DECENTRALIZED FILE STORAGE: EMPOWERING SECURITY AND TRANSPARENCY THROUGH INTER PLANETARY FILE SYSTEM AND BLOCKCHAIN

A PROJECT REPORT

Submitted by

GINNELA JAYADEEP [Reg No:RA2011003010397]

Y. BHARATH SIMHA[RegNo:RA2011003010407]

Under the Guidance of

Dr. C. VIJAYAKUMARAN

Associate Professor, Department of Computing Technologies

in partial fulfilment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTING TECHNOLOGIES

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR– 603 203

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR–603 203

BONAFIDE CERTIFICATE

Certified that 18CSP109L project report titled "**Decentralized File Storage: Empowering Security and Transparency Through Inter Planetary File System and Blockchain**" is the bonafide work of GINNELA JAYADEEP [RA2011003010397] and Y. BHARATH SIMHA [RA2011003010407] who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

Dr. C. VIJAYAKUMARAN
SUPERVISOR
Associate Professor
Department of Computing Technologies

Dr. C. VIJAYAKUMARAN
PANEL HEAD
Associate Professor
Department of Computing Technologies

Prof. Dr. M. PUSHPALATHA
HEAD OF THE DEPARTMENT
Department of Computing Technologies

INTERNAL EXAMINER

EXTERNAL EXAMINER

Department of Computing Technologies

SRM Institute of Science and Technology

Own Work Declaration Form

Degree/Course	**:** B. Tech in Computer Science and Engineering
Student Names	**:** GINNELA JAYADEEP, Y BHARATH SIMHA
Registration Number	**:** RA2011003010397, RA2011003010407
Title of Work	**:** Decentralized File Storage: Empowering Security and Transparency Through Inter Planetary File System and Blockchain

We here by certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is our own except where indicated, and that we have met the following conditions:

- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web,etc.)
- Given the sources of all pictures, data etc that are not my own.
- Not made any use of the report(s) or essay(s) of any other student(s)either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course hand book / University website

We understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:

I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.

Student 1 Signature:

Student 2 Signature:

Date:

ACKNOWLEDGEMENT

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr. T. V. Gopal**, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor and Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the Department, **Dr. M. Pushpalatha,** Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Panel Members, **Dr. C. Vijayakumaran,** Panel Head**, Dr. B. Amutha,** Assistant Professor, **Dr. Ramkumar Jayaraman,** Assistant Professor and **Dr. R. Jeya,** Assistant Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor, **Dr. S. Jagadeesan**, Assistant Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Dr. C. Vijayakumaran**, Associate Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under his mentorship. He provided us with the freedom and support to explore the research topics of our interest. His passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank all the staff and students of Computing Technologies Department, School of Computing, S.R.M Institute of Science and Technology, for their help during our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support and encouragement.

GINNELA JAYADEEP [Reg. No: RA2011003010397]

Y. BHARATH SIMHA [Reg. No: RA2011003010407]

ABSTRACT

The large part of assessment literature suggests that lot of people face storage issues. A decentralized file storage platform could offer a myriad of benefits, including increased security, privacy, efficiency, and accessibility. By leveraging the decentralized nature of blockchain, this platform could eliminate the need for a centralized authority, reduce costs, and provide a more equitable and transparent environment for storage and computing services. A decentralized file storage system using blockchain and IPFS has the potential to revolutionize the file storage industry. The proposed system leverages the security and transparency features of blockchain technology to provide a decentralized platform for file storage. The proposed system's benefits include increased transparency, improved security, reduced transaction costs, and a seamless rental experience for both renters and providers. Algorithms involved in building this system are SHA 256 and IPFS gives a unique address derived from a hash of the file’s content, so hash algorithm also plays a vital role in the system. Some other algorithms used in payment gateway are Data Encryption Standard (DES), Advanced Encryption Standard (AES) which are used for the encrypting the details securely. All these algorithms are symmetric key algorithms. The main aim of this application is to provide a new avenue for individuals to monetize their unused hard disk space while maintaining control and ownership over their data. It caters not only to people who are tech-savvy and seasoned with the concept of Blockchain and IPFS but also users that want a hasslefree website to make money.

Table of Contents

List of Figures

List of Tables

List of Symbols and Abbreviations

IPFS - Inter Planetary File System

BaaS - Blockchain as a Service

IOT - Internet of Things

NPM - Node Package Manager

C Id - Content Identifier

CHAPTER 1
INTRODUCTION

1.1Purpose

The objective of this initiative is to establish a decentralized platform where individuals can trade their surplus hard disk space directly, eliminating the need for third-party mediators. This platform seeks to address the contemporary challenge of storage scarcity amidst the escalating reliance on technology. Through the establishment of a dependable and safeguarded marketplace, individuals can generate income from their unused storage, while customers can procure storage without relying on conventional cloud storage providers. Additionally, blockchain technology is integrated into the project to ensure the security of transactions and user information. Ultimately, the aim is to offer an effective and economical resolution to the issue of limited storage capacity.

1.2 Scope

1. Establishing a decentralized platform facilitating the trade of unused hard disk space among individuals.

2. Employing a blockchain-powered mechanism to validate the credibility of sellers.

3. Utilizing cryptographic protocols to encode user data, thereby guaranteeing its protection.

4. Integrating a payment gateway that accepts Bitcoin and various other payment methods.

5. Crafting a user-friendly interface enabling smooth business transactions.

6. Developing a resilient backend infrastructure to manage user registrations, data storage, and transactions.

7. Offering a comprehensive framework for user identity verification and authentication.

8. Ensuring the system's scalability to accommodate a growing user base and transaction volume.

1.3 Motivation

Third-party cloud services like AWS or Microsoft Azure offer advantages such as scalability, flexibility, and cost-effectiveness. However, they come with drawbacks that drive individuals to seek alternatives like a decentralized marketplace for trading unused hard disk space. One significant drawback is the loss of data control. When data is stored on third-party cloud services, users surrender control over its location and accessibility, raising security concerns regarding potential breaches. Vendor lock-in is another issue, making it challenging for users to switch providers due to migration complexities and costs, limiting adaptability. Additionally, while these services are affordable for large enterprises, they can be costly for smaller businesses or individuals. Hence, the development of alternatives like a decentralized marketplace addresses these concerns by providing data control, reducing lock-in, offering cost-effective solutions, and enhancing data accessibility.

1.4 Problem Statement

Conventional storage setups often rely on centralized structures, involving intermediaries and incurring high transaction expenses. Furthermore, persistent concerns regarding data privacy and security deter many users from sharing sensitive information online. Additionally, opacity in rental agreements and payment methods can lead to disputes and inefficiencies. In response to these challenges, a file storage system employing blockchain and cryptocurrency proposes a decentralized and secure platform. The objective is to offer users a transparent and streamlined storage solution while reducing transaction costs and eliminating the need for intermediaries. Thus, the focus lies in developing a decentralized and secure file storage platform that harnesses blockchain and cryptocurrency technologies to enhance efficiency, reliability, and accessibility.

CHAPTER 2

LITERATURE SURVEY

After reading a number of studies, a conclusion is made that there are a variety of ways to approach the modelling of a Decentralized file storage system using IPFS and Blockchain.

2.1 NutBaaS: A Blockchain-as-a-Service Platform

This paper proposes a Blockchain-as-a-Service (BaaS) platform [1]that enables users to deploy, manage, and use their own blockchain [2]networks. The NutBaaS platform is designed to provide a user-friendly, customizable, and scalable environment for building decentralized applications (dApps). The NutBaaS platform is built using open-source technologies and is designed to support multiple blockchain frameworks such as Ethereum, Fabric, and Corda.

2.1.1. Features:

This paper introduces a Blockchain-as-Service (BaaS) platform named NutBaaS, aiming to facilitate users in deploying, overseeing, and utilizing their individual blockchain networks. NutBaaS is tailored to offer a user friendly, adaptable, and expandable setting for constructing decentralized applications (dApps). It's constructed using open-source tools and is intended to accommodate various blockchain frameworks like Ethereum, Fabric, and Corda.

The three main features of this paper include:

- **Architecture design**: The document suggests a design for an architecture aimed at a blockchain-as-a-service (BaaS) platform, which utilizes both cloud computing and blockchain technology. This architecture is intended to furnish developers with the necessary infrastructure for the creation and deployment of applications based on blockchain. It comprises multiple layers, namely a blockchain layer, a service layer, and a cloud infrastructure layer.

- **Smart contract management:** Additionally, the paper suggests a system for managing smart contracts, allowing developers to generate, deploy, and oversee smart contracts on the BaaS platform. This system encompasses a library of smart contract templates, a testing environment for smart contracts, and a tool for deploying and managing smart contracts.

- **Security and privacy:**

The document underscores the significance of security and privacy within blockchain-driven applications and suggests multiple strategies to safeguard the security and privacy of data stored and handled on the BaaS platform.

2.1.2 Limitations

Here are some potential limitations of the proposed platform and future work:

Dependency on Third-Party Providers: Despite efforts to enhance reliability and security, the platform still relies on third-party service providers for its infrastructure. This dependency introduces a level of vulnerability as the platform's performance and security are contingent upon the reliability and integrity of these providers. Future work could explore strategies to minimize this dependency or mitigate its potential risks.

Complexity of Implementation: Introducing advanced technical services such as Identity-Chain technology and smart contract security vulnerability detection may increase the complexity of the platform's implementation. This complexity could pose challenges for developers and users in understanding and effectively utilizing these features. Future work should consider simplifying the integration process and providing comprehensive documentation and support to facilitate adoption.

Scalability Concerns: While the proposed platform addresses reliability and security concerns, scalability remains a potential limitation. As the platform grows and accommodates more users and transactions, it may face challenges in maintaining performance and responsiveness. Future work should focus on optimizing the platform's architecture and protocols to ensure scalability without compromising on reliability and security.

Regulatory and Compliance Issues: The paper does not address potential regulatory or compliance challenges associated with the deployment and operation of the platform. Depending on the jurisdiction, there may be legal requirements or industry standards that the platform must adhere to, which could impact its design and functionality. Future work should involve collaboration with legal experts to ensure compliance with relevant regulations and standards.

Limited Adoption and Ecosystem Support: Despite the proposed improvements, the success of the platform ultimately depends on its adoption by developers and users. Without sufficient ecosystem support and community engagement, the platform may struggle to gain traction in the market. Future work should include strategies for promoting the platform, fostering developer communities, and incentivizing participation to drive adoption.

These limitations provide areas for further research and development to enhance the proposed platform and address potential challenges in its implementation and adoption.

2.2 Blockchain and IoT Integration

This paper offers insights into the utilization of blockchain technology and smart contracts within Internet of Things (IoT) applications [2]. It explores how blockchain can effectively tackle key hurdles encountered by the IoT, including security, privacy, and interoperability issues. Furthermore, the paper explores the potential of smart contracts to streamline and automate diverse processes within IoT applications.

2.2.1. Features

- Introduction of a blockchain-centered method for ensuring secure communication and sharing of data within IoT networks.
- Incorporation of smart contracts to streamline transaction execution and uphold regulations.
- Advocacy for a decentralized framework to deliver scalability, resilience, and confidentiality in IoT networks.

2.2.2 Limitations

While the integration of blockchains and IoT offers promising benefits, several limitations persist. These include scalability issues due to the large volume of IoT devices, potential security vulnerabilities in IoT device communication, and the challenge of reconciling real-time IoT data with blockchain transaction speeds. Additionally, concerns regarding data privacy and regulatory compliance may hinder widespread adoption. Moreover, the complexity of implementing and managing blockchain-IoT systems could pose barriers for smaller organizations. Addressing these limitations will be crucial for realizing the full potential of blockchain-enabled IoT solutions across industries.

2.3 Decentralized Storage Framework

This paper suggests a framework based on blockchain for sharing data in decentralized storage systems, incorporating fine-grained access control. The primary goal is to establish a secure and effective platform for data sharing, empowering data owners to manage access to their data through detailed access policies [3]. The proposed framework employs smart contracts for overseeing data access and storage, alongside a decentralized file system to securely store data.

2.3.1. Features

• **Fine-grained access control**: The framework allows data owners to establish access policies for their data with precise control over who can access, alter, or remove the data.

• **Decentralized storage system**: The framework utilizes the benefits of decentralized storage systems, including enhanced availability, resilience against faults, and scalability, to deliver a sturdy and dependable platform for sharing data.

• **Flexibility:** The framework offers flexibility regarding the variety of data permissible for storage and sharing, as well as the range of access control policies applicable to the data. This flexibility facilitates customization and adjustment to various use cases and situations.

2.3.2 Limitations

Despite advancements, the proposed framework still has limitations. Firstly, it lacks mechanisms for user attribute revocation and access policy updates, which are crucial for dynamic access control. Secondly, while decentralized storage mitigates single points of failure, it introduces challenges regarding data availability and reliability, especially in terms of ensuring consistent search results. Additionally, integrating IPFS and Ethereum blockchain may raise concerns about scalability and transaction costs, particularly as the system scales. Moreover, real-world implementation may face interoperability issues with existing infrastructure and regulatory challenges. Addressing these limitations will be essential for enhancing the effectiveness and practicality of the proposed framework in decentralized storage systems.

2.4 User Collusion Avoidance CP-ABE for Cloud Storage

User Collusion Avoidance CP-ABE (ciphertext-policy attribute-based encryption) for Cloud Storage introduces a novel method for cloud storage management. This scheme incorporates a hierarchical attribute framework and a revocation system that selectively revokes compromised or unnecessary attributes, rather than revoking all attributes linked to a user. This strategy enhances attribute revocation efficiency and minimizes the risk of user collusion, thereby enhancing the confidentiality and integrity of data stored in the cloud.

2.4.1. Features

- An achieved result is the introduction of an effective attribute revocation system, ensuring the revocation of user attributes while preserving the system's security and efficiency [4]. This system facilitates dynamic alterations to access control policies and prevents unauthorized data access by revoked users.
- Another achieved outcome of this study is the proposition of a user collision prevention mechanism utilizing efficient attribute-based encryption featuring ciphertexts of constant size. This mechanism delivers secure and effective data access control within cloud storage settings.

2.4.2 Limitations

While the proposed User Collusion Avoidance CP-ABE scheme presents several advancements in cloud storage management, it also exhibits limitations:

Complexity and Implementation Challenges: Implementing a hierarchical attribute framework and a selective attribute revocation system may introduce complexity, potentially increasing the difficulty of system deployment and management.

Scalability Concerns: The effectiveness of the scheme in larger-scale deployments remains uncertain. As the number of users and attributes grows, the overhead of attribute management and revocation may become significant, potentially impacting system scalability and performance.

Interoperability Issues: Integration with existing cloud storage infrastructures and systems may pose challenges. Compatibility issues could arise when attempting to implement the proposed scheme alongside other access control mechanisms or in heterogeneous cloud environments.

Security and Trust Assumptions: The scheme relies on assumptions of trustworthiness regarding attribute revocation and enforcement mechanisms. If these mechanisms are compromised or improperly implemented, the security of the system could be compromised.

Potential Overhead: While the scheme aims to achieve efficient attribute-based encryption with ciphertexts of constant size, there may still be overhead associated with encryption, decryption, and attribute management operations, impacting system performance and resource utilization.

Addressing these limitations will be crucial to realizing the full potential of the proposed scheme in enhancing cloud storage security and access control mechanisms.

Table 2.1 Literature Survey Summary

Paper Title	Main Contribution	Key Features/Outcomes	Limitations
Blockchain-As-A-Service Platform Enhancement Authors: Shangping,Yinglong Zhang June 2018	Improved BaaS platform reliability and security	Introduction of Identity-Chain tech, smart contract vulnerability detection	Dependency on third-party providers, complexity of implementation, scalability concerns, regulatory compliance issues, limited adoption and ecosystem support
Blockchain and IoT Integration Authors: Jinglin Zou, Debiao He October 2021	Potential transformations in various industries	Automation of workflows, cryptographic verifiability, cost and time savings	Scalability issues, security vulnerabilities, regulatory compliance challenges, limited adoption and ecosystem support
Decentralized Storage Framework Authors: S.Uthayashangar, T.Dhanya July 2021	Combination of IPFS, Ethereum, and ABE for storage	Elimination of trusted PKG, fine-grained access control, searchable encryption	Lack of attribute revocation and policy update functions, challenges in ensuring

			consistent search results, potential scalability and interoperability issues, regulatory and trust concerns
User Collusion Avoidance CP-ABE for Cloud Storage Baodong Qin Qinglan Zhao July 2019	Novel attribute revocation system and user collusion prevention	Selective attribute revocation, constant-size ciphertexts, dynamic access control	Complexity in implementation, scalability concerns, interoperability issues, security and trust assumptions, potential overhead

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

3.1 Architecture

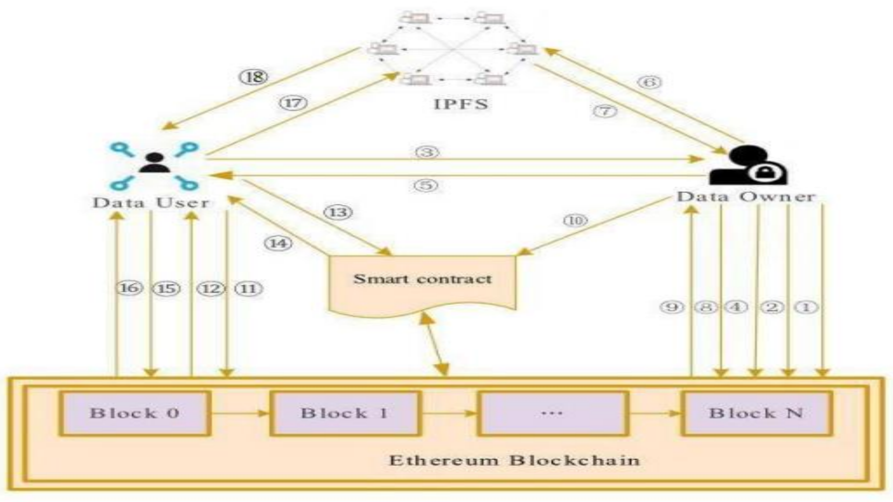


Fig 3.1 Architecture for Decentralized Storage System

In a decentralized storage system, several components collaborate to facilitate the storage and retrieval of data in a distributed manner.

IPFS (InterPlanetary File System):

IPFS serves as the foundational layer for decentralized storage. It operates as a protocol and network that enables a peer-to-peer method of storing and sharing hypermedia. IPFS allows data to be addressed by its content rather than its location, ensuring decentralization and redundancy.

Data User:

The Data User refers to individuals or applications interacting with the decentralized storage system. These users seek to retrieve or store data within the network. They can be anyone requiring access to the stored information.

Data Owner:

Data Owners are entities responsible for uploading data onto the decentralized storage system. They possess control over the access rights and permissions associated with the data they upload. This control ensures data integrity and security within the system.

Smart Contract:

Smart contracts are self-executing contracts encoded with predefined terms and conditions. Within the context of the decentralized storage system, smart contracts play a crucial role in automating processes related to data storage and access. For instance, they can define access permissions, manage payments for storage services, and enforce rules governing data usage.

Ethereum Blockchain:

The Ethereum blockchain serves as the underlying infrastructure for executing smart contracts. It acts as a decentralized platform where transactions and interactions with smart contracts are recorded immutably. In the decentralized storage system, Ethereum is utilized to maintain records of data ownership, access permissions, and transactions related to storage services.

Integration and Workflow:

Data Upload:

A Data Owner uploads data onto the IPFS network, ensuring decentralization and redundancy.

Upon upload, a smart contract deployed on the Ethereum blockchain is triggered. This smart contract records essential details such as data ownership, access permissions, and associated storage fees.

Data Retrieval:

When a Data User requests access to specific data stored on IPFS, the system checks their access permissions against the relevant smart contract on the Ethereum blockchain.

If access is authorized, the requested data is retrieved from IPFS and provided to the Data User, ensuring seamless access to stored information**.**

Payments and Transactions:

Smart contracts deployed on the Ethereum blockchain manage transactions related to data storage and access within the system.

These transactions may involve payments from Data Users to Data Owners for accessing data or payments from Data Owners to storage providers for hosting data, all executed transparently and securely on the Ethereum network.

By integrating IPFS for decentralized storage, Ethereum for smart contract execution and transaction management, and providing user-friendly interfaces for interaction, the decentralized storage system offers a robust, secure, and efficient solution for storing and accessing data in a distributed manner.

