

**BLOCKCHAIN-BASED DECENTRALIZED DATA
STORAGE USING IPFS
A PROJECT REPORT**

Submitted by

S. DINESH (921019104013)

K.GOPALA KRISHNAN (921019104015)

J. HARI PRASAD (921019104017)

S.S. SANTHOSH SIVAN (921019104044)

In partial fulfillment for the award of the degree

Of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING



**NADAR SARASWATHI COLLEGE OF ENGINEERING AND
TECHNOLOGY, THENI-625531
ANNA UNIVERSITY: CHENNAI**

MAY 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certificated that this project “**BLOCKCHAIN-BASED DECENTRALIZED DATA STORAGE USING IPFSA PROJECT REPORT**” is the bonafide work of **DINESH.S (921019104013), GOPALA KRISHNA.K (921019104015), HARI PRASAD.J (921019104017) and SANTHOSH SIVAN.S.S (921019104044)** who carried out the project work under my supervision.

SIGNATURE

Mr. J.MATHALAIRAJ M.E.,(Ph.D).,
HEAD OF THE DEPARTMENT

Department of Computer
Science & Engineering,
Nadar Saraswathi College of
Engineering and Technology,
Theni-625531

SIGNATURE

Mr. L.S. VIGNESH M.E.,
SUPERVISOR

Department of Computer
Science & Engineering,
Nadar Saraswathi College of
Engineering and Technology,
Theni-625531

Submitted for the project viva voce Examination held on_____ at
Nadar Saraswathi College of Engineering and Technology, Theni.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

At this pleasing moment of having successfully completed our project, we wish to convey our sincere thanks and gratitude to the management of our college and our beloved Secretary, **Mr. A.S.R. MAHESWARAN, B.Sc.,** and **Mr. A. RAJKUMAR, B.B.A.,** and Joint Secretary **Mr. S. NAVEEN RAM B.E., MBA.**

We would like to express our sincere gratitude for the enthusiastic support and professional suggestions provided by our Principal, **Prof. Dr. C. MATHALAI SUNDARAM, M.E., M.B.A., M.I.S.T.E., Ph.D.,** to complete our project fruitfully.

Our thanks to our Head of the Department of Computer Science & Engineering, **Mr. J.MATHALAIRAJ M.E.,(Ph.D).,** for his valuable advice regarding our project and his untiring effort to complete our project effectively.

Gratitude in memory of heart, words cannot adequately express our thanks for our Project Coordinator **Dr. A. SOLAIRAJ M.E.,Ph.D.,** and Project Guide **Mr. L.S. VIGNESH M.E.,** Assistant Professor, Department of Computer Science and Engineering and all the staff members of Computer Science and Engineering department who all provided us adequate schedule for completing our project.

Without the blessings of god almighty, nothing is possible, we thank god almighty for showering his blessings over us to complete this work meticulously.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	i
	LIST OF FIGURES	ii
1	INTRODUCTION	1
	1.1 BLOCKCHAIN	1
	1.2 ABOUT THE PROJECT	3
2	LITERATURE SURVEY	5
3	SYSTEM ANALYSIS	10
	3.1 EXISTING SYSTEM	10
	3.2 PROPOSED SYSTEM	11
4	SYSTEM SPECIFICATION	13
	4.1 HARDWARE SPECIFICATION	13
	4.2 SOFTWARE SPECIFICATION	13
	4.3 SOFTWARE DESCRIPTION	14
	4.3.1 WINDOWS 7	14
	4.3.2 SOLIDITY	16
	4.3.3 WEB 3	17

	4.3.4 REACT	18
	4.3.5 IPFS LIBRARIES	19
	4.3.6 NODE JS	20
	4.3.7 IPFS	22
5	SYSTEM DESIGN	23
	5.1 DESIGN GOALS	23
	5.1.1 INPUT DESIGN	24
	5.1.2 OBJECTIVES	24
	5.1.3 OUTPUT DESIGN	25
	5.2 SYSTEM ARCHITECTURE	25
6	SYSTEM IMPLEMENTATION	26
	6.1 USER TRAINING	26
	6.1.1 TRAINING ON THE APPLICATION SOFTWARE	26
	6.1.2 OPERATIONAL DOCUMENTATION	27
	6.1.3 SYSTEM MAINTENANCE	27
	6.1.4 CORRECTIVE MAINTENANCE	27
	6.1.5 ADAPTIVE MAINTENANCE	28
	6.1.6 PERCEPTIVE MAINTENANCE	28

	6.2 MODULES	28
	6.3 MODULE DESCRIPTION	29
	6.3.1 FRONT-END USER INTERFACE USING REACT	30
	6.3.2 IPFS CLIENT LIBRARIES	31
	6.3.3 SOLIDITY IN POLYGON NETWORK	32
	6.3.4 ETHEREUM NETWORK, GANACHE CONNECTION	33
7	TESTING	34
	7.1 INTRODUCTION ABOUT TESTING	34
	7.1.1 TESTING OBJECTIVES	34
	7.1.2 TESTING METHODOLOGIES	34
	7.1.2.1 UNIT TESTING	35
	7.1.2.2 INTEGRATION TESTING	35
	7.1.2.3 SYSTEM TESTING	36
	7.1.2.4 BLACK BOX TESTING	36
	7.1.2.5 WHITE BOX TESTING	37
	7.1.2.6 PERFORMANCE TESTING	37
	7.1.2.7 APPENDIX TESTING	38

8	CONCLUSION	39
	8.1 CONCLUSION	39
	8.2 FUTURE WORK	
9	APPENDIX-I (SOURCE CODE)	40
	CODING (SOLITIDY FILES)	40
10	APPENDIX-II	43
	SCREENSHOTS	43
11	APPENDIX-III	47
	REFERENCES	47

ABSTRACT

Blockchain-based decentralized storage using IPFS is a novel approach to data storage that leverages blockchain technology and the Inter Planetary File System (IPFS) to provide a secure, decentralized, and scalable solution. Data is stored on the IPFS network and is protected by the immutability and security features of the blockchain network. The decentralized nature of the IPFS network ensures that data is distributed across multiple nodes, providing redundancy and reducing the risk of data loss.

Here, the user can login to their account using metamask account, and after they logged in. Then, they can upload their files by clicking upload files. After uploading the file, the user have to pay a fee for the transaction using metamask account. After the successful transaction the files or data which is uploaded by the user, will be stored in the IPFS. Furthermore, since the data is encrypted, only authorized users with the IPFS hash can access it, providing an additional layer of security. This approach has the potential to revolutionize the way data is stored and shared, providing a more secure, scalable, and cost-effective alternative to traditional centralized storage solutions.

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
1.1	WORKING OF BLOCKCHAIN	2
1.2	DISTRIBUTED DATABASE	2
1.3	NETWORK OF NODES	3
5.2	PROPOSED ARCHITECTURE	23
6.3.1	FRONT-END USER INTERFACE USING REACT	29
6.3.4	ETHEREUM NETWORK, GANACHE CONNECTION	33
10.1	HOME PAGE	43
10.2	FILE UPLOAD	44
10.3	METAMASK TRANSACTION	45
10.3.1	WALLET	45
10.4	VIEW THE FILE UPLOADED	46

CHAPTER 1

INTRODUCTION

1.1 BLOCKCHAIN

Blockchain could be a data structure that could be a growing list of information blocks. The knowledge blocks are unit coupled along, such recent blocks can't be removed or altered. Blockchain is the backbone Technology of the Digital Crypto Currency BitCoin. The blockchain is a distributed database of records of all transactions or digital events that have been executed and shared among participating parties. Each transaction is verified by the majority of participants of the system. It contains every single record of each transaction.

Bitcoin is the most popular cryptocurrency an example of the blockchain. Blockchain Technology first came to light when a person or Group of individuals name 'Satoshi Nakamoto' published a white paper on "BitCoin: A peer-to-peer electronic cash system" in 2008. Blockchain Technology Records Transaction in Digital Ledger which is distributed over the Network thus making it incorruptible. Anything of value like Land Assets, Cars, etc. can be recorded on Blockchain as a Transaction.

How does Blockchain Technology Work?

One of the famous use of Blockchain is Bitcoin. Bitcoin is a cryptocurrency and is used to exchange digital assets online. Bitcoin uses cryptographic proof instead of third-party trust for two parties to execute transactions over the Internet. Each transaction protects through digital signature.

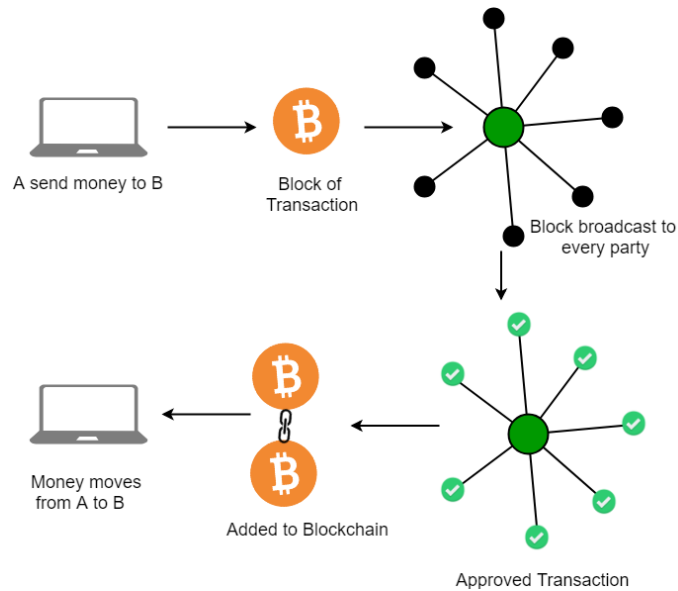


Fig.1.1. Working of Blockchain

Distributed Database: There is no Central Server or System which keeps the data of the Blockchain. The data is distributed over Millions of Computers around the world which are connected to the Blockchain. This system allows the Notarization of Data as it is present on every Node and is publicly verifiable.

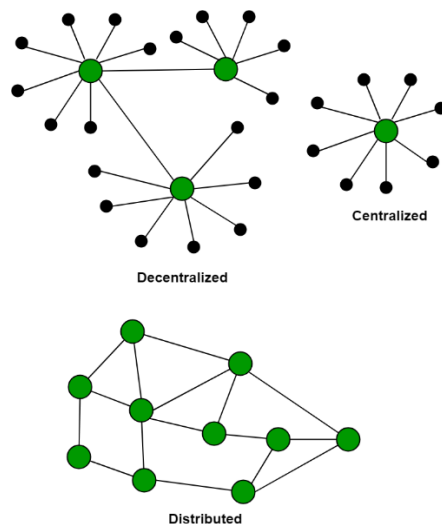


Fig.1.2.Distributed Database

A network of nodes: A node is a computer connected to the Blockchain Network. Node gets connected with Blockchain using the client. Client helps in validating and propagating transactions on to the Blockchain. When a computer connects to the Blockchain, a copy of the Blockchain data gets downloaded into the system and the node comes in sync with the latest block of data on Blockchain. The Node connected to the Blockchain which helps in the execution of a Transaction in return for an incentive is called Miners.

