**GENBA SOPANRAO MOZE COLLEGE OF ENGINEERING BALEWADI, PUNE-411045.**

****

**DEPARTMENT OF COMPUTER ENGINEERING**

**A**

**PROJECT (STAGE-II) REPORT ON**

**“SAFEGUARD: Blockchain Based File Storage System”**

Submitted in the partial fulfilment of the requirement for BE

in Computer Engineering during 2024-25

**Submitted by**

**Name Seat No.**

Rupika Patil B400300260

Esha Nalawade B400300249

Tejal Tayade B400300302

**UNDER THE GUIDANCE OF**

**Dr. Vajid Khan**

**GENBA SOPANRAO MOZE COLLEGE OF ENGINEERING BALEWADI, PUNE-411045.**

****

**DEPARTMENT OF COMPUTER ENGINEERING**

**CERTIFICATE**

**This is to certify that the project entitled**

**“SAFEGUARD: Blockchain Based File Storage System "**

**Submitted by**

**Name Seat No.**

Rupika Patil B400300260

Esha Nalawade B400300249

Tejal Tayade B400300302

It is a bonafide work carried out by them under the guidance of **Dr. Vajid Khan** and is approved for the partial fulfilment of the requirement of SPPU for the award of BE in Computer Engineering.

Date:…./…../……

**Dr. Vajid Khan Prof.Sai Sudha Dorabala Dr. Ratnarajkumar Jambi**

**(Internal Guide) (HOD, CS) (Principal, GSMCOE)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Internal Examiner External Examiner

**ACKNOWLEDGMENT**

This is a great pleasure & immense satisfaction to express our deepest sense of Gratitude & thanks to everyone who has directly or indirectly helped us in partial completion of Project stage-II successfully. It gives us great pleasure in presenting project report on:

**“SAFEGUARD: Blockchain Based File Storage System”**

We would like to take this opportunity to thank our guide **Dr. Vajid Khan** for giving us the guidance & all the help that we needed. We are really grateful to him for his kind support. We would also like to express our gratitude towards our parents for their kind cooperation and encouragement which helped us in completion of this project.

We are grateful to our HOD Prof. Sai Sudha Dorabala for her kind support & guidance. We express our sincere thanks to our beloved Principal, Dr. Ratnaraja Kumar Jambi & the management for providing us the resources required in successful completion of the project.

Last but not the least, many thanks go to the coordinator of the project, **Dr. Vajid Khan** who has invested his full effort in guiding the team for achieving the goal. We are also grateful to his dispensable support and suggestions. We express our gratitude to all the teaching, non-teaching & our dear friends for their support.

Regards,

Rupika Patil

Esha Nalawade

Tejal Tayade

I

**ABSTRACT**

As data generation grows exponentially across industries, traditional centralized storage solutions face critical limitations, including security vulnerabilities, high operational costs, and lack of user control. These systems are prone to single points of failure, data breaches, and censorship, often controlled by a few major providers. This project explores the design and implementation of a Decentralized Storage Network (DSN), aimed at addressing these challenges by distributing data across a peer-to-peer network. Leveraging blockchain technology and advanced encryption methods, this decentralized approach enhances data security, resilience, and transparency. By fragmenting and distributing data across multiple nodes, the system mitigates the risk of data loss due to node failure or cyberattacks. Users maintain control over their data, which is protected by encryption, and storage transactions are secured through an immutable ledger, ensuring data integrity and privacy. In addition, the decentralized architecture reduces operational costs by utilizing unused storage capacity across the network, offering a more cost-efficient alternative to traditional cloud storage. This project outlines the key components and protocols needed to develop a secure, scalable, and user-controlled storage solution, demonstrating the potential to disrupt centralized data storage models and establish a more resilient, equitable, and privacy-focused storage ecosystem.

**KEYWORDS:** Decentralized storage, blockchain, storage networks, blockchain storage, Cloud Technology.

II

**TABLE OF CONTENTS**

**CHAPTER NO. TITLE PAGE NO.**

**ACKNOWLEDGEMENT I**

**ABSTRACT II**

**TABLE OF CONTENTS III**

**LIST OF FIGURES IV**

**LIST OF TABLES V**

**1. INTRODUCTION 9**

**1.1 Overview of the Project 10**

**1.2 Problem Statement 10**

**1.3 Scope of the Project 10**

**2. LITERATURE SURVEY 11**

**3. METHODOLOGY 14**

**3.1 Existing System 14**

**3.2 Proposed System 17**

**4. SYSTEM DESIGN 20**

**5. SYSTEM REQUIREMENTS 27**

**5.1 Software Requirements 28**

**5.2 Hardware Requirements 29**

**6. IMPLEMENTATION 30**

**6.1 Tools Used 30**

**6.2 Algorithms Used 32**

**6.3 Functions Used 36**

**7. RESULTS 40**

**8. SYSTEM TESTING 42**

**8.1 Advantages 44**

**8.2 Disadvantages 45**

**8.3 Applications 46**

**8.4 Challenges 48**

**9. CONCLUSION 49**

**FUTURE SCOPE 50**

**10. REFERENCES 51**

III

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **FIGURE NAME** | **PAGE NO.** |
| **Fig 1** | **System Architecture** | **20** |
| **Fig 2** | **DFD 0** | **21** |
| **Fig 3** | **DFD 1** | **22** |
| **Fig 4** | **UML** | **23** |
| **Fig 5** | **Sequence Diagram** | **24** |
| **Fig 6** | **Result** | **40** |

**IV**

**LIST OF TABLES**

**V**

|  |  |  |
| --- | --- | --- |
| **TABLE NO.** | **TITLE** | **PAGE NO.** |
| **1** | **Comparison Table** | **16** |
| **2** | **Function Table** | **38** |
| **3** | **Test Cases** | **43** |

**CHAPTER I**

**INTRODUCTION**

In today’s digital-first world, data has become the most valuable asset. Every sector—from healthcare to finance, education, government, and personal use—relies heavily on secure data storage and retrieval. Traditionally, file storage systems have been either local (hard drives, network-attached storage) or cloud-based (Google Drive, Dropbox, AWS S3). While these systems have proven useful and scalable, they carry significant limitations. Centralized storage creates a single point of failure, making data vulnerable to hacking, tampering, data loss, or unauthorized access.

Traditional file storage systems, whether cloud-based or local, often suffer from issues like centralization, vulnerability to data breaches, unauthorized access, and potential data loss. To address these challenges, this project introduces a **Blockchain-based File Storage System**, which leverages the decentralized and tamper-resistant nature of blockchain technology to offer a more secure and transparent method of storing files.

This project demonstrates how blockchain can be used not just for cryptocurrencies, but also for decentralized data storage. It is implemented using Python and Flask, providing users with a simple web interface for uploading and downloading files. Each uploaded file is embedded into a block on the blockchain, along with metadata such as the filename, user information, and timestamp. This ensures that the data becomes immutable and traceable, as every block is cryptographically linked to the previous one.

A key feature of the system is its **Proof of Work (PoW)** consensus mechanism, which secures the blockchain by requiring computational effort before a new block is added. The project compares two PoW strategies—random nonce and incremental nonce—to evaluate their efficiency. Furthermore, all files are stored directly **on-chain**, showcasing a fully decentralized approach without relying on third-party storage services.

By integrating core blockchain principles—decentralization, immutability, and security—this project offers a foundational model for building trustless and robust file storage systems that can be applied in various domains, from legal documentation to academic archives and beyond.

* 1. **Overview of the Project**

The **Blockchain-based File Storage** project is a decentralized web application developed to provide secure and tamper-proof file storage using blockchain technology. Traditional centralized storage systems are often vulnerable to data breaches, single points of failure, and unauthorized access. This project addresses those concerns by leveraging a peer-to-peer (P2P) network where files are distributed across multiple nodes, ensuring data redundancy and availability. Each file upload is recorded as a transaction on the blockchain, making the storage process transparent and immutable. The system uses a Proof of Work (PoW) consensus mechanism to validate and secure transactions, further enhancing trust in the data stored. Built with Python and Flask for the backend, along with HTML and CSS for the frontend, the application offers a user-friendly interface for uploading and downloading files. Modules such as Block.py, Blockchain.py, and peer.py handle the core blockchain logic and peer communication. This project serves as a practical implementation of decentralized storage, demonstrating how blockchain can be used beyond cryptocurrency to improve data security and integrity in real-world applications.