3.2 Use Case Diagram

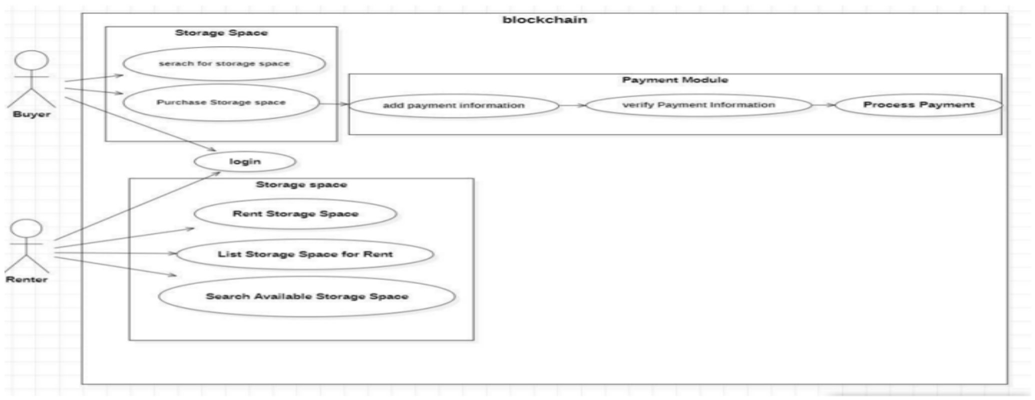


Fig 3.2 Use Case Diagram

Buyer:

The Buyer represents individuals or entities seeking to either purchase or rent storage space within the system. They are responsible for searching for available storage spaces, initiating transactions, and managing their account credentials through the login mechanism.

Renter:

Similar to the Buyer, the Renter represents users interested in renting storage space. They share similar responsibilities to the Buyer, including searching for available spaces, initiating rental transactions, and managing their account authentication through the login process.

Storage Space:

This component represents the inventory of available storage spaces within the system. Its responsibilities include listing available spaces for rent, providing detailed information about each space, and facilitating transactions between Buyers/Renters and space owners.

Login:

The Login component serves as the authentication mechanism, ensuring secure access to user accounts and system functionalities. It verifies user credentials and grants access to authorized users, maintaining the integrity and security of the system.

Blockchain (Payment Module):

The Blockchain Payment Module integrates blockchain technology into the system to manage payment-related functionalities. Its responsibilities include securely handling payment transactions, adding and verifying payment information, and processing payments using blockchain technology. This ensures transparency, security, and immutability in all payment transactions within the system.

This architecture enables seamless interaction between Buyers/Renters, storage space providers, and the payment system, facilitating a smooth and secure experience for all stakeholders involved in the storage space marketplace.

3.3. Sequence Diagram

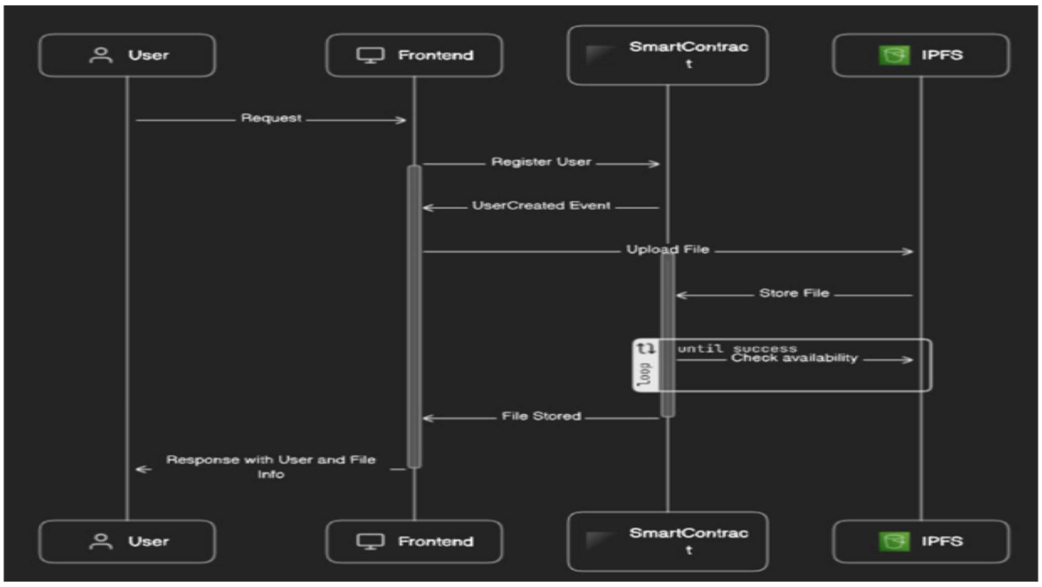


Fig 3.3 Sequence Diagram

Sequence Diagram:

User Interaction:

The sequence initiates when a user engages with the frontend interface, accessing functionalities like browsing the storage space marketplace or executing a transaction.

Frontend Request:

Following the user's action, the frontend dispatches a request message to the smart contract deployed on the Ethereum blockchain. This request encapsulates details of the user's activity, such as searching for available storage spaces or initiating a transaction.

Smart Contract Processing:

Upon receipt of the request, the smart contract undertakes processing tasks. These include validating user permissions, interacting with data stored on IPFS if necessary, or executing predefined algorithms linked to the requested action.

Interaction with IPFS:

If the user's request involves data manipulation on IPFS, the smart contract interacts with the IPFS network. Tasks may include data retrieval, uploading new information, or updating existing content.

Response Generation:

Following request processing and IPFS interaction, the smart contract generates a response message. This message typically communicates the outcome of the user's request, such as confirming a successful transaction or furnishing a list of available storage spaces.

Frontend Update:

Upon receipt of the response from the smart contract, the frontend interface undergoes updating. This involves adjusting its display elements to reflect pertinent information, such as transaction status updates or presenting search results for available storage spaces.

User Interaction Continues:

The sequence persists as the user continues interacting with the frontend interface. Subsequent actions might include initiating additional transactions or navigating through different sections of the storage space marketplace.

This sequence delineates the orchestrated flow of interactions among the user, frontend interface, smart contract, and IPFS network, ensuring a cohesive and efficient user experience within the decentralized storage space marketplace.

3.4 Packaging Diagram

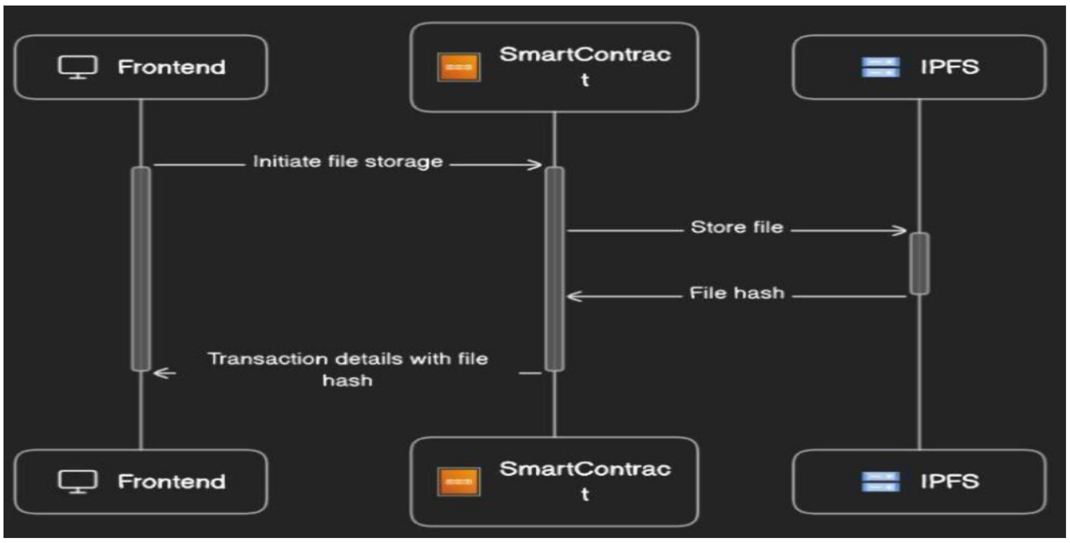


Fig 3.4 Packaging Diagram

Frontend: The user interface (UI) component of the application, allowing users to interact with the system. It displays information, accepts user inputs, and communicates with the backend.

Smart Contracts: Self-executing contracts with the terms of the agreement directly written into code. They automate transactions and enforce rules on the blockchain.

IPFS: InterPlanetary File System, a decentralized protocol for storing and sharing data in a peer-to-peer network. It provides a distributed method for storing and accessing files, enhancing data resilience and availability.

Transactions: Actions or operations performed on the blockchain, such as transferring tokens, executing smart contracts, or recording data. They are validated, confirmed, and permanently recorded on the blockchain ledger.

3.5. Deployment Diagram

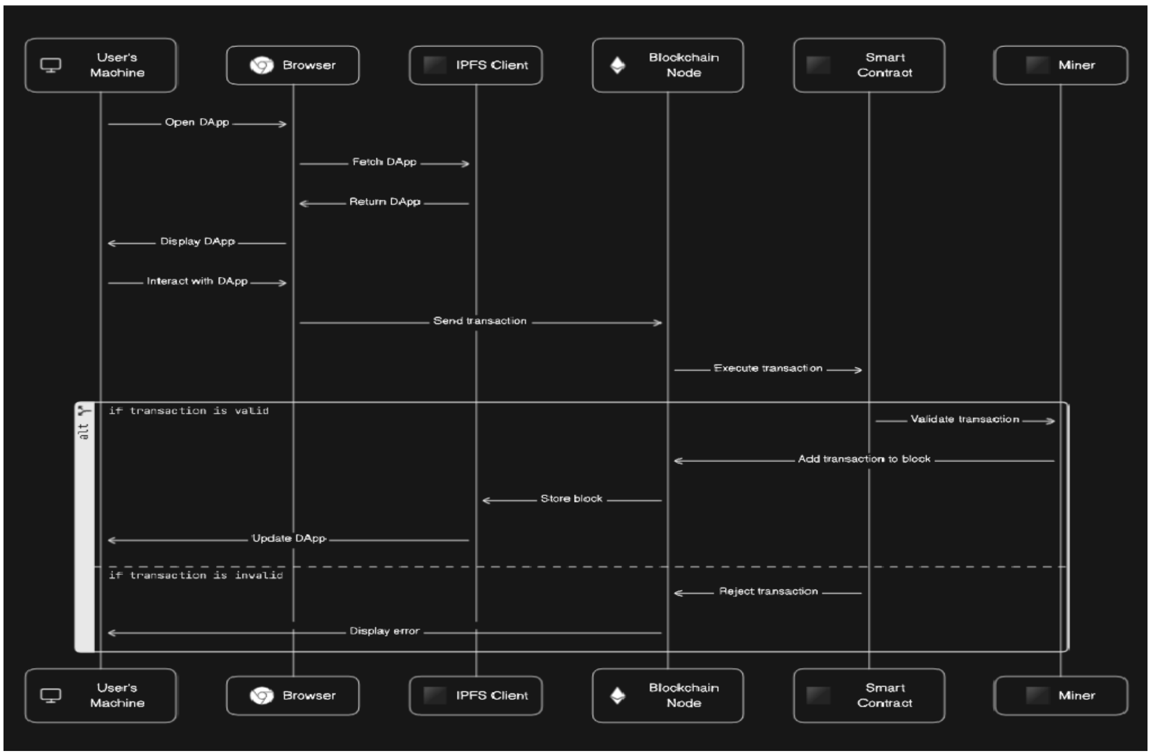


Fig 3.5 Deployment Diagram

Deployment Diagram:

User's Machine:

This component symbolizes the device, whether it's a computer or a mobile device, utilized by users to access the decentralized storage space marketplace. It serves as the platform where users interact with the system, hosting their operating system and preferred browser for accessing the marketplace interface.

Browser:

The browser acts as the gateway for users to navigate and interact with the decentralized storage space marketplace. Installed on the user's machine, the browser facilitates access to the frontend interface of the marketplace. Users utilize the browser to browse storage space listings, initiate transactions, and engage with various features provided by the platform.

IPFS Client:

An essential software component installed on the user's machine, the IPFS client enables seamless interaction with the InterPlanetary File System (IPFS) network. Through the IPFS client, users can upload, download, and manage files stored on IPFS. This functionality allows users to access and retrieve data associated with storage spaces within the marketplace.

Blockchain Node:

Representing a node within the Ethereum blockchain network, this component plays a crucial role in facilitating interactions between users and the blockchain. Users may opt to run blockchain node software on their machines, providing them with access to the Ethereum blockchain. This enables users to interact with smart contracts deployed on the blockchain, ensuring the execution of transactions related to storage space rentals and purchases.

Smart Contract:

Deployed on the Ethereum blockchain, smart contracts serve as the backbone of the decentralized storage space marketplace. They contain predefined rules and logic governing various transactions and interactions within the platform. Users engage with smart contracts through transactions, such as renting or purchasing storage spaces, ensuring transparency, security, and automation within the marketplace.

Miner:

Miners form an integral part of the blockchain network, responsible for validating and adding new blocks of transactions to the blockchain through the mining process. By contributing computational resources, miners help secure the network and validate transactions, ensuring the integrity and reliability of the decentralized storage space marketplace.

This deployment configuration ensures the seamless operation of the decentralized storage space marketplace, enabling users to access and utilize the platform securely and efficiently while leveraging the benefits of blockchain technology and IPFS for data management.

CHAPTER 4

METHODOLOGY

4.1 Product Perspective

4.1.1 Emergence of Decentralized File Storage System

The concept of a decentralized file storage system employing IPFS and blockchain has emerged in response to the escalating demand for digital storage space and the rapid expansion of blockchain technologies. As businesses and individuals increasingly rely on digital data for various purposes, ranging from personal files to enterprise-level data storage, there has been a corresponding rise in the need for secure, reliable, and scalable storage solutions. However, despite the proliferation of storage options, a significant portion of available storage capacity remains underutilized or inaccessible to potential users.

4.1.2 Growing Demand for Digital Storage

The growing reliance on digital data across diverse sectors has fueled the demand for digital storage solutions. Businesses, organizations, and individuals are generating vast amounts of data daily, encompassing everything from documents and media files to application data and sensor data. This proliferation of data necessitates robust storage infrastructure capable of accommodating large volumes of data while ensuring accessibility, security, and scalability. However, traditional centralized storage solutions often struggle to meet these requirements, leading to issues such as data silos, security vulnerabilities, and scalability limitations.

4.1.3 Addressing Storage Challenges

To address the challenges posed by traditional centralized storage solutions and the growing demand for digital storage, the concept of a decentralized file storage system has emerged. This system aims to leverage the collective storage resources of network participants to create a distributed, resilient, and cost-effective storage infrastructure. By harnessing the surplus storage capacity of individual users and organizations, the decentralized file storage system seeks to provide a scalable, efficient, and affordable solution for storing and accessing digital data.

4.1.4 Utilizing Surplus Storage Capacity

One of the key principles underlying the decentralized file storage system is the utilization of surplus storage capacity. Many individuals and organizations possess excess storage capacity that remains unused or underutilized for extended periods. By enabling these stakeholders to rent out their surplus storage space to others in need of additional storage, the decentralized file storage system creates a marketplace for storage resources, optimizing the utilization of available storage capacity and unlocking value for both providers and consumers.

4.1.5 Leveraging IPFS and Blockchain Technology

In developing the decentralized file storage system, IPFS and blockchain technology were identified as ideal solutions to support its functionality and infrastructure. IPFS, or the InterPlanetary File System, provides a decentralized protocol for storing and accessing files, utilizing a content-addressable peer-to-peer network to distribute and retrieve data efficiently. By leveraging IPFS, the decentralized file storage system can benefit from its decentralized architecture, data redundancy, and tamper-resistant properties, ensuring the security and integrity of stored data.

4.1.6 Ensuring Secure and Transparent Transactions

Blockchain technology plays a pivotal role in ensuring secure and transparent transactions within the decentralized file storage system. By employing blockchain technology, the system can facilitate secure transactions and compensation mechanisms for all involved parties, including storage providers and consumers. Smart contracts, programmable self-executing contracts deployed on the blockchain, can automate and enforce the terms of storage agreements, ensuring that storage providers are fairly compensated for their services and that consumers have access to reliable and affordable storage solutions.

In conclusion, the concept of a decentralized file storage system employing IPFS and blockchain technology represents a innovative solution to the challenges posed by traditional centralized storage systems. By leveraging surplus storage capacity, utilizing IPFS for decentralized data storage, and harnessing blockchain technology for secure and transparent transactions, the decentralized file storage system offers a scalable, efficient, and cost-effective solution for storing and accessing digital data. As businesses and individuals continue to generate and rely on vast amounts of digital data, the decentralized file storage system stands poised to revolutionize the storage industry, unlocking new opportunities for data storage, sharing, and monetization.

4.2 Product Features

The cloud space rental system, integrating both cloud computing and blockchain technology, encompasses a range of essential features and functionalities that facilitate the seamless renting and accessing of digital storage space. These features and functions are designed to provide users with a secure, efficient, and transparent storage solution. The key components of the product features can be categorized as follows:

4.2.1 Block chain Module:

User Registration:

Users are provided with the option to sign up on the blockchain to establish their accounts within the cloud space rental system. Each user is assigned distinctive identifiers, such as Ethereum addresses, which serve as unique identifiers within the blockchain network.

User Authentication:

The system implements secure login procedures using cryptographic techniques to authenticate users. A verification process is employed to ensure that only registered users can access their data, enhancing security and privacy for user accounts and stored data.

4.2.2 IPFS Module:

Decentralized File Storage:

User files are stored within the IPFS (InterPlanetary File System) network, a decentralized protocol for storing and sharing content in a peer-to-peer manner. Files undergo segmentation into blocks and are distributed across numerous nodes within the IPFS network to ensure redundancy and fault tolerance.

Content Addressing:

The system utilizes content-based addressing for file retrieval, utilizing the unique hashes generated for each file stored within the IPFS network. This ensures efficient and reliable retrieval of files based on their content address, eliminating the need for centralized servers or intermediaries.

4.2.3 Payment Module:

Crypto Transactions:

Users are provided with the capability to conduct transactions using cryptocurrency, such as Ether, within the cloud space rental system. Smart contracts are employed to manage payments for services or storage, ensuring transparency and immutability of transaction records on the blockchain.

Transaction Security:

The system guarantees secure and transparent transactions through the utilization of blockchain technology. By leveraging blockchain's inherent properties, such as decentralized consensus and cryptographic security, the system ensures the integrity and security of all transactional activities, including payments and data access.

4.2.4 Login/Signup Module:

User Registration:

New users can register with the cloud space rental system using a username and password or other authentication methods. The registration process collects essential user information and generates unique identifiers, such as Ethereum addresses, to establish user accounts within the blockchain network.

User Authentication:

Secure login procedures are implemented to authenticate users and grant access to their accounts. Users can securely log in using their registered credentials, which are verified through cryptographic techniques to prevent unauthorized access and protect user privacy.

Expanding on Product Features:

The integration of blockchain technology into the cloud space rental system brings several advantages, including enhanced security, transparency, and efficiency in data storage and transaction processing. By leveraging blockchain's decentralized architecture and cryptographic security mechanisms, the system provides users with a secure and reliable platform for renting and accessing digital storage space. Additionally, the utilization of

IPFS for decentralized file storage ensures high availability and fault tolerance, enabling users to store and retrieve their files efficiently from anywhere in the world.

User Registration and Authentication:

The user registration process on the blockchain provides users with a secure and tamper-proof method of creating and managing their accounts within the cloud space rental system. By registering on the blockchain, users receive unique identifiers, such as Ethereum addresses, which serve as their digital identities within the system. This eliminates the need for centralized authentication servers and minimizes the risk of unauthorized access or data breaches.

Decentralized File Storage:

The use of IPFS for decentralized file storage offers numerous benefits, including data redundancy, fault tolerance, and censorship resistance. Files stored within the IPFS network are segmented into blocks and distributed across multiple nodes, ensuring redundancy and resilience against data loss or network failures. Additionally, content-based addressing allows for efficient retrieval of files based on their unique hashes, eliminating the reliance on centralized servers or intermediaries for file storage and access.

Payment Module:

The integration of cryptocurrency transactions within the cloud space rental system provides users with a convenient and secure method of conducting transactions for services or storage. By leveraging smart contracts deployed on the blockchain, users can execute transactions autonomously, without the need for intermediaries or third-party payment processors. This ensures transparency and immutability of transaction records, as all transactional activities are recorded on the blockchain and can be verified by all network participants.

Overall, the cloud space rental system offers a comprehensive and innovative solution for storing and accessing digital storage space securely and efficiently. By integrating blockchain technology and IPFS, the system provides users with a decentralized, transparent, and resilient platform for managing their digital assets. With features such as user registration and authentication, decentralized file storage, and cryptocurrency transactions, the system delivers a robust and user-friendly experience for individuals and businesses seeking reliable storage solutions in the digital age.

4.3. Functional Requirements

4.3.1 User Registration:

Description: Users have the option to register on the platform by supplying a distinct username and a password for security.

Functionality: The smart contract logs user registration particulars and triggers an event signaling successful registration.

In the realm of our decentralized file storage system project, the user registration functionality serves as the gateway for individuals to access the platform securely. Through this feature, users are empowered to create unique accounts by providing essential credentials, including a distinct username and password. These credentials are pivotal for ensuring the security and integrity of user accounts within the system. Upon submission of registration details, the smart contract associated with our platform diligently records and verifies the provided information. Subsequently, upon successful validation, an event is triggered, signifying the completion of the registration process. This event-driven mechanism not only enhances the efficiency of user registration but also ensures that all registered users are seamlessly onboarded onto the platform.

4.3.2 File Upload and Storage:

Description: Users have the capability to upload files onto the decentralized file storage system.

Functionality: The files are stored within the IPFS network, with their content addresses documented on the blockchain to facilitate retrieval.

