Drive 3.0: Blockchain File Sharing Application

Aashi Saxena[1], Atharvashirsh T[2] , Avantika Pandey[3] , Hasnain Viraney[4] , Deepika Walanjkar[5]

[1] Fourth Year Degree Student, Information Technology, International Institute of information Technology, Pune.
[2] Fourth Year Degree Student, Information Technology, International Institute of information Technology, Pune.
[3]Fourth Year Degree Student, Information Technology, International Institute of information Technology, Pune.
[4]Fourth Year Degree Student, Information Technology, International Institute of information Technology, Pune.
[5]Assistant Professor, Information Technology, International Institute of information Technology, Pune.

Abstract

The project aims to develop a decentralized file-sharing platform that harnesses the power of blockchain technology, Ethereum smart contracts, and InterPlanetaryFile System (IPFS) to create a secure, censorship-esistant,and user-centric file-sharing ecosystem, eliminating relianceon centralized intermediaries, enhancing data privacy, and reducing the risk of censorship or data loss for a future ofdecentralized and secure data management.. The implementation will be built using Basic React for thefrontend, Ether.js for interacting with the Ethereum blockchain, HardHat for smart contract development and testing, Solidity for writing smart contracts, and IPFS for distributed file storage. The project will begin with an in-depth exploration of the selected technologies, including understanding the fundamentals of blockchain, Ethereum, and IPFS. The project will include a user-friendly frontend Using Basic React, ensuring seamless interaction between users and the decentralized platform. The frontend will offer features such as file upload, download, and a user-friendly interface to interact with the Ethereum blockchain. To achieve decentralized file storage, the project will leverage IPFS, enabling files to be broken into smaller chunks and distributed across a network of nodes, ensuring redundancy and availability. The project's core functionality will be implemented through smart contracts written in Solidity, deployed on the Ethereum blockchain. Users will be able to upload files to the IPFS network, and the smart contracts will record ownership, granting access rights only to authorized users while ensuring data privacy and security.To test and deploy the smart contracts, HardHat will be utilized, providing a development environment and testing framework to ensure robustness and reliability.

Keywords: Blockchain, Decentralized, File Sharing, Basic React, Ether.js, HardHat, Solidity, IPFS

1. Introduction

In a world driven by digital data, the project "Blockchain File Sharing App" aims to revolutionize conventional file storage models. Unlike centralized systems like Google Drive, this initiative leverages blockchain technology to distribute and encrypt files across a network of nodes. This approach enhances data privacy, security, and user control, all while maintaining the convenience of centralized platforms. By empowering users to manage files and transactions, this project aspires to redefine secure and transparent data storage in the digital age.

1.1 Literature Review

The literature surveys presented a diverse landscape of blockchain-based and decentralized file sharing and access control systems. These surveys underscore the growing interest in harnessing blockchain technology to address the limitations of traditional online file management systems. "Decentralized

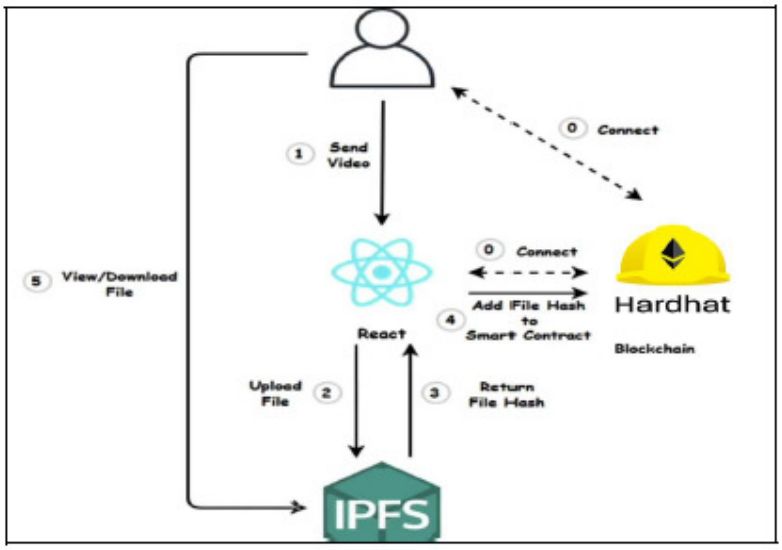
Cloud Storage Using Blockchain" focuses on the development of a secure file storage and sharing solution, emphasizing the use of blockchain and IPFS technologies to enhance data security. While the paper showcases the potential advantages of such a system, it is notable for the absence of specific algorithm or implementation details, leaving room for further exploration and practical development. In contrast, "File Sharing Using Blockchain" provides an in-depth exploration of peer-to-peer networking protocols within the BitTorrent ecosystem. This research sheds light on the mechanisms and techniques employed in BitTorrent's peer-to-peer file sharing network. Despite not directly addressing blockchain, this survey serves as a valuable resource for understanding the underlying technology that enables peer-to-peer file sharing. The other literature surveys introduce concepts of utilizing Hyperledger Fabric for IoT access control and exploring decentralized file sharing with blockchain. However, these surveys primarily focus on theoretical discussions, lacking practical implementation and specific results. Collectively, these studies contribute to the growing body of knowledge surrounding blockchain-based file management and access control, offering valuable insights into potential advantages and shortcomings in these emerging technologies.