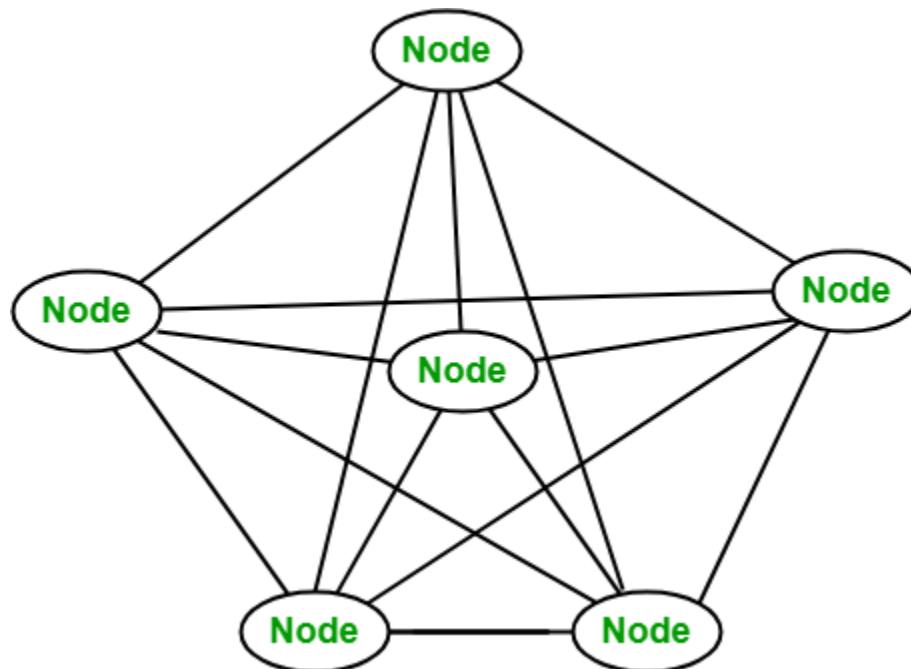


Fig.1.3.Network of Nodes

1.2 ABOUT THE PROJECT

A blockchain is a decentralized, peer-to-peer system where every node in the network keeps a copy of the blockchain, making it immutable. In the proposed system, users' files are encrypted and stored on multiple peers in the network using the IPFS (Interplanetary File System) protocol.

IPFS generates hash values. The hash value specifies the path to the file and is stored on the blockchain. This white paper focuses on distributed secure data storage, data high availability and efficient use of storage resources.

Large files cannot be stored efficiently on the blockchain. On the one hand, the blockchain is full of data that needs to be distributed on the blockchain network. On the other hand, since blockchains are replicated across many nodes, they require a lot of storage space for no immediate purpose, especially if node operators don't need to see all the files stored on the blockchain. It also increases the cost of blockchain operating nodes as more data needs to be processed, transmitted and stored. IPFS is a file sharing system that can be used to store and share large files. It is based on a cryptographic hash that can be easily stored on the blockchain. However, IPFS does not allow users to share files with parties of their choosing.

Required if sensitive or personal data needs to be shared. Therefore, this article presents a modified version of the Interplanetary File System (IPFS) that uses Ethereum smart contracts to enable access-controlled file sharing.

Smart contracts are used to maintain access control lists, and modified IPFS software enforces them. To do this, it interacts with the smart contract whenever a file is uploaded, uploaded or transmitted. Blockchain applications interact directly with a blockchain or smart contract to reach consensus on the execution of transactions, data or code.

A network of heterogeneous nodes stores blockchains, processes transactions, and executes smart contracts when needed. This causes the following problems when working with large data files: Normally, blockchain nodes don't need files to function, which leads to blockchain bloat, resulting in data being replicated across a large number of nodes. On the one hand, storing large files on the blockchain is inefficient. Due to block size limitations, files must be split and reassembled off-chain.

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 portfolio.

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 would thus require connections with higher bandwidths and more storage space to store the blockchain, even partially, thus leading to increased costs.

CHAPTER 2

LITERATURE SURVEY

1. DECENTRALIZED DATA STORAGE USING BLOCKCHAIN

Author: Niraj Wadile

Abstract:

This paper introduces a methodology that uses Blockchain applications to interact either directly with blockchains or with smart contracts in order to achieve consensus on transactions, data or 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 number of nodes data is analysed. The blockchain contains a series of transactions organized logically inside a block, these blocks are tied together using encryption in order to form a chain.

The technology itself forms a decentralized consensus allowing peers to reach consensus on the state of a transaction. Information is usually stored in private databases maintained by each organization. The database technology is commonly described as the CAP theorem and relates to Consistency (C), Availability (A), and Partition Tolerance (P). No database, whether SQL or non-SQL, can simultaneously implement all three properties. In the case of blockchain, data is organized using linked lists using hash pointers instead of plain pointers.

The transaction has size variables and includes a reference to the previous block if it is not a genesis block. The genesis block is the first block of the blockchain that was hardcoded at the time of the release of the blockchain, and its structure varies depending on which blockchain technology is used requires an address to add a block to the blockchain. The address is derived from public key. These are unique identifiers that identify the sender and receiver.

A sender (i.e. a node on) creates a transaction by digitally signing it with a private key to transfer the value of from one address to another.

2. INTEGRATING BLOCKCHAIN AND THE INTERPLANETARY FILE SYSTEM

Author: Jitesh Shamdasani

Abstract:

This paper introduces integrating blockchain and Interplanetary File System (IPFS), hence understanding what the terminologies mean is mandatory. Blockchain is a decentralized database that keeps a permanent record of information and stores it in blocks in consecutive, cryptographic patterns linking these blocks to form a chain. It almost nullifies the chances of hacking the database due to the cryptographic encryption technique used in the blockchain. The digital ledger transaction is duplicated and spread across various nodes of the computer which makes it transparent, decentralized, and immune to tampering network. Since blockchain cannot store large files due to size limitations and high cost, an alternative is necessary to store large files and get the benefits of the decentralized database.

This can be achieved with distributed file storage systems such as the Interplanetary File System (IPFS). Interplanetary File System (IPFS) uses content addressing. That is, check what is in the file and share it with all nodes to save the file. Therefore, the Interplanetary File System (IPFS) copies files over the network.

Here is a simple chart comparing a traditional file storage system to the Interplanetary File System (IPFS). In the Interplanetary File System (IPFS), all computers are interconnected according to the standard, and all users on the network have copies of downloaded files. Additionally, copies of the files are accessible only to those with the Interplanetary File System (IPFS) hash. So, providing security as a hash, acts like a password. The Interplanetary File System (IPFS) is called the "Persistent Web" because files always reside at a specific address, unlike the classic Hypertext Transfer Protocol (HTTP), which protects against unauthorized access. Copies of files are distributed over a peer-to-peer (P2P) network. Each copy has a cryptographic hash of that cannot be decrypted. Data is shared across the network from these peer nodes, called nodes. Each node in the network uses extra information indexing to store relevant content. This will help you find out which node stores which content needs these cryptographic hashes to get complete data. The hash receives data from all different nodes where the data is split and combined to form the downloaded file. Data scrambling here is prevented using a Distributed Hash Table (DHT). Blockchain stores hash addresses in smart contracts. Increase security by hiding the contents of file behind a series of letters and numbers.

Any change, regardless of motive, completely changes this sequence of letters and numbers, which proves the lack of consistency between the requested file and the file stored on the Interplanetary File System (IPFS).

3. BLOCKCHAIN BASED DECENTRALIZED STORAGE SCHEME

Author: Sanket Deshmukh

Abstract:

The paper intends to propose a decentralized storage system based on blockchain technology, which can make full use of the remaining space of personal hard disks of users around the world. The storage provider performs a data integrity certificate for the user, and after verifying that the verification is passed, the user pays the storage fee to the storage provider through lightning network technology. All proofs and payment information are stored in the blockchain, which guarantees the security and credibility of the system.

Compared with the current mainstream distributed storage systems this scheme has been improved in terms of system access and payment method. a distributed storage scheme based on blockchain. The user uploads the encrypted data to the middleman, and the middleman sends the data to the storage provider and informs the user of the data storage location. After the data integrity certificate is completed between the user and the storage provider, the user uses lightning network technology to pay the storage fee to the storage provider.

Blockchain technology provides immutable data storage in that it allows transactions to only be appended and never to be modified or removed. However, the data storage on the blockchain has a cost model that differs from conventional data storage in terms of size and cost. For instance, in terms of size, the Bitcoin blockchain provides the store of arbitrary data in transactions. It was limited to 80 bytes which has been reduced to 40 in February 2014. In terms of cost, storing 80 bytes on the Bitcoin blockchain costs roughly US\$0.03617 and US\$0.007 when using Ethereum. As described, blockchain transactions hold only very small amounts of data, wherefore it is crucial to choose what data should be placed On-chain and what should be kept Off-chain.

There are several Off-chain data storage solutions designed to be friendly to blockchains, such as Storj, FileCoin, Sia and IPFS. These solutions share the idea of a peer-to-peer distributed file system, where the data is shredded, encrypted and distributed to multiple nodes in the network to ensure their safety and availability.

One main issue with these systems is the lack of access control. These features are extremely important if blockchain-based applications equipped with off-chain data storage solutions are deployed in operational and sensitive environments such as the financial and public sectors.

4. SHARING SYSTEM USING BLOCKCHAIN NETWORKS

Author: Mobaashir Sayyed

Abstract:

This white paper is intended to supplement our previous review of insecure data storage on Android smartphones by expanding the range of security threats and solutions. I think thorough testing of the Android storage model is warranted.

So, we are looking at attacks, threats and solutions for Android from 2013-2018. We also propose a phased separation of the threat model for Android data storage based on physical and software threats, and consider several issues for each class.

We are also exploring solutions to reduce each category of. Therefore, the Android app is designed to securely store and secure using BlockChain and IPFS.

There have been dramatic changes in information technology over the past few years. Easy to use and easy to develop, the is open source and has caught the attention of developers who want to create content for the masses. Cloud computing is known as the next big step in all its forms using standard technology.

From businesses to non-profit organizations and individual users, has many programs that can use cloud computing to provide better, faster and smarter computing. Seems to. This document aims to combine these two tasks to create an Android cloud system and provide users with a cloud computing experience in the palm of their hands.

The performance of mobile devices, most of which are smartphones, has improved rapidly in recent years. Many users have high- performance smartphones and enjoy content longer on their smartphones than on other devices.

As a result users are constantly exchanging content, and the requirements for exchanging files through the extension call have increased significantly. To overcome these problems, we introduce seamless file sharing app for Android devices. We expect the proposed application to be a reliable and cost-effective file sharing solution for mobile devices.

5. DESIGN OF DATA SHARING PLATFORM BASED ON BLOCKCHAIN AND IPFS TECHNOLOGY

Author: Mobaashir Sayyed

Abstract:

This essay focuses on the rapid development of information technology and digitalization, as well as the rising amount of data produced by people in their daily lives, including office documents, photos, and videos. With the advancement of blockchain technology, it now exhibits the following features: Providing a fresh approach to the issue of data security is "traceability," "hard to tamper with," and "decentralisation."