In the dynamic landscape of our project, file upload and storage functionality serve as the cornerstone for users to leverage the decentralized file storage system effectively. Through this feature, users are bestowed with the autonomy to upload a diverse array of files onto the platform, ranging from documents and images to videos and audio files. Once a file is uploaded, our system harnesses the capabilities of the InterPlanetary File System (IPFS) to securely store the file data across a decentralized network of nodes.

Simultaneously, the content address (CID) of each uploaded file is meticulously documented on the blockchain. This strategic integration between IPFS and blockchain technology ensures the immutable recording of file metadata, thereby facilitating seamless file retrieval for users across the platform.

4.3.3 File Retrieval:

Description: Users have the ability to retrieve files by supplying the content address (CID) of the intended file.

Functionality: The system employs IPFS to retrieve the file linked to the provided CID and delivers it to the user.

In the ecosystem of our decentralized file storage system, file retrieval functionality stands as a beacon of accessibility and convenience for users seeking to retrieve their stored files. Leveraging this feature, users possess the capability to effortlessly retrieve files by supplying the content address (CID) of their desired file. Subsequently, our system orchestrates a streamlined retrieval process by harnessing the power of the InterPlanetary File System (IPFS). Through IPFS, the system seamlessly locates and retrieves the file associated with the provided CID, ensuring prompt delivery to the requesting user. This robust file retrieval mechanism underscores our commitment to enhancing user experience and accessibility within the platform.

4.3.4 Decentralized Identity and Authentication:

Description: Decentralized identity solutions are utilized for user authentication.

Functionality: Users validate their ownership of decentralized identity, thereby bolstering security and privacy throughout interactions.

Within the fabric of our project, decentralized identity and authentication functionality serve as the bedrock for establishing trust and security among platform users. Through this feature, users are empowered to authenticate their identity and ownership securely, leveraging decentralized identity solutions. By adopting decentralized identity frameworks, our system ensures that user authentication processes are executed in a manner that upholds privacy, security, and user sovereignty. Through cryptographic mechanisms and decentralized identifiers (DIDs), users validate their ownership of identity credentials, fostering a trust-based environment conducive to secure interactions and transactions within the platform.

4.3.5 Blockchain Transactions:

Description: Users have the capability to engage in transactions on the blockchain, which encompass activities such as establishing user profiles, documenting file uploads, and executing payments.

Functionality: Smart contracts oversee and authenticate user engagements, guaranteeing the integrity and transparency of transactions.

In the multifaceted landscape of our project, blockchain transactions functionality emerges as a pivotal conduit for facilitating a myriad of user engagements and interactions. Through this feature, users are bestowed with the ability to partake in diverse transactions on the blockchain, ranging from establishing user profiles and documenting file uploads to executing payments and transactions. Central to this functionality is the role played by smart contracts, which diligently oversee and authenticate user engagements. By leveraging smart contracts, our system ensures the integrity, transparency, and immutability of transactions, thereby instilling confidence and trust among platform users.

4.3.6 Payment Module:

Description: The platform facilitates cryptocurrency payments for specific transactions.

Functionality: Users have the option to conduct payments using cryptocurrency, such as Ether, via blockchain transactions documented on smart contracts.

Within the framework of our project, the payment module functionality emerges as a pivotal enabler for facilitating seamless and secure transactions within the platform. Through this feature, users are provided with a robust infrastructure for conducting payments using cryptocurrency, such as Ether, thereby unlocking a myriad of possibilities for financial transactions and interactions. Leveraging blockchain technology, our system ensures the integrity, transparency, and security of payment transactions, thereby fostering a trust-based environment conducive to financial interactions. Additionally, by documenting payment transactions on smart contracts, our platform guarantees the immutability and auditability of payment records, instilling confidence and reliability among users engaging in financial transactions.

In essence, the functional requirements outlined above serve as the building blocks for realizing the vision of our decentralized file storage system project. By meticulously designing and implementing these functionalities, our platform endeavors to deliver a seamless, secure, and user-centric experience, thereby unlocking new frontiers in decentralized file storage and digital innovation.

4.4 Non-Functional Requirements

4.4.1 Security:

Security stands as a paramount concern in any system handling sensitive data, particularly personal and financial information. Given the nature of the platform's operations, robust security measures are imperative to safeguard user data and prevent unauthorized access. While blockchain technology inherently provides an additional layer of security through its decentralized and immutable nature, supplementary measures such as SSL encryption and secure authentication protocols should be integrated. SSL encryption ensures that data transmitted between the user's device and the platform's servers remains encrypted and secure from potential eavesdropping or interception. Secure authentication protocols, such as multi-factor authentication or biometric authentication, add an extra layer of security by requiring users to provide additional verification beyond just a username and password. By implementing these security measures, the platform can bolster the protection of user data and mitigate the risk of security breaches or data leaks.

4.4.2 Performance:

Performance optimization is crucial to ensuring the platform can efficiently manage substantial data volumes and handle user traffic without experiencing performance degradation. To achieve optimal performance, emphasis should be placed on optimizing response times and minimizing latency to deliver a seamless user experience. This involves fine-tuning the platform's architecture, optimizing database queries, and leveraging caching mechanisms to reduce load times and improve overall system responsiveness. Additionally, implementing scalable infrastructure solutions, such as cloud-based hosting services or containerization technologies, can help dynamically allocate resources based on demand to maintain consistent performance levels even during peak usage periods. By prioritizing performance optimization, the platform can deliver a responsive and reliable user experience, enhancing user satisfaction and engagement.

4.4.3 Availability:

Ensuring 24/7 availability with minimal downtime is crucial to enable users to access their data and manage storage requirements at any time conveniently. Achieving high availability involves implementing redundant systems and failover mechanisms to minimize service disruptions in the event of hardware failures or maintenance activities. This may include deploying redundant servers in geographically diverse locations, utilizing load balancers to distribute traffic evenly across multiple servers, and implementing automated monitoring and alerting systems to promptly detect and respond to any issues that may arise. By prioritizing availability, the platform can instill confidence in users that their data will always be accessible whenever they need it, enhancing trust and reliability.

4.4.4 Scalability:

The platform must possess scalability capabilities to accommodate growing user numbers and data volumes without compromising performance or security. Scalability involves designing the platform's architecture in a modular and flexible manner, allowing it to scale horizontally by adding additional resources or vertically by upgrading existing hardware components. Implementing scalable database solutions, such as NoSQL databases or sharding techniques, can help distribute data across multiple servers to handle increased loads effectively. Additionally, employing containerization technologies like Docker or Kubernetes enables the platform to dynamically scale resources up or down based on demand, ensuring optimal resource utilization and performance efficiency. By prioritizing scalability, the platform can accommodate future growth and expansion seamlessly, without encountering bottlenecks or limitations.

4.4.5 Usability:

User-friendliness and intuitive navigation are paramount to ensuring a positive user experience. Clear instructions and user-friendly interfaces should be provided to enable users, regardless of their technical expertise, to manage their storage requirements seamlessly. This involves conducting user research and usability testing to understand user preferences and behaviors and iteratively refining the platform's design and functionality based on user feedback. Implementing intuitive navigation features, such as logical menu structures and prominent call-to-action buttons, can help users easily navigate the platform and access desired features or functionalities. Additionally, providing comprehensive user documentation and support resources can further empower users to effectively utilize the platform's capabilities and troubleshoot any issues they may encounter. By prioritizing usability, the platform can enhance user satisfaction and adoption rates, driving engagement and retention.

4.4.6 Blockchain Integration:

Utilizing blockchain technology is essential for ensuring secure storage and data sharing among users. Blockchain technology provides a decentralized and immutable ledger that records all transactions and data modifications, ensuring transparency, traceability, and data integrity. Integrating blockchain technology into the platform involves establishing and upholding secure contracts, known as smart contracts, that govern the interactions and transactions between users. Smart contracts are self-executing contracts with the terms of the agreement directly written into code, ensuring that transactions are automatically executed according to predefined rules and conditions. By leveraging blockchain technology, the platform can enhance security, transparency, and trust among users, enabling them to securely store and share data without relying on centralized intermediaries. Additionally, blockchain technology enables users to monitor and verify transactions in real-time, providing increased visibility and accountability throughout the data sharing process. By prioritizing blockchain integration, the platform can establish a secure and trustworthy environment for users to interact and exchange data, fostering collaboration and innovation.

4.5 Software Requirements

4.5.1 Blockchain Network

Ethereum:

The decentralized file storage system leverages the Ethereum blockchain for its smart contract operations, ensuring compatibility with Ethereum clients like Geth or Infura. Ethereum, renowned for its robustness and flexibility, provides a decentralized platform for executing smart contracts and facilitating secure transactions across a distributed network of nodes. By utilizing Ethereum as the underlying blockchain infrastructure, the decentralized file storage system ensures transparency, immutability, and tamper-resistance, crucial for maintaining the integrity of stored data and transactions.

4.5.2 Smart Contracts

Solidity:

Smart contracts, the cornerstone of Ethereum's decentralized applications (DApps), are coded in Solidity, a purpose-built programming language designed for developing Ethereum smart contracts. Solidity's syntax is tailored to enable developers to define smart contract logic, data structures, and transactional behaviors effectively. Its resemblance to popular programming languages like JavaScript makes it accessible to developers familiar with web development, facilitating seamless integration with existing codebases and frameworks.

Truffle:

Truffle, an Ethereum development framework, serves as a comprehensive suite of tools for compiling, deploying, and testing smart contracts. Truffle streamlines the development process by providing built-in support for tasks such as contract compilation, migration, and testing, reducing development overhead and ensuring consistency across different stages of the development lifecycle. Additionally, Truffle's integrated testing framework enables developers to write and execute automated tests for smart contracts, ensuring their functionality and reliability under various conditions.

4.5.3 Web Development

HTML, CSS, JavaScript:

The frontend of the decentralized file storage system is constructed using conventional web technologies, including HTML for markup, CSS for styling, and JavaScript for dynamic functionality. HTML provides the structural foundation for organizing content, while CSS enhances the visual presentation by defining styles and layouts. JavaScript, as a versatile scripting language, enables interactive and dynamic elements to be integrated into the user interface, enhancing user experience and interactivity.

Web3.js:

Web3.js, a JavaScript library tailored for Ethereum interaction, is integrated into the decentralized file storage system to facilitate communication with Ethereum nodes directly from the web application. Web3.js acts as a bridge between the frontend user interface and the Ethereum blockchain, enabling seamless integration of blockchain functionality such as transaction submission, contract interaction, and event monitoring into web-based applications. By leveraging Web3.js, developers can harness the full power of Ethereum's capabilities within their web applications, unlocking a wide range of decentralized features and functionalities.

4.5.4 Web Framework

Bootstrap:

Bootstrap, a popular front-end framework developed by Twitter, is utilized to enhance the user interface design and ensure responsiveness in the decentralized file storage system. Bootstrap provides a comprehensive set of pre-designed UI components, layout grids, and responsive utilities, allowing developers to quickly create visually appealing and mobile-friendly web interfaces with minimal effort. By adopting Bootstrap, the platform ensures consistency in design across different devices and screen sizes, improving usability and accessibility for users accessing the application from various devices and platforms.

4.5.5 IPFS Integration

IPFS Client Library:

Integration of an IPFS client library, such as ipfs-http-client, is essential for enabling interaction with the InterPlanetary File System (IPFS) network from the application. IPFS, a peer-to-peer distributed file system, revolutionizes the way data is stored and accessed by utilizing content-addressable storage and decentralized distribution mechanisms. The IPFS client library facilitates seamless integration of IPFS functionality into the decentralized file storage system, allowing users to upload, retrieve, and share files securely and efficiently across the IPFS network.

4.5.6 Payment Module

Web3.js (Ether Transfer):

The platform leverages Web3.js to facilitate transactions involving cryptocurrency (Ether) on the Ethereum blockchain. Web3.js enables users to initiate and authorize Ether transfers directly from the web application, interacting with Ethereum smart contracts to execute payment transactions securely and transparently. By integrating Web3.js for cryptocurrency payments, the platform provides users with a seamless and decentralized payment experience, enabling them to transact with confidence and privacy on the blockchain.

4.5.7 User Authentication

Solidity:

User authentication features are extended to smart contract functionality to ensure secure and decentralized authentication mechanisms. By incorporating authentication logic into smart contracts coded in Solidity, the platform eliminates the need for centralized authentication servers or third-party authentication providers, enhancing security and privacy throughout user interactions.

Web3.js:

User authentication and authorization are implemented through Web3.js interactions with smart contracts. Web3.js enables users to authenticate their identities and authorize transactions securely and transparently, leveraging cryptographic techniques and blockchain technology to verify ownership of decentralized identities and access rights. By integrating Web3.js for user authentication, the platform ensures that only authorized users can access and interact with sensitive data and functionalities, safeguarding against unauthorized access and malicious activities.

4.5.8 Build Tools

Node.js:

Node.js serves as the server-side scripting platform and runtime environment for build tools used in the development of the decentralized file storage system. Node.js enables developers to write server-side code in JavaScript, leveraging its event-driven architecture and non-blocking I/O model to build scalable and high-performance web applications.

By utilizing Node.js, developers can streamline the development process, share code between the frontend and backend, and leverage a vast ecosystem of npm packages and libraries to enhance functionality and efficiency.

npm (Node Package Manager):

npm, the default package manager for Node.js, is employed to manage project dependencies and scripts in the decentralized file storage system. npm simplifies dependency management by providing a centralized repository of reusable code modules and libraries, enabling developers to easily integrate thirdparty libraries and tools into their projects. Additionally, npm facilitates the installation, updating, and removal of dependencies, ensuring consistency and reliability in the project's development environment. By leveraging npm, developers can expedite the development process, reduce dependency conflicts, and maintain project dependencies efficiently.

4.5.9 Version Control

Git:

Git, a distributed version control system, is employed for version control in the development of the decentralized file storage system. Git enables developers to track changes to the project's source code, collaborate with team members, and manage codebase versions effectively. By utilizing Git, developers can create branches to work on new features or bug fixes independently, merge changes seamlessly, and revert to previous versions if needed. Additionally, Git provides robust branching and merging capabilities, facilitating parallel development and code collaboration among team members.

4.5.10 Integrated Development Environment (IDE)

Visual Studio Code:

Visual Studio Code, a lightweight and versatile code editor developed by Microsoft, is utilized as the primary integrated development environment (IDE) for Solidity smart contract development and web application coding in the decentralized file storage system. Visual Studio Code offers a wide range of features and extensions tailored for web development and blockchain development, including syntax highlighting, code completion, debugging tools, and Git integration. Its intuitive user interface and customizable workflow make it a popular choice among developers for writing, editing, and debugging code efficiently.

4.5.11 Testing Frameworks

Truffle Test:

Truffle's testing framework is employed for writing and running tests for smart contracts in the decentralized file storage system. Truffle Test provides a comprehensive suite of testing utilities and assertions for verifying the functionality and behavior of smart contracts under various conditions. By writing automated tests with Truffle Test, developers can ensure the reliability and correctness of smart contract logic, identify and address potential bugs or vulnerabilities, and maintain code quality throughout the development lifecycle.

4.5.12 Browser Compatibility

Cross-Browser Testing:

Ensuring compatibility with major web browsers such as Chrome, Firefox, and Safari is essential to guarantee a consistent and seamless user experience across different platforms and devices. Cross-browser testing involves validating the functionality and appearance of the decentralized file storage system in various web browsers to identify and address any compatibility issues or inconsistencies. By conducting thorough cross-browser testing, developers can ensure that the platform performs optimally and remains accessible to users regardless of their choice of web browser or device.

DApp Browser or Extension

MetaMask:

MetaMask, a popular DApp browser extension for Ethereum interactions, is utilized to test the decentralized application in the development environment. MetaMask provides a user-friendly interface for interacting with Ethereum-based DApps directly from the web browser, allowing users to manage their Ethereum accounts, sign transactions, and interact with smart contracts seamlessly. By integrating MetaMask into the development workflow, developers can validate the functionality and user experience of the decentralized file storage system in a real-world Ethereum environment, ensuring compatibility and reliability before deployment to production.

CHAPTER 5
IMPLEMENTATION

5.1 Implementation Overview:

Frontend Interface:

User Interface and User Experience (UI/UX):

The implementation of the user interface and user experience (UI/UX) focuses on creating a seamless and intuitive experience for users interacting with the decentralized storage space marketplace. The frontend application, typically built using modern web technologies such as HTML, CSS, and JavaScript, presents a user-friendly design for accessing storage space listings, initiating transactions, and managing user accounts. The interface should be responsive to accommodate various devices, including desktops, tablets, and mobile phones. Ensuring accessibility and ease of navigation is paramount for a positive user experience. Additionally, user authentication and authorization mechanisms should be in place to safeguard user data and maintain secure interactions with the marketplace.

Integration with IPFS:

Integration with the InterPlanetary File System (IPFS) allows the project to leverage decentralized file storage and sharing capabilities. The IPFS client installed on users' machines facilitates interactions with the IPFS network, enabling users to upload, download, and manage files related to storage spaces. By utilizing IPFS, the project achieves data redundancy and resilience, as files are distributed across multiple nodes in the network. This reduces dependency on centralized servers and enhances the overall security and availability of data. Additionally, IPFS's content-addressing feature ensures that files are uniquely identified and easily accessible based on their content, rather than their location.

Smart Contract Development:

Smart contracts are a central part of the project's implementation, automating key aspects of the storage space marketplace. These self-executing contracts are deployed on the Ethereum blockchain and contain predefined rules and logic for managing transactions and interactions. Smart contracts handle tasks such as verifying user permissions, processing storage space rentals and purchases, and managing payments. By automating these processes, smart contracts minimize the need for intermediaries and enhance transparency and trust in the system. The development of smart contracts requires a strong understanding of blockchain technology and programming languages such as Solidity.

Blockchain Integration:

The project's integration with the Ethereum blockchain ensures secure and transparent transaction processing. Blockchain nodes facilitate communication with the Ethereum network, enabling users to interact with smart contracts and access blockchain ledger data. The use of blockchain provides benefits such as immutability, decentralized control, and tamper-resistant records. Additionally, blockchain-based payment modules allow for secure handling of transactions, including adding, verifying, and processing payment information. This integration enhances the overall security and reliability of the marketplace, fostering trust among users and stakeholders.

Mining and Network Security:

Mining plays a crucial role in maintaining network security and transaction validation within the blockchain. Miners validate and add new blocks of transactions to the blockchain through the mining process, contributing computational resources to solve complex mathematical puzzles. This ensures the integrity of the blockchain and the security of transactions within the decentralized storage space marketplace. Additionally, the project's implementation should consider measures to protect against potential attacks and vulnerabilities, such as double-spending and 51% attacks. By securing the blockchain network, the project enhances trust and reliability for users engaging in transactions.

User Data Management:

Effective user data management is a key component of the project's implementation. The system should prioritize data privacy and security, ensuring that user information is protected according to relevant regulations and best practices. This involves the use of encryption to secure user data during storage and transmission, as well as implementing access controls to restrict data access to authorized individuals. User data may include personal information, transaction history, and preferences related to storage space rentals or purchases. By safeguarding user data, the project can maintain user trust and comply with data protection standards.

Monitoring and Analytics:

Monitoring and analytics tools are essential for the ongoing success of the project. These tools allow for real-time tracking of system performance, user engagement, and potential issues within the marketplace. Metrics such as transaction volume, user activity, and storage space utilization provide valuable insights that can inform decisions on system optimization and enhancements. Additionally, monitoring can help identify potential security threats or anomalies, enabling proactive measures to maintain system stability and security. Regular data analysis can guide improvements in the user experience and overall system efficiency.

Scalability and Performance:

Scalability and performance considerations are important for the long-term success of the project. As the number of users and transactions increases, the system must be capable of handling larger workloads without compromising performance. This involves optimizing the architecture to support increased demand, such as distributing data storage and processing across multiple nodes or servers. Caching mechanisms and load balancing techniques can enhance responsiveness and reduce latency. Additionally, the project's implementation should include plans for scaling blockchain infrastructure and IPFS nodes as needed to support growing usage.

Testing and Quality Assurance:

Testing and quality assurance are crucial for ensuring the project's reliability and functionality. This includes unit testing, integration testing, and end-to-end testing to verify that all components of the system work as intended. Smart contracts should undergo rigorous testing to validate their logic and security, while frontend and backend applications should be tested for usability and performance. Automated testing frameworks can streamline the testing process and help identify potential bugs or issues early in development. Quality assurance practices contribute to a stable and high-quality user experience.

Community and Ecosystem Engagement:

Engaging with the community and ecosystem surrounding the project can provide valuable feedback and support. This includes active participation in developer communities, blockchain forums, and IPFS networks. Community engagement can lead to collaborations and partnerships that enhance the project's features and reach. Additionally, gathering input from users and stakeholders can guide future updates and improvements, ensuring that the project remains aligned with user needs and industry trends. By fostering a strong community presence, the project can build a network of support and advocacy that contributes to its long-term success.