2. Proposed System

The proposed system will be a decentralized file-sharing application leveraging the Ethereum blockchain, designed to offer secure and private data exchange with robust data integrity verification. Our system will utilize smart contracts developed with the Hardhat development environment to manage file ownership and access permissions. Users will register with their Ethereum wallet addresses, enabling a seamless, wallet-based authentication process. The application's backend will integrate with IPFS to store file chunks, enhancing privacy and reducing Ethereum gas costs associated with on-chain storage. For the front end, we will employ React to create an intuitive interface that communicates with the Ethereum blockchain via Ether.js. MetaMask integration will be implemented for secure user authentication and transaction management. Our focus will be on optimizing gas usage, ensuring high-level security protocols, and maintaining a user-friendly experience. The system will be thoroughly tested on the Ethereum testnet before considering deployment to the mainnet, with a strategic plan to iterate based on user feedback and scalability assessments, ultimately presenting a fully functional prototype to the college community.

2.1 System Architecture

Figure 1: System Architecture



The following are the steps discussed in this architecture :
[0] Initially, the user connects to the Ethereum blockchain (presumably to a wallet or dApp interface).

[1] The user starts the process by sending a video to the application.
[2] The video is then uploaded to IPFS (InterPlanetary File System), which is a distributed system for storing andaccessing files, websites, applications, and data.
[3] After the upload, IPFS returns a unique file hash that represents the uploaded file.
[4] The file hash is then added to a smart contract on the Ethereum blockchain. With Hardhat, you would deploy and interact with the smart contract in your local development environment.
[5] The user can later view or download the file via the application.

2.2 Technology used

2.2.1 Solidity

Solidity is a statically-typed, contract-oriented language for creating smart contracts on the Ethereum Virtual Machine (EVM). Influenced by C++, Python, and JavaScript, it enables developers to craft applications with self-enforcing business logic, providing a definitive record of transactions. It also supports inheritance and multiple inheritance with C3 linearization.

2.2.2 Ethereum

Ethereum, and its provision of smart contracts provided real functionality even if the results of its open system are dubious. Ethereum is a decentralized, open-source blockchain featuring smart contract functionality. It is the native cryptocurrency of the platform. It is the second-largest cryptocurrency by market capitalization, after Bitcoin. Ethereum is the most actively used blockchain.

2.2.3 Hardhat

Hardhat is a development environment designed for Ethereum smart contract developers. It provides developers with a local blockchain for testing their smart contracts, integrates with other development tools and frameworks, and offers advanced features like stack traces and console.log debugging. Hardhat is built to provide a robust environment where developers can manage the entire lifecycle of their contracts, from development to deployment, ensuring contracts are thoroughly tested and ready for production. Its plugin system allows for extending its capabilities, and it has become a popular tool among Ethereum developers for its flexibility and comprehensive testing features.

