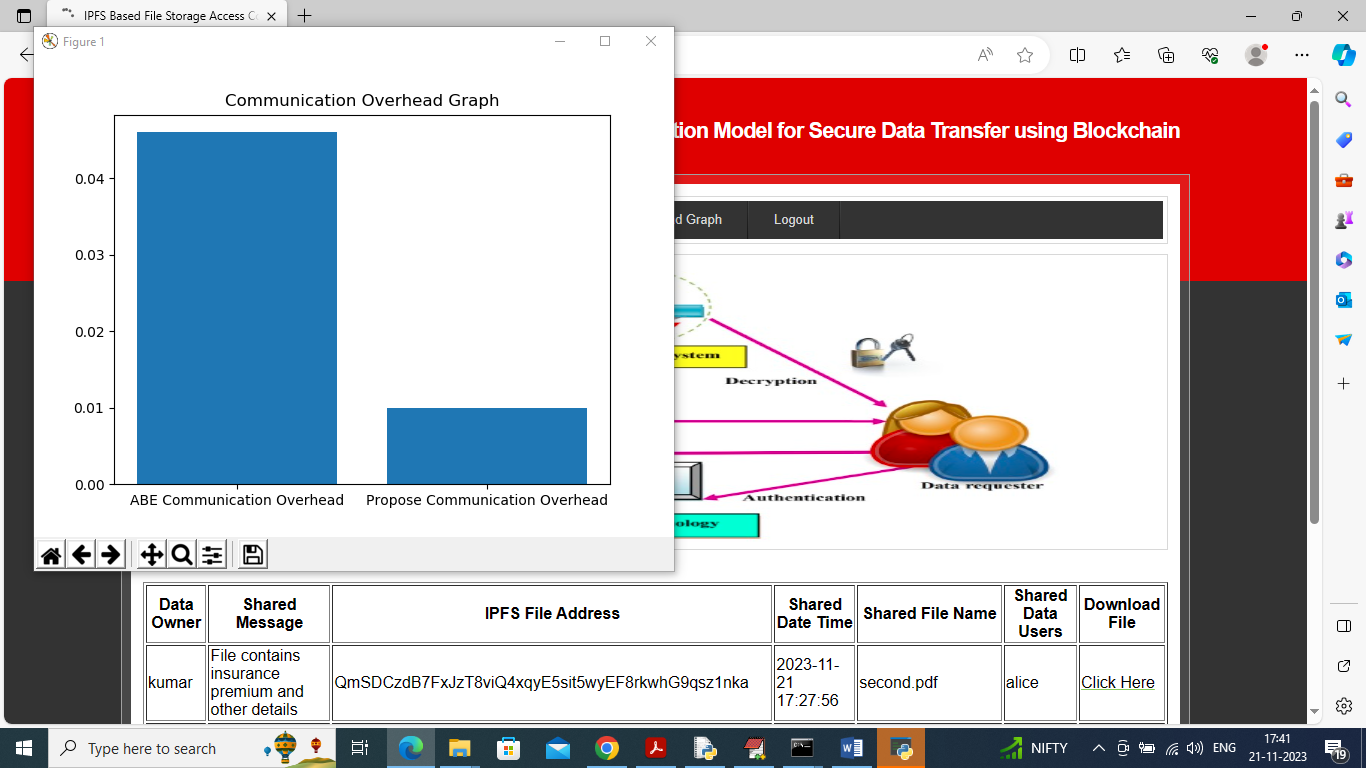
**View file details:** In below table data requester Alice can view all details and then can click on ‘Click Here’ link to decrypt and download file like below screen