A data sharing plan based on blockchain technology has been presented by certain academics to guarantee data privacy security and sharing through access permission setting and data storage. To address issues like data loss and data tampering, some academics have created a platform for exchanging data that combines blockchain and machine learning technology.

The data unchained storage mechanism of cloud + blockchain has been suggested by certain academics to realise data transfer and storage.

The transaction has size variables and includes a reference to the previous block if it is not a genesis block. The genesis block is the first block of the blockchain that was hardcoded at the time of the release of the blockchain, and its structure varies depending on which blockchain technology is used requires an address to add a block to the blockchain. The address is derived from public key. These are unique identifiers that identify the sender and receiver.

A sender (i.e. a node on) creates a transaction by digitally signing it with a private key to transfer the value of from one address to another.

Blockchain technology provides immutable data storage in that it allows transactions to only be appended and never to be modified or removed. However, the data storage on the blockchain has a cost model that differs from conventional data storage in terms of size and cost. For instance, in terms of size, the Bitcoin blockchain provides the store of arbitrary data in transactions. It was limited to 80 bytes which has been reduced to 40 in February 2014.

CHAPTER 3

3. SYSTEM ANALYSIS

The creative and challenging phase of the system life cycle is system analysis. The term analysis describes the initial stage of the system. The term design describes the final system generated and process by which it is developed. It refers to the technical specification that will be applied in implementing the system. It also includes the construction of programs and program testing.

System analysis and design is a solution a “how to” approach to the creation of a new system. This important phase is composed of several steps.

It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study.

3.1 EXISTING SYSTEM

1. Censorship:

Censorship in the cloud refers to the practice of restricting access to certain content or services hosted on cloud platforms, Providers have the ability to monitor and filter content, which can be used to censor or restrict access to certain types of information.

2. Architecture:

Blockchain-based decentralized cloud storage typically involves the use of a blockchain network as well as a cloud storage system and the data is stored on both the blockchain network and a cloud storage system.

3. Scalability:

Blockchain-based decentralized cloud storage may be limited in scalability due to the transaction processing capacity of the blockchain network and the capacity of the cloud storage system.

Eg: Storj, Filecoin

3.2 PROPOSED SYSTEM

With the ongoing development of information technology, people's need for computing and resource storage has recently demonstrated a tendency of rapid rise. In order to satisfy their desire for greater computational power and larger storage, people are continually investigating novel methods of storage room Cloud computing has drawn the attention of academic and commercial circles since the introduction of computing paradigms like peer-to-peer, grid, utility, and distributed computing.

As a result of the way cloud computing divides up processing work among huge groups of computers, different application systems can access computational resources, storage capacity, and software services as needed. Blockchain technology was presented by Nakamoto in 2008 when he released "Bitcoin: a peer-to-peer electronic cash system," and it was built and used in Bitcoin. The use of blockchain technology.

Whenever a node uploads a file, as per IPFS the node creates chunks and the associated Merkle DAG. During this process all content identifiers corresponding to the file are collected. These are then passed on to the permissions package, which registers the files in the smart contract, using the add Block functions.

This is done by forming the transactions necessary for this purpose the permissions system further more allows the verification of transactions based on the corresponding content identifiers. If the transactions succeed, the execution returns to the IPFS code which stores the blockchain the local storage and then registers itself as a provider of the content identifiers. If the transaction fails, ownership of at least one of the content identifiers and therefore the whole file could not be claimed.

Censorship:

A storage without censorship would allow for complete freedom of expression and information sharing, this would provide an open and transparent environment for people to express their opinions and ideas without fear of censorship or persecution.

Architecture:

Blockchain-based decentralized data storage using IPFS involves the use of the Inter Planetary File System (IPFS) as the underlying storage system and data is stored only on the IPFS network.

Scalability:

IPFS has the potential to be more scalable since it is not limited by the capacity of a blockchain network and can leverage the distributed storage capacity of the IPFS network. Cost: IPFS is relatively less expensive since it is based on a distributed storage network that may be less costly to maintain.

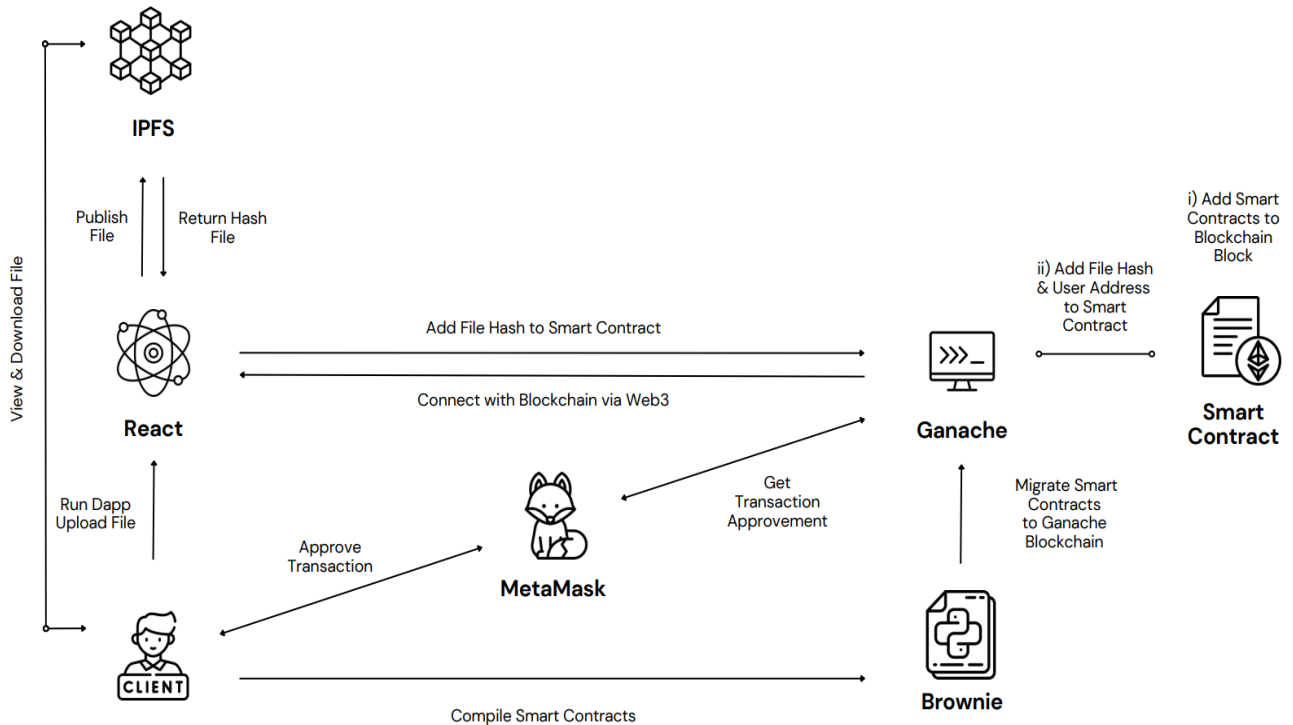


Fig.3.2. Proposed Architecture

CHAPTER 4

4.1 HARDWARE SPECIFICATION

Hardware is a general term for the physical artifacts of a technology. It may also mean the physical components of a system. These are the devices, which are made to perform specific functions and can do only that works as instructed.

They can be both electronic devices and mechanical systems.

- System : Pentium IV 2.4 GHz
- Hard Disk : 160 GB
- Monitor : 15 VGA Color
- Mouse : Logitech.
- Keyboard : 110 keys enhanced

4.2 SOFTWARE SPECIFICATION

Computer software, or just software, is a collection of computer programs and related data that provide the instructions for telling a computer what to do and how to do it.

- O/S : Windows 7 and above
- Language : Solidity
- Client Side : Web 3, React, IPFS Libraries
- Server : Node.js, IPFS

4.3 SOFTWARE DESCRIPTION

4.3.1 Windows 7

Windows 7 is a popular operating system developed by Microsoft. It was released on July 22, 2009, and was designed to be a significant improvement over its predecessor, Windows Vista.

Windows 7 introduced several new features, including improved performance, a streamlined user interface, enhanced security, and new multimedia tools. It also included a redesigned taskbar that allowed users to pin their favorite applications for quick access, as well as new Jump Lists that provided quick access to frequently used files and tasks.

Windows 7 was well-received by users and was widely adopted, becoming one of the most popular operating systems in history. It was also the last version of Windows to have a traditional Start menu and the Aero user interface.

However, as of January 14, 2020, Microsoft ended support for Windows 7, which means that it no longer receives security updates or technical support from Microsoft. Users still running Windows 7 are encouraged to upgrade to a newer version of Windows, such as Windows 10, to ensure that their computer remains secure and up-to-date.

Windows 7 was a significant improvement over its predecessor, Windows Vista, and included several new features that made it a popular operating system. Some of the notable features of Windows 7 include:

- **Improved performance:** Windows 7 was designed to run faster and more efficiently than Windows Vista, thanks to its improved memory management and optimized system resources.

- **Streamlined user interface:** Windows 7 featured a new taskbar and Start menu, which allowed users to pin their favorite applications for quick access and provided more convenient access to common tasks.
- **Enhanced security:** Windows 7 introduced several new security features, such as BitLocker encryption, improved firewall protection, and User Account Control (UAC), which helped prevent unauthorized changes to the system.
- **Multimedia tools:** Windows 7 included several new multimedia tools, such as Windows Media Player 12, which supported new audio and video formats, and Windows DVD Maker, which allowed users to create and burn DVDs from their home videos.
- **Home Group:** Windows 7 introduced Home Group, a feature that allowed users to easily share files and printers between computers on the same network.
- **Improved compatibility:** Windows 7 was designed to be more compatible with older software and hardware, making it easier for users to upgrade from previous versions of Windows.
- **Touch and handwriting recognition:** Windows 7 included support for touchscreens and handwriting recognition, making it easier to use on tablets and other touch-enabled devices.

Overall, Windows 7 was a well-received operating system that introduced several new features and improvements over its predecessor, making it a popular choice among users.

4.3.2 Solidity

Solidity is a programming language used for writing smart contracts on the Ethereum blockchain. It is a high-level, contract-oriented language that is designed to be easy to learn and to allow developers to write secure and efficient code.

Solidity is an object-oriented language, and its syntax is similar to that of JavaScript. It supports various data types, including integers, strings, arrays, and custom types. It also supports various control structures, such as if/else statements, loops, and switch statements.

One of the key features of Solidity is its ability to define and manage smart contracts. Smart contracts are self-executing contracts with the terms of the agreement between buyer and seller being directly written into lines of code. Solidity allows developers to write these contracts in a way that ensures they are secure and tamper-proof.

Solidity is used to write code that is deployed on the Ethereum blockchain and executed by the Ethereum Virtual Machine (EVM). Smart contracts written in Solidity are used for a variety of purposes, including creating and managing digital assets, enforcing business logic, and executing financial transactions.