* 1. **Problem Statement**

Centralized file storage is prone to data loss, tampering, and unauthorized access. This project aims to develop a blockchain-based file storage system that ensures secure, decentralized, and tamper-proof file management.

* 1. **Scope of the Project**

The potential scope for a project: The project aims to create a comprehensive and scalable system designed to address the security, affordability, and efficient management of data in educational institutions. The scope of the project would cover several key areas, ensuring that the data infrastructure within schools, colleges, and universities is secure, scalable, and cost effective.

**CHAPTER II**

**LITERATURE SURVEY**

**1) A Comprehensive Survey on Blockchain-Based Decentralized Storage Networks [Muhammad Irfan Khalid1 , Ibtisam Ehsan 2 , Ayman Khallel Al-Ani 3 , Jawaid Iqbal 4 , Saddam Hussain 5, Syed Sajid Ullah6, And Nayab7]**

**Description:**

Blockchain are a new approach to creating distributed networks that were first introduced in 2008. It allows the formation of peer-to-peer networks based on consensus, forming chains from accepted blocks without requiring a central authority or centralized controller. A prominent application of this technology is its use in decentralized storage systems. Individuals in decentralized storage networks rent unused hardware storage space to other individuals. A decentralized network utilizing end-to-end encryption eliminates the risk of data loss associated with centralized data control by enabling clients to transmit their files securely. The storage providers must prove that they have kept unaltered files in this network for this time. Many studies have been conducted in this specific domain, most targeting storage capacity and efficiency, but a security, integrity and privacy loop hole need to be addressed.

**Advantages:**

* Enhanced Privacy & Security: Data is encrypted and distributed, reducing risks of breaches.
* Censorship Resistance: No central authority can alter or remove data.
* Data Integrity: Immutable data ensures it cannot be tampered with.
* Cost Efficiency: Eliminates the need for central servers, reducing costs.

#### 2) Secure Data Storage and Recovery in Industrial Blockchain Network Environments [W. Liang, Yongkai Fan, +2 Authors J. Gaudiot Published In IEEE Transactions On… 13 January 2020, Computer Science, Engineering Zhang.]

#### Description :

The massive redundant data storage and communication in network 4.0 environments have issues of low integrity, high cost, and easy tampering. To address these issues, in this article, a secure data storage and recovery scheme in the blockchain-based network is proposed by improving the decentration, tampering-proof, real-time monitoring, and management of storage systems, as such design supports the dynamic storage, fast repair, and update of distributed data in the data storage system of industrial nodes. A local regenerative code technology is used to repair and store data between failed nodes while ensuring the privacy of user data. That is, as the data stored are found to be damaged, multiple local repair groups constructed by vector code can simultaneously yet efficiently repair multiple distributed data storage nodes.

**Advantages:**

* Secure data storage and recovery in industrial blockchain network environments offers numerous advantages, including enhanced security through immutability and encryption, and a decentralized architecture that reduces the risk of tampering or single points of failure.
* The distributed nature of blockchain ensures redundant data storage, allowing for quick and reliable recovery in case of system failures.

**3) "A Data Sharing Scheme for GDPR-Compliance Based on Consortium Blockchain" [Yangheran Piao & Kai Ye & Xiaohui Cui, 2021. Future Internet, MDPI, vol. 13(8), pages 1-17, August.]**

#### Description :

After the General Data Protection Regulation (GDPR) was introduced, some organizations and big data companies shared data without conducting any privacy protection and compliance authentication, which endangered user data security, and were punished financially for this reason. This study proposes a blockchain-based GDPR compliance data sharing scheme, aiming to promote compliance with regulations and provide a tool for interaction between users and service providers to achieve data security sharing. It has superiority in efficiency and privacy protection compared with other schemes.

**Advantages :**

* It ensures robust data security through encryption and controlled access, protecting personal information while providing transparency with immutable audit trails for easy compliance verification.
* Automated smart contracts enforce GDPR rules, such as data retention and consent management, streamlining operations. The use of off-chain storage enables compliance with the "right to be forgotten," while the system tracks cross-border data flows to ensure GDPR standards are upheld across different jurisdictions.

**CHAPTER III**

**METHODOLOGY**

**3.1 Existing System**

**3.1.1 Centralized Storage Solutions**

Traditional file storage systems, such as Google Drive, Dropbox, and OneDrive, operate on centralized architectures where data is stored on servers managed by third-party providers. While these systems offer user-friendly interfaces and scalability, they are accompanied by several drawbacks:

* **Single Point of Failure**: Centralized servers are susceptible to outages or failures, potentially leading to data inaccessibility.
* **Data Breaches**: Central repositories are prime targets for cyberattacks, risking unauthorized access to sensitive information.
* **Limited Transparency**: Users have minimal insight into how their data is managed, stored, or shared by service providers.
* **Data Ownership Concerns**: Users often relinquish control over their data, subject to the terms and conditions set by the providers.
* **Censorship and Manipulation:** Data stored on centralized servers can be altered, censored, or deleted without the user's knowledge.

### ****3.1.2 Limitations of Existing Systems****

The centralized nature of these storage solutions introduces vulnerabilities:

* **Security Risks**: Centralized databases are attractive targets for hackers, leading to potential data theft or loss.
* **Data Integrity Issues**: Without robust verification mechanisms, data can be altered without detection.

**Difference Between Centralized and Decentralized System:**

| **Basis** | **Centralization** | **Decentralization** |
| --- | --- | --- |
| **Meaning** | The concentration of authority at the top level is known as Centralizaon. | The evenly and systematic distribution of authority at all levels is known as Decentralization. |
| **Delegation of authority** | There is no delegation of authority as all the authority for taking decisions is vested in the hands of top-level management. | There is a systematic delegation of authority at all levels. |
| **Suitability** | It is suitable for small organisations. | It is suitable for large organisations. |
| **Freedom of decision making** | There is no freedom of decision-making at the middle and lower level. | There is freedom of decision-making at all levels of management. |
| **Flow of Information** | There is a vertical flow of information. | There is an open and free flow of information. |
| **Employee Motivation** | Employees are demotivated as compared to decentralization. | Employees are motivated as compared to centralization. |
| **Conflict in Decision** | e are least chances of any conflict in decision as only top-level management is involved. | There are chances of conflict in decision as many people are involved. |
| **Burden** | The burden of work is not shared and only one group carries the burden. | The burden of work is shared amongst all levels. |

**Table No.1**

**3**.**2 Proposed System**

### ****Overview****

The proposed system is a decentralized file storage application utilizing blockchain technology to ensure secure, immutable, and tamper-proof file management. By distributing data across a peer-to-peer (P2P) network and recording transactions on a blockchain, the system mitigates the risks associated with centralized storage

### ****3.2.1 Key Features****

* **Decentralization**: Eliminates reliance on a central authority by distributing data across multiple nodes, enhancing fault tolerance and availability.
* **Immutable Ledger**: Utilizes blockchain to record file transactions, ensuring a transparent and unalterable history of data operations.
* **Enhanced Security**: Employs cryptographic techniques and consensus mechanisms to protect data from unauthorized access and tampering.
* **User Autonomy**: Empowers users with control over their data, eliminating dependence on third-party providers.
* **Blockchain Integration:** Each file upload is recorded as a transaction on the blockchain, providing a transparent and immutable history of file actions.
* **User Control:** Users retain full control and ownership of their files, as there is no dependency on third-party providers to store or retrieve data.
* **Transparency and Trust:** The decentralized ledger allows anyone in the network to verify the authenticity and history of stored files, fostering trust without the need for intermediaries.

### ****3.3 System Architecture****

* **Frontend**: Developed using HTML and CSS, providing a user-friendly interface for file upload and download operations.
* **Backend**: Implemented in Python using the Flask framework, handling file processing, blockchain interactions, and network communications.
* **Blockchain Components**:
  + **Block.py**: Defines the structure and properties of individual blocks in the blockchain.
  + **Blockchain.py**: Manages the chain of blocks, including block addition and validation processes.
  + **Peer.py**: Handles peer-to-peer networking, facilitating communication between nodes in the decentralized network.