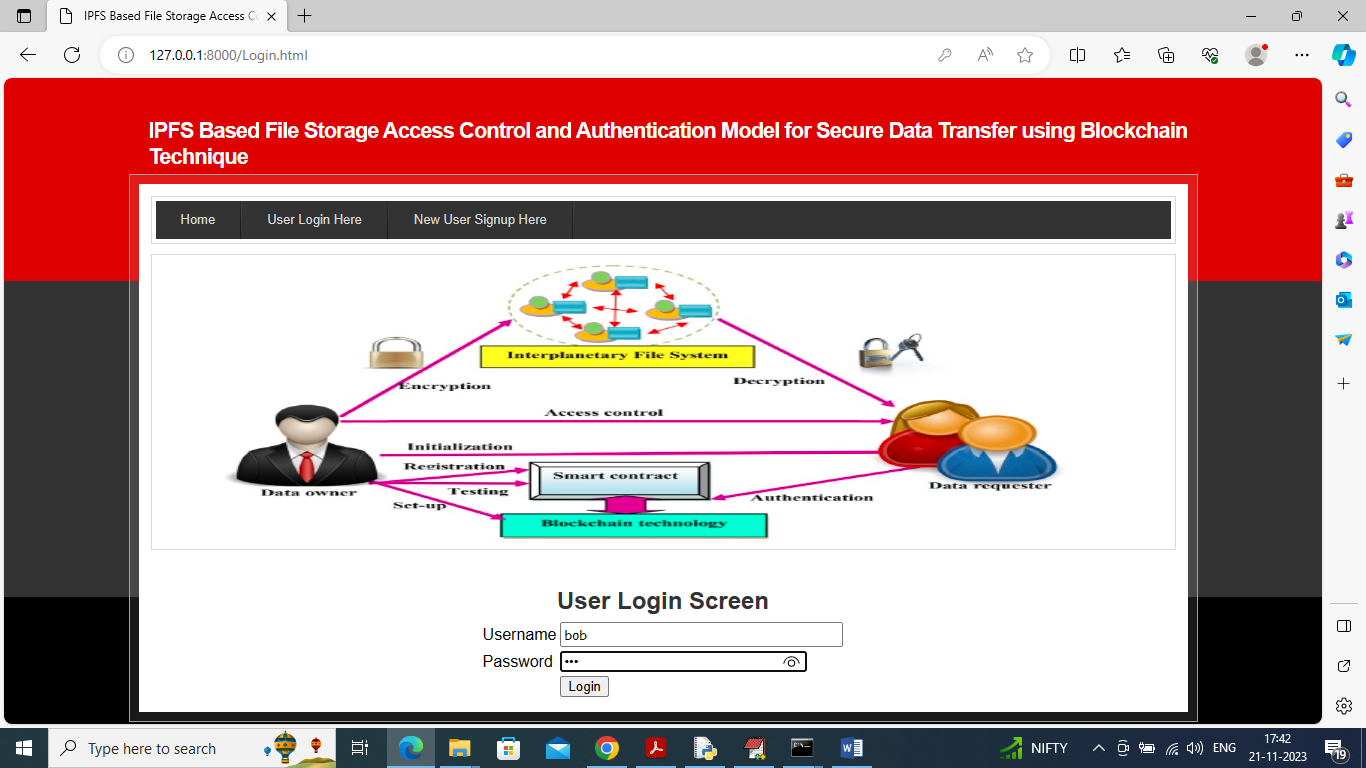
**View downloaded file:** In below screen in browser top dialog box can see file is downloaded and now click on ‘Storage Overhead Graph’ link to get below page



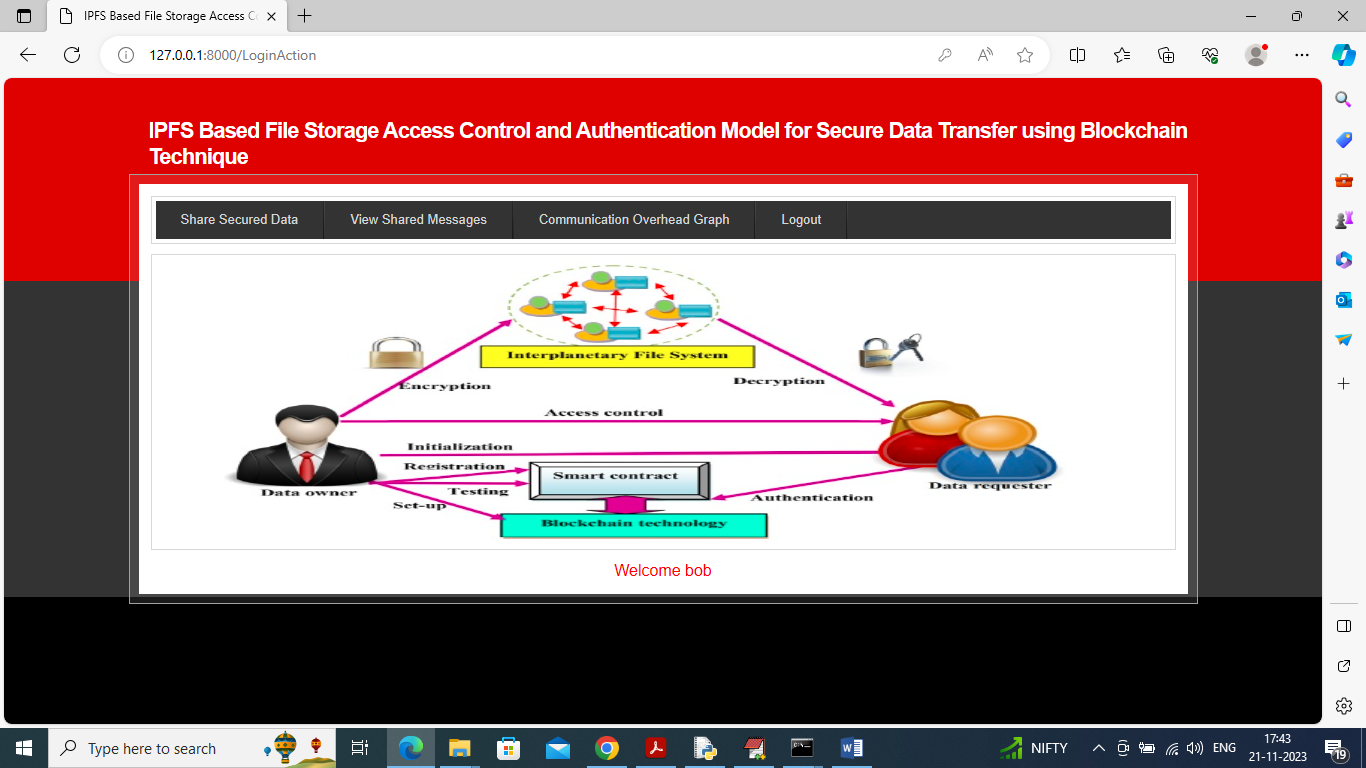
**View graph results:** In above graph x-axis represents ABE and propose algorithms and y-axis represents communication over or execution time and in both algorithms propose has less communication overhead. Now logout and login as ‘bob’ and bob don’t have access keys so he cannot view files.