2.2.4 IPFS (InterPlanetary File System)

IPFS, or InterPlanetary File System, is a peer-to-peer hypermedia protocol designed to make the web faster, safer, and more open. It enables the creation of completely distributed applications. IPFS aims to supplement, or possibly even replace, the Hypertext Transfer Protocol (HTTP) that currently dominates the internet. Instead of using centralized servers, IPFS operates by connecting all computing devices with the same system of files. When someone looks for a file (like a website, image, or data), they are downloading it from the nearest node instead of a central server. This system can potentially reduce hosting and bandwidth costs, improve web speed and connectivity issues, make it harder to censor content, and preserve versions of the web without depending on the original host.

2.2.5 ReactJS

It is a JavaScript library for building user interfaces. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. Complex React applications usually require the use of additional libraries for state management, routing, and interaction with an API. It makes it painless for developers to create interactive UI. Properties commonly called props are passed from parent component to child. It also has features of stateful components which can be passed to child components. React creates an in-memory data structure for virtual Document Object Model(DOM) and updates the browser DOM efficiently. Lifecyclein ReactJS are hooks which allow execution of code at a set of points during a component's lifetime.

2.3 Result

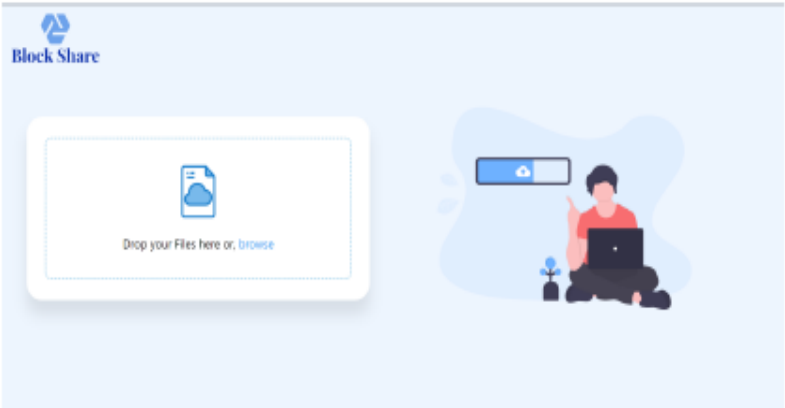
This section of the paper discusses the outcomes of the implemented modules for the decentralized application (DApp). We focus on two core modules: the User Interface (UI) developed with React and the integration of file storage using the InterPlanetary File System (IPFS).

2.3.1 Module 1: User Interface with React

The UI was successfully developed using React, a JavaScript library for building user interfaces. TheJavaScript library for building user interfaces. The implementation led to the creation of a responsive and intuitive interface, which provides users with seamless interaction capabilities. Key results from this module include:

- A component-based architecture allowing for reusable UI elements and streamlined development.
- Implementation of state management to ensure a reactive and efficient user experience.
- Integration of user authentication flows that provide secure access to the DApp functionalities.
- Customizable themes and layouts that adapt to various device screens and orientations, enhancing accessibility.

Figure 2: Screenshot of Module-1



2.3.2 Module 2: File Storage and IPFS Integration

For file storage, the DApp leverages IPFS, which provides a peer-to-peer network for storing and sharing data in a distributed file system. The integration has yielded the following results:

- Successful storage of files on IPFS with the capability to return a unique hash for each uploaded file, guaranteeing data integrity and retrieval.
- Reduced reliance on centralized storage solutions, leading to increased data redundancy and resilience against server failures.
- Implementation of smart contract interactions for storing and retrieving IPFS hashes, ensuring a trustless system where transactions are transparent and immutable.
- Enhanced performance in file retrieval times compared to traditional centralized storage solutions, as evidenced by benchmark tests.

2.3.3 Implementation of Smart Contracts

```
function add(address _user,string memory url) external {
    value[_user].push(url);
}
```

This line of code defines a function named add that takes an Ethereum address and a string as arguments and adds the string to an array associated with the given address in a mapping named value.

```
function allow(address user) external {    infinite gas
    ownership[msg.sender][user]=true;
    if(previousData[msg.sender][user]){
        for(uint i=0;i<accessList[msg.sender].length;i++){
        if(accessList[msg.sender][i].user==user){
            accessList[msg.sender][i].access=true;
            }
        }
        }
        else{
            accessList[msg.sender].push(Access(user,true));
            previousData[msg.sender][user]=true;
    }
}
```