Overall, Solidity is a powerful programming language that is essential for anyone interested in developing decentralized applications or working with smart contracts on the Ethereum blockchain.

4.3.3 Web 3

Web 3, also known as the decentralized web, is the next evolution of the internet, which aims to decentralize control over online activities and data. It is based on the idea of using blockchain technology, peer-to-peer networks, and decentralized applications (dApps) to create a more open, transparent, and secure web.

Web 3 is often referred to as the "Internet of Value" as it allows for the exchange of value, such as digital assets or cryptocurrencies, without the need for intermediaries like banks or financial institutions. It is built on decentralized protocols that enable trust, security, and privacy without the need for centralized entities to manage them.

Some of the key features of Web 3 include:

- **Decentralization:** Web 3 is built on a decentralized network, which means that data and applications are not controlled by a single entity, making it more resilient to censorship and hacking.
- **Blockchain Technology:** Web 3 is based on blockchain technology, which allows for secure and transparent record-keeping and transactions.
- **Smart Contracts:** Web 3 supports the use of smart contracts, which are self-executing contracts with the terms of the agreement between buyer and seller being directly written into lines of code. This makes it possible to automate complex financial transactions and enforce business logic.
- **Decentralized Applications (DAPP):** Web 3 allows for the development of decentralized applications that are built on top of decentralized protocols. These

DAPP are the open-source and run on a peer-to-peer network, making them more secure and resistant to censorship.

- **Interoperability:** Web 3 allows for interoperability between different blockchains and decentralized networks, making it possible to exchange value and data between them.

Overall, Web 3 represents a major shift in the way we think about the internet and our digital lives. It has the potential to revolutionize industries such as finance, healthcare, and social media by enabling trust, security, and transparency without the need for intermediaries.

4.3.4 React

React is an open-source JavaScript library used for building user interfaces (UIs) on the web. It was developed by Facebook and is now maintained by a community of developers and companies.

React is known for its component-based architecture, which allows developers to break down a complex UI into smaller, reusable parts called components. These components can be combined to create more complex UIs, making it easier to manage and maintain large-scale web applications.

One of the key benefits of React is its use of a virtual DOM (Document Object Model), which is an in-memory representation of the actual DOM. This allows React to update the UI efficiently by minimizing the number of actual DOM manipulations. When a component's state changes, React only updates the parts of the DOM that need to be updated, which results in faster rendering and better performance.

React also supports server-side rendering, which allows for faster initial load times and better search engine optimization (SEO). This is because the server can render the initial HTML and CSS of the application before sending it to the client, which reduces the time it takes to render the page.

React is used by many companies and organizations, including Facebook, Instagram, Netflix, Airbnb, and more. It is widely considered to be one of the most popular and powerful UI libraries for building web applications.

Overall, React has revolutionized the way developers build web applications by providing a powerful, efficient, and scalable way to manage complex user interfaces. Its component-based architecture, virtual DOM, and server-side rendering capabilities have made it a popular choice among developers looking to build high-performance web applications.

4.3.5 IPFS Libraries

IPFS (Inter Planetary File System) is a distributed protocol for storing and accessing files on a peer-to-peer network. There are several libraries available for developers to use when building applications that utilize IPFS. Here are some of the most popular IPFS libraries:

- **js-ipfs:** This is the official JavaScript implementation of the IPFS protocol. It allows developers to build decentralized applications that can store and retrieve data from the IPFS network using JavaScript.
- **go-ipfs:** This is the official Go implementation of the IPFS protocol. It provides a command-line interface (CLI) and a set of APIs for developers to use when building applications that utilize IPFS.

- **ipfs-api:** This is a JavaScript library that provides a simple API for interacting with an IPFS node. It allows developers to add, get, and remove files from the IPFS network using JavaScript.
- **ipfs-http-client:** This is another JavaScript library that provides a higher-level API for interacting with an IPFS node. It supports async/await and allows developers to easily manage files on the IPFS network.
- **ipfs-core:** This is a modular JavaScript library that allows developers to use only the parts of IPFS that they need for their application. It provides a set of modular components that can be combined to build custom IPFS applications.
- **py-ipfs:** This is a Python implementation of the IPFS protocol. It allows developers to interact with the IPFS network using Python and provides a set of APIs for managing files on the network.

These are just a few examples of the many IPFS libraries available for developers to use. Each library provides a different set of features and capabilities, so developers can choose the one that best fits their needs when building applications that utilize IPFS.

4.3.6 Nodejs

Node.js is an open-source, cross-platform JavaScript runtime environment that allows developers to run JavaScript code on the server-side. It was initially developed by Ryan Dahl in 2009 and has since become a popular choice for building web applications and server-side APIs.

Node.js is built on top of the V8 JavaScript engine from Google Chrome, which allows it to

execute JavaScript code very quickly. It provides a range of built-in modules and libraries that make it easy to perform common tasks such as reading and writing files, creating HTTP servers, and working with databases.

One of the key benefits of Node.js is its non-blocking I/O model, which allows it to handle large numbers of simultaneous connections without slowing down or blocking the server. This makes it well-suited for building real-time applications such as chat applications, online games, and streaming services.

Node.js also has a large and active community of developers who contribute to its development and create a wide range of third-party modules and libraries that can be used to extend its functionality. This makes it easy for developers to find and use existing solutions to common problems, which can save time and effort when building applications.

Some of the popular frameworks and libraries built on top of Node.js include Express.js for building web applications, Socket.io for real-time communication, Mongoose for working with MongoDB databases, and many others.

Overall, Node.js has become a popular choice for building web applications and server-side APIs due to its speed, scalability, and large community. Its non-blocking I/O model and extensive library of modules and libraries make it a powerful and versatile tool for building a wide range of applications.

4.3.7 IPFS

IPFS (Inter Planetary File System) is a distributed protocol for storing and accessing files on a peer-to-peer network. It was developed by Juan Benet and his team at Protocol Labs in 2014 and has since gained popularity as a decentralized alternative to traditional HTTP-based protocols for content distribution.

IPFS uses a content-addressed system, which means that each file is assigned a unique hash based on its content. This hash can be used to retrieve the file from any node on the IPFS network, regardless of where it was originally stored. This makes it easier to distribute content and ensures that files are always available, even if the original source goes offline.

IPFS also uses a decentralized network architecture, which means that there is no central server or point of control. Instead, each node on the network contributes to the storage and distribution of files, which helps to ensure the network's resilience and fault tolerance.

Another key feature of IPFS is its support for mutable content, which allows files to be updated and changed over time while still maintaining their unique content address. This is achieved through the use of IPNS (Inter Planetary Naming System), which provides a way to map mutable content to a fixed content address.

IPFS has many potential use cases, including decentralized file sharing, content distribution, and version control. It can also be used to build decentralized applications that are not reliant on centralized servers or infrastructure.

Overall, IPFS represents a significant shift in the way we think about content distribution and storage on the web. Its decentralized architecture and content-addressed system provide a more secure, resilient, and efficient way to share and distribute files, making it a promising technology for the future of the web.

CHAPTER 5

5 SYSTEM DESIGN

Designing is the most important phase of software development. It requires a careful planning and thinking on the part of the system designer. Designing software means to plan how the various parts of the software are going to achieve the desired goal. It should be done with utmost care because if the phase contains any error then will affect the performance of the system, as a result it may take more processing time, more response time, extra coding workload, etc.,

Software design sits at the technical kernel of the software engineering process and is applied regardless of the software process model that is used. After the software requirements have been analyzed and specified, software design is the first of the three technical activities designing, coding and testing that are required to build and verify the software. Each activity transforms in such a manner that ultimately results in validated computer software.

5.1 DESIGN GOALS

The following goals were kept in mind while designing the system: make system user-friendly. This was necessary so that system could be used efficiently and system could act as catalyst in achieving objectives. Make system compatible and it should fit in the total integrated system. Future maintenance and enhancement must be less. Make the system reliable, understandable and cost- effective.

5.1.1 Input Design

The input design is the link between the information system and the user. Input design is the process of converting user-oriented input into computer-based format. The goal of designing input data is to make data entry as easy as possible and free from errors. Errors in the input are handled in the input design. The input data is the lifeblood of a system and have to be analyzed and designed with utmost care and consideration. The decisions made during the input design are:

- ✓ To provide cost effective method of input
- ✓ To achieve the highest possible level of accuracy
- ✓ To ensure that input is understood by the user

Taking into account, all the above needs of the input designing, the new process designing could have the following features. In all cases processing should be automatic and manual work should be kept minimum. Similarly, the user input should also be kept minimum or least extend possible. By giving the above input calculations should be automatic. The auto calculation is the motto of this new process, while providing the auto calculation the results with minimum time operation, should be accurate and free from errors.

5.1.2 Objectives

Input design is the process of converting a user-oriented description of the input into a computer based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system. It is achieved by entering the user's sensitive and insensitive data. This also leads to higher security.

5.1.3 Output Design

Output design generally refers to the results and information's that are generated by the system for many end-users; output is the main reason for developing the system and the basis on which they evaluate the usefulness of the application. The basic requirements of output are that it should be accurate, timely and appropriate, in terms of content, medium and layout for its intended purpose.

The important task of any system lies in the capability of producing high quality output and reports. Computer output is the most important and direct source of information to the user. A good output design contains all required information and well formatted for the better visualization and avoids the complexity in displaying the data this system was developed by keeping in mind and proper output is displayed for the users accurately.

5.2 SYSTEM ARCHITECTURE

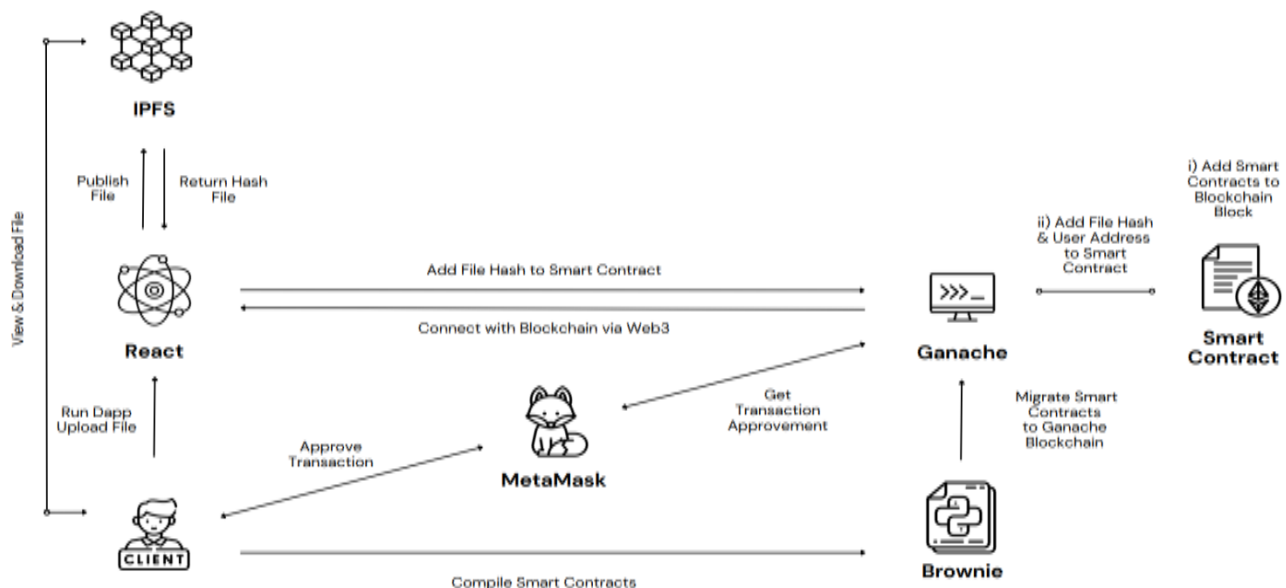


Fig 5.2.1 Proposed Architecture

CHAPTER 6

6 SYSTEM IMPLEMENTATION

Implementation of the software refers to the final installation of the package in its real environment, to the satisfaction of the intended users and the operation of the system. The people are not sure that the software is meant to make their job easier. The active user must be aware of the benefits of using the system. Their confidence in the software built up. Proper guidance is impaired to the user so that he is comfortable in using the application.

Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not running on the server, the actual processes will not take place.

6.1 USER TRAINING

To achieve the objectives and benefits expected from the proposed system it is essential for the people who will be involved to be confident of their role in the new system. As system becomes more complex, the need for education and training is more and more important.

6.1.1 Training on The Application Software

After providing the necessary basic training on the computer awareness, the user will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design, type of help on the screen, type of error while entering the data, the corresponding validation check at each entry and the ways to correct the data enter.

6.1.2 Operational Documentation

Once the implementation plan is decided, it is essential that the user of the system is made familiar and comfortable with the environment. A documentation providing the whole operations of the system is being developed. Useful tips and guidance is given inside the application itself to the user. The system is developed user friendly so that the user can work the system from the tips given in the application itself.

6.1.3 System Maintenance

The maintenance phase of the software cycle is the time in which software performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is to make adaptable to the changes in the system environment. There may be social, technical and other environmental changes, which affects a system which is being implemented. Software product enhancements may involve providing new functional capabilities, improving user displays and mode of interaction, upgrading the performance characteristics of the system.

6.1.4 Corrective Maintenance

The first maintenance activity occurs because it is unreasonable to assume that software testing will uncover all latent errors in a large software system. During the use of any large program, errors will occur and be reported to the developer. The process that includes the diagnosis and correction of one or more errors is called corrective maintenance.

6.1.5 Adaptive Maintenance

The second activity that contributes to a definition of maintenance occurs because of the rapid change that is encountered in every aspect of computing. Therefore adaptive maintenance termed as an activity that modifies software to properly interfere with a changing environment is both necessary and common place.

6.1.6 Perceptive Maintenance

The third activity that may be applied to a definition of maintenance occurs when a software package is successful. As the software is used, recommendations for new capabilities, modifications to existing functions, and general enhancement are received from users. To satisfy requests in this category, perceptive maintenance is performed. This activity accounts for the majority of all efforts expended on software maintenance.

6.2 MODULES

- Front-end User Interface using React
- IPFS Client Libraries
- Solidity in polygon network
- Ethereum Network, Ganache Connection

6.3 MODULE DESCRIPTION

6.3.1 Front-end User Interface using React

In this the frontend is designed by using the react framework. Where the user can login to his metamask account. After logged in to their account, they can upload the file in our IPFS storage (INTER PLANETARY FILE SYSTEM), where we can store our data of any format like pdf, docs, images, videos, zip files, etc.

In frontend there will be an upload button, where the user can upload their file and it redirects to the page where the transaction fees has to be pay from the metamask. After the payment has been done successfully, the file will be uploaded in this storage. The user can view the file or user can download the file using the hash value generated for the file, which is uploaded.

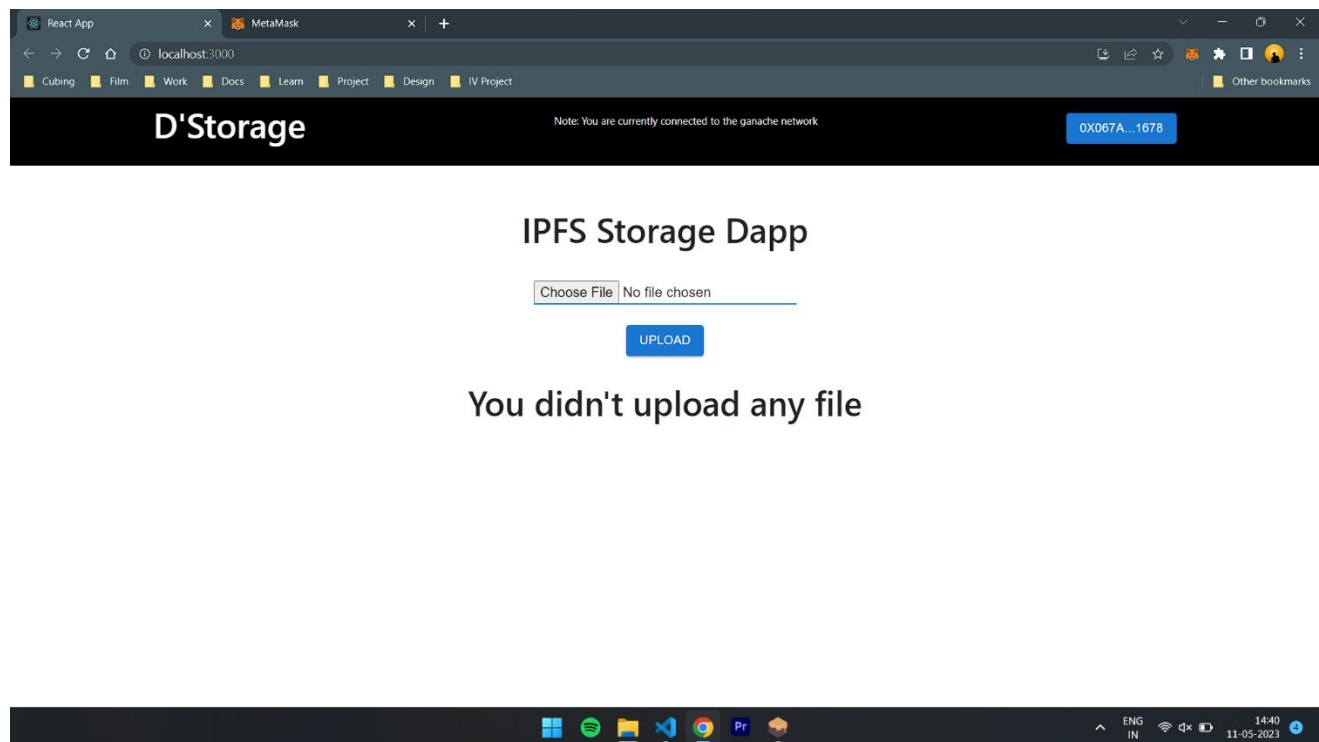


Fig. 6.3.1 Frontend User Interface

6.3.2 IPFS Client Libraries

IPFS (Inter Planetary File System) is a protocol and network designed for decentralized storage and sharing of files. It allows users to store and access data without relying on a centralized server, which can improve data privacy, security, and availability.

One important aspect of IPFS is the availability of client libraries, which are software libraries that allow developers to interact with the IPFS network. These libraries provide a high-level abstraction of the IPFS protocol, making it easier for developers to use IPFS in their applications.

There are several IPFS client libraries available in different programming languages, including JavaScript, Go, Python, and Rust. These libraries provide a variety of functionalities, including adding and retrieving files, publishing and resolving IPNS (Inter Planetary Naming System) names, and managing IPFS nodes.

JavaScript is one of the most widely used programming languages for web development, and there are several IPFS client libraries available for it. For example, `ipfs-api` and `ipfs-http-client` provide a simple way to interact with an IPFS node over HTTP. On the other hand, `js-ipfs` is a pure JavaScript implementation of IPFS that can run both in the browser and in Node.js.

In summary, IPFS client libraries are an important aspect of decentralized data storage with IPFS. They provide a high-level abstraction of the IPFS protocol, making it easier for developers to use IPFS in their applications. There are several IPFS client libraries available in different programming languages, and each library provides a unique set of functionalities and advantages.

6.3.3 Solidity in Polygon Network

In this project, we define a Solidity contract called IPFS Storage. This contract has two functions: `uploadFile` and `getFile`.

The `uploadFile` function takes a string parameter `_fileHash`, which represents the hash of the file we want to upload to IPFS. This function stores the file hash in the `fileHash` variable of the contract.

The `getFile` function is a getter function that returns the `fileHash` variable, which represents the hash of the uploaded file.

To upload a file to IPFS, we first need to use an IPFS client library in our application to interact with the IPFS network. In this example, we're assuming that we have already uploaded the file to IPFS and obtained its hash.

To store the file on the Polygon network, we would need to deploy this Solidity contract to the Polygon network using a tool like Remix or Truffle. Once the contract is deployed, we can call the `uploadFile` function and pass in the file hash as a parameter. This will store the file hash in the `fileHash` variable of the contract.

To retrieve the file from the Polygon network, we can call the `getFile` function, which will return the file hash. We can then use an IPFS client library to retrieve the file from IPFS using its hash. Note that storing large files on the blockchain may not be practical due to gas costs and storage limitations. It's recommended to store only the hash of the file on the blockchain, and store the actual file on IPFS or another decentralized storage network.

6.3.4 Ethereum Connection and Ganache

To connect IPFS with Solidity on the Ethereum network, you can use an IPFS client library in your smart contract code. There are several IPFS client libraries available for Solidity, including `ipfs-api` and `js-ipfs`.

We're using the IPFS contract from the `ipfs-api` library to interact with IPFS. The IPFS contract provides a `add` function that allows us to add data to IPFS and get its hash in return.

To use the IPFS contract, we need to pass an instance of it to our `MyContract` contract's constructor. This can be done in the deployment code of the contract.

In the `uploadFile` function, we take a string parameter `_fileName` to represent the name of the file, and a bytes parameter `_fileData` to represent the actual file data. We then call the `add` function of the IPFS contract to add the file data to IPFS and get its hash in return.

Once we have the file hash, we can store it in contract storage or emit an event to make it accessible to other contracts or external applications.

Note that storing large files on IPFS may result in high gas costs, so it's recommended to store only the hash of the file on the blockchain, and store the actual file on IPFS or another decentralized storage network.