### ****3.3.1 Operational Workflow****

1. **File Upload**:
   * User selects a file to upload via the web interface.
   * The system computes the file's hash and creates a new block containing the file data and metadata.
   * A Proof of Work (PoW) algorithm validates the block, which is then added to the blockchain.
   * The file is distributed across the P2P network, ensuring redundancy and availability.
2. **File Download**:
   * User requests a file through the interface.
   * The system retrieves the corresponding block from the blockchain and verifies its integrity using the stored hash.
   * The file is reconstructed and made available for download.

## ****3.3.2 Advantages of the Proposed System****

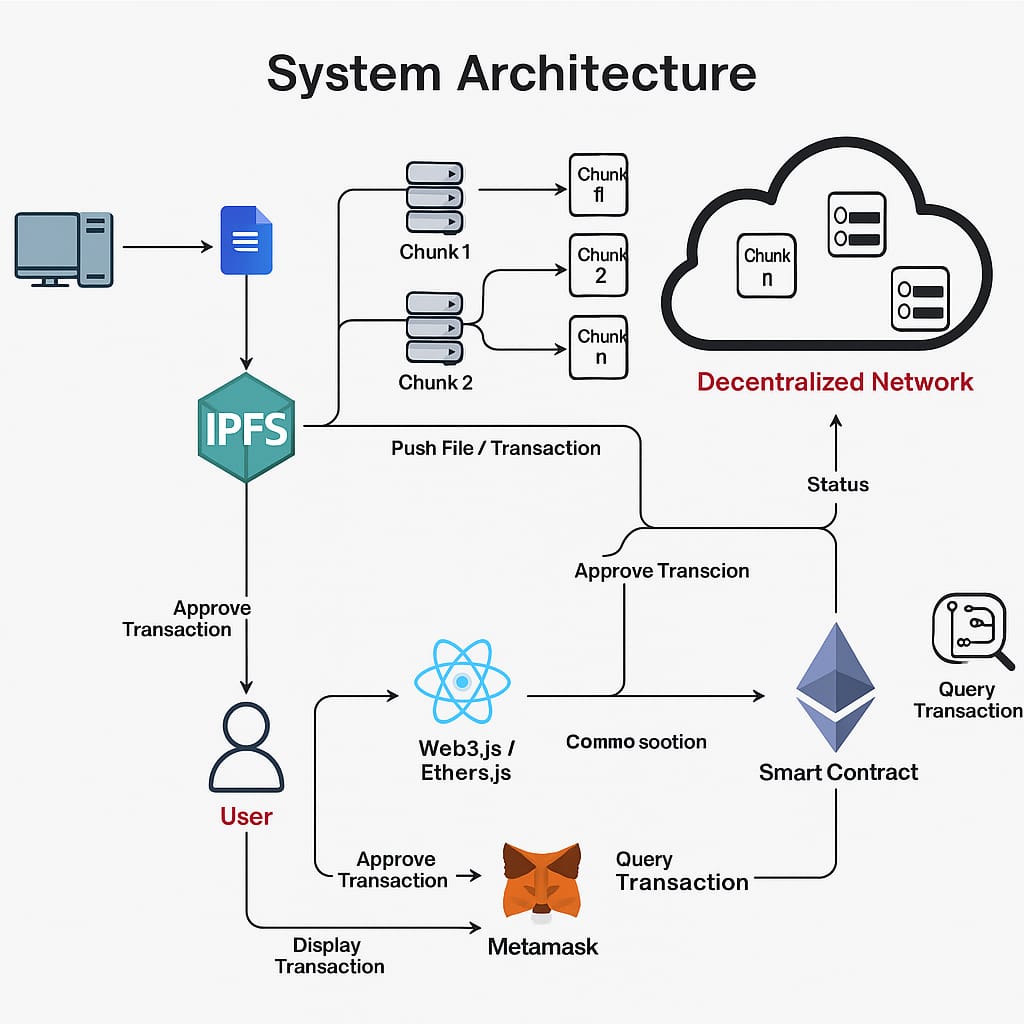
* **Security**: Decentralization and cryptographic techniques protect against unauthorized access and data breaches.
* **Data Integrity**: Blockchain's immutable ledger ensures that any alterations to data are easily detectable.
* **Transparency**: All transactions are recorded on the blockchain, providing a clear audit trail.
* **Resilience**: Distributed storage enhances system robustness against failures and attacks.
* **User Empowerment**: Users maintain control over their data without reliance on third-party services.

The blockchain-based file storage system presents a robust alternative to traditional centralized storage solutions. By leveraging the decentralized nature of blockchain, the system enhances data security, integrity, and user autonomy. This approach addresses the inherent vulnerabilities of centralized systems, offering a more resilient and trustworthy platform for file storage and management.

