

Prototype of Implementation of Smart Contract for Blockchain-Based Document Storage

Annes Maria Pangidoan ^{1*}, Putu Wira Buana ^{2*}, Fajar Purnama ^{3*}

* Teknologi Informasi, Universitas Udayana

Annessimanjuntakk@gmail.com ¹, wbhuanan@it.unud.ac.id ², fajarpurnama@unud.ac.id ³

Article Info

Article history:

Received 2025-04-24

Revised 2025-07-03

Accepted 2025-07-04

Keyword:

Blockchain,
Smart Contract,
IPFS,
Document Storage.

ABSTRACT

Data, including digital and physical documents, is a valuable asset often vulnerable to forgery, theft, and reliance on centralized servers, which are costly and prone to failure. This study develops a prototype of a decentralized document storage application by combining blockchain and the InterPlanetary File System (IPFS). The system is designed as a web-based decentralized application (DApp), integrating Ethereum smart contracts to immutably record document metadata and access history, while the actual files are stored in IPFS and identified using unique Content Identifiers (CIDs). User interactions are facilitated through MetaMask for authentication and transaction approval. The system is developed using the Waterfall methodology. Functional testing is conducted through unit tests using Ganache as a local Ethereum blockchain, and the smart contract is also deployed to the Sepolia Ethereum testnet. The results show that the system successfully stores documents via IPFS and records metadata and access activities transparently on the blockchain. Access and download tracking features enhance document accountability. This solution provides a secure, efficient, and transparent alternative to centralized document storage and contributes to the advancement of distributed digital archiving systems.



This is an open access article under the [CC-BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.

I. INTRODUCTION

In the digital era, data storage and management have become an important part of various aspects of daily life. Proper data storage is crucial, as data holds essential and critical value that must be maintained to prevent misuse by unauthorised parties. One significant category of data is document storage, such as work-related documents, financial records, personal data, and other essential archives.

Document storage can be implemented in various ways, ranging from traditional methods such as local storage and physical file archives to centralised storage that relies on cloud providers and private servers. However, these storage methods come with several challenges and limitations, including vulnerability to data leaks, server failures, data manipulation, misuse, and risks associated with natural disasters. The most basic challenge is data storage which still uses a centralized system [1], where users rely on third parties that have control over the user's data, increasing the risks.

Given these challenges, transitioning from centralized and traditional document storage to a distributed system is essential to mitigate vulnerabilities such as cyberattacks, data leak, and server failures. This study successfully implemented the use of a decentralized application (DApp) that integrates IPFS and blockchain through smart contracts. This approach leverages the blockchain's strengths in decentralization, transparency, and security to manage and store documents without reliance on centralized providers system that is shared across the entire blockchain network [2], to enable document storage and management without reliance on centralized storage providers. Unlike traditional cloud storage, where users depend on third-party services prone to centralized failure and unauthorized access, IPFS offers a peer-to-peer content-addressable storage mechanism that enhances data integrity and decentralization by identifying files through unique cryptographic hashes (CIDs).

Smart contracts have the property of being able to execute contracts directly without involving a third party or the server

that opens them [3]. This eliminates the dependency on centralized authorities, thereby improving the trustworthiness and autonomy of document workflows in the DApp, and also self-sufficient executing functions autonomously without server maintenance. Once deployed, the smart contract will remain active on the blockchain and utilize tokens to manage resources efficiently [4]. A smart contract on the Ethereum blockchain, written in the Solidity programming language, consists of codes that regulate business logic that can be executed automatically based on predetermined conditions running on a blockchain network [6].

The structure of a blockchain consists of a series of interconnected "blocks" forming a "chain", where each block contains a collection of transactions. Each transaction is represented by a hash, which serves as a unique identifier and ensures the integrity of the stored data [7]. Since blockchain operates as a distributed system, every participating node maintains a copy of the entire blockchain [8]. This enhances the reliability of document access because data is stored and distributed across all nodes, eliminating reliance on a single central server. Transparency is ensured because all transactions are recorded on the blockchain are immutable and can be accessed by public [9]. Data security is increased as each transaction is encrypted and stored across all nodes in the form of cryptographic hashes, making it difficult to manipulate.