GANACHE:

ACCOUNTS

BLOCKS

TRANSACTIONS

CONTRACTS

EVENTS

LOGS

CURRENT BLOCK0

GAS PRICE20000000000

GAS LIMIT6721975

HARDFORKMUIRGLACIER

NETWORK ID5777

RPC SERVERHTTP://127.0.0.1:7545

MINING STATUSAUTOMINING

WORKSPACECOLD-TICKET

SWITCH

MNEMONIC ?

convince coil shrimp soup one about property paddle gold unusual invite woman

HD PATH

m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX	
0x35C80c73982c116985c2F6Eab1B47CCaFD2CB13b	100.00 ETH	0	0	
0x78E0DD58101056CdDe719356E8320CE2Ed26cE00	100.00 ETH	0	1	
0x186d8ecF388C2FE8f028EdB4ED9Ab7234e4d76C3	100.00 ETH	0	2	
0x15b6Eec0160B4Af4581937b7949A7F9789EB698b	100.00 ETH	0	3	
0x43420744E79A228d1354f13Ef10F0cb67Ec8cc05	100.00 ETH	0	4	
0x69a6a62dC129574bD90cba41878206aFA9FbD991	100.00 ETH	0	5	
0xc2204da9AA94a58D6EC24916AcB2aD558033A0ab	100.00 ETH	0	6	

Fig.6.3.4.1 Ganache

CHAPTER 7

TESTING

7.1 INTRODUCTION ABOUT TESTING

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is executing a system in order to identify any gaps, errors or missing requirements in contrary to the actual requirements.

Testing is a process used to identify the correctness, completeness and quality of developed computer software. With that in mind, testing can never completely establish the correctness of computer software.

Testing helps in verifying and validating if the software is working as it is intended to be working. This involves using static and dynamic methodologies to test the application.

7.1.1 Testing Objectives

- To make sure that the end result meets the business and user requirements.
- To ensure that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specifications.
- To gain the confidence of the customers by providing them a quality product.

7.1.2 Testing Methodologies

Software Testing Methodology is defined as strategies and testing types used to certify that the Application under Test meets client expectations. Test Methodologies include functional and non-functional testing to validate the AUT.

There are different types of techniques and methodologies involved in testing:

- Unit Testing
- Integration Testing
- System Testing
- Black Box Testing
- White Box Testing
- Performance Testing
- Acceptance Testing

7.1.2.1 Unit Testing

This type of testing is performed by the developers before the setup is handed over to the testing team to formally execute the test cases. Unit testing is performed by the respective developers on the individual units of source code assigned areas. The developers use test data that is different from the test data of the quality assurance team. The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

7.1.2.2 Integration Testing

Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. Software development environment, bottom-up testing is usually done first, followed by top-down testing. The process concludes with multiple tests of the complete application, preferably in scenarios designed to mimic actual situations. Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems.

7.1.2.3 System Testing

System testing tests the system as a whole. Once all the components are integrates, the application as a whole is tested rigorously to see that it meets the specified Quality Standards. This type of testing is performed by a specialized testing team.

System testing is the first step in the Software Development life Cycle, where the application is tested as a whole. The application is tested thoroughly to verify that it meets the functional and technical specifications.

7.1.2.4 Blackbox Testing

Black Box Testing, also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories:

Incorrect or missing functions

- Interface errors
- Errors in data structures or external database access
- Behavior or performance errors
- Initialization and termination errors

7.1.2.5 Whitebox Testing

White Box Testing (Also Known As Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing Or Structural Testing). In a software testing method in which the internal Structure/Design/Implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential. White box testing is testing beyond the user interface and into the Nitty-Gritty of a system.

This Method Is Named So Because The Software Program, In The Eyes Of The Tester, Is Like A White/Transparent Box; Inside Which One Clearly Sees.

- **White-Box Testing:** Testing Based On An Analysis Of The Internal Structure Of The Component Or System.
- **White-Box Test Design Technique:** Procedure To Derive And/or Select Test Cases Based On An Analysis Of The Internal Structure Of A Component Or System.

7.1.2.6 Performance Testing

A type of Physical test covering a wide range of engineering or functional evaluations where a material, product, or system is not specified by detailed material or component specifications: rather, emphasis is on the final measurable performance characteristics.

7.1.2.6 Acceptance Testing

The types of acceptance testing are:

- The **User Acceptance test:** focuses mainly on the functionality thereby validating the fitness-for-use of the system by the business user. The user acceptance test is performed by the users and application managers.
- The **Operational Acceptance test:** also known as Production acceptance test validates whether the system meets the requirements for operation. In most of the organization the operational acceptance test is performed by the system administration before the system is released. The operational acceptance test may include testing of backup/restore, disaster recovery, maintenance tasks and periodic check of security vulnerabilities
- **Contract Acceptance testing:** It is performed against the contract's acceptance criteria for producing custom developed software. Acceptance should be formally defined when the contract is agreed.
- **Compliance Acceptance testing:** It is also known as regulation acceptance testing is performed against the regulations which must be adhered to, such as governmental, legal or safety regulations.

CHAPTER 8

8.1 CONCLUSION

In this paper, first investigated the state-of-art designs of smart home systems and then propose a generic smart home architecture for the future development of smart home systems. The basic concept behind the development of this system is to improve the communication between the IOT devices in smart home. The proposed approach provides better transmission support to entire network and removes the excessive overheads of transmission via LMA even after authentication. This allows more traffic to pass efficiently without much delay and loss. Our system provides Route Optimization (RO) in which the speed of the data transmission increases by the use of IPV6 communication protocol. It also provides secured data transmission by the use of weight and light symmetric encryption techniques which preserves the privacy of the users.

8.2 FUTURE WORK

In future, the proposed system has some low UI designs, so in future, there are some updating will be done, such as,

- UI Enhancement
- Bug clearance

CHAPTER 9
APPENDIX – I
SOURCE CODE

9.1 SOLIDITY FILES:
IPFS Storage

```
pragma solidity ^0.5.0;

contract DStorage {
    string public name = 'DStorage';
    uint public fileCount = 0;
    mapping (uint => File) public files;

    struct File {
        uint fileId;
        string fileHash;
        uint fileSize;
        string fileType;
        string fileName;
        string fileDescription;
        uint uploadTime;
        address payable uploader;
    }

    event FileUploaded(
        uint fileId,
        string fileHash,
        uint fileSize,
        string fileType,
        string fileName,
        string fileDescription,
        uint uploadTime,
        address payable uploader
    );

    constructor() public {
    }
```

```

function uploadFile(string memory _fileHash, uint _fileSize, string memory
_fileType, string memory _fileName, string memory _fileDescription) public {

    // Make sure the file hash exists
    require(bytes(_fileHash).length > 0);

    // Make sure file type exists
    require(bytes(_fileType).length > 0);

    // Make sure file description exists
    require(bytes(_fileDescription).length > 0);

    // Make sure file fileName exists
    require(bytes(_fileName).length > 0);

    // Make sure uploader address exists
    require(msg.sender!=address(0));

    // Make sure file size is more than 0
    require(_fileSize>0);

    // Increment file id
    fileCount ++;

    // Add File to the contract
    files[fileCount] = File(fileCount, _fileHash, _fileSize, _fileType, _fileName,
_fileDescription, now, msg.sender);
    // Trigger an event
    emit FileUploaded(fileCount, _fileHash, _fileSize, _fileType, _fileName,
_fileDescription, now, msg.sender);
}
}

```

Migration.sol:

```
pragma solidity >=0.4.21 <0.6.0;

contract Migrations {
    address public owner;
    uint public last_completed_migration;

    constructor() public {
        owner = msg.sender;
    }

    modifier restricted() {
        if (msg.sender == owner) _;
    }

    function setCompleted(uint completed) public restricted {
        last_completed_migration = completed;
    }

    function upgrade(address new_address) public restricted {
        Migrations upgraded = Migrations(new_address);
        upgraded.setCompleted(last_completed_migration);
    }
}
```