**CHAPTER IV**

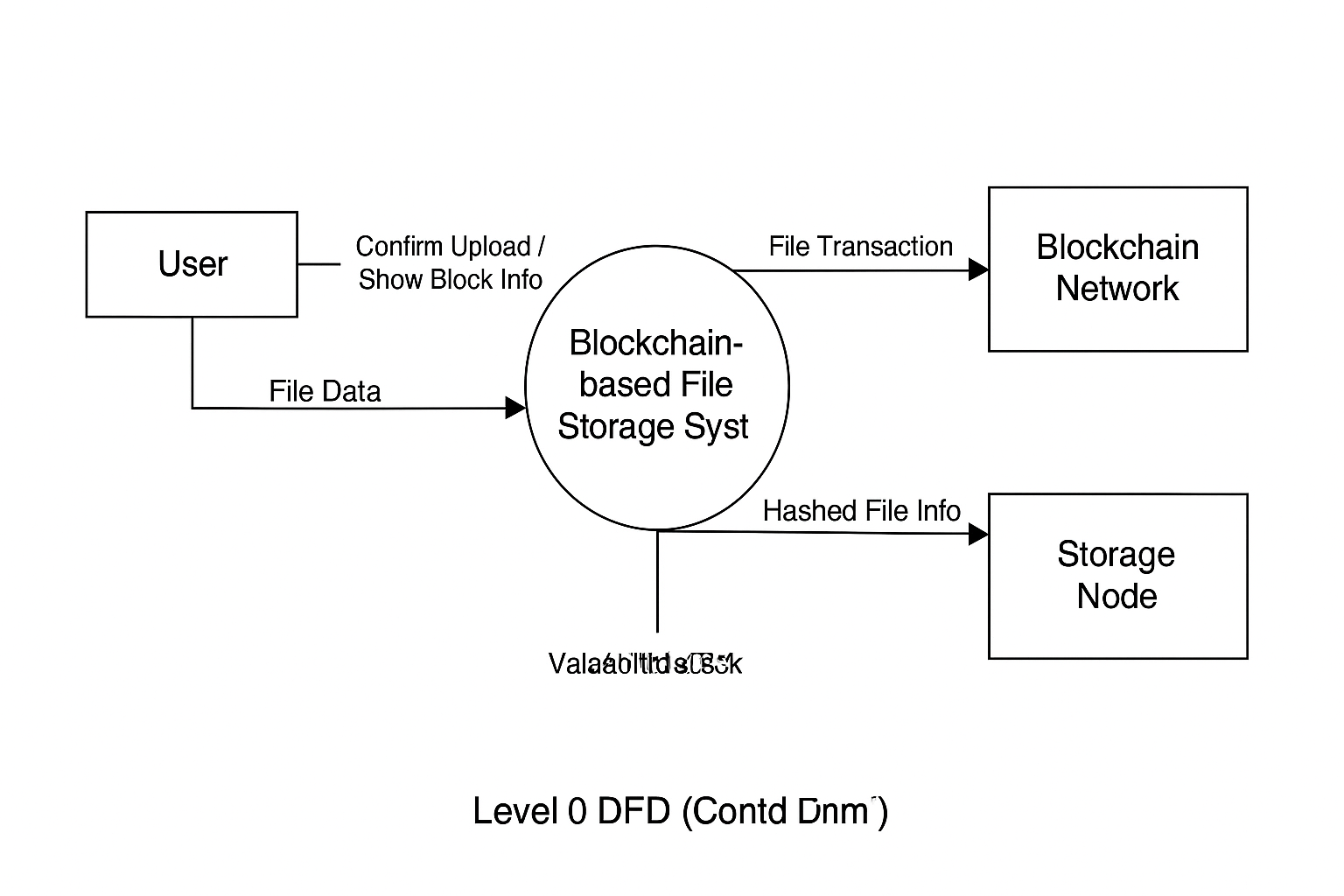
**SYSTEM DESIGN**

**System Architecture:**



**Fig no:01**

**DFD 0 :**

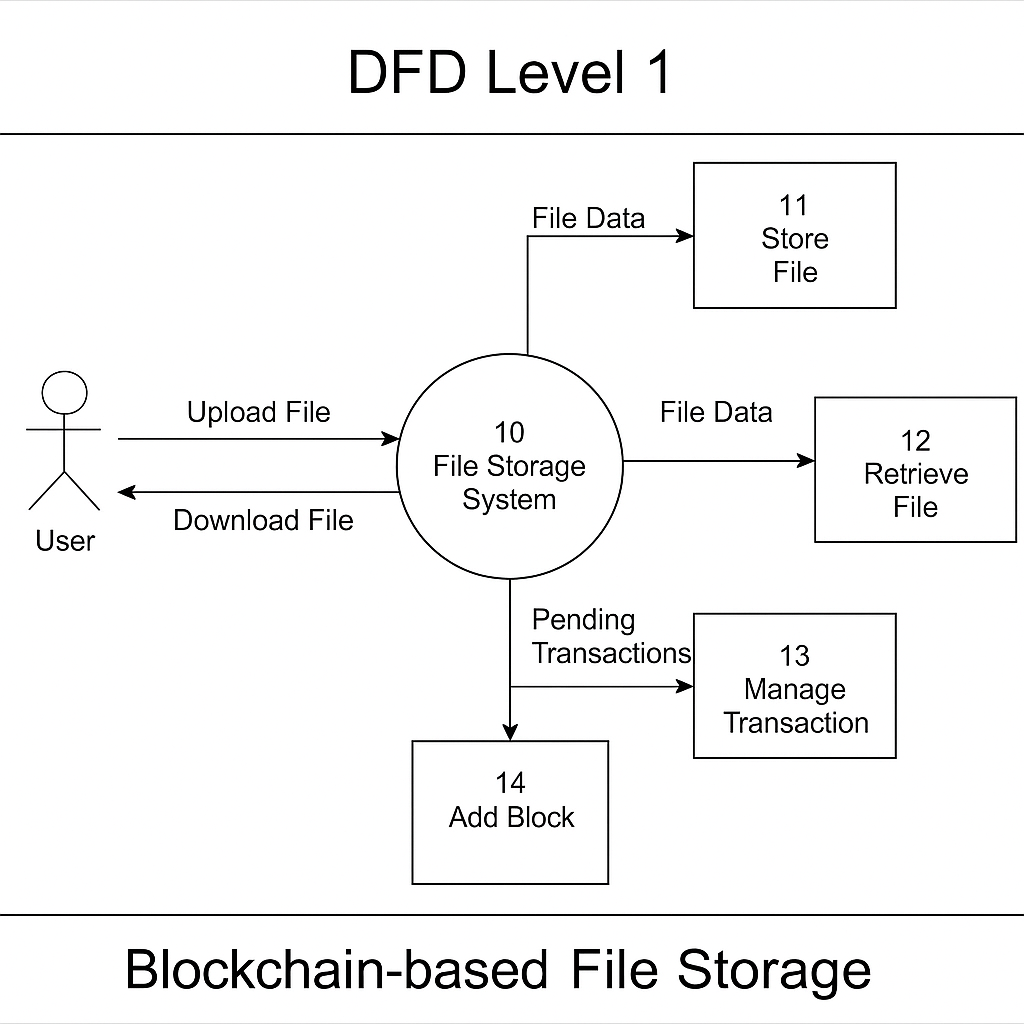


**DFD LEVEL 0**

**Fig no:02**

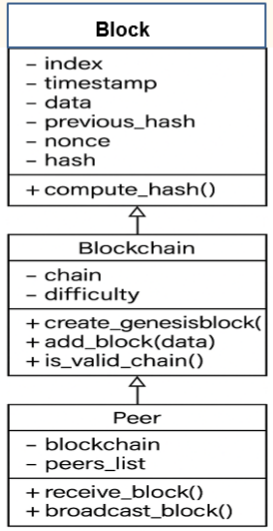
**Fig No. 2**

**DFD 1 :**



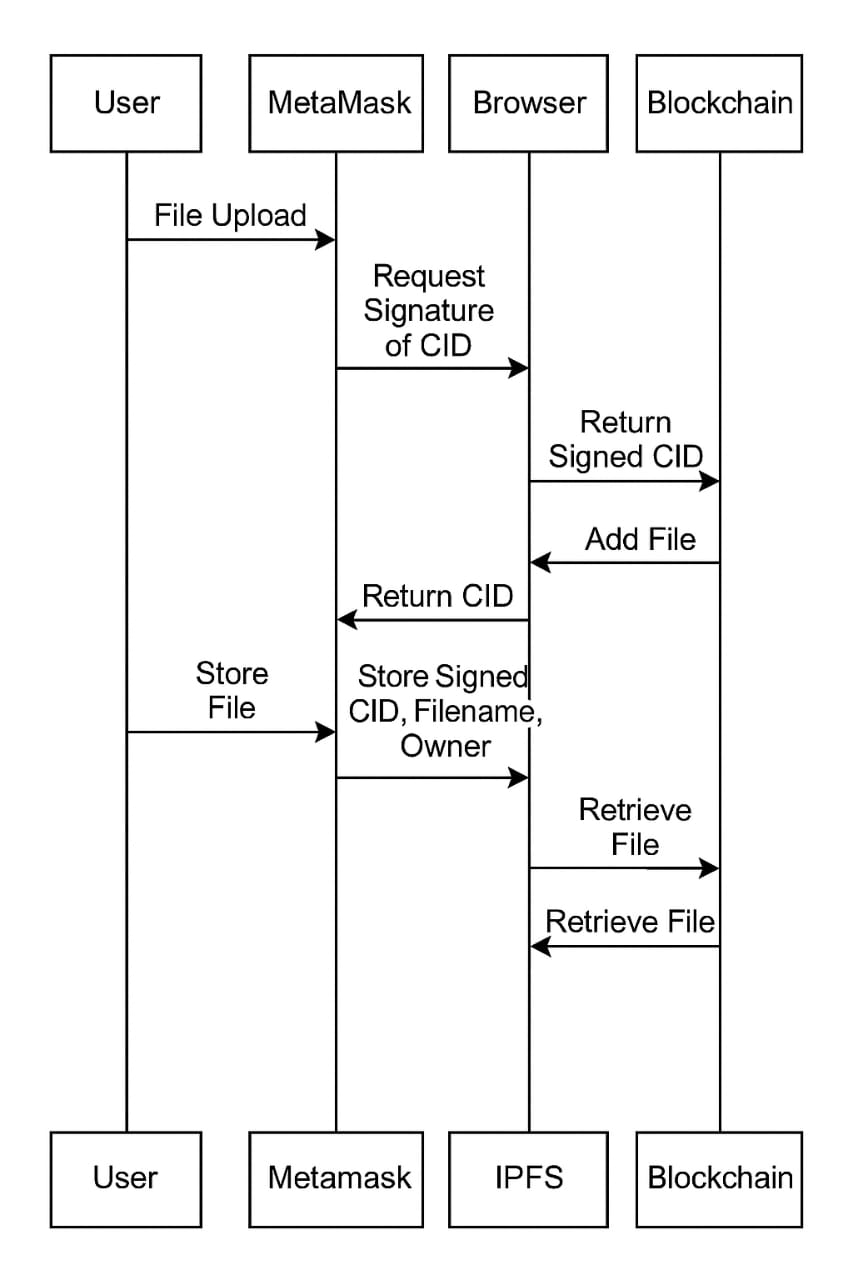
**Fig no:03**

**UML Diagram:**

****

**Fig no:04**

**Sequence Dig:**



**Fig no:05**

* 1. **Approach and Methodology:**

The methodology diagram showcases a system for secure and efficient file storage and transactions using decentralized networks and blockchain technology. Here's a detailed breakdown:

1. **File Transfer and Storage (IPFS Integration)**

* File Operations: A user initiates file operations (such as uploading files) on a system.
* File Transfer Efficiency: The file is split into smaller chunks (Chunk 1, Chunk 2, etc.) to improve efficiency and reliability.
* IPFS (Inter-Planetary File System): The file chunks are transferred via IPFS, a decentralized storage system. IPFS pushes the files and returns a transaction hash (unique identifier for the operation). This hash is crucial for tracking file uploads and ensuring integrity.
* Decentralized Network-The file chunks are distributed across a decentralized network to ensure higher data security and availability. Each chunk of the file is stored on different nodes, making the system resilient to failures or data corruption.