Tools Used:

User Interface and User Experience (UI/UX):

To create an effective user interface and user experience, developers utilize a combination of design tools and frameworks. Popular design tools such as Figma, Adobe XD, or Sketch enable designers to create wireframes and prototypes, visualizing the user journey and layout of the application. Once the design is finalized, frontend frameworks like React, Vue.js, or Angular are used to implement the interface, ensuring a responsive and interactive user experience. CSS preprocessors like Sass or Less can enhance styling, while JavaScript libraries such as jQuery simplify DOM manipulation and event handling. Accessibility testing tools, including WAVE or aXe, help ensure the application meets accessibility standards for all users. By combining these tools, developers can create an intuitive, visually appealing, and responsive UI that enhances user interaction with the decentralized storage space marketplace.

Integration with IPFS:

Integration with the InterPlanetary File System (IPFS) is achieved using specialized tools and libraries. IPFS provides its own software suite, including the IPFS command-line interface (CLI) and IPFS desktop application, to manage files and interact with the network. Additionally, the IPFS JavaScript client (ipfs-http-client) allows developers to integrate IPFS capabilities directly into web applications. This library provides methods for uploading, downloading, and managing files stored on IPFS. Furthermore, tools such as IPFS Cluster enable the creation of distributed IPFS networks for improved data redundancy and load balancing. By leveraging these tools, the project can seamlessly integrate IPFS functionality, offering users efficient access to decentralized data storage.

Smart Contract Development:

Smart contract development requires specific tools and environments tailored to blockchain programming. Solidity, the primary programming language for Ethereum smart contracts, is used to write contract code. Developers can leverage tools like Remix, an online IDE, to write, compile, and deploy smart contracts directly on the Ethereum blockchain. Additionally, Truffle Suite provides a development environment and framework for managing smart contracts, including testing and deployment scripts. Ganache, a personal Ethereum blockchain, can simulate blockchain environments for local testing and debugging of smart contracts. For security analysis, tools such as MythX and Oyente help identify vulnerabilities and potential issues within contract code. By utilizing these specialized tools, developers can create, test, and deploy robust smart contracts for the decentralized storage space marketplace.

Blockchain Integration:

Blockchain integration involves using a range of tools and libraries to interact with the Ethereum blockchain. Web3.js, a JavaScript library, facilitates communication between applications and the Ethereum network, enabling the execution of smart contract methods and retrieval of blockchain data. Similarly, ethers.js provides a lightweight alternative for blockchain interaction and contract management. For blockchain node setup and management, tools like Geth (Go Ethereum) and Parity can be used to run Ethereum clients, allowing nodes to participate in the blockchain network. Additionally, Metamask and other wallet solutions enable users to manage their accounts and interact with the blockchain securely. These tools work together to enable smooth integration with the Ethereum blockchain, providing the necessary infrastructure for decentralized transactions and smart contract execution.

Mining and Network Security:

The project's mining and network security rely on a combination of hardware and software tools. Specialized mining hardware, such as GPUs or ASICs, is used for efficient mining, while mining software like Ethminer or Claymore's Dual Ethereum Miner facilitates the mining process. Monitoring tools like Hashrate OS or Hive OS provide insights into mining performance and help optimize mining operations. For network security, tools such as Snort and Suricata enable real-time intrusion detection, monitoring network traffic for potential threats. Additionally, blockchain-specific tools like Etherscan and BlockScout allow for detailed analysis of blockchain transactions and activity, aiding in the detection of suspicious behavior. By combining these tools, the project can maintain efficient mining operations while ensuring the security and integrity of the blockchain network.

User Data Management:

Effective user data management involves using a suite of tools that ensure data privacy, security, and compliance with relevant regulations. Data encryption tools such as OpenSSL or WebCrypto API help secure user data during storage and transmission. Access control and identity management tools, including OAuth or OpenID Connect, allow developers to implement secure authentication and authorization mechanisms. Additionally, privacy-focused databases such as MongoDB or PostgreSQL offer features like role-based access control and encryption to safeguard user information. Data masking and anonymization tools can also be used to protect sensitive data from unauthorized access. Furthermore, compliance tools such as OneTrust assist in managing user data according to regulations like GDPR or CCPA, helping maintain trust and transparency with users.

Monitoring and Analytics:

To maintain an efficient and reliable decentralized storage space marketplace, developers use monitoring and analytics tools to track system performance and user engagement. Tools like Prometheus and Grafana provide real-time monitoring and visualization of system metrics, including CPU and memory usage, transaction rates, and latency. Logging frameworks such as ELK Stack (Elasticsearch, Logstash, Kibana) enable the collection and analysis of application logs for debugging and troubleshooting. Additionally, user behavior analytics tools like Google Analytics or Mixpanel help gather insights into user interactions and preferences, guiding improvements to the user experience. By leveraging these tools, developers can proactively identify and address potential issues, optimizing the system for better performance and user satisfaction.

Monitoring and Analytics:

To maintain an efficient and reliable decentralized storage space marketplace, developers use monitoring and analytics tools to track system performance and user engagement. Tools like Prometheus and Grafana provide real-time monitoring and visualization of system metrics, including CPU and memory usage, transaction rates, and latency. Logging frameworks such as ELK Stack (Elasticsearch, Logstash, Kibana) enable the collection and analysis of application logs for debugging and troubleshooting. Additionally, user behavior analytics tools like Google Analytics or Mixpanel help gather insights into user interactions and preferences, guiding improvements to the user experience. By leveraging these tools, developers can proactively identify and address potential issues, optimizing the system for better performance and user satisfaction.

Scalability and Performance:

Scalability and performance tools are essential for ensuring the system can handle increasing workloads and provide a smooth user experience. Load balancing tools such as Nginx or HAProxy help distribute traffic across multiple servers, reducing bottlenecks and improving responsiveness. Auto-scaling services like Kubernetes or Docker Swarm dynamically adjust resource allocation based on demand, ensuring efficient resource utilization. Caching mechanisms, including Redis or Memcached, speed up data retrieval and reduce database load by storing frequently accessed data in memory. Additionally, tools like New Relic or AppDynamics offer application performance monitoring, providing insights into response times, error rates, and other key performance indicators. By incorporating these tools, the project can achieve high availability and responsiveness, accommodating growth while maintaining performance.

Testing and Quality Assurance:

A robust testing and quality assurance process involves the use of various tools to ensure the reliability and stability of the decentralized storage space marketplace. Unit testing frameworks such as Jest or Mocha allow developers to validate individual components of the application. Integration testing tools like Postman or Insomnia enable the testing of API endpoints and interactions between different system components. For smart contract testing, tools such as Hardhat or Truffle's built-in testing suite allow developers to simulate blockchain environments and verify contract logic. Additionally, end-to-end testing frameworks like Cypress or Selenium test the entire user journey within the application, ensuring a seamless experience. Automated testing tools such as GitHub Actions or GitLab CI/CD streamline the testing process and provide continuous feedback. By leveraging these tools, developers can maintain high-quality code and deliver a stable and reliable system to users.

Community and Ecosystem Engagement:

Engaging with the community and ecosystem requires tools that facilitate collaboration and communication. Platforms like GitHub or GitLab provide a space for developers to collaborate on the project, share code, and manage version control. Discussion forums and community chat platforms such as Discord or Slack enable direct communication with users and other developers, fostering knowledge sharing and support. Additionally, social media management tools like Hootsuite or Buffer help maintain a strong online presence and reach a wider audience. Surveys and feedback tools like SurveyMonkey or Typeform allow developers to gather insights from users and stakeholders, guiding future improvements to the project. By actively engaging with the community and leveraging these tools, the project can build a supportive network that contributes to its long-term success.

Algorithms Used:

Search for Storage Space

The search algorithm for storage space is essential in providing users with relevant and accurate results when they query available storage options. The algorithm often involves a combination of filtering, ranking, and relevance matching based on the user's query criteria and preferences.

Query Parsing: The algorithm starts by parsing the user's search query to identify keywords, location preferences, space requirements, and other parameters such as duration, price range, and storage type.

Data Retrieval: Next, the algorithm retrieves data from the database or the IPFS network containing the storage listings. It fetches all listings that meet the user's specified criteria.

Filtering: The retrieved data is filtered based on the user’s requirements such as location, type of storage, capacity, and price range. This step ensures that the user is presented with listings relevant to their needs.

Ranking and Relevance Matching: The algorithm ranks the filtered listings based on relevance to the user’s query. This ranking can consider various factors such as proximity to the user's specified location, user reviews or ratings, and availability.

Results Presentation: The top-ranked and most relevant results are presented to the user through the frontend interface. Pagination or infinite scrolling may be implemented to handle large result sets and improve user experience.

The algorithm may utilize machine learning techniques to improve search relevance over time by learning from user interactions and feedback. This continuous improvement ensures that the algorithm adapts to evolving user preferences and behaviors.

Rent and Purchase of Storage Space

The rent and purchase algorithm involves facilitating transactions between users and storage space providers. This includes reserving and confirming storage space, as well as handling payments.

User Selection: The algorithm begins with the user selecting a desired storage space listing based on their search results. The user may specify the rental duration, start date, and any other preferences.

Availability Check: Before finalizing the transaction, the algorithm checks the availability of the chosen storage space. This involves verifying the listing against the provider’s schedule to ensure the space is not already reserved.

Contract Creation: Once availability is confirmed, the algorithm creates a contract or agreement detailing the terms of the rental or purchase. This contract includes information such as rental duration, price, and any additional terms.

Payment Handling: The algorithm handles the user's payment for the storage space, which may involve interaction with the blockchain-based payment module. This includes adding, verifying, and processing the user's payment information.

Confirmation and Receipt: After payment is processed, the algorithm sends a confirmation to the user and provider, officially reserving or transferring ownership of the storage space. A receipt is generated and stored for future reference.

This algorithm is designed to ensure secure and transparent transactions between users and providers, leveraging blockchain technology for reliable and immutable record-keeping.

Blockchain-Based Payment Module

The blockchain-based payment module uses cryptographic algorithms to ensure secure and transparent transactions between users and providers. The module leverages smart contracts and distributed ledger technology.

Smart Contract Interaction: The algorithm begins with interacting with a smart contract on the blockchain. This contract governs the terms of the transaction, including payment details and amounts.

Adding Payment Information: The algorithm securely adds the user's payment information to the smart contract, which may involve creating a digital wallet for the user if they do not already have one.

Verify Payment Information: The smart contract verifies the user's payment information, including the amount and source of funds. This step ensures that the user has sufficient funds and that the payment is valid.

Process Payment: Once verification is complete, the algorithm processes the payment by transferring the specified amount from the user's wallet to the provider's wallet. This transaction is recorded on the blockchain for transparency.

Receipt and Confirmation: After the payment is successfully processed, the algorithm generates a transaction receipt and confirmation. This receipt is stored on the blockchain, providing an immutable record of the transaction.

The use of blockchain technology in this module ensures secure and transparent payment handling, with smart contracts automating many aspects of the transaction process.

User Authentication and Authorization

User authentication and authorization are essential algorithms for ensuring secure access to the system and protecting user data and transactions.

User Authentication: The algorithm begins by verifying the user's identity when they attempt to access the system. This is typically done using methods such as username and password, biometric authentication, or single sign-on (SSO) solutions.

Session Management: Once the user is authenticated, the algorithm creates a secure session for the user, allowing them to remain logged in while navigating the application. Techniques such as token-based authentication (e.g., JSON Web Tokens) are commonly used.

Role-Based Access Control (RBAC): The algorithm checks the user's role or permissions to determine which resources and actions they are authorized to access. This helps enforce security policies and protect sensitive data.

Multi-Factor Authentication (MFA): To enhance security, the algorithm may require the user to provide additional verification, such as a code sent to their mobile device, especially for critical actions like transactions.

Authorization Checks: Throughout the user's session, the algorithm performs authorization checks to ensure the user has the necessary permissions to perform certain actions, such as accessing specific data or initiating transactions.

By implementing robust authentication and authorization algorithms, the system can protect user accounts and data from unauthorized access, while also ensuring compliance with security best practices.

IPFS Data Management

The IPFS data management algorithm involves efficiently handling data storage, retrieval, and updates using the InterPlanetary File System (IPFS).

Data Upload: The algorithm starts by receiving data from the user for storage on the IPFS network. The data is then hashed to create a unique content identifier (CID), which serves as its address in the network.

Data Storage: Once the data is uploaded, it is stored across the distributed IPFS network. The algorithm may leverage pinning services to ensure the data remains available and accessible over time.

Data Retrieval: When a user requests data stored on IPFS, the algorithm uses the CID to locate and retrieve the data from the network. IPFS's peer-to-peer architecture allows for efficient data distribution and retrieval.

Data Updates: If the user wants to update or modify data, the algorithm will handle the process of uploading the new version of the data and generating a new CID. The old version can be kept or deleted based on the user's preference.

Data Integrity: Throughout the data management process, the algorithm ensures data integrity and security by using IPFS's content-addressing system and cryptographic hashing.

By efficiently managing data on IPFS, the algorithm provides users with reliable and decentralized storage options while leveraging the benefits of peer-to-peer networking.

Miner Operations and Block Validation

Miner operations and block validation are critical algorithms that ensure the security and integrity of the blockchain network.

Mining Setup: The algorithm begins with the miner setting up the mining hardware and software to participate in the blockchain network. This includes joining a mining pool or operating as a solo miner.

Block Proposal: When a miner receives a set of unconfirmed transactions, the algorithm creates a candidate block that includes these transactions along with a reference to the previous block in the chain.

Proof of Work: The algorithm attempts to solve the cryptographic puzzle associated with the candidate block, known as the proof-of-work process. This involves generating a hash that meets the network's difficulty level.

Block Validation: Once the puzzle is solved, the algorithm broadcasts the new block to the network for validation. Other nodes verify the block's transactions and proof-of-work before adding it to the chain.

Login and Authentication Module

The login and authentication module employs algorithms that ensure secure access to the decentralized storage space marketplace while protecting user privacy.

User Credentials Validation: The algorithm starts by validating user credentials (e.g., username and password) provided during login. It uses encryption algorithms such as SHA-256 to securely hash and compare user-entered passwords with stored hashes.

Two-Factor Authentication (2FA): For added security, the algorithm may require a second authentication factor (e.g., a code sent to the user's phone) to verify the user's identity. This step helps prevent unauthorized access, even if a user's password is compromised.

Session Management: After successful authentication, the algorithm creates a session for the user. This may involve generating a session token using algorithms such as JSON Web Token (JWT). The token is encrypted and stored securely on the client side.

Token Validation and Renewal: The algorithm validates the session token on each user request to maintain the user's authenticated state. Tokens may be renewed periodically to ensure session security and prevent unauthorized access.

Logout and Session Termination: Upon user logout or session expiration, the algorithm terminates the session, invalidates the session token, and clears any sensitive data from the client side.

By leveraging secure hashing, encryption, and token-based authentication, the login and authentication module provides a robust mechanism for user access control.

Smart Contract Execution

The smart contract execution module involves algorithms for deploying, interacting with, and managing smart contracts on the blockchain.

Contract Deployment: The algorithm begins with deploying a smart contract to the Ethereum blockchain. This process involves compiling the contract code, creating a transaction with the compiled code, and submitting it to the blockchain for execution.

Contract Interaction: Once the contract is deployed, the algorithm allows users and other smart contracts to interact with it. This involves calling contract methods using blockchain libraries such as Web3.js or ethers.js, passing input parameters as needed.

State Management: The algorithm manages the contract's state, ensuring that data changes according to contract logic. Changes in contract state, such as updating rental agreements or payment records, are securely recorded on the blockchain.

Event Emission: Smart contracts may emit events to signal specific occurrences, such as successful transactions or contract state changes. The algorithm listens for these events to trigger subsequent actions or notify users.

Error Handling: The algorithm includes error handling routines to manage exceptions during contract execution. This may involve reverting transactions, retrying operations, or providing meaningful error messages to users.

Smart contract execution algorithms leverage blockchain technology for secure, transparent, and automated management of storage space marketplace transactions.

IPFS Data Management

The IPFS data management module encompasses algorithms for efficiently storing, retrieving, and managing data using the InterPlanetary File System (IPFS).

Data Upload: The algorithm allows users to upload data to the IPFS network. Data is split into chunks, and each chunk is hashed using a cryptographic algorithm such as SHA-256. The resulting content identifier (CID) is used to reference the data.

Data Retrieval: Users can retrieve data from the IPFS network using the CID. The algorithm queries the IPFS network for the data and retrieves the requested chunks, reconstructing the original file for the user.

Data Pinning: To ensure data availability, the algorithm may pin data (i.e., keep it stored locally) on IPFS nodes. Pinning helps maintain data accessibility even if the original provider goes offline.

Data Replication and Redundancy: The algorithm may replicate data across multiple IPFS nodes to ensure redundancy and improve data availability. This approach enhances data resilience and reliability.

Access Control and Permissions: The algorithm can use encryption and permission management to control access to data. This ensures that only authorized users can access specific data, preserving privacy and data security.

The monitoring and analytics module is essential for maintaining the performance, security, and efficiency of the decentralized storage system. It employs a combination of algorithms and tools to track system metrics, user behavior, and potential issues in real-time. Here's a detailed explanation of each aspect:

Real-Time Monitoring:

Real-time monitoring algorithms continuously observe various system metrics, providing immediate insights into the system's performance. Key metrics include:

CPU and Memory Usage: By monitoring CPU and memory usage, the system can identify potential bottlenecks or resource-intensive processes that may affect performance.

Transaction Rates: Tracking transaction rates helps understand user activity and the load on the system. Unusual spikes in transaction rates may indicate a security threat or performance issue.

Response Times: Monitoring response times helps ensure the system remains responsive and efficient. Delays in response times could indicate problems with system components or network connectivity.

Real-time monitoring allows for proactive identification and resolution of performance issues, ensuring the system remains stable and efficient.

Event Logging

Event logging involves recording significant application events and errors for analysis and troubleshooting. Logs may include:

Timestamps: Each log entry is timestamped, providing a clear timeline of events and errors.

Severity Levels: Logs can be classified by severity (e.g., info, warning, error) to prioritize issues and responses.

Stack Traces: For errors, stack traces provide detailed information about where in the code the error occurred, aiding in debugging.

Effective logging provides a comprehensive record of system activities, enabling developers and administrators to trace issues and improve system reliability.

User Behaviour Analysis

User behavior analysis involves tracking and analyzing user interactions with the platform to optimize the user experience. Key areas include:

Navigation Paths: Analyzing how users navigate the platform helps identify popular paths and areas for improvement.

Session Durations: Tracking session durations can provide insights into user engagement and potential areas where users might be dropping off.

Frequently Used Features: Understanding which features are most commonly used can guide prioritization in feature development and improvements.

By understanding user behavior, the system can be optimized to provide a better user experience and improve overall satisfaction.

Alerts and Notifications

Alerts and notifications are triggered when monitoring algorithms detect that predefined thresholds are exceeded. Examples include:

High Error Rates: If the system detects an unusually high rate of errors, an alert can be triggered to prompt investigation and resolution.

Resource Usage: Alerts can be set for excessive CPU or memory usage, which may indicate a potential performance issue.

Security Events: Alerts can also be set for security events, such as suspicious login attempts or unauthorized access.

These alerts enable administrators to respond quickly to potential issues, minimizing downtime and maintaining system reliability.

Data Visualization

Data visualization algorithms transform raw data into charts, graphs, and reports that provide actionable insights. Key areas include:

Performance Metrics: Visualizations of performance metrics (e.g., response times, error rates) help administrators quickly assess the health of the system.

User Behavior: Reports on user behavior can inform decisions on user experience improvements and feature development.

Trend Analysis: Visualizations help identify trends over time, such as seasonal variations in user activity or emerging performance issues.