CHAPTER 10

APPENDIX II

SCREENSHOTS

10.1 HOME PAGE

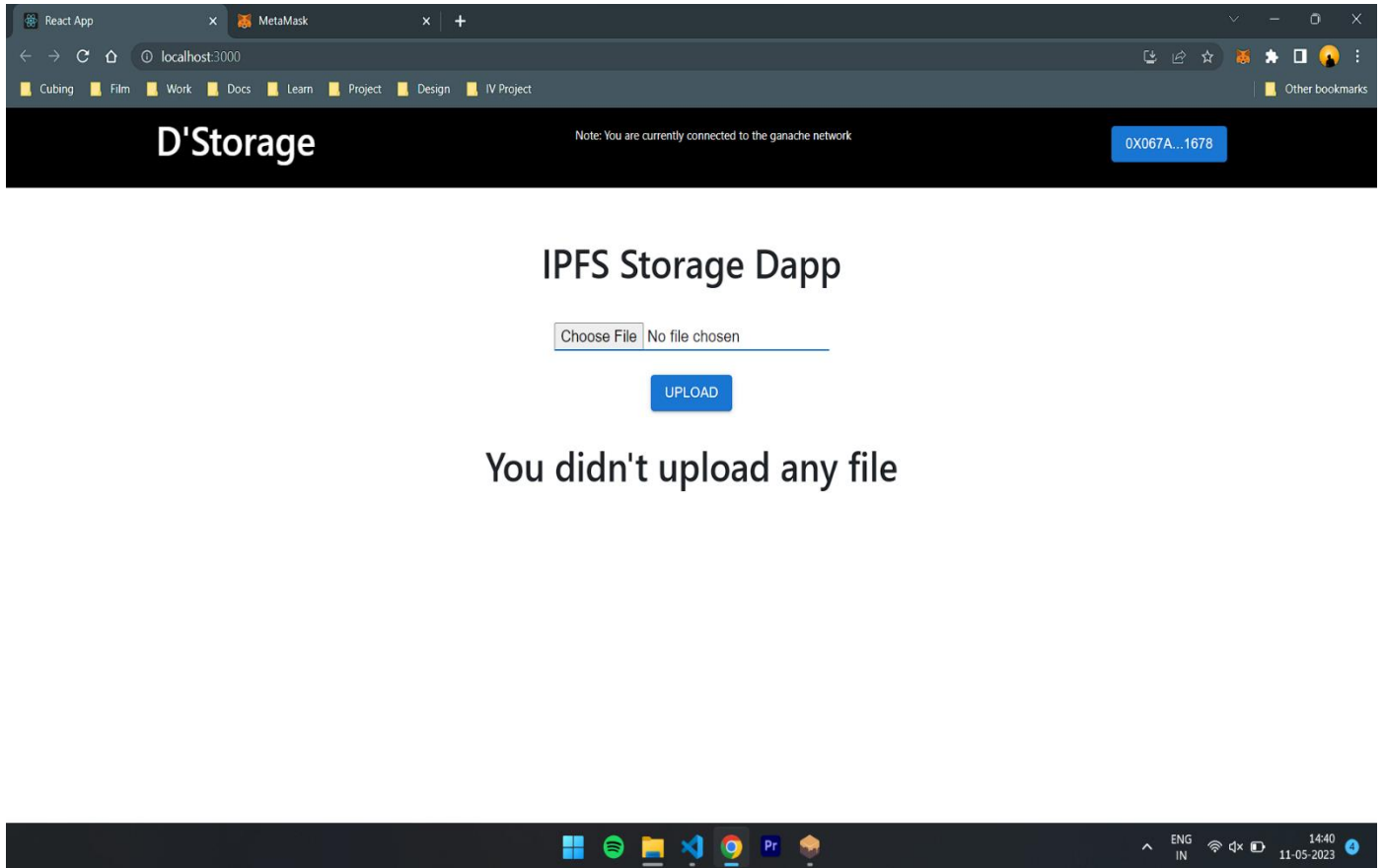


Fig 10.1 Home Page

10.2 File Upload

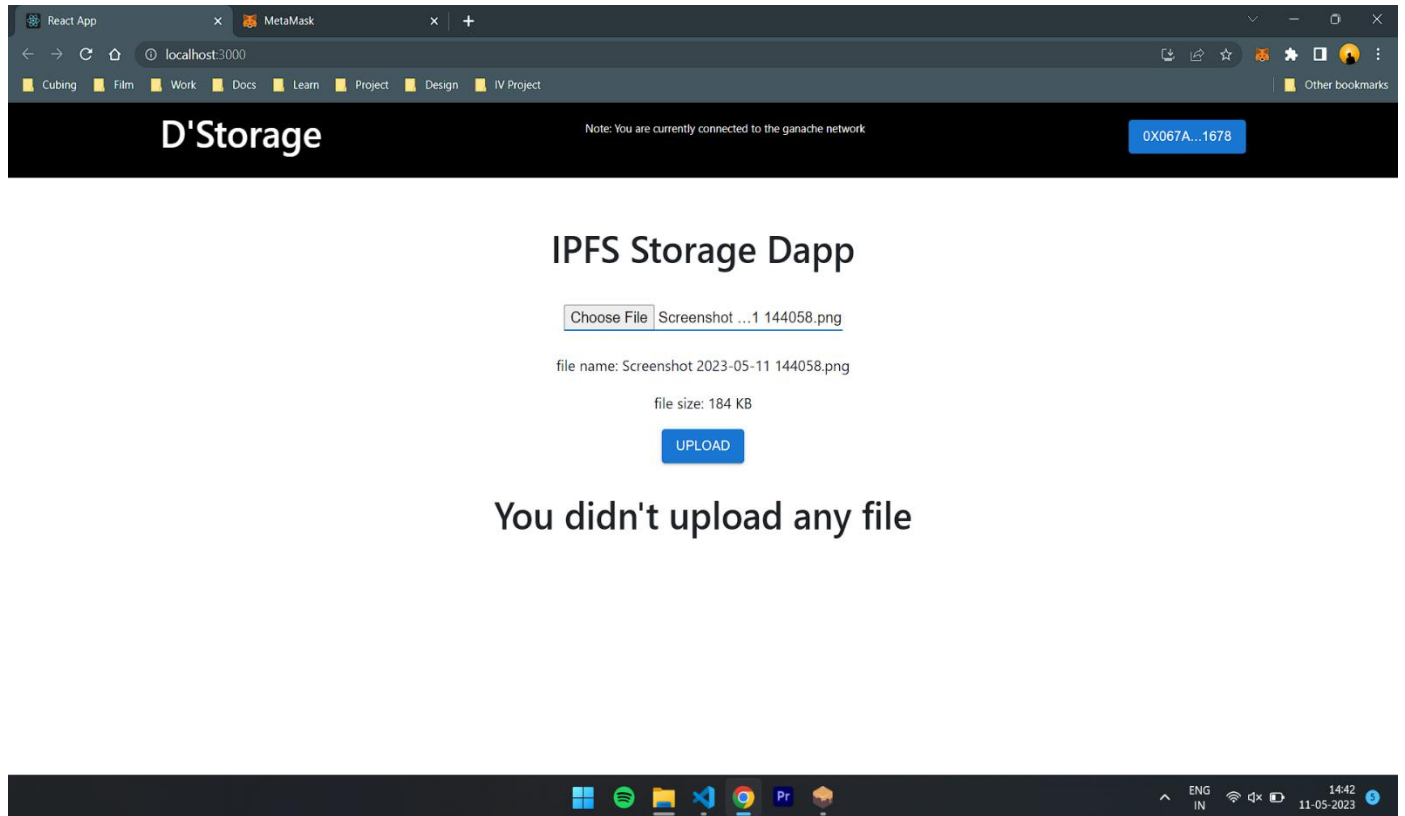


Fig 10.2 File Upload

10.3 Metamask Transaction

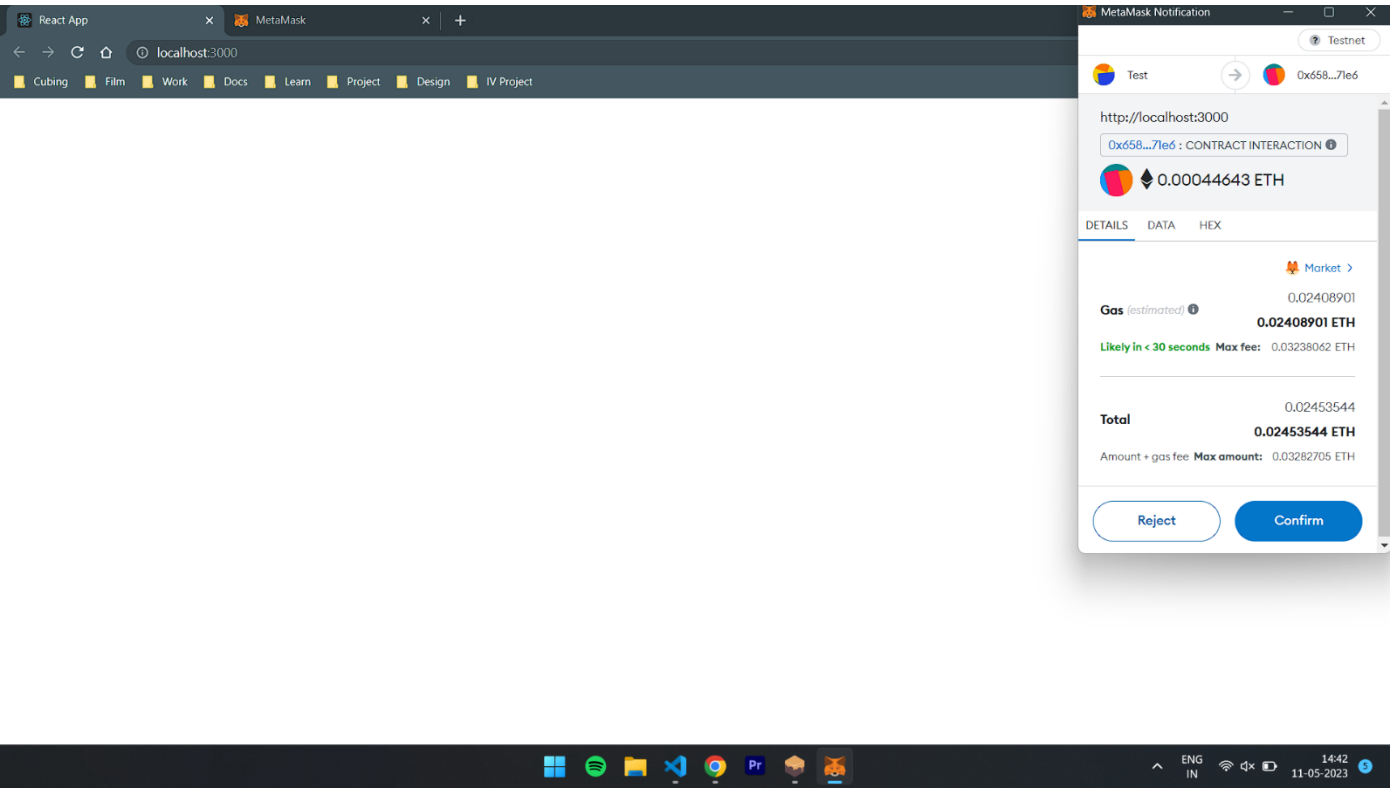


Fig 10.3 Metamask Transaction

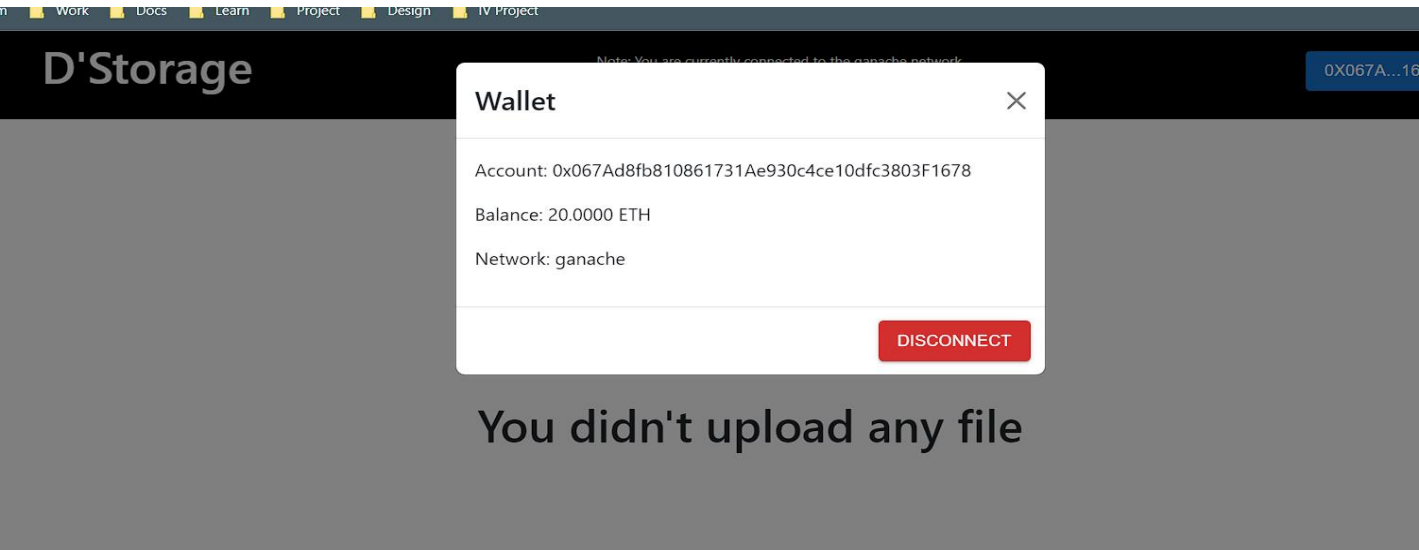


Fig 10.3.1 Wallet

10.4 VIEW the Uploaded file

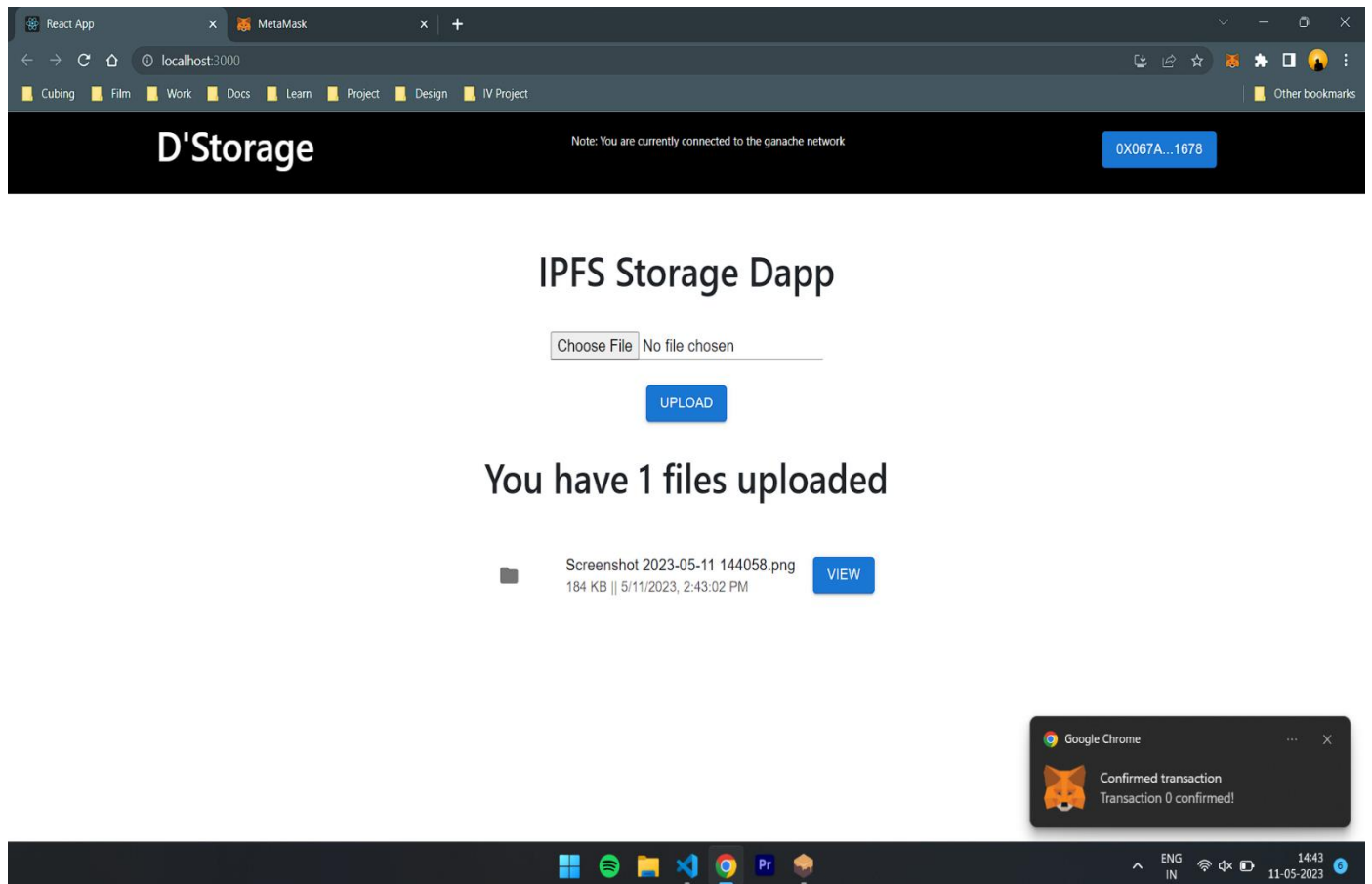


Fig 10.4 View the File Uploaded

CHAPTER 11
APPENDIX 3
REFERENCES

- [1] C. Kelly, N. Pitropakis, A. Mylonas, S. McKeown, and W. J. Buchanan, “A comparative analysis of honeypots on different cloud platforms,” *Sensors*, vol. 21, no. 7, p. 2433, 2021.
- [2] P. R. Carnley and H. Kettani, “Identity and access management for the internet of things,” *International Journal of Future Computer and Communication*, vol. 8, no. 4, pp. 129–133, 2019.
- [3] R. Kuhn, D. Yaga, and J. Voas, “Rethinking distributed ledger technology,” *Computer*, vol. 52, no. 2, pp. 68–72, 2019.
- [4] J. Sousa, A. Bessani, and M. Vukolic, “A byzantine fault tolerant or-deriving service for the hyper ledger fabric blockchain platform,” in *Proceedings of the 2018 48th annual ieee/ifip international conference on dependable systems and networks (dsn)*, pp. 51–58, Luxembourg, Europe, July 2018.
- [5] J. A. Ramirez and E. Rodriguez, “A singular value decomposition approach for testing the efficiency of Bitcoin and Ethereum markets,” *Economics Letters*, vol. 206, Article ID 109997, 2021.
- [6] S. H and D. R. Prasath, “Bi-fitness Swarm optimizer: blockchain assisted secure Swarm intelligence routing Protocol for MANET,” *Indian Journal of Computer Science and Engineering*, vol. 12, no. 5, pp. 1442–1458, 2021.

CONFERENCE CERTIFICATES

NSCET/2022-23/ICNSCET - 143



Theni Melapettai Hindu Nadargal Uravinmurai
NADAR SARASWATHI COLLEGE OF
ENGINEERING & TECHNOLOGY
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai
Annanji(po), Vadapudupatti, Theni - 625 531.

International Conference on
New Scientific Creations in Engineering and Technology
“ICNSCET 2K23”
CERTIFICATE OF APPRECIATION

This is to certify that Dr./Ms./Mr. **DINESH S** of
Nadar Saraswathi College of Engineering and Technology
has presented a paper titled **DECENTRALIZED DATA STORAGE USING BLOCKCHAIN**
in the International Conference on New Scientific Creations in Engineering and Technology –
ICNSCET 2K23 organized by Department of CSE, IT , AI&DS, Nadar Saraswathi College of
Engineering and Technology, Theni during 9th & 10th May 2023.


Mr.J.Mathalainaraj, M.E. (Ph.D.)
Convenor & Head Incharge /CSE.


Mr.L.S.Vignesh, M.E. (Ph.D.)
Convenor & Head Incharge / AI&DS.


Dr.M.Sathya, M.Tech, Ph.D.,
Convenor & Head Incharge /IT.


Dr.C. Mathalai Sundaram, M.E. M.B.A. Ph.D.,
Principal, NSCET.


Er.S.Naveen Ram, B.E. M.B.A.,
Joint Secretary, NSCET.


Mr.A.S.R.Maheswaran, B.Sc.,
Secretary, NSCET.


Mr.A.Rajkumar, B.B.A.,
Secretary, NSCET.



Theni Melapettai Hindu Nadargal Uravinmurai
**NADAR SARASWATHI COLLEGE OF
 ENGINEERING & TECHNOLOGY**

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai
 Annanji(po), Vadapudupatti, Theni - 625 531.

International Conference on
 New Scientific Creations in Engineering and Technology