1. **Blockchain Integration**

* File Hash Creation: Once a file is processed by IPFS, a file hash is generated. This hash serves as a unique identifier for the file and is critical for verifying file integrity on the blockchain.

1. Ganache and Blockchain Connection: The system connects to the Ganache Blockchain. The file hash, along with the user’s address, is stored in a smart contract.
2. Smart Contract Deployment: The smart contract is migrated to the blockchain using the Truffle framework. This ensures the hash is securely stored and can be retrieved or verified later on.
3. **User Interaction**

* Frontend (React.js): The user interacts with a front-end system built with React.js, which communicates with the blockchain and IPFS.
* Metamask (Transaction Approval): The user approves transactions via Metamask, a digital wallet used to interact with decentralized applications. Once approved, the transaction is displayed on the front-end interface.
* Smart Contracts & Web3.js.
* The file hash and user’s address are stored in a smart contract, which is compiled and deployed using Web3.js. The contract is added to the blockchain, securing the integrity of the transaction
* This methodology combines decentralized file storage (IPFS) with blockchain technology to create a secure and efficient system for managing files and transactions. The decentralized network

**PROOF OF WORK(POW):**

Proof of Work consensus is the mechanism of choice for the majority of cryptocurrencies currently in circulation. The algorithm is used to verify the transaction and create a new block in the blockchain. The idea for Proof of Work(PoW) was first published in 1993 by Cynthia Dwork and Moni Naor and was later applied by Satoshi Nakamoto in the Bitcoin paper in 2008. The term “proof of work” was first used by **Markus Jakobsson** and **Ari Juels** in a publication in 1999.

Cryptocurrencies like Litecoin, and Bitcoin are currently using PoW.

**Purpose of PoW:**

The **purpose** of a consensus mechanism is to bring all the nodes in agreement, that is, trust one another, in an environment where the nodes don't trust each other.

All the transactions in the new block are then validated and the new block is then added to the blockchain.

The block will get added to the chain which has the longest block height(see [blockchain forks](https://www.geeksforgeeks.org/blockchain-forks/) to understand how multiple chains can exist at a point in time).

Miners(special computers on the network) perform computation work in solving a complex mathematical problem to add the block to the network, hence named, Proof-of-Work.

With time, the mathematical problem becomes more complex.

**Features of PoW:**

**There are mainly two features that have contributed to the wide popularity of this consensus protocol and they are:**

It is hard to find a solution to a mathematical problem.

It is easy to verify the correctness of that solution.

**CHAPTER V**

**SYSTEM REQUIREMENTS**

**4.1 Software Requirements**

|  |  |
| --- | --- |
| **Operating System** | **:** Windows 10/11, Linux (Ubuntu preferred), or macOS |

|  |  |
| --- | --- |
| **Programming Language** | **:** Python 3.x |

|  |  |
| --- | --- |
| **Framework** | **:** Flask (for backend server and API handling) |

|  |  |
| --- | --- |
| **Web Technologies** | **:**HTML, CSS (for frontend) |

|  |  |
| --- | --- |
| **Python Libraries** | **:**flask, requests, hashlib, json, threading, uuid |

|  |  |
| --- | --- |
| **Web Browser** | **:**Google Chrome, Mozilla Firefox (for frontend access) |

|  |  |
| --- | --- |
| **IDE / Code Editor** | **:**Visual Studio Code, PyCharm, or any preferred text editor |

|  |  |
| --- | --- |
| **Git** | **:**For version control and project collaboration |

**4.2 Hardware Requirements**

1.CPU Dual-core processor (Intel i3 / Ryzen 3)

2.RAM 4 GB

3.Storage 10 GB free disk space

4.OS Windows / Linux / macO

**CHAPTER VI**

**IMPLEMENTATION**

**5.1 Tools Used:**

The Blockchain-based File Storage project utilizes several modern tools and technologies to achieve decentralized and secure file storage on the blockchain. Below is an overview of the primary tools used:

1. **Solidity**  
   Solidity is the smart contract programming language used to write the blockchain contracts for this project. It enables defining the logic for file storage references, ownership management, and permission controls on the Ethereum blockchain.
2. **Ethereum Blockchain (Ganache / Testnet)**  
   The project leverages the Ethereum blockchain platform to deploy smart contracts that manage the file metadata and storage access rights. Ganache, a personal blockchain for Ethereum development, is typically used during development and testing.
3. **IPFS (Inter Planetary File System)**  
   IPFS is a peer-to-peer distributed file system that allows storing and sharing files in a decentralized manner. This project stores the actual files on IPFS and uses the Ethereum blockchain to store the file hashes (CIDs) ensuring data integrity and decentralization.
4. **React.js**  
   React.js is used to build the frontend user interface of the application. It offers a responsive and dynamic user experience for uploading, retrieving, and managing files
5. **Ethers.js**  
   Ethers.js is a JavaScript library that facilitates interaction between the frontend React application and the Ethereum blockchain. It is used to connect to the user’s wallet (such as MetaMask), send transactions, and read smart contract data.
6. **Node.js and npm**  
   Node.js runtime and npm package manager are used to manage project dependencies and run scripts for building, testing, and deploying the application.
7. **MetaMask**  
   MetaMask is a browser extension cryptocurrency wallet used by the users to authenticate transactions and interact with the deployed smart contracts on Ethereum blockchain.
8. **Truffle / Hardhat (if applicable)**  
   These are development frameworks that help in compiling, deploying, and testing Solidity smart contracts. One of these might be used to simplify contract management during development.

**5.2 Algorithm Used:**

## Algorithm 1: ****PoW with Random Nonce****

### Step-by-Step:

1. **Initialize** the block's nonce with 0.
2. Enter a loop:
   * Generate a **random number** as the new nonce.
   * Assign this nonce to the block.
   * **Generate the hash** of the block using .generate\_hash().
3. **Check if hash is valid**:
   * A hash is valid if it starts with n leading zeros (where n = difficulty level).
4. If the hash is **not valid**, repeat from Step 2 with a new random nonce.
5. If a **valid hash is found**, the block is ready to be added to the blockchain.

## Algorithm 2: ****PoW with Incremental Nonce****

### Step-by-Step:

1. **Start nonce at 0**.
2. Enter a loop:
   * Assign current nonce to the block.
   * **Generate the hash** using .generate\_hash().
3. **Check if the hash is valid** (starts with required number of zeros).
4. If not valid:
   * Increment nonce by 1.
   * Go back to Step 2.
5. If valid:
   * The block is mined and ready to be added.

**2.SHA-256 Cryptographic Hashing**

**Step-by-Step Algorithm for SHA-256 in Blockchain-based File Storage**

**SHA-256 (Secure Hash Algorithm 256-bit) is a cryptographic hash function used in this project to generate unique and secure hash values for each block. Here’s how the SHA-256 algorithm is applied step-by-step in the project:**

**1. Collect Block Data**

- Gather all relevant data for the block:

- Block index

- Nonce (number used for PoW)

- Previous block’s hash

- List of transactions (file information)

**2. Combine Block Data**

- Concatenate all the above data into a single string (data serialization).