CHAPTER 6

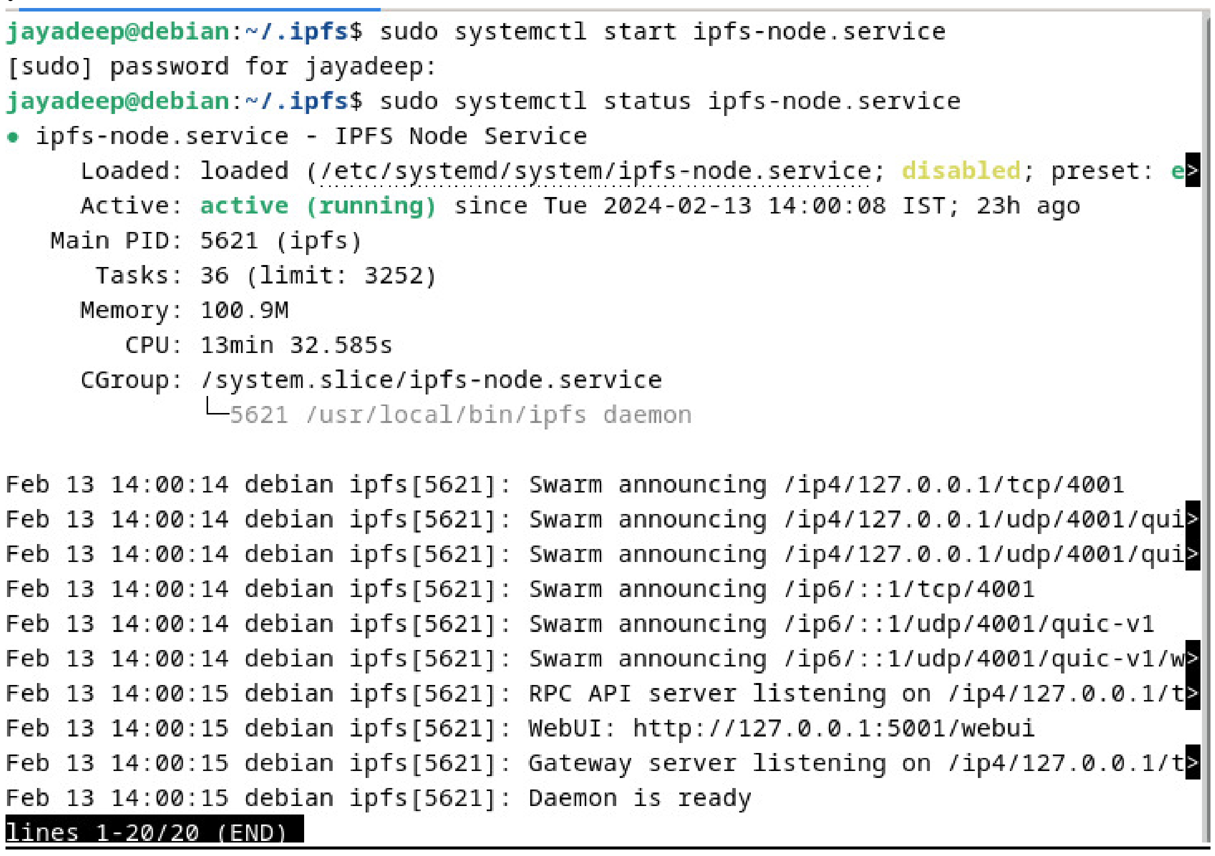
RESULTS AND DISCUSSIONS

Running the "Decentralized File Storage" project involves several configurations to execute the program effectively. Here's an overview of the platform and configurations used:

Platform: Debian Linux 12 Terminal

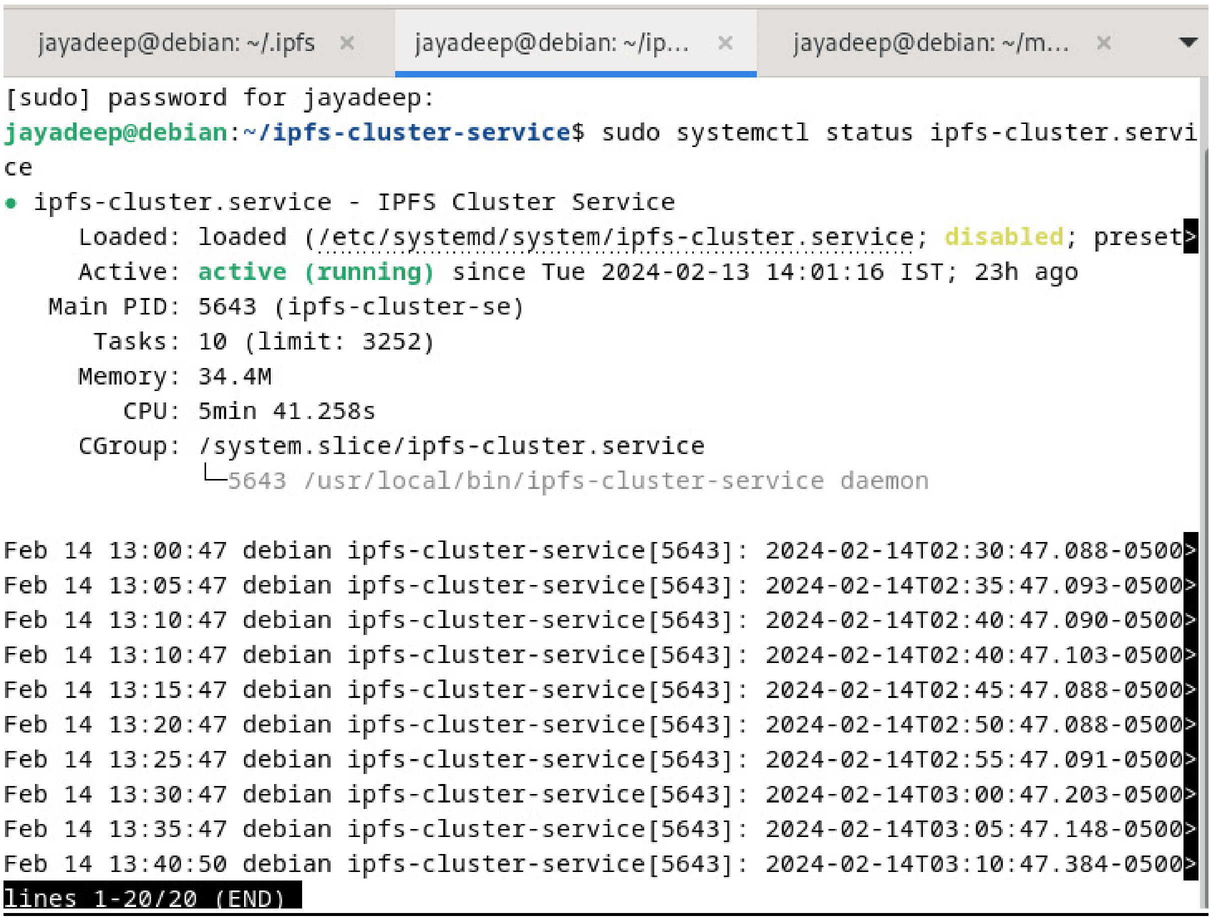
The Debian Linux 12 Terminal serves as the primary environment for running the program. It provides a stable and secure operating system for executing commands and managing the project.

•



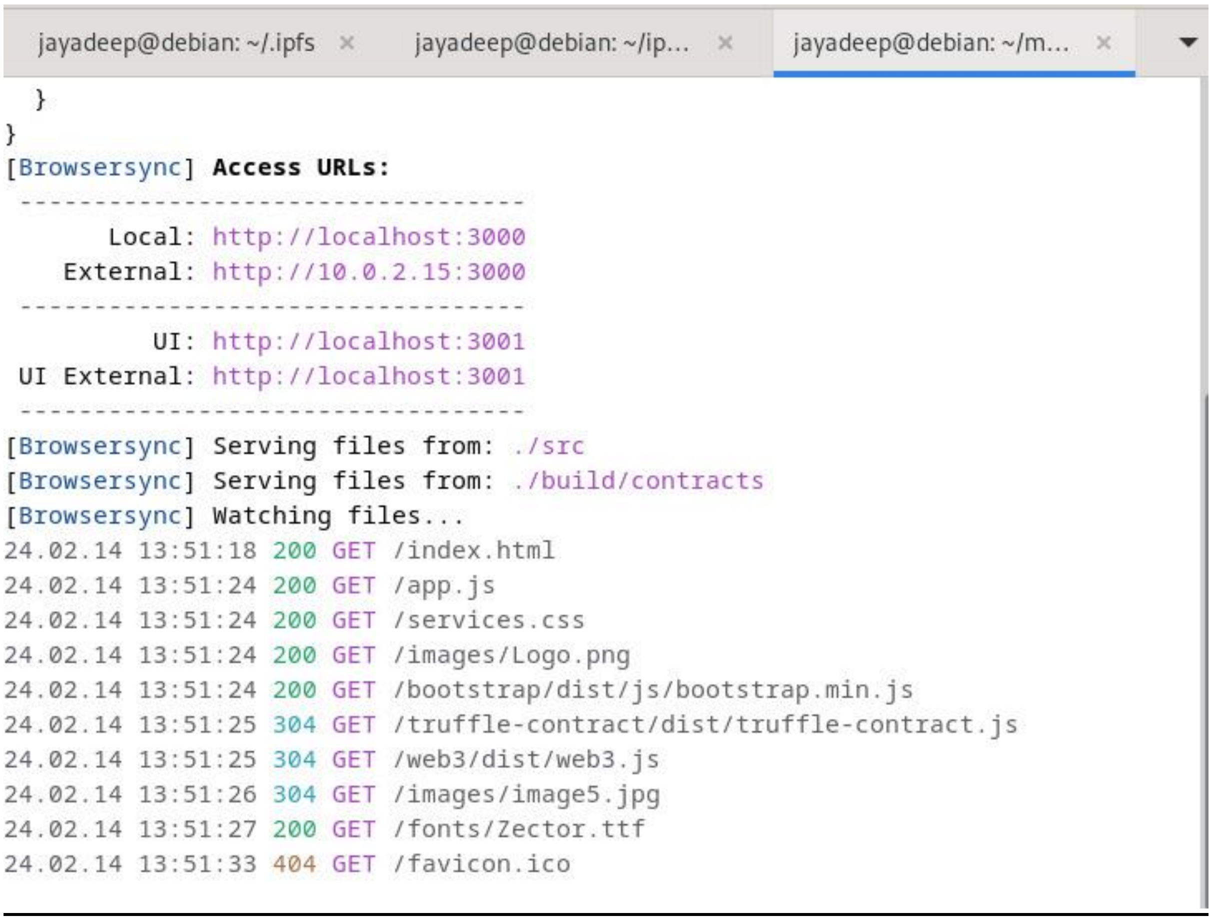
```
jayadeep@debian:~/.ipfs$ sudo systemctl start ipfs-node.service
[sudo] password for jayadeep:
jayadeep@debian:~/.ipfs$ sudo systemctl status ipfs-node.service
● ipfs-node.service - IPFS Node Service
     Loaded: loaded (/etc/systemd/system/ipfs-node.service; disabled; preset: e>
     Active: active (running) since Tue 2024-02-13 14:00:08 IST; 23h ago
   Main PID: 5621 (ipfs)
      Tasks: 36 (limit: 3252)
     Memory: 100.9M
        CPU: 13min 32.585s
     CGroup: /system.slice/ipfs-node.service
             └─5621 /usr/local/bin/ipfs daemon

Feb 13 14:00:14 debian ipfs[5621]: Swarm announcing /ip4/127.0.0.1/tcp/4001
Feb 13 14:00:14 debian ipfs[5621]: Swarm announcing /ip4/127.0.0.1/udp/4001/qui>
Feb 13 14:00:14 debian ipfs[5621]: Swarm announcing /ip4/127.0.0.1/udp/4001/qui>
Feb 13 14:00:14 debian ipfs[5621]: Swarm announcing /ip6/::1/tcp/4001
Feb 13 14:00:14 debian ipfs[5621]: Swarm announcing /ip6/::1/udp/4001/quic-v1
Feb 13 14:00:14 debian ipfs[5621]: Swarm announcing /ip6/::1/udp/4001/quic-v1/w>
Feb 13 14:00:15 debian ipfs[5621]: RPC API server listening on /ip4/127.0.0.1/t>
Feb 13 14:00:15 debian ipfs[5621]: WebUI: http://127.0.0.1:5001/webui
Feb 13 14:00:15 debian ipfs[5621]: Gateway server listening on /ip4/127.0.0.1/t>
Feb 13 14:00:15 debian ipfs[5621]: Daemon is ready
lines 1-20/20 (END)
```

Fig 6.1 Node Service Activation



```
[sudo] password for jayadeep:
jayadeep@debian:~/ipfs-cluster-service$ sudo systemctl status ipfs-cluster.servi
ce
● ipfs-cluster.service - IPFS Cluster Service
     Loaded: loaded (/etc/systemd/system/ipfs-cluster.service; disabled; preset>
     Active: active (running) since Tue 2024-02-13 14:01:16 IST; 23h ago
   Main PID: 5643 (ipfs-cluster-se)
      Tasks: 10 (limit: 3252)
     Memory: 34.4M
        CPU: 5min 41.258s
     CGroup: /system.slice/ipfs-cluster.service
             └─5643 /usr/local/bin/ipfs-cluster-service daemon

Feb 14 13:00:47 debian ipfs-cluster-service[5643]: 2024-02-14T02:30:47.088-0500>
Feb 14 13:05:47 debian ipfs-cluster-service[5643]: 2024-02-14T02:35:47.093-0500>
Feb 14 13:10:47 debian ipfs-cluster-service[5643]: 2024-02-14T02:40:47.090-0500>
Feb 14 13:10:47 debian ipfs-cluster-service[5643]: 2024-02-14T02:40:47.103-0500>
Feb 14 13:15:47 debian ipfs-cluster-service[5643]: 2024-02-14T02:45:47.088-0500>
Feb 14 13:20:47 debian ipfs-cluster-service[5643]: 2024-02-14T02:50:47.088-0500>
Feb 14 13:25:47 debian ipfs-cluster-service[5643]: 2024-02-14T02:55:47.091-0500>
Feb 14 13:30:47 debian ipfs-cluster-service[5643]: 2024-02-14T03:00:47.203-0500>
Feb 14 13:35:47 debian ipfs-cluster-service[5643]: 2024-02-14T03:05:47.148-0500>
Feb 14 13:40:50 debian ipfs-cluster-service[5643]: 2024-02-14T03:10:47.384-0500>
lines 1-20/20 (END)
```

Fig 6.2 Cluster Service Activation



```
  }
}
[Browsersync] Access URLs:
 ----------------------------------
       Local: http://localhost:3000
    External: http://10.0.2.15:3000
 ----------------------------------
          UI: http://localhost:3001
 UI External: http://localhost:3001
 ----------------------------------
[Browsersync] Serving files from: ./src
[Browsersync] Serving files from: ./build/contracts
[Browsersync] Watching files...
24.02.14 13:51:18 200 GET /index.html
24.02.14 13:51:24 200 GET /app.js
24.02.14 13:51:24 200 GET /services.css
24.02.14 13:51:24 200 GET /images/Logo.png
24.02.14 13:51:24 200 GET /bootstrap/dist/js/bootstrap.min.js
24.02.14 13:51:25 304 GET /truffle-contract/dist/truffle-contract.js
24.02.14 13:51:25 304 GET /web3/dist/web3.js
24.02.14 13:51:26 304 GET /images/image5.jpg
24.02.14 13:51:27 200 GET /fonts/Zector.ttf
24.02.14 13:51:33 404 GET /favicon.ico
```

Fig 6.3 Browser Sync Connection

Components:

a. **Metamask:** Installed as a browser extension, Metamask acts as a cryptocurrency wallet and facilitates interactions with Ethereum-based applications. It enables secure transactions and interactions with the blockchain network.

b. **Browser Sync:** Used to connect the terminal to the browser, Browser Sync ensures seamless communication between the project environment and web-based applications, enhancing the user experience and facilitating testing and development.

c. **Oracle VM VirtualBox:** Utilized to create a virtualized environment for running Debian Linux. It enables isolation and flexibility in managing the project's infrastructure, ensuring compatibility and resource efficiency.

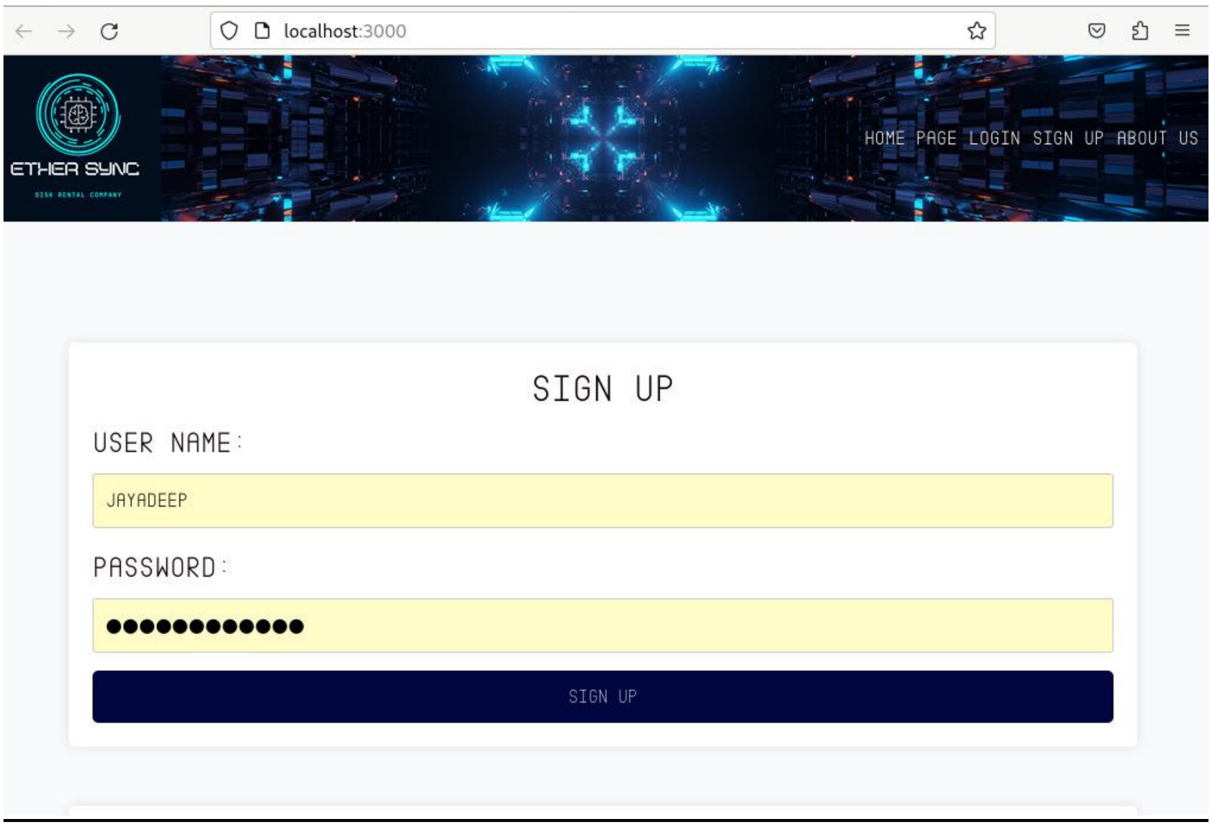
d. **Ganache:** Employed as a test Ethereum network, Ganache provides a local blockchain environment for development and testing purposes. It simulates Ethereum network behavior, allowing for rapid prototyping and debugging of smart contracts and decentralized applications.

Configuration:

Ensure that Metamask is properly configured and connected to the Ganache network to facilitate interactions with Ethereum smart contracts.

Configure Browser Sync to establish a connection between the Debian Linux terminal and the browser for seamless integration and testing.

Configure Oracle VM VirtualBox to ensure that both nodes are private and under the same network, facilitating communication and collaboration between the decentralized storage nodes.



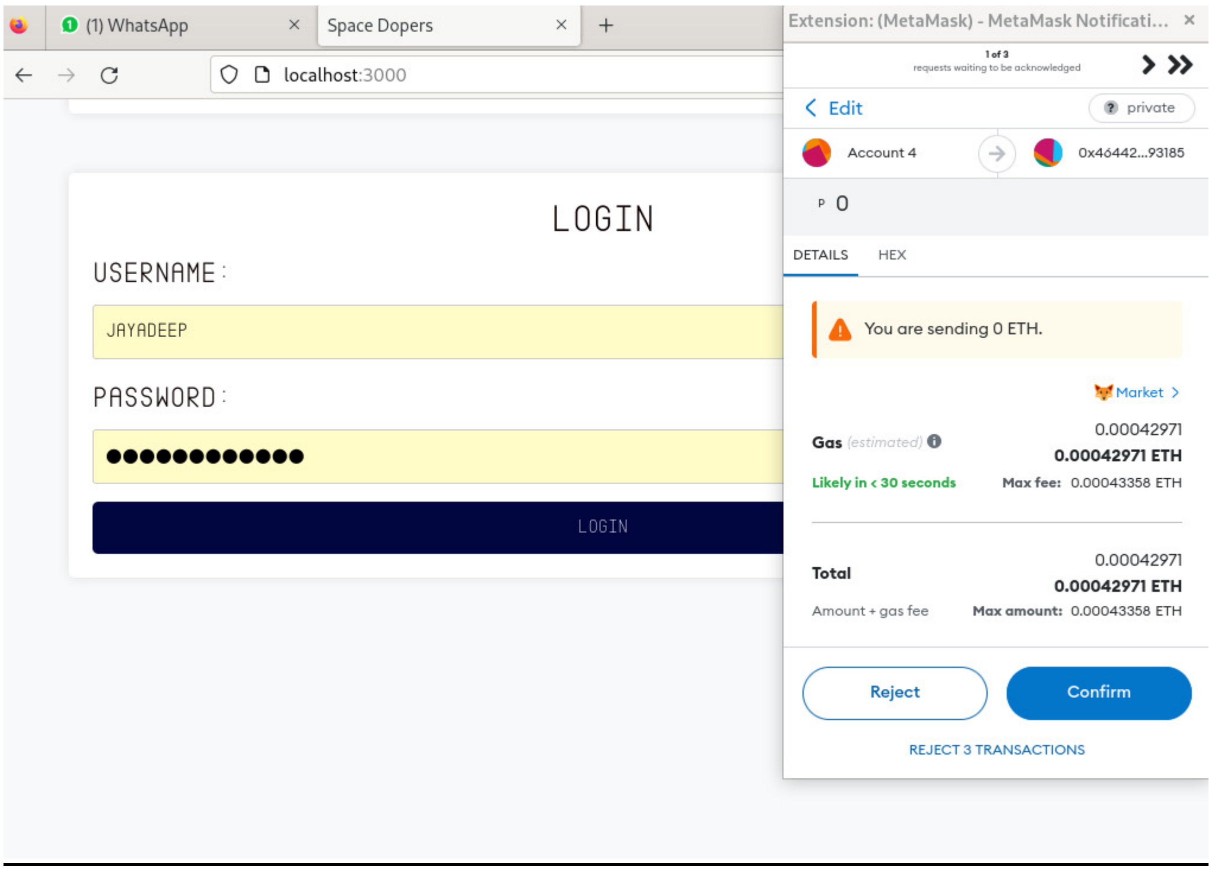
USER NAME:

JAYADEEP

PASSWORD:

●●●●●●●●●●●●●

Fig 6.4 Sign up Page


(1) WhatsApp | Space Dopers

localhost:3000

LOGIN

USERNAME :

JAYADEEP

PASSWORD :

●●●●●●●●●●●●●

LOGIN

Extension: (MetaMask) - MetaMask Notificati...