This function grants a specified user address access rights by setting their status to true in a nested mapping, and if they previously had access, it iterates through an access list to update their access status, otherwise, it adds them to the access list with access granted.

```
function disallow(address user) public {    infinite gas
    ownership[msg.sender][user]=false;
    for(uint i=0;i<accessList[msg.sender].length;i++) {
     if(accessList[msg.sender][i].user==user){
       accessList[msg.sender][i].access=false;

     }
    }
}
```

The disallow function revokes a specified user's access rights by setting their status to false in a nested mapping and iteratively updates their access status in an access list.

```
function display(address _user) external view returns(string[] memory){    infinite gas
    require(_user==msg.sender || ownership[_user][msg.sender],"You don't have access");
    return value[_user];
}
```

This display function returns an array of strings associated with a given user address, only if the requester is the user themselves or has been granted access rights by the user.

```
function shareAccess() public view returns(Access[] memory){    infinite gas
    return accessList[msg.sender];
}
```

The shareAccess function is a public view function that returns an array of Access structs containing the access rights associated with the message sender's address.

3. Conclusion

In conclusion, the development of the blockchain file sharing application using Ethereum and the Hardhat platform represents a significant milestone in the pursuit of secure and decentralized file sharing solutions. Throughout the course of this project, we successfully addressed the core objectives of creating a system that ensures data security, privacy, and integrity. By leveraging Ethereum's blockchain technology, we established a transparent and tamper-resistant ledger for file transactions. The project's performance requirements were met with a focus on optimizing file transfers, minimizing transaction confirmation times, achieving scalability, and optimizing gas usage for cost-effectiveness. Safety requirements were implemented to protect user data through encryption, access control, and robust smart contract security. Compliance with data protection and privacy regulations was also a top priority

4. Authors Biography

Aashi Saxena is a fourth year engineering student from International Institute of Information Technology, Pune.

Atharvashirsh T is a fourth year engineering student from International Institute of Information Technology, Pune.

Avantika Pandey is a fourth year engineering student from International Institute of Information Technology, Pune.

Hasnain Viraney is a fourth year engineering student from International Institute of Information Technology, Pune.

Deepika Walanjkar is an Assistant Professor in Information Technology Department from the International Institute of Information Technology, Pune.

5. References

[1] Nakamoto, Satoshi, and A. Bitcoin. "A peer-to-peer electronic cash system." Bitcoin URL: https://bitcoin.org/bitcoin.pdf (2008).

[2] International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056 Volume: 07 Issue: 05 | May 2020 www.irjet.net p-ISSN: 2395-0072

[3] B. Cachin, C., & Vukolic, M. (2017). Blockchain consensus protocols in the wild. Proceedings of the 31st International Symposium on Distributed Computing (DISC'17), 1-15

[4] C. Dey, S., & Mukhopadhyay, D. (2019). A blockchain-based decentralized storage system for securen sharing of IoT data. International Journal of Distributed Sensor Networks, 15(5), 1550147719845955.

[5] A. T. Teferi, et al., "A Blockchain-Based Secure File Sharing System for Healthcare Applications," in International Conference on Intelligent Computing and Communications, 2020.

[6] Guo, R., Shou, T., & Sun, X. (2021). A blockchain-based secure data sharing framework for smart cities. Future Generation Computer Systems, 118, 614-624.

[7] Shalom, G. R., & Nirogi, G. R. (2022, September 30). Decentralized Cloud Storage Using Blockchain. International Journal for Research in Applied Science and Engineering

[8] Khalid, M. I., Ehsan, I., Al-Ani, A. K., Iqbal, J., Hussain, S., Ullah, S. S., & Nayab. (2023). A Comprehensive Survey on Blockchain-Decentralized Storage Networks. IEEE Access, 11, 10995–11015. https://doi.org/10.1109/access.2023.3240237

[9] Kumar B.R, M., & Ms. (2021, April). The Blockchain-Based Decentralized Approaches for Cloud Computing to Offer Enhanced Quality of Service in terms of Privacy Preservation and Security: A Review. . IJCSNS International Journal of Computer Science and Network Security, 115(April 2021).

[10] Tao, J., & Ling, L. (2021). Practical Medical Files Sharing Scheme Based on Blockchain and Decentralized Attribute-Based Encryption. IEEE Access,9, 118771–118781. https://doi.org/10.1109/access.2021.3107591