- Example:

all\_data\_combined = str(index) + str(nonce) + prev\_hash + str(transactions)

**3. Encode Data**

- Convert the combined data string to bytes, which is the required format for hashing.

- Example (Python): data\_bytes = all\_data\_combined.encode()

**4. Apply SHA-256 Hash Function**

- Use the SHA-256 algorithm to generate a hash of the encoded data.

- Example (Python):

hash\_value = sha256(data\_bytes).hexdigest()

**5. Get the Hash Output**

- The output is a 64-character hexadecimal string representing the 256-bit hash value.

- This hash uniquely identifies the block and is used in PoW validation**.**

**6. Use Hash in Blockchain**

- The computed hash is checked against the difficulty requirement (e.g., must start with three zeros).

- If valid, the hash is set as the block’s hash and the block is added to the chain.

**5.3 Functions used:**

### Functions Used in Blockchain-based File Storage Project

This project primarily involves a smart contract in Solidity to manage file storage on the blockchain and a frontend to interact with it.

#### 1. ****Smart Contract Functions****

Here are the key Solidity functions typically used in the project (based on standard blockchain file storage contracts and the repo):

* **uploadFile(string memory \_fileHash, string memory \_fileName, uint \_fileSize, string memory \_fileType, string memory \_fileDescription)**
  + Purpose: Uploads file metadata to the blockchain.
  + Stores the file hash and metadata on the contract.
  + Emits an event to notify frontend about the new file upload.
* **getFile(uint \_fileId) public view returns (File memory)**
  + Purpose: Retrieves file metadata by its ID.
  + Used to display stored files to users.
* **getFileCount() public view returns (uint)**
  + Purpose: Returns the total number of files uploaded.
  + Helps frontend iterate through all files.
* **Constructor**
  + Purpose: Initializes the contract, possibly setting the owner.
* **Modifiers (e.g., onlyOwner)**
  + Purpose: Restricts access to certain functions.
* **Events**
  + Example: FileUploaded event that is emitted when a file is uploaded.

#### 2. ****Frontend Functions (React + ethers.js)****

* **connectWallet()**
  + Connects the user's Ethereum wallet (e.g., MetaMask).
* **uploadFileHandler(event)**
  + Handles file upload input from the user.
  + Reads file and calls uploadFile function on the smart contract.
* **fetchFiles()**
  + Calls smart contract to fetch the list of files.
  + Updates UI to display file details.
* **downloadFile(fileHash)**
  + Provides link or triggers file download using IPFS hash.