1 of 3
requests waiting to be acknowledged

Edit | private

Account 4 → 0x46442...93185

0

DETAILS | HEX

You are sending 0 ETH.

Market

Gas (estimated) 0.00042971
0.00042971 ETH
Likely in < 30 seconds | Max fee: 0.00043358 ETH

Total 0.00042971
0.00042971 ETH
Amount + gas fee | Max amount: 0.00043358 ETH

Reject | Confirm

REJECT 3 TRANSACTIONS

Fig 6.5 Login Page

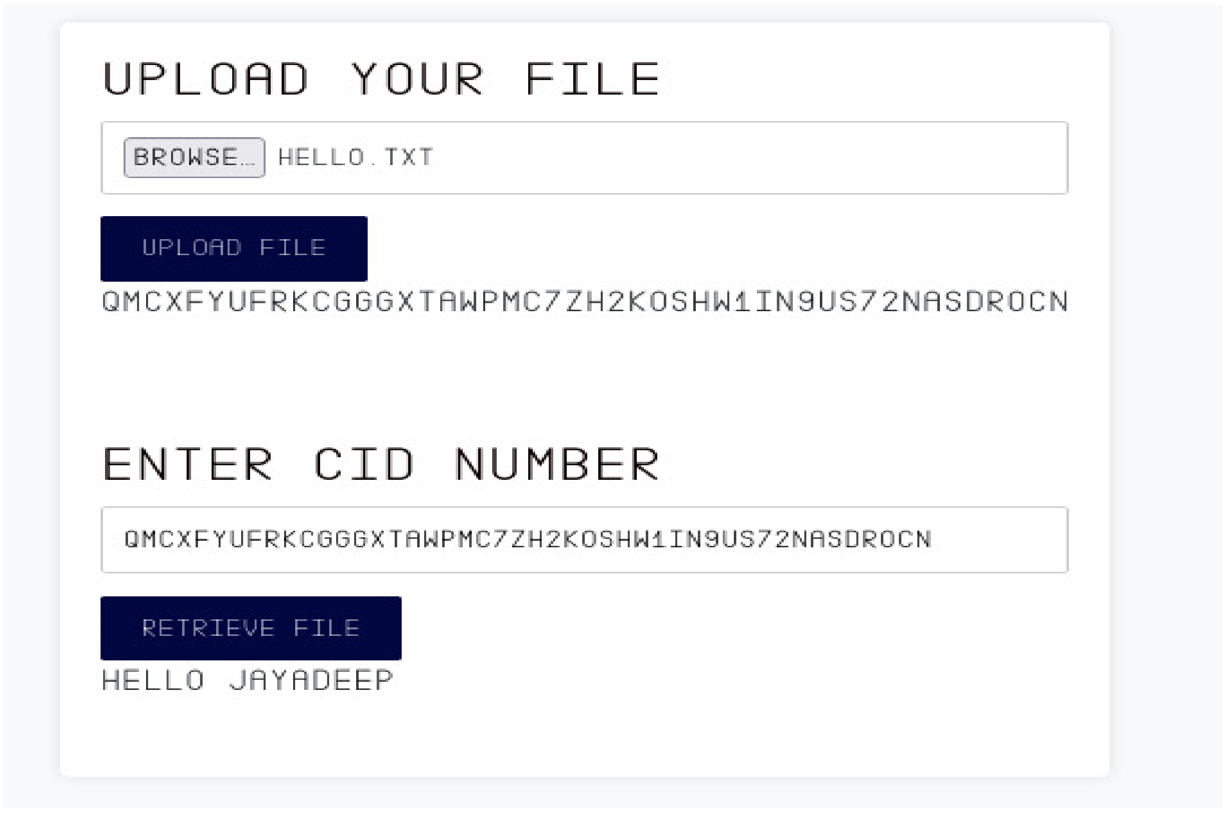
Significance of Results:

The output/results obtained from running the program on this configured platform are significant as they demonstrate the feasibility and effectiveness of the decentralized file storage solution proposed in the project.

By leveraging InterPlanetary File System (IPFS) and blockchain technology, the project aims to empower security and transparency in file storage, addressing common challenges such as data privacy, integrity, and accessibility.

The results obtained validate the functionality and performance of the decentralized file storage system, showcasing its ability to securely store and retrieve files while ensuring transparency and accountability through blockchain-based mechanisms.

Insights gained from analyzing the output/results contribute to further refining and optimizing the decentralized file storage solution, driving innovation and advancement in the field of decentralized technologies and secure data management


UPLOAD YOUR FILE

BROWSE... HELLO.TXT

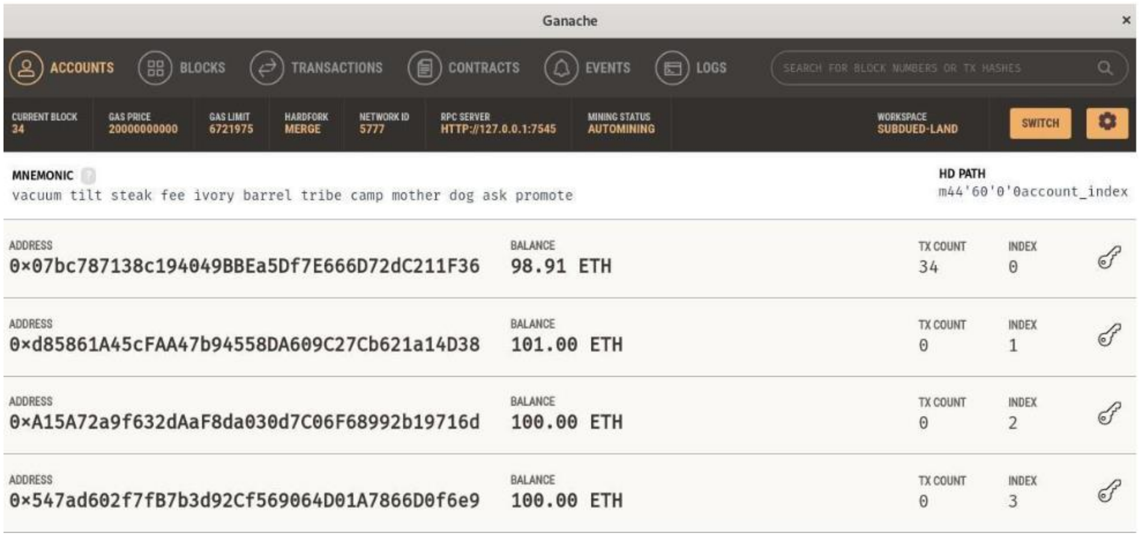
QMCXFYUFRKCGGGXTAWPMC7ZH2KOSHW1IN9US72NASDROCN

ENTER CID NUMBER

QMCXFYUFRKCGGGXTAWPMC7ZH2KOSHW1IN9US72NASDROCN

HELLO JAYADEEP

Fig 6.6 Uploading File



CURRENT BLOCK	GAS PRICE	GAS LIMIT	HARDFORK	NETWORK ID	RPC SERVER	MINING STATUS	WORKSPACE
34	20000000000	6721975	MERGE	5777	HTTP://127.0.0.1:7545	AUTOMINING	SUBDUED-LAND

MNEMONIC
vacuum tilt steak fee ivory barrel tribe camp mother dog ask promote

HD PATH
m44'60'0'0account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×07bc787138c194049BBEa5Df7E666D72dC211F36	98.91 ETH	34	0
0×d85861A45cFAA47b94558DA609C27Cb621a14D38	101.00 ETH	0	1
0×A15A72a9f632dAaF8da030d7C06F68992b19716d	100.00 ETH	0	2
0×547ad602f7fB7b3d92Cf569064D01A7866D0f6e9	100.00 ETH	0	3

Fig 6.7 Ganache Addresses

CHAPTER 7
CONCLUSION AND FUTURE ENHANCEMENT

Future Prospects for the Application

As the cloud space rental system continues to evolve, there are several exciting future prospects and opportunities for enhancement. These include:

Integration with Other Blockchains:

One of the key future prospects for the application is the exploration of compatibility with various other blockchain platforms. While the system currently utilizes Ethereum for its smart contract operations and cryptocurrency transactions, integrating with other blockchain networks could provide users with broader choices and enhance interoperability. By exploring compatibility with blockchains such as Binance Smart Chain, Polkadot, or Solana, the system can offer users additional options for storing and accessing digital storage space, catering to diverse preferences and requirements.

Improved Payment Options:

Another area of focus for future development is the enhancement of payment options within the cloud space rental system. While the system currently supports cryptocurrency payments using Ether, there is potential to introduce additional cryptocurrency payment methods to accommodate a wider range of user preferences. Additionally, exploring scalability solutions for layer 2 blockchain networks could help improve transaction throughput and reduce transaction fees, making cryptocurrency payments more efficient and cost-effective for users. By expanding payment options and optimizing transaction processing, the system can further enhance user experience and accessibility, fostering greater adoption and utilization of the platform.

Conclusion:

In conclusion, the future prospects for the cloud space rental system are promising, with opportunities for integration with other blockchains and improvements to payment options on the horizon. By exploring compatibility with various blockchain platforms and introducing additional cryptocurrency payment methods, the system can expand its capabilities and cater to a broader user base.

REFERENCES

[1] J. Zou et al., “Integrated blockchain and cloud computing systems: A systematic survey, solutions, and challenges,” ACM Comput. Surv., vol. 54, no. 8, pp. 1-36, 2022

[2] J. Li et al., “User collusion avoidance CP-ABE with efficient attribute revocation for cloud storage” in *IEEE Syst. J.*, vol. 12, no. 2, pp. 1767-1777, Jun. 2018

[3] S. Uthayashangar et al., 2021, “Decentralized blockchain based system for secure data storage in cloud,”.Available at: https://www.semanticscholar.org/paper/Decentralized-Blockchain-Based-System-for-Secure-in-S.Uthayashangar-T.Dhanya/57ed1f4c8b9cd4afec53154bb1d3b238ac07c72d

[4] C. Gray, Mar. 01, 2021 [Online], “Storj vs. Dropbox: Why Decentralized Storage is the future,” Bitcoin Magazine – Bitcoin News, Articles and Expert Insights.

[5] Z. Zheng et al., An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends, vol. 2017. Honolulu, HI, USA: IEEE International Congress on Big Data (BigData Congress), 2017, pp. 557-564

[6] S. Ohashi et al., “Token-based sharing control for IPFS,” IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, 2019, 2019, pp. 361-367

[7] M. Steichen et al., “Blockchain-based, decentralized access control for IPFS,” vol. 2018, pp. 1499-1506, 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada

[8] R. Kumar and R. Tripathi, “Implementation of distributed file storage and access framework using IPFS and blockchain,” Fifth International Conference on Image Information Processing (ICIIP), Shimla, India, 2019, 2019, pp. 246-251

[9] I.-V. Gingu, D. Hursan, and RebelDot Solutions, "'IPFS: Decentralized storage in a centralized world,' technical brief" [Online].

[10] B. Guidi et al., "Data Persistence in Decentralized Social Applications: The IPFS approach," 18th Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, vol. 2021. IEEE, *2021*, pp. 1-4,

[11] Q. Xu et al., "Building an Ethereum and IPFS-based decentralized social network system," IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS), Singapore, 2018, *2018*, pp. 1-6

[12] K. R. Houerbi et al., "Blockchain for ridesharing: A systematic literature review," International Conference on Smart Applications, Communications and Networking (SmartNets), Istanbul, Turkiye, vol. 2023, *2023*, pp. 1-6.

[13] M. D. Praveen et al., "Scalable Blockchain Architecture using off-chain IPFS for Marks Card Validation," Procedia Comput. *Sci.*, vol. 215, pp. 370-379, Jan. 2022

[14] S. Babu et al., "Blockchain and IPFS: A permanent fix for tracking farm produce," Rev. Intell. Artif., vol. 37, no. 2, pp. 267-274, Apr. 2023,

[15] J. Jayabalan and N. Jeyanthi, "Scalable blockchain model using off-chain IPFS storage for healthcare data security and privacy," J. Parallel Distrib. Comput., vol. 164, pp. 152-167, Jun. 2022,

[16] Ethersync, "GitHub – ethersync/ethersync: Ethersync enables real-time co-editing of local text files," GIThub.

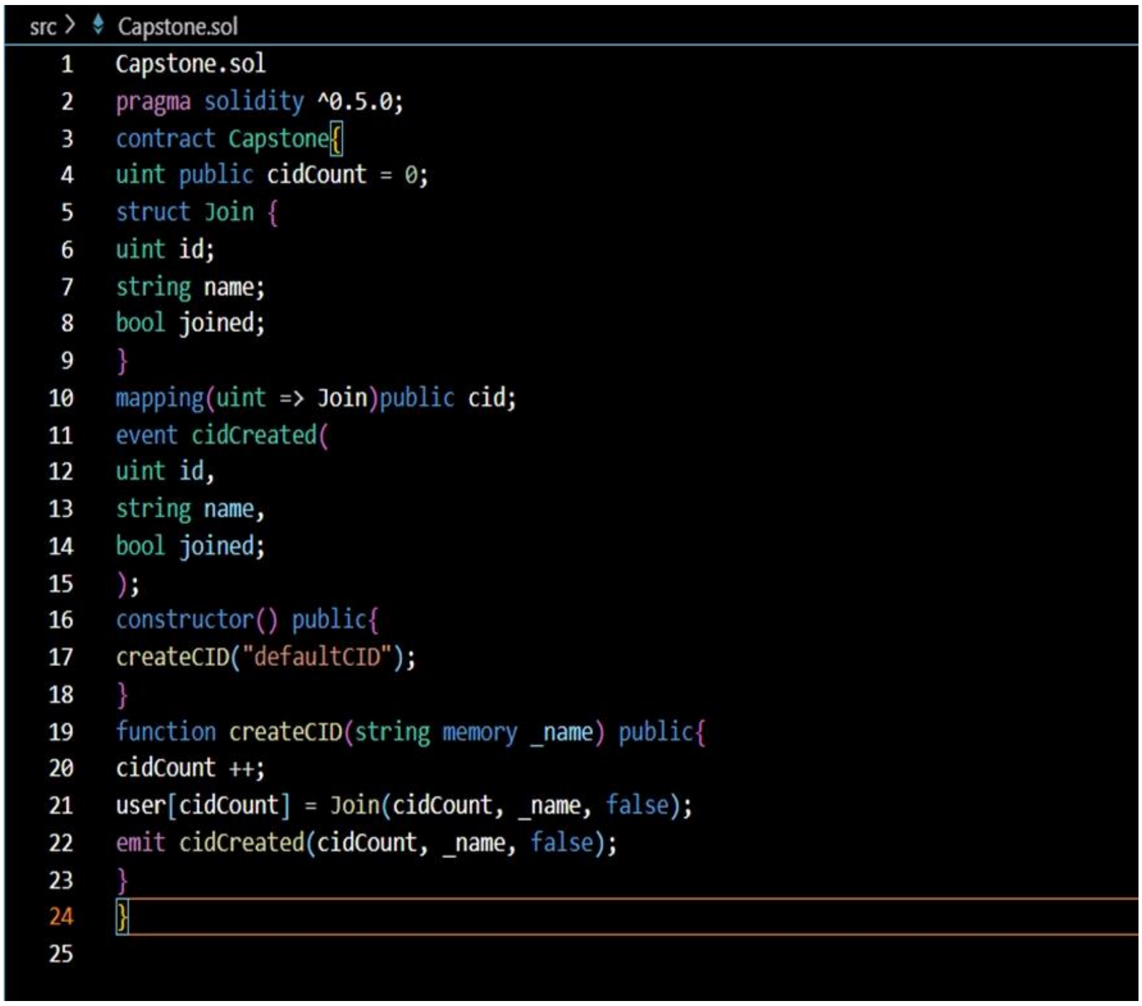
[17] R. Ramadoss, "Blockchain technology: An overview" in IEEE Potentials, vol. 41, no. 6, pp. 6-12, Nov.-Dec. 2022

[18] T. V. Doan et al., “Toward decentralized cloud storage with IPFS: Opportunities, challenges, and future considerations” in *IEEE* Internet Comput., vol. 26, no. 6, Nov. 1, pp. 7-15, Dec. 2022

[19] P. Sharma et al., “Blockchain-based decentralized architecture for cloud storage system,” J. Inf. Sec. Appl., vol. 62, p. 102970, Nov. 2021