IPFS is a decentralized file storage protocol where documents are represented by a unique hash according to the content of the file [10]. Each piece of content stored in IPFS has a content identifier, or CID (Content Identification) [11]. Each document has a different hash because any modification to the document results in a new CID. This ensures data integrity, as any unauthorized changes can be detected.

This decentralized application (DApp) integrates IPFS for document storage, where each uploaded document generates a CID that serves as a unique representation, ensuring document integrity. Metadata and document activity logs are stored on the Ethereum blockchain. By storing metadata on the blockchain, the data remains immutable and resistant to unauthorized modifications, as it is distributed across multiple nodes. A blockchain is composed of cryptographic algorithms, timestamp data, and transaction information, all of which serve to connect each block to its predecessor [12], enhancing transparency and security. Metadata storage and document activity tracking are managed via smart contracts, where predefined functions execute automatically upon user interaction.

Previous studies have implemented smart contracts for document storage, such as the research titled "Design and Development of Distributed Storage Using Blockchain with the InterPlanetary File System (IPFS) Protocol" by Muhammad Fadhil Abrar Lasawedi [13], which focused on the development of a decentralized storage architecture combining IPFS and blockchain. However, this research only creates an integrated storage system with IPFS and blockchain, there is no document traceability feature.

The difference is, this study not only implements decentralized application (DApp) that integrates IPFS and Ethereum smart contracts also a real-time document activity logging through smart contract events, and decentralized metadata storage.

By building a document storage system using the IPFS protocol can enhance document integrity, as each file is represented by a unique Content Identifier (CID) derived from the file's content. In addition, storing metadata in a distributed manner on the blockchain through smart contracts can improve the authenticity, transparency, and security of document management.

II. PROPOSED METHOD

Implementation process, the system uses React as the front-end framework, Ganache as the local Ethereum blockchain network, and Truffle as the smart contract development framework. The programming languages used include Solidity for developing smart contracts, as well as HTML, CSS, and JavaScript for building the user interface. To connect the interface with the blockchain network, Metamask is used as the user's digital wallet. In addition, documents are stored using the IPFS protocol through the available API. The methodology used for the implementation of the smart contract in document storage is illustrated in Figure 1.

A. System Development Waterfall

The system development in this research adopted the waterfall method. The system development was built as a prototype developed in the laboratory to find out how smart contracts work to store document metadata in a system integrated with IPFS. Based on the consideration that blockchain records each stage of the process accurately. In a project that applies the waterfall method, where the process flows sequentially from one phase to the next, blockchain plays an important role in storing historical data that cannot be changed or audited [15]. Therefore, the Waterfall method was chosen in this project to ensure that application development runs in a structured and systematic.

The waterfall method is used so that the development of the system becomes more systematic. Waterfall method development is carried out linear flow in one direction, "down" like a waterfall from the communication process to the distribution stage, with several stages [16].

Figure 1 the first stages used in the system development process of this research begin with a system requirements analysis, conducted through literature studies of previous studies that have implemented smart contracts and the Interplanetary File System (IPFS) with blockchain technology. The results of the literature study are used as a reference for developing existing features. It was found that previous systems can be further developed by adding document management features such as access and download

logging. Thus, the resulting system is not only able to store documents, but also allows users to manage their documents.

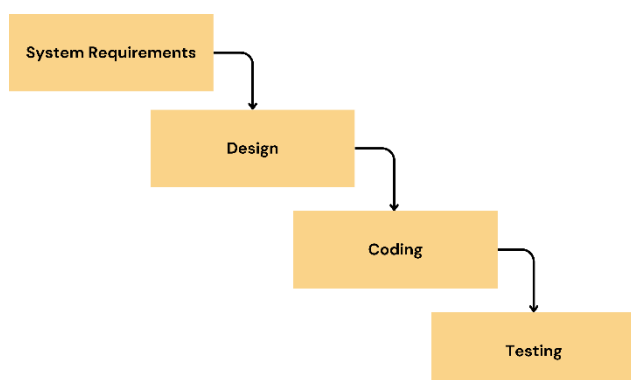


Figure 1. Waterfall System Development

The system design was carried out based on the requirements that had been previously analysed. This stage includes modelling and structuring the features to be implemented, utilising programming languages and supporting tools. Several diagrams were used in the design process, including flowcharts, use case diagrams, system overviews, and the structure of the blockchain employed.