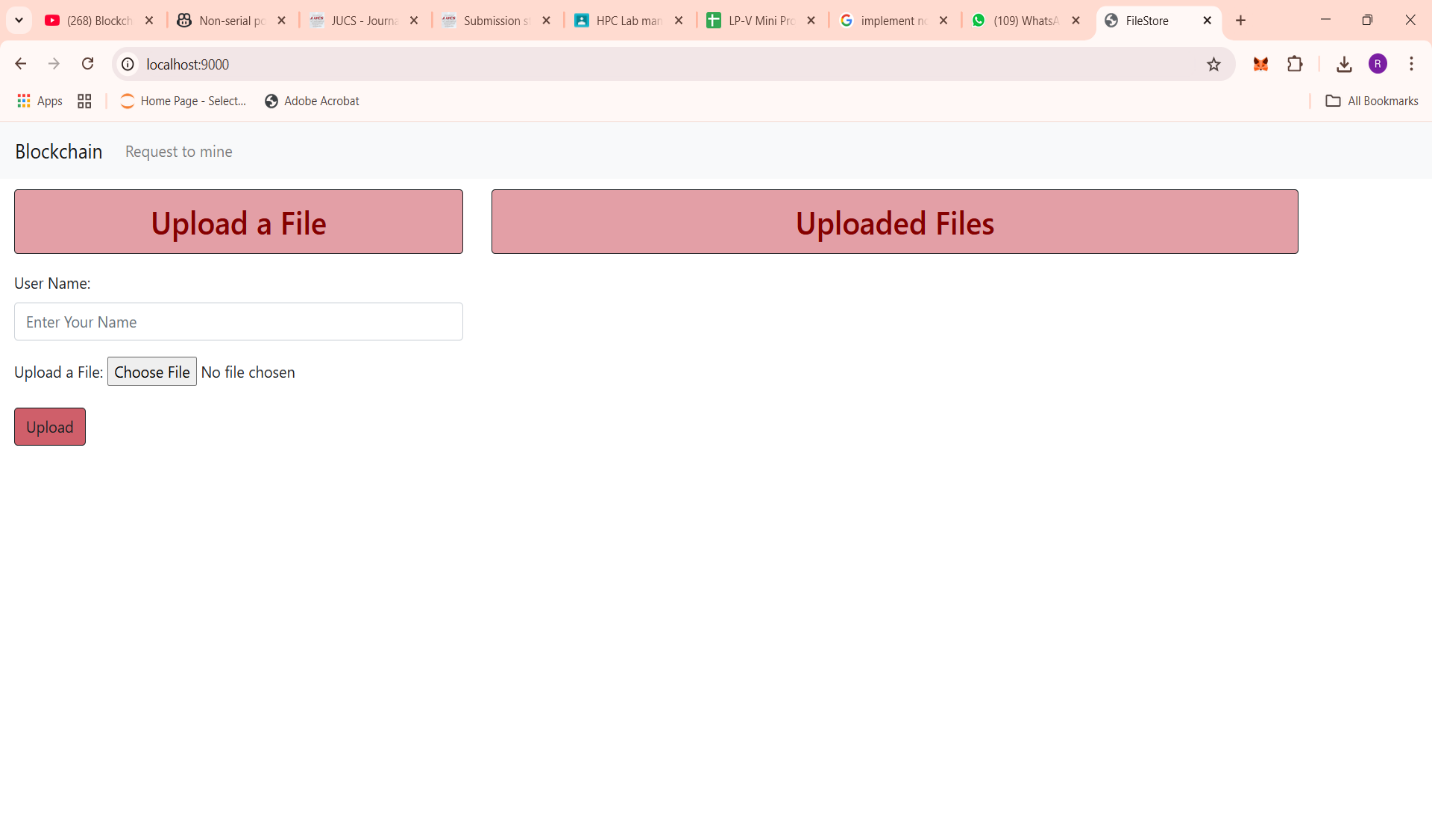
**Functions Table:**

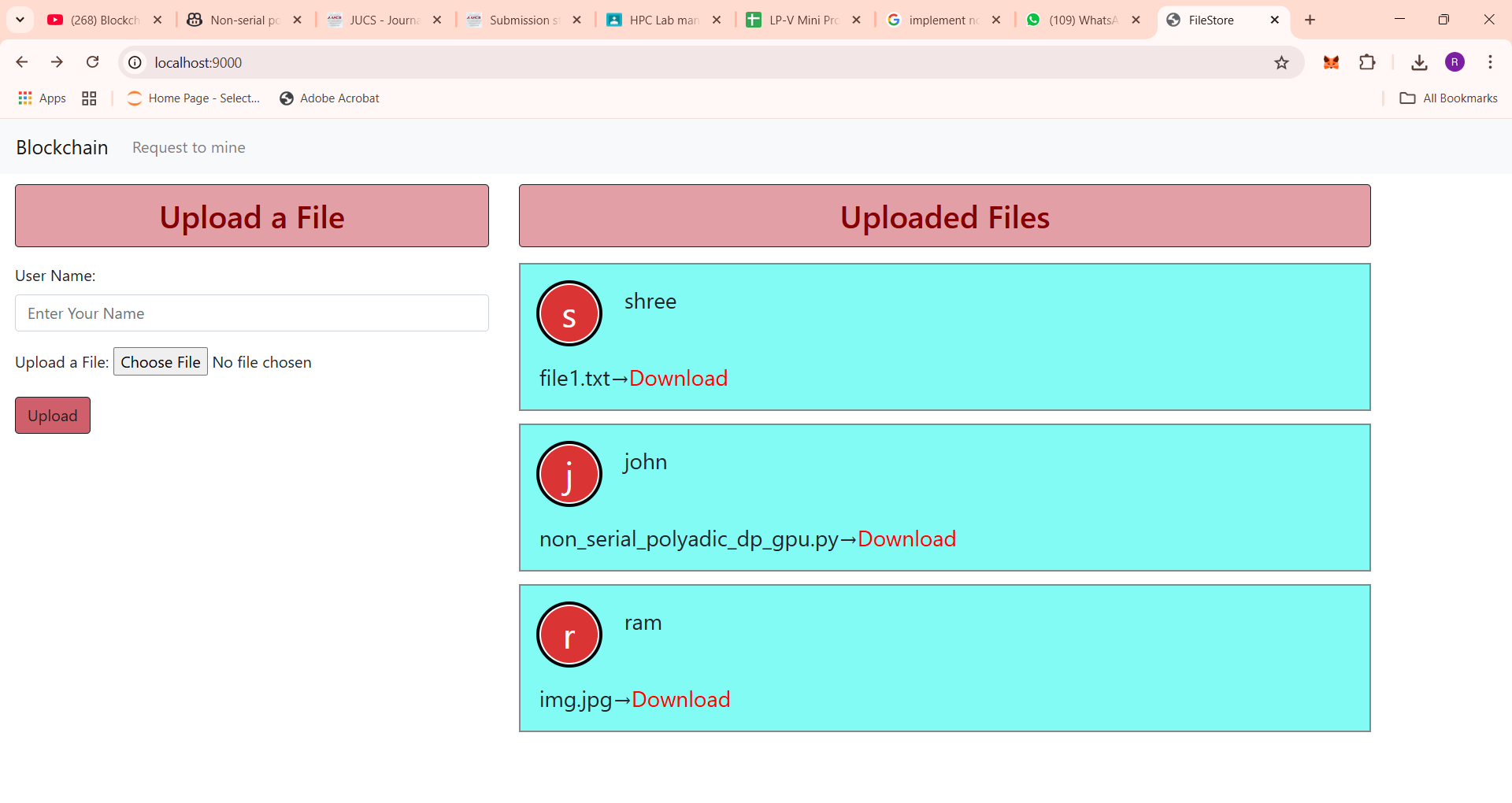
| **Function Name** | **Location** | **Parameters** | **Return Type** | **Description** |
| --- | --- | --- | --- | --- |
| uploadFile | FileStorage.sol (Solidity) | \_fileHash, \_fileSize, \_fileType, \_fileName, \_fileDescription | void | Uploads file metadata to the blockchain and emits a FileUploaded event. |
| getFileCount | FileStorage.sol (Solidity) | None | Uint | Returns the total number of files uploaded to the blockchain. |
| getFile | FileStorage.sol (Solidity) | uint fileId | File struct | Returns file metadata by ID. |
| constructor | FileStorage.sol (Solidity) | None | N/A | Initializes the smart contract. |
| FileUploaded (event) | FileStorage.sol (Solidity) | \_fileHash, \_fileSize, \_fileType, \_fileName, \_fileDescription, uploader, uploadTime | Event | Emits data about each uploaded file for the frontend to listen. |
| connectWalletHandler | App.js (React) | None | void | Connects the user's MetaMask wallet to the DApp. |
| loadBlockchainData | App.js (React) | None | void | Loads user account and contract data from the blockchain. |
| loadFiles | App.js (React) | None | void | Calls getFileCount() and loops through getFile() to fetch all file data. |
| uploadFileHandler | App.js (React) | event | void | Captures user file input, uploads it to IPFS, then stores metadata on-chain. |
| captureFile | App.js (React) | Event | void | Reads the file from the input and converts it into a buffer for IPFS upload. |
| createFile | App.js (React) | fileHash, fileSize, fileType, fileName, fileDescription | Transaction | Calls the smart contract's uploadFile method using ethers.js. |
| displayFiles | App.js / JSX Components | None | JSX | Renders the uploaded files as cards with name, type, and download links. |
| downloadFile (via IPFS) | Not a separate function | IPFS hash passed into <a href> tag | N/A | Enables users to download files directly from IPFS. |

**Table No. 2**

**CHAPTER VII**

**RESULTS**

**Fig No. 6**

****

**Fig No. 7**

**CHAPTER VIII**

**SYSTEM TESTING**

**System Testing:**

System testing is the final phase of testing where the complete and integrated software is tested to verify that it meets the specified requirements. For the **Blockchain-based File Storage** project, system testing involved testing all components — the smart contract on the blockchain, the frontend user interface, and their interaction with each other.

**Testing Objectives:**

Verify that users can upload files and their metadata correctly stores on the blockchain.

Ensure the retrieval and display of stored files is accurate and reliable.

Confirm that files can be downloaded successfully via the IPFS hashes.

Test wallet connection and transaction handling through MetaMask or similar wallets.

Check for correct error handling and user notifications during failures.

**Test Environment:**

**Blockchain network:** Local Ethereum blockchain (e.g., Ganache) or testnet (Ropsten, Mumbai).

**Frontend:** React app running on localhost.

**Wallet:** MetaMask connected to the chosen blockchain network.

**Tools:** Remix IDE for contract deployment, Ganache for local blockchain, ethers.js for frontend interaction.

**Test Cases:**

| **Test Case** | **Description** | **Expected Result** |
| --- | --- | --- |
| Connect Wallet | Connect MetaMask wallet to the app | Wallet connects successfully |
| Upload File | Upload a file with metadata (name, type, description, hash) | File metadata stored on blockchain; Event emitted |
| Retrieve Files | Fetch all files stored on blockchain | Correct list of files displayed |
| Download File | Download file using IPFS hash | File downloads without errors |
| Upload Invalid File | Try uploading unsupported or corrupted file | Error displayed, upload fails |
| Wallet Transaction Failure | Simulate failed transaction (e.g., reject in MetaMask) | Transaction rejected, user notified |
| Multiple Users | Multiple users upload and retrieve files | Each user’s files are stored/retrieved correctly |

**Table no:3**

**8.1 ADVANTAGES**

**1. Decentralization**

This system eliminates the need for a central authority or server. By distributing file references across the blockchain and possibly using decentralized storage solutions (like IPFS), it ensures that no single entity controls the data. This reduces the risk of system failure or censorship.

**2. Data Integrity and Immutability**

Blockchain technology guarantees immutability. When a file’s hash is stored on the blockchain, it cannot be altered. This ensures that the file has not been tampered with or corrupted, preserving its authenticity over time.

**3.Enhanced Security**

Blockchain uses cryptographic techniques to secure data. Each transaction or file record is linked using secure cryptographic hashes, making it extremely difficult for malicious actors to manipulate or forge records.

**4. Transparency and Auditability**

Every file-related action (such as uploads) is recorded on a public or permissioned ledger. This allows for complete transparency and the ability to audit changes or activities associated with each file.

**5. User Ownership and Control**

Users can manage their own files without relying on a third party. Smart contracts can assign ownership and access rights, giving users full control over who can view or interact with their files.

**6. Resistance to Data Loss**

In decentralized systems, data or file references are often stored redundantly across multiple nodes. This redundancy reduces the chance of data loss due to server crashes or shutdowns.

**8.2 DISADVANTAGES**

1. Limited Storage Capabilities

Blockchains are not suitable for storing large files directly due to size and cost constraints. Instead, only the hash or pointer to the file is stored on-chain, with the actual file stored off-chain (e.g., in IPFS). This adds complexity and requires managing multiple systems.

2. High Operational Costs

Smart contract interactions often require gas fees (especially on networks like Ethereum). These costs can become significant when uploading or interacting with files frequently.

3. Scalability Issues

Public blockchains have limitations in terms of transaction throughput. If the system handles a large number of users or files, performance may degrade due to network congestion or long confirmation times.4. \*Complexity for End Users\*

The average user may find it difficult to interact with blockchain-based systems. Wallet setup, understanding gas fees, and managing decentralized IDs are often barriers to entry.

5. Legal and Compliance Challenges

Once data (or even just metadata) is written to the blockchain, it cannot be changed or deleted. This raises concerns for data protection laws like GDPR, which require the ability to delete personal data upon request.