**Login User Bob:** In above screen bob is login and after login will get below page

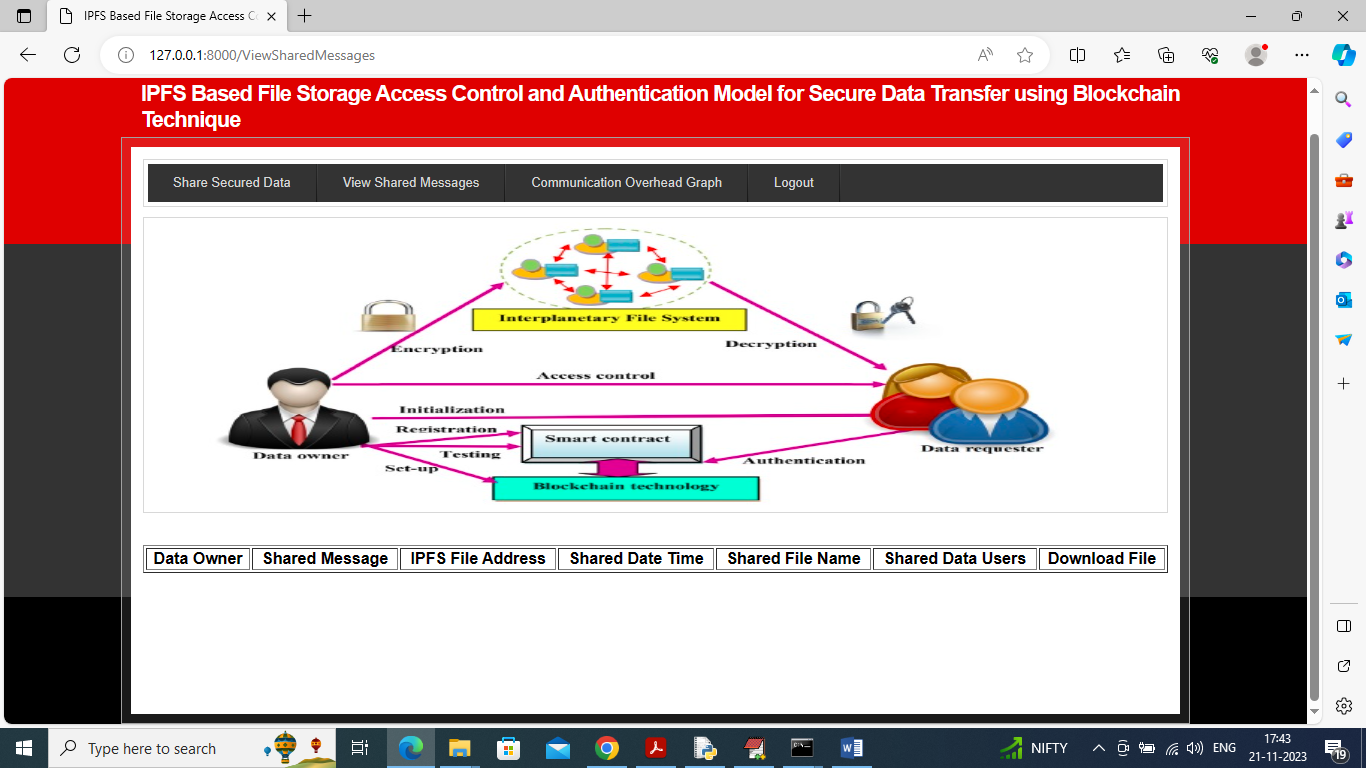


**View Shared Messages’**: In below screen bob can click on ‘View Shared Messages’ link to get below page



**View no access user:** In below screen we can see bob has no access permission so he cannot view or download file.

Similarly, by following above screens you can control shared data permissions



**7. SYSTEM STUDY AND TESTING**

**7.1 Feasibility Study**

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* Economical feasibility
* Technical feasibility
* Social feasibility

**Economic Feasibility**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### **Technical Feasibility**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**Social Feasibility**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**System Testing**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**7.2 Types of Tests**

**7.2.1 Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**7.2.2 Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**7.2.3 Functional testing**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**7.2.4 White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**7.2.5 Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page**.**
* **TEST CASES:**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Result** |
| Input text files | File upload or not | Success |

* **Test cases Model building:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.NO** | **Test cases** | **I/O** | **Expected O/T** | **Actual O/T** | **P/F** |
| 1 | Register Data owner | Enter name, email, password, mobile number, profile, address. | Registration successful | Registration successful | P |
| 2 | Login data owner | Enter email, password | Login successful | Login successful | P |
| 3 | Login data owner | Enter email, password | Login successful | Login Fail | F |
| 4 | Upload file | Enter file path | Data uploaded successful | Data uploaded successful | F |
| 5 | Register Data user | Enter name, email, password, mobile number, profile, address. | Registration successful | Registration successful | P |
| 6 | Login data user | Enter email, password | Login successful | Login successful | P |
| 7 | Login data user | Enter email, password | Login successful | Login Fail | F |

1. **CONCLUSION:**

In conclusion, the project successfully addresses the challenges associated with storing large files on the blockchain by introducing acl-IPFS, a modified InterPlanetary File System (IPFS) integrated with Ethereum smart contracts for access control. By leveraging Ethereum's blockchain capabilities, the acl-IPFS system efficiently manages permissions, allowing users to securely register, access, and modify files while mitigating the scalability issues of traditional blockchain storage. This innovative approach not only ensures a controlled and secure environment for large file sharing but also highlights the synergies between decentralized file systems and blockchain technologies, offering a practical solution for applications requiring access-controlled and tamper-resistant file storage.