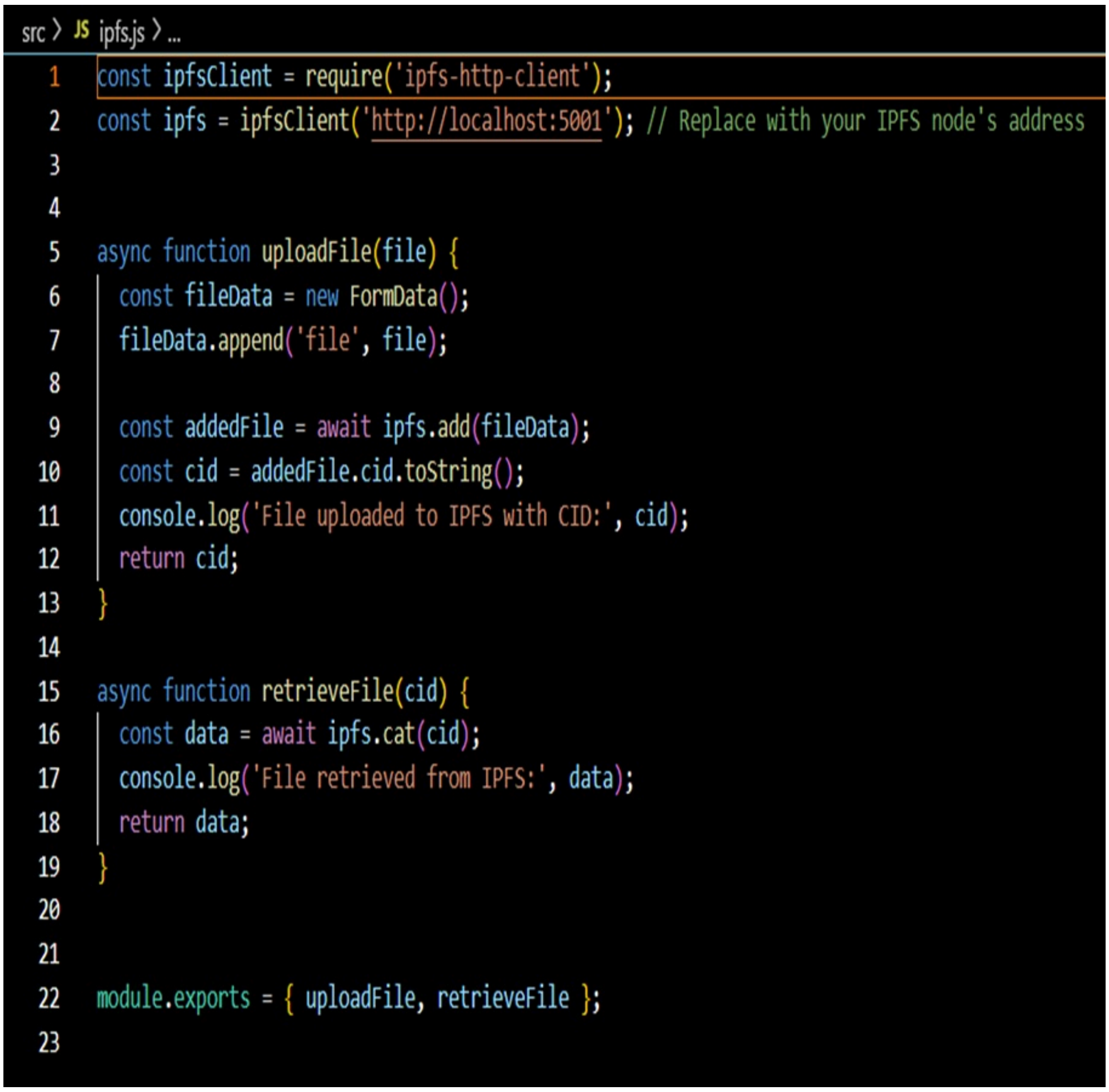
APPENDIX 1

CODE



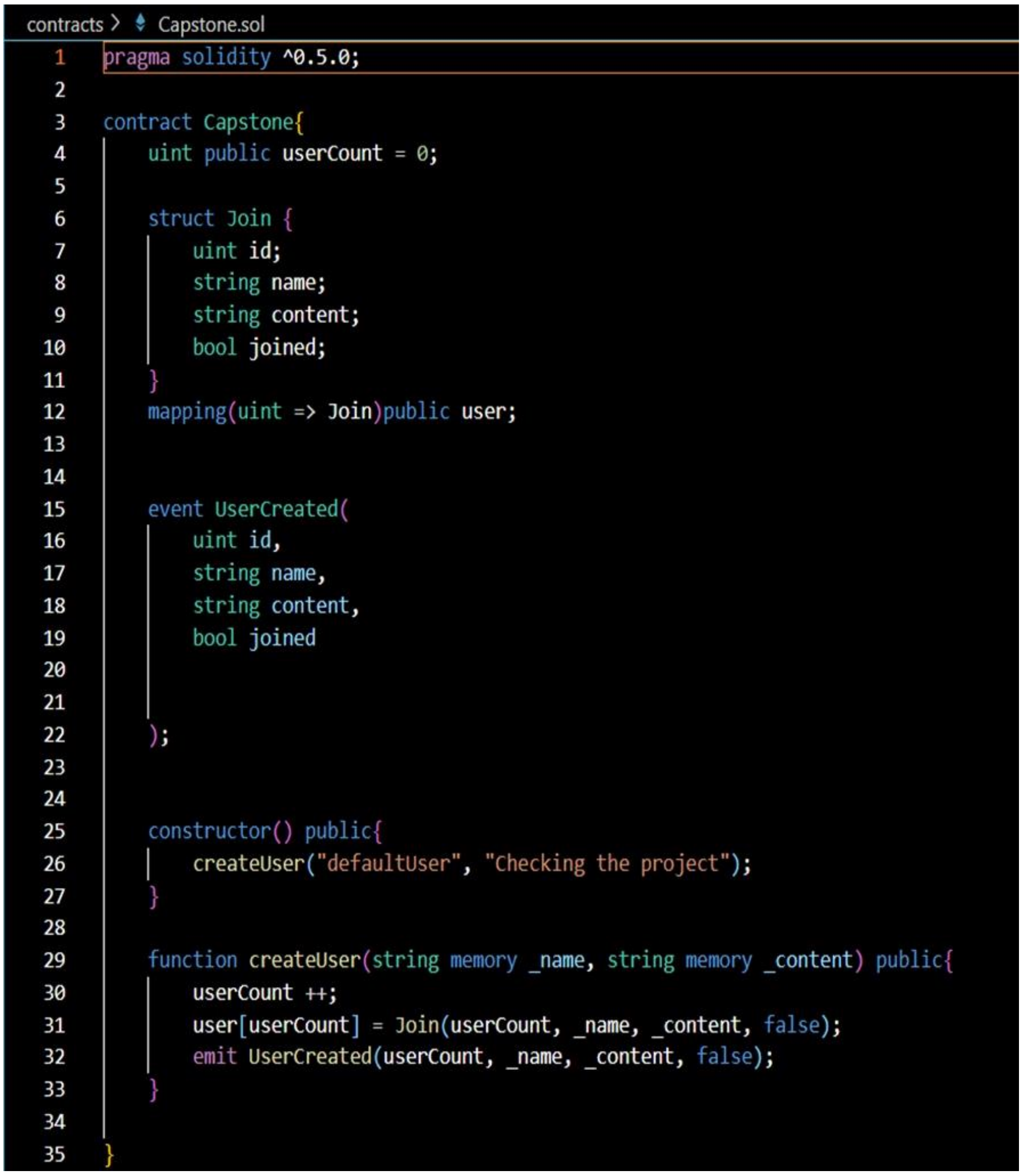
src > Capstone.sol

```
Capstone.sol
pragma solidity ^0.5.0;
contract Capstone{
uint public cidCount = 0;
struct Join {
uint id;
string name;
bool joined;
}
mapping(uint => Join)public cid;
event cidCreated(
uint id,
string name,
bool joined;
);
constructor() public{
createCID("defaultCID");
}
function createCID(string memory _name) public{
cidCount ++;
user[cidCount] = Join(cidCount, _name, false);
emit cidCreated(cidCount, _name, false);
}
}

```



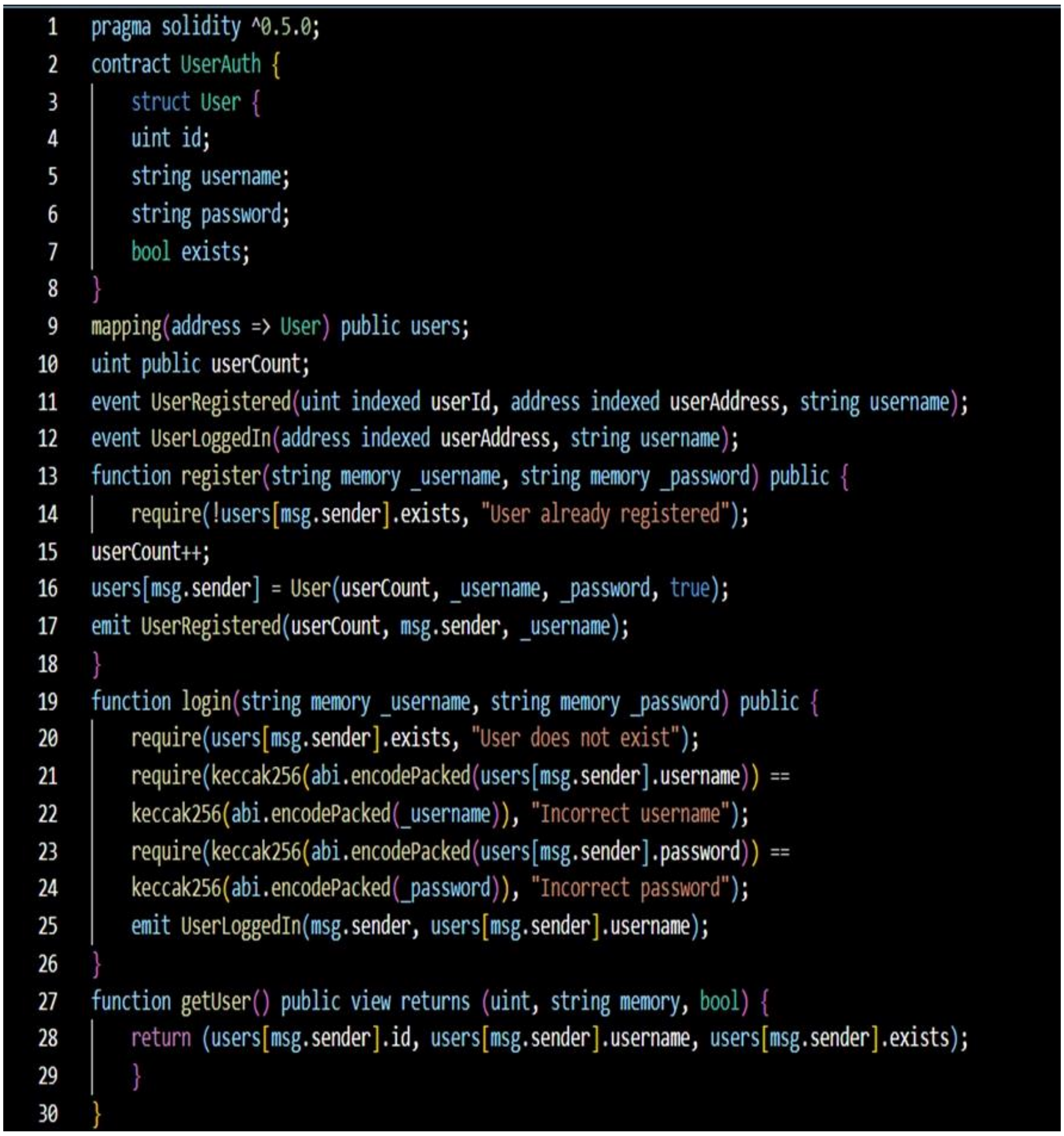
src > JS ipfs.js > ...

```javascript
const ipfsClient = require('ipfs-http-client');
const ipfs = ipfsClient('http://localhost:5001'); // Replace with your IPFS node's address


async function uploadFile(file) {
  const fileData = new FormData();
  fileData.append('file', file);

  const addedFile = await ipfs.add(fileData);
  const cid = addedFile.cid.toString();
  console.log('File uploaded to IPFS with CID:', cid);
  return cid;
}

async function retrieveFile(cid) {
  const data = await ipfs.cat(cid);
  console.log('File retrieved from IPFS:', data);
  return data;
}


module.exports = { uploadFile, retrieveFile };

```

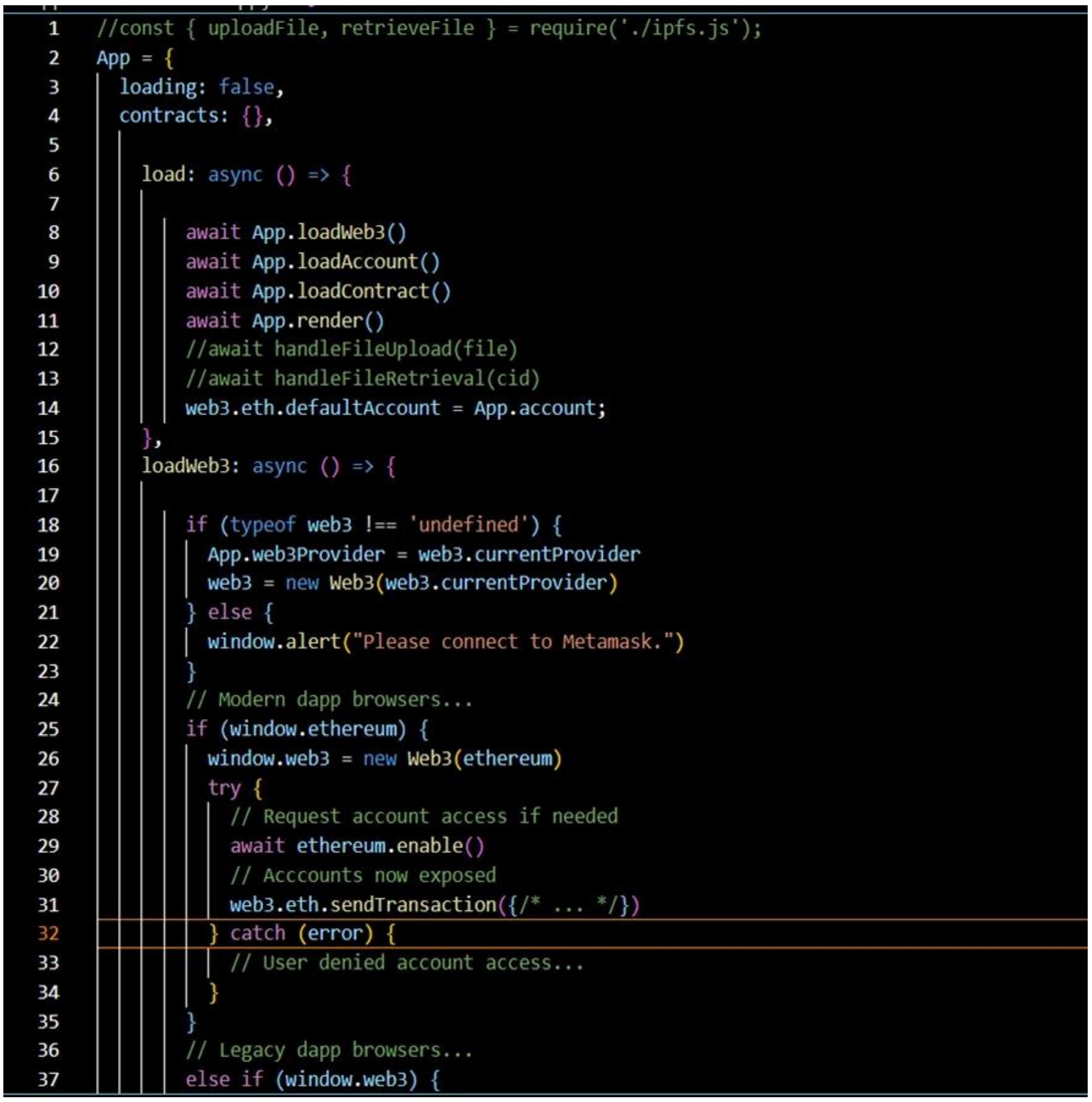


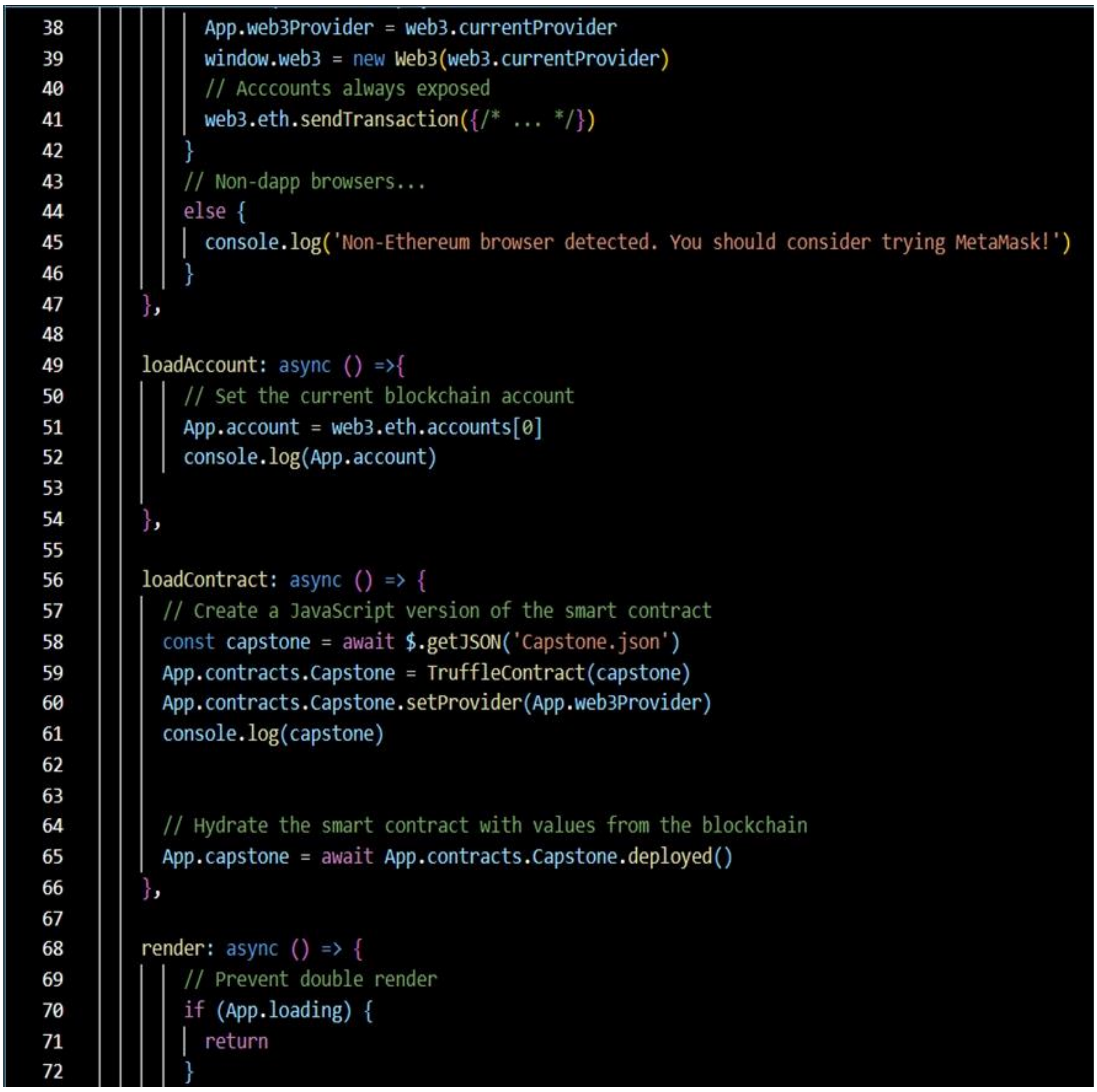
contracts > Capstone.sol

```
pragma solidity ^0.5.0;

contract Capstone{
    uint public userCount = 0;

    struct Join {
        uint id;
        string name;
        string content;
        bool joined;
    }
    mapping(uint => Join)public user;


    event UserCreated(
        uint id,
        string name,
        string content,
        bool joined


    );


    constructor() public{
        createUser("defaultUser", "Checking the project");
    }

    function createUser(string memory _name, string memory _content) public{
        userCount ++;
        user[userCount] = Join(userCount, _name, _content, false);
        emit UserCreated(userCount, _name, _content, false);
    }

}
```

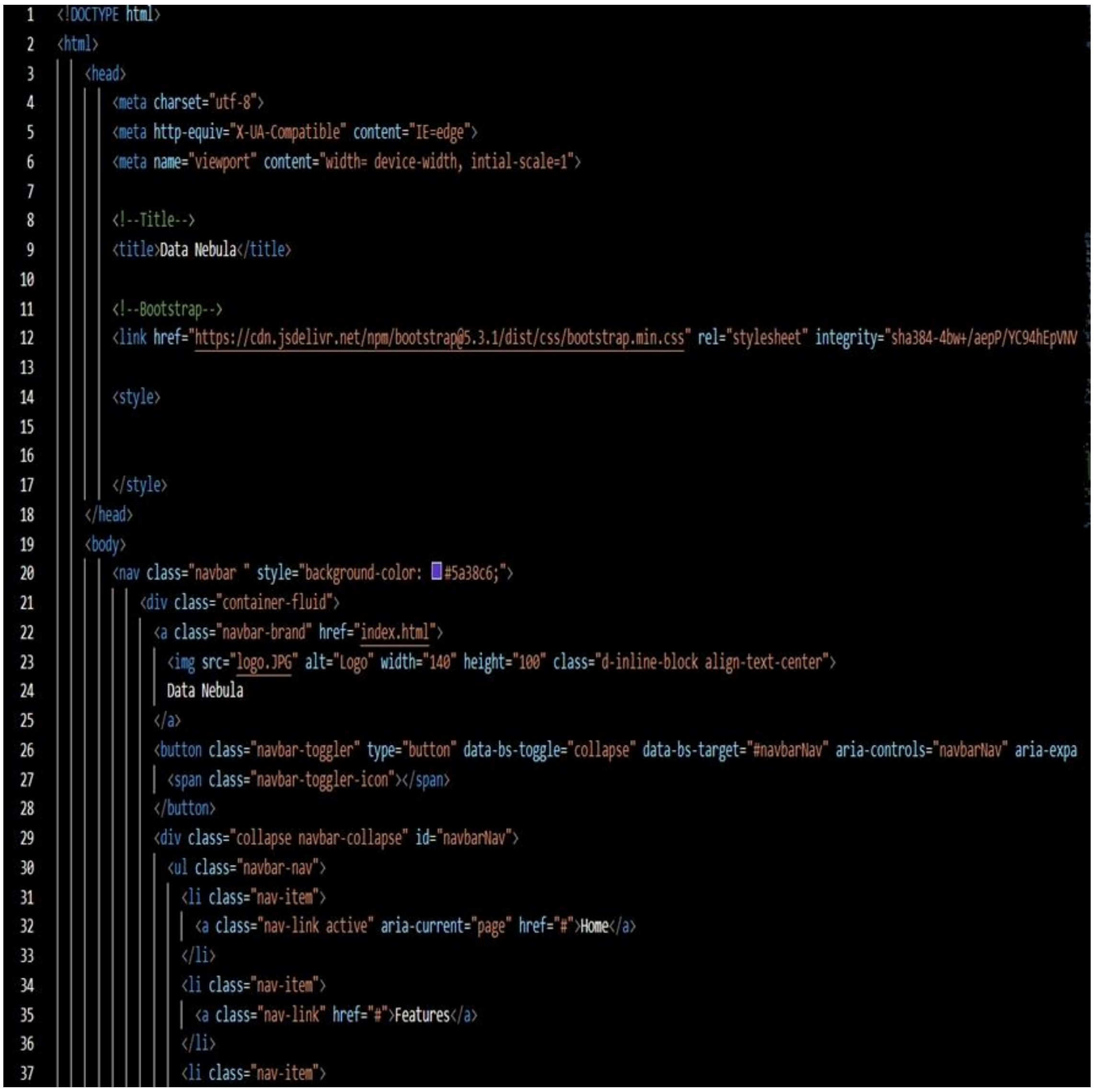


```
pragma solidity ^0.5.0;
contract UserAuth {
    struct User {
    uint id;
    string username;
    string password;
    bool exists;
}
mapping(address => User) public users;
uint public userCount;
event UserRegistered(uint indexed userId, address indexed userAddress, string username);
event UserLoggedIn(address indexed userAddress, string username);
function register(string memory _username, string memory _password) public {
    require(!users[msg.sender].exists, "User already registered");
userCount++;
users[msg.sender] = User(userCount, _username, _password, true);
emit UserRegistered(userCount, msg.sender, _username);
}
function login(string memory _username, string memory _password) public {
    require(users[msg.sender].exists, "User does not exist");
    require(keccak256(abi.encodePacked(users[msg.sender].username)) ==
    keccak256(abi.encodePacked(_username)), "Incorrect username");
    require(keccak256(abi.encodePacked(users[msg.sender].password)) ==
    keccak256(abi.encodePacked(_password)), "Incorrect password");
    emit UserLoggedIn(msg.sender, users[msg.sender].username);
}
function getUser() public view returns (uint, string memory, bool) {
    return (users[msg.sender].id, users[msg.sender].username, users[msg.sender].exists);
    }
}
```