"ICNSCET 2K23"

CERTIFICATE OF APPRECIATION

This is to certify that Dr./Ms./Mr. **GOPALA KRISHNANK** of
Nadar Saraswathi College of Engineering and Technology
 has presented a paper titled **DECENTRALIZED DATA STORAGE USING BLOCKCHAIN**
 in the International Conference on New Scientific Creations in Engineering and Technology –
 ICNSCET 2K23 organized by Department of CSE, IT , AI&DS, Nadar Saraswathi College of
 Engineering and Technology, Theni during 9th & 10th May 2023.



 Mr.J.Mathalairaj, M.E. (Ph.D.)
 Convenor & Head Incharge /CSE.


 Mr.L.S.Vignesh, M.E. (Ph.D.)
 Convenor & Head Incharge / AI&DS.


 Dr.M.Sathya, M.Tech. Ph.D.,
 Convenor & Head Incharge / IT.


 Dr.C. Mathalai Sundaram, M.E. MBA, Ph.D.
 Principal, NSCET.


 Er.S.Naveen Ram, B.E. MBA,
 Joint Secretary, NSCET.


 Mr.A.S.R.Maheswaran, B.Sc.,
 Secretary, NSCET.


 Mr.A.Rajkumar, BBA,
 Secretary, NSCET.



Theni Melapettai Hindu Nadargal Uravinmurai
**NADAR SARASWATHI COLLEGE OF
 ENGINEERING & TECHNOLOGY**

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai
 Annanji(po), Vadapudupatti, Theni - 625 531.

International Conference on
 New Scientific Creations in Engineering and Technology

“ICNSCET 2K23”

CERTIFICATE OF APPRECIATION

This is to certify that Dr./Ms./Mr. **HARI PRASAD J** of
Nadar Saraswathi College of Engineering and Technology
 has presented a paper titled **DECENTRALIZED DATA STORAGE USING BLOCKCHAIN**
 in the International Conference on New Scientific Creations in Engineering and Technology –
 ICNSCET 2K23 organized by Department of CSE, IT , AI&DS, Nadar Saraswathi College of
 Engineering and Technology, Theni during 9th & 10th May 2023.



 Mr.J.Mathalairaj, M.E. (Ph.D.)
 Convenor & Head Incharge /CSE.


 Mr.L.S.Vignesh, M.E. (Ph.D.)
 Convenor & Head Incharge / AI&DS.


 Dr.M.Sathya, M.Tech. Ph.D.,
 Convenor & Head Incharge / IT.


 Dr.C. Mathalai Sundaram, M.E. MBA, Ph.D.
 Principal, NSCET.


 Er.S.Naveen Ram, B.E. MBA,
 Joint Secretary, NSCET.


 Mr.A.S.R.Maheswaran, B.Sc.,
 Secretary, NSCET.


 Mr.A.Rajkumar, BBA,
 Secretary, NSCET.



Theni Melapettai Hindu Nadargal Uravinmurai
**NADAR SARASWATHI COLLEGE OF
 ENGINEERING & TECHNOLOGY**

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai
 Annanji(po), Vadapudupatti, Theni - 625 531.

International Conference on
 New Scientific Creations in Engineering and Technology

"ICNSCET 2K23"

CERTIFICATE OF APPRECIATION

This is to certify that Dr./Ms./Mr. **SANTHOSH SIVAN S.S** of
Nadar Saraswathi College of Engineering and Technology
 has presented a paper titled **DECENTRALIZED DATA STORAGE USING BLOCKCHAIN**
 in the International Conference on New Scientific Creations in Engineering and Technology –
 ICNSCET 2K23 organized by Department of CSE, IT , AI&DS, Nadar Saraswathi College of
 Engineering and Technology, Theni during 9th & 10th May 2023.



 Mr.J.Mathalairaj, M.E. (Ph.D.)
 Convenor & Head Incharge /CSE.


 Mr.L.S.Vignesh, M.E. (Ph.D.)
 Convenor & Head Incharge / AI&DS.


 Dr.M.Sathya, M.Tech. Ph.D.,
 Convenor & Head Incharge / IT.


 Dr.C. Mathalai Sundaram, M.E. MBA, Ph.D.
 Principal, NSCET.


 Er.S.Naveen Ram, B.E. MBA,
 Joint Secretary, NSCET.


 Mr.A.S.R.Maheswaran, B.Sc.,
 Secretary, NSCET.


 Mr.A.Rajkumar, BBA,
 Secretary, NSCET.