**9. FUTURE ENHANCEMENT**

To further enhance the acl-IPFS project, future developments could focus on optimizing access control mechanisms and expanding interoperability. Integration with decentralized identity solutions, such as Self-Sovereign Identity (SSI), could enhance user authentication and authorization processes. Implementing decentralized storage solutions or leveraging off-chain storage options could address scalability concerns associated with large file storage on the Ethereum blockchain. Additionally, exploring privacy-preserving techniques, such as zero-knowledge proofs, could bolster the confidentiality of sensitive data. Continuous collaboration with the broader blockchain and decentralized storage communities will be essential to stay abreast of emerging technologies and ensure acl-IPFS remains a secure and efficient solution for controlled access to large files.

**10.REFERANCES**

[1] G. Zyskind, O. Nathan, and A. Pentland, “Decentralizing privacy: Using blockchain to protect personal data,” in 2015 IEEE Security and Privacy Workshops, May 2015, pp. 180–184.

[2] G. Wood. (2018) Ethereum: A secure, decentralized generalized transaction ledger. [Online

[3] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008.

[4] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, “A taxonomy of blockchain-based systems for architecture design,” in 2017 IEEE International Conference on Software Architecture (ICSA), April 2017, pp. 243–252.

[5] C. K. and T. M. V., Distributed Access Control in Cloud Computing Systems. Wiley-Blackwell, 2016, ch. 35, pp. 417– 432. [Online].

[6] S. Alansari, F. Paci, and V. Sassone, “A distributed access control system for cloud federations,” in 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), June 2017, pp. 2131–2136.

[7] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan, “Smart contract-based access control for the internet of things,” CoRR, vol. abs/1802.04410, 2018. [Online]. Available: [http://arxiv.org/abs/1802. 04410](http://arxiv.org/abs/1802.%2004410).

[8] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, “Zerocash: Decentralized anonymous payments from bitcoin,” in 20.

[9] I. Baumgart and S. Mies, “S/kademlia: A practicable approach towards secure key-based routing,” in Proceedings of the 13th International Conference on Parallel and Distributed Systems - Volume 02, ser. ICPADS ’07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 1–8. [Online]. Available: http://dx.doi.org/10.1109/ICPADS.2007. 4447808.