6. Security Depends on Implementation

Although blockchain itself is secure, poor implementation of smart contracts, wallet integration, or storage connections (like IPFS) can introduce vulnerabilities. Smart contract bugs may lead to data loss or unauthorized access.

**8.3 APPLICATIONS**

**Application of Blockchain-based File Storage System:**

**1. Secure File Storage**

The primary application of this system is to securely store files using a decentralized model. Traditional cloud storage systems are centralized and prone to:

Data breaches

Single point of failure

Censorship

Using IPFS for file storage and Ethereum smart contracts for metadata and access control, this system ensures:

Tamper-proof storage

High availability

Data integrity

**2. Proof of Ownership & Timestamping**

Files uploaded to IPFS and registered on the blockchain come with a cryptographic timestamp and ownership record:

Useful for legal documents, research work, or digital content

Owners can prove when they uploaded the file and that it hasn’t been modified

**3. Decentralized File Sharing**

Users can share file access with others via the IPFS hash, while the smart contract keeps a secure log of who owns and accesses what file.

Eliminates dependency on centralized authorities

Promotes peer-to-peer sharing

**4. Enterprise and Legal Use-Cases**

This system can be extended for:

Legal document archives

Healthcare record storage

Government recordkeeping

Where auditability, transparency, and immutability are critical.

**5. Monetized File Access (Future Scope)**

With enhancements, smart contracts can support:

Access control

Pay-per-access models

Tokenized data markets

Example: A photographer uploads images and charges users ETH to view or download

**8.4 Key Challenges in Blockchain-Based File Storage**

1. Scalability and Performance

Storing large files directly on the blockchain is inefficient due to storage limitations and high costs. This can lead to network congestion and increased transaction fees. Off-chain storage solutions, like IPFS, are often used to mitigate this, but they introduce additional complexity in ensuring data integrity and retrieval speed.

2. Data Privacy and Security

While blockchain offers transparency, it can conflict with the need for data privacy. Ensuring that sensitive information is securely stored and accessed without compromising the decentralized nature of the system is a significant challenge. Encryption and access control mechanisms are essential but can add overhead. ([Quadrant Technologies][2], [Investopedia][3])

3. Complexity in Integration

Integrating blockchain with existing file storage systems requires careful design to ensure seamless interoperability. This includes managing metadata on-chain, handling off-chain data storage, and ensuring consistent access controls across the system. ([ResearchGate][4])

4. Cost and Resource Consumption

Blockchain transactions, especially those involving smart contracts, can incur significant costs. Additionally, the computational resources required for operations like Proof of Work can be substantial, leading to increased energy consumption and operational expenses.

**CHAPTER IX**

**CONCLUSION**

The Blockchain-based File Storage system provides a secure, transparent, and decentralized method for storing and accessing files. By embedding file data directly into the blockchain, the system eliminates the need for centralized or cloud-based storage. The use of the Proof of Work consensus mechanism ensures data integrity and prevents tampering. This project highlights the potential of blockchain technology beyond cryptocurrencies, especially in secure data management. Although on-chain storage has limitations with large files, it is highly effective for small, sensitive data. The system demonstrates a working prototype that can be enhanced with IPFS or smart contracts in future. Overall, it is a promising approach toward trustless and tamper-proof file storage.

**FUTURE SCOPE**

1.In future work, we will apply several methods for training and testing model.

2.Our experimental results show the practicality of our scheme. To put it in a nutshell, we leave the following questions for future research: Further optimizing the security and privacy of the proposed scheme based on the above technologies, Adding more functions to better comply with GDPR requirements, Exploring how to improve the efficiency of generation and verification of zero- knowledge proof

**CHAPTER X**

**REFERENCES**

1)Yangheran Piao, Kai Ye, Xiaohui Cui designed "A Data Sharing Scheme for GDPR Compliance Based on Consortium Blockchain to achieve data security sharing", Future Internet 2021, 13, 217.

2)Abhirup Khanna, Anushree Sah, Vadim Bolshev, Alessandro Burgio, Vladimir Panchenko, Marek Jaisinski designing Blockchain–Cloud Integration: "A survey for blockchain–cloud services being offered by existing Cloud Service providers ",Sensors 2022, 22, 5238.

3)Sarada Prasad Gochhayat, Sachin Shetty, Ravi Mukkamala designed "Measuring Decentrality in Blockchain Based Systems", 2020.

4)G. Subathra, A. Antonidoss and B. K. Singh, “Decentralized Consensus Blockchain and IPFSBased Data Aggregation for Efficient Data Storage Scheme”, 2022.

5)Seyednima Khezr, Md Moniruzzaman, Abdulsalam Yassine and Rachid Benlamri designed “ Blockchain Technology in Healthcare: A Comprehensive Review and Direction for Future Research”, 2019. 29

6)Moez Krichen, Meryem Ammi, Alaeddine Mihoub , 2022, “Blockchain for Modern Application : A Survey.

7)Pratima Sharma, Rajni Jindal, Malaya Dutta Borah, “ Blockchain Technology for Cloud Storage: A Systematic Literature Review”,2020.

8)J. Benet, "Filecoin : A decentralized storage network", 2017.

9)Firdaus M, Kyung-Hyune Rhee, “Blockchain-Enhanced Secure data storage and sharing in vehicular edge computing networks”, Appl.Sci.2021,11,414.

10)Wei Liang, Yongkai Fan, “Secure data storage and recovery in industrial blockchain network environments”, 2021.

11)M. Firdaus and K. H. Rhee, ‘‘On blockchain-enhanced secure data storage and sharing in vehicular edge computing networks,’’ Appl. Sci., vol. 11, no. 1, pp. 1–21, Jan. 2021, doi:10.3390/app11010414.

12)N. Lipusch, ‘‘Initial coin offerings—A paradigm shift in funding disruptive innovation,’’ SSRN Electron. J., vol. 139, pp. 1–21, Mar. 2018, doi: 10.2139/ssrn.3148181.

13)S. Yang, A. Hareedy, R. Calderbank, and L. Dolecek, ‘‘Topology-aware cooperative data protection in blockchain-based decentralized storage networks,’’ in Proc. IEEE Int. Symp. Inf. Theory (ISIT), Jun. 2020, pp. 1-6.

14)Y. Zhu, G. Zheng, and K.-K. Wong, ‘‘Blockchain-empowered decentralized storage in Air-to-Ground industrial networks,’’ IEEE Trans. Ind. Informat., vol. 15, no. 6, pp. 3593–3601, Jun. 2019, doi: 10.1109/TII.2019.2903559.

15)A. S. de Pedro, D. Levi, and L. I. Cuende, ‘‘Witnet: A decentralized Oracle network protocol,’’ 2017, arXiv:1711.09756.

16)S. He, Y. Lu, Q. Tang, G. Wang, and C. Q. Wu, ‘‘Fair peer-to-peer content delivery via blockchain,’’ 2021, arXiv:2102.04685.

17)K. Aldriwish, ‘‘A double-blockchain architecture for secure storage and transaction on the Internet of Things networks,’’ Int. J. Comput. Sci. Netw. Secur., vol. 21, no. 6, pp. 119–126, 2021, doi: 10.22937/IJCSNS.2021.21.6.16.

18)M. Campanelli, D. Fiore, N. Greco, D. Kolonelos, and L. Nizzardo, ‘‘Vector commitment techniques and applications to verifiable decentralized storage,’’ Elsevier, Tech. Rep., 2020.

19)I. Vakilinia, W. Wang, and J. Xin, ‘‘An incentive-compatible mechanism for decentralized storage network,’’ 2022, arXiv:2208.09937.

20)Y. Du, H. Duan, A. Zhou, C. Wang, M. Ho Au, and Q. Wang, ‘‘Enabling secure and efficient decentralized storage auditing with blockchain,’’ IEEE Trans. Dependable Secure Comput., vol. 19, no. 5, pp. 3038–3054, Oct. 2022, doi: 10.1109/TDSC.2021.3081826.

21)Protocol Labs, ‘‘Filecoin: A decentralized storage network,’’ IEEE, Tech. Rep., 2017, vol. 9, no. 15.