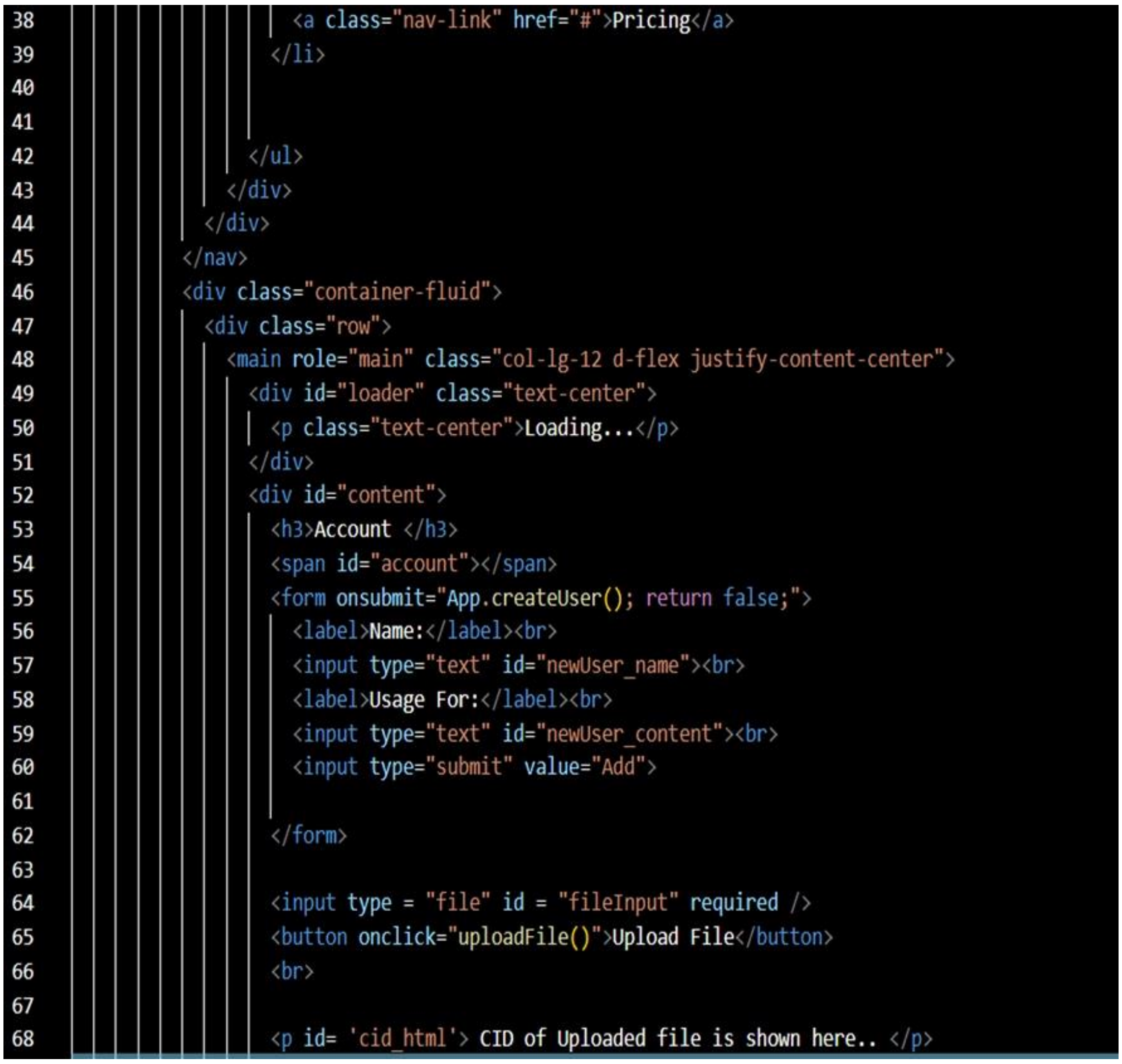

```
//const { uploadFile, retrieveFile } = require('./ipfs.js');
App = {
  loading: false,
  contracts: {},

    load: async () => {

        await App.loadWeb3()
        await App.loadAccount()
        await App.loadContract()
        await App.render()
        //await handleFileUpload(file)
        //await handleFileRetrieval(cid)
        web3.eth.defaultAccount = App.account;
    },
    loadWeb3: async () => {

        if (typeof web3 !== 'undefined') {
          App.web3Provider = web3.currentProvider
          web3 = new Web3(web3.currentProvider)
        } else {
          window.alert("Please connect to Metamask.")
        }
        // Modern dapp browsers...
        if (window.ethereum) {
          window.web3 = new Web3(ethereum)
          try {
            // Request account access if needed
            await ethereum.enable()
            // Acccounts now exposed
            web3.eth.sendTransaction({/* ... */})
          } catch (error) {
            // User denied account access...
          }
        }
        // Legacy dapp browsers...
        else if (window.web3) {
```



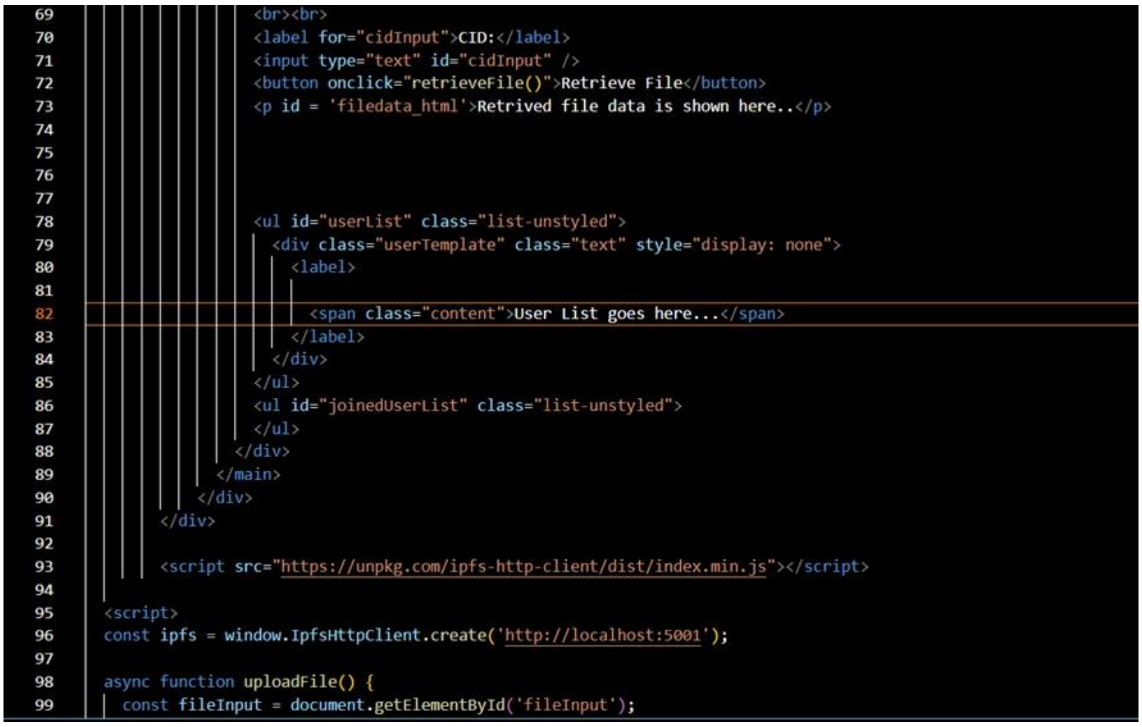
```
            App.web3Provider = web3.currentProvider
            window.web3 = new Web3(web3.currentProvider)
            // Acccounts always exposed
            web3.eth.sendTransaction({/* ... */})
          }
          // Non-dapp browsers...
          else {
            console.log('Non-Ethereum browser detected. You should consider trying MetaMask!')
          }
      },

      loadAccount: async () =>{
          // Set the current blockchain account
          App.account = web3.eth.accounts[0]
          console.log(App.account)

      },

      loadContract: async () => {
        // Create a JavaScript version of the smart contract
        const capstone = await $.getJSON('Capstone.json')
        App.contracts.Capstone = TruffleContract(capstone)
        App.contracts.Capstone.setProvider(App.web3Provider)
        console.log(capstone)


        // Hydrate the smart contract with values from the blockchain
        App.capstone = await App.contracts.Capstone.deployed()
      },

      render: async () => {
          // Prevent double render
          if (App.loading) {
            return
          }
```



```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width= device-width, intial-scale=1">

        <!--Title-->
        <title>Data Nebula</title>

        <!--Bootstrap-->
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-4bw+/aepP/YC94hEpVNV

        <style>


        </style>
    </head>
    <body>
        <nav class="navbar " style="background-color: #5a38c6;">
            <div class="container-fluid">
              <a class="navbar-brand" href="index.html">
                <img src="logo.JPG" alt="Logo" width="140" height="100" class="d-inline-block align-text-center">
                Data Nebula
              </a>
              <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expa
                <span class="navbar-toggler-icon"></span>
              </button>
              <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav">
                  <li class="nav-item">
                    <a class="nav-link active" aria-current="page" href="#">Home</a>
                  </li>
                  <li class="nav-item">
                    <a class="nav-link" href="#">Features</a>
                  </li>
                  <li class="nav-item">
```



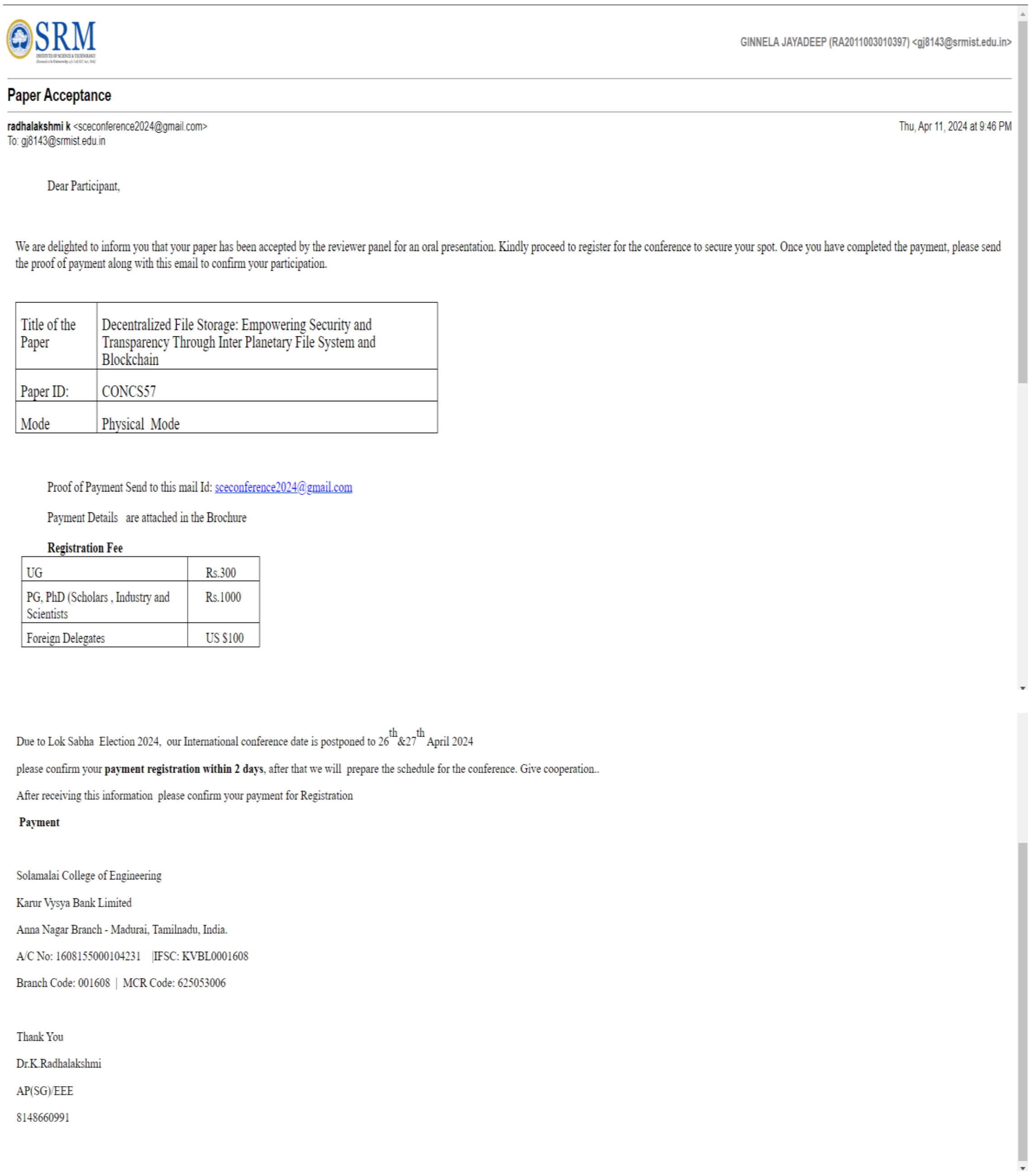
```
                    <a class="nav-link" href="#">Pricing</a>
                  </li>


                </ul>
              </div>
            </div>
          </nav>
          <div class="container-fluid">
            <div class="row">
              <main role="main" class="col-lg-12 d-flex justify-content-center">
                <div id="loader" class="text-center">
                  <p class="text-center">Loading...</p>
                </div>
                <div id="content">
                  <h3>Account </h3>
                  <span id="account"></span>
                  <form onsubmit="App.createUser(); return false;">
                    <label>Name:</label><br>
                    <input type="text" id="newUser_name"><br>
                    <label>Usage For:</label><br>
                    <input type="text" id="newUser_content"><br>
                    <input type="submit" value="Add">

                  </form>

                  <input type = "file" id = "fileInput" required />
                  <button onclick="uploadFile()">Upload File</button>
                  <br>

                  <p id= 'cid_html'> CID of Uploaded file is shown here.. </p>
```



```
                  <br><br>
                  <label for="cidInput">CID:</label>
                  <input type="text" id="cidInput" />
                  <button onclick="retrieveFile()">Retrieve File</button>
                  <p id = 'filedata_html'>Retrived file data is shown here..</p>




                  <ul id="userList" class="list-unstyled">
                    <div class="userTemplate" class="text" style="display: none">
                      <label>

                        <span class="content">User List goes here...</span>
                      </label>
                    </div>
                  </ul>
                  <ul id="joinedUserList" class="list-unstyled">
                  </ul>
                </div>
              </main>
            </div>
        </div>

        <script src="https://unpkg.com/ipfs-http-client/dist/index.min.js"></script>

  <script>
  const ipfs = window.IpfsHttpClient.create('http://localhost:5001');

  async function uploadFile() {
    const fileInput = document.getElementById('fileInput');
```

APPENDIX 2

CONFERENCE PUBLICATION



GINNELA JAYADEEP (RA2011003010397) <gj8143@srmist.edu.in>

Paper Acceptance

radhalakshmi k <sceconference2024@gmail.com>
To: gj8143@srmist.edu.in

Thu, Apr 11, 2024 at 9:46 PM

Dear Participant,

We are delighted to inform you that your paper has been accepted by the reviewer panel for an oral presentation. Kindly proceed to register for the conference to secure your spot. Once you have completed the payment, please send the proof of payment along with this email to confirm your participation.

Title of the Paper	Decentralized File Storage: Empowering Security and Transparency Through Inter Planetary File System and Blockchain
Paper ID:	CONCS57
Mode	Physical Mode

Proof of Payment Send to this mail Id: sceconference2024@gmail.com

Payment Details are attached in the Brochure

Registration Fee

UG	Rs.300
PG, PhD (Scholars , Industry and Scientists	Rs.1000
Foreign Delegates	US $100

Due to Lok Sabha Election 2024, our International conference date is postponed to 26th &27th April 2024

please confirm your **payment registration within 2 days**, after that we will prepare the schedule for the conference. Give cooperation..

After receiving this information please confirm your payment for Registration

Payment

Solamalai College of Engineering

Karur Vysya Bank Limited

Anna Nagar Branch - Madurai, Tamilnadu, India.

A/C No: 1608155000104231 |IFSC: KVBL0001608

Branch Code: 001608 | MCR Code: 625053006

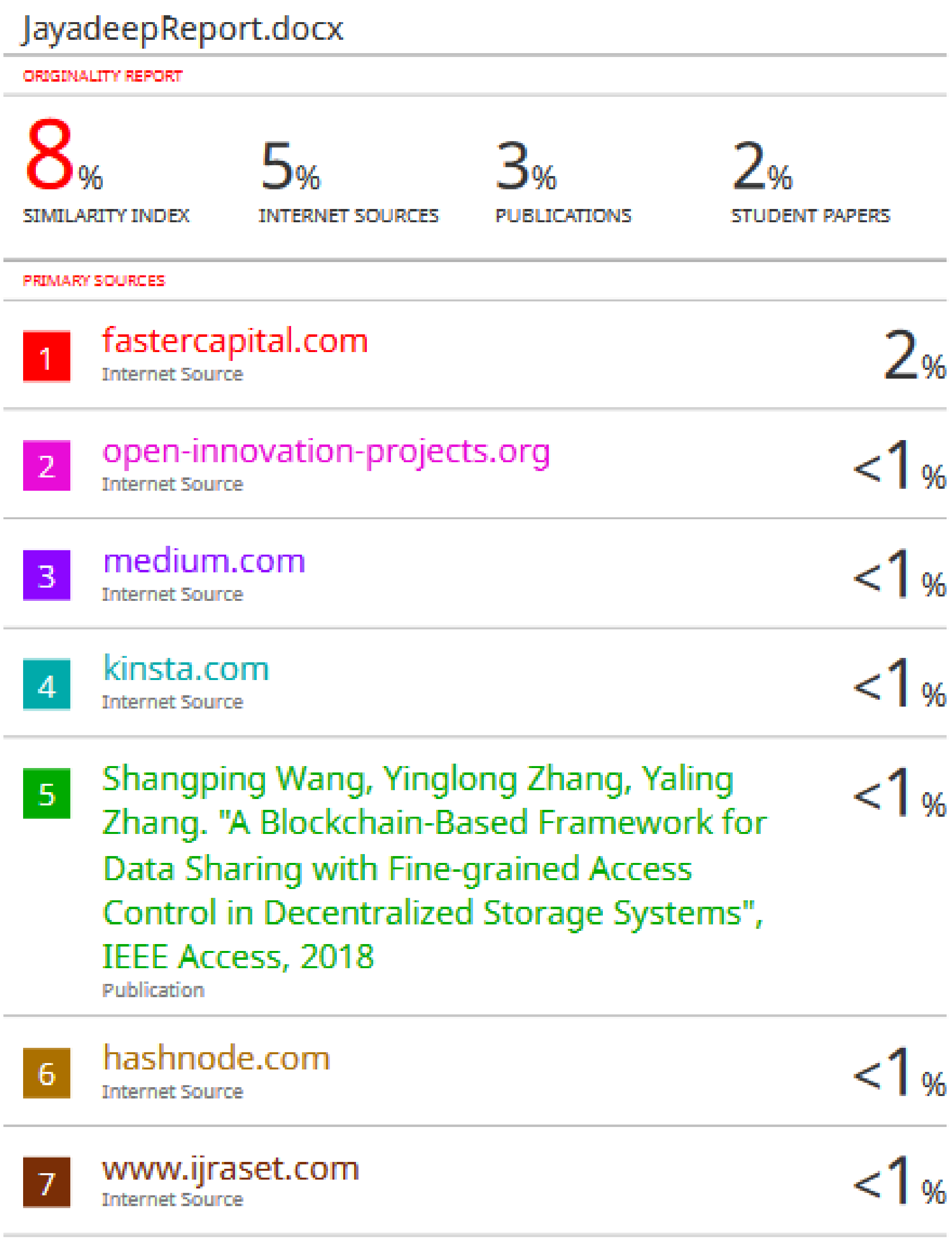
Thank You

Dr.K.Radhalakshmi

AP(SG)/EEE

8148660991

PLAGIARISM REPORT



JayadeepReport.docx

ORIGINALITY REPORT

8%	5%	3%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	fastercapital.com Internet Source	2%
2	open-innovation-projects.org Internet Source	<1%
3	medium.com Internet Source	<1%
4	kinsta.com Internet Source	<1%
5	Shangping Wang, Yinglong Zhang, Yaling Zhang. "A Blockchain-Based Framework for Data Sharing with Fine-grained Access Control in Decentralized Storage Systems", IEEE Access, 2018 Publication	<1%
6	hashnode.com Internet Source	<1%
7	www.ijraset.com Internet Source	<1%