The implementation stage is the realisation of the results of the system design into the form of program code. In the development of this system, the coding stages include, smart contract built using the Solidity programming language, the interface (frontend) is developed with the React framework using JavaScript, HTML, and CSS, and document storage is carried out through system integration with the IPFS protocol. All components are integrated into the local blockchain network through integration with the Metamask platform.

After the system was fully developed, testing was conducted to ensure that it functions properly and meets the specified requirements. The testing process involved unit testing of the smart contract. This testing aims to evaluate the smart contract functions feasibility before it is widely used by end users.

B. General System Overview

The overview aims to provide information on the outline of how the system works as a whole, to provide understanding and information related to the system to be created. The overview of this plan provides an explanation of how the relationship between blockchain, Metamask, and IPFS works on DApp with users.

Decentralized Application (DApp) is a system that has the ability to implement blockchain in the system to support business processes that can be executed in a decentralized manner, then run and audited by all nodes in the system, and run through smart contracts [17].

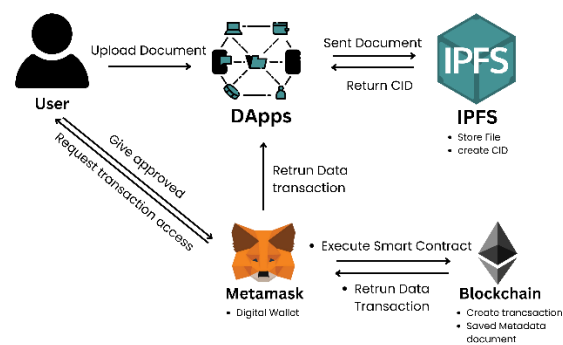


Figure 2. General System Overview

The DApp shown in figure 2 above is a platform for executing smart contracts and as a document management system. When a smart contract is executed, verification is required Metamask allows users to using and manage Ethereum accounts, send and receive ETH, and interact with DApps and smart contract [18]. Metamask account serves as both the user's identity and digital wallet within the system. The smart contract implementation in the DApp is written using the Solidity programming language and deployed on a local blockchain network using Ganache as a test blockchain, which changes the mining and speed and transaction fees on blockchain network transactions [24].

The system workflow begins when a user uploads a file through the DApp web interface. The file is then stored on IPFS using the API. IPFS processes the file and generates a Content Identifier (CID), which is a hash representing the document. The CID along with the document metadata can then be stored on the blockchain through the DApp, in accordance with the rules defined in the smart contract, after the transaction is approved through Metamask. All activities related to the document are permanently recorded on the blockchain. The stored data can later be accessed by the user through the DApp by authorizing the transaction via Metamask.

The implementation stage, the system architecture adopts a partially decentralized approach tailored to development and testing requirements. For document storage, the system utilizes the InterPlanetary File System (IPFS) protocol through the Pinata API service. Consequently, IPFS nodes are not independently hosted by the developer but instead rely on third-party infrastructure for file pinning to ensure availability within the IPFS network. Uploaded files can be accessed globally via public gateways such as <https://ipfs.io/ipfs/{CID}>, meaning that even without self-hosting, the files are stored within a peer-to-peer distributed network.

On the blockchain side, the smart contract—developed using the Solidity language—has been successfully deployed to the Ethereum Sepolia testnet. However, within the DApp implementation, all transactions involving the smart contract—such as document uploads, access logging, download logging, and metadata deletion—are still executed

locally using Ganache. In other words, these transactions are not yet processed or permanently recorded on the public Ethereum network; they are only simulated for system testing purposes.

Given these conditions, the decentralization claim in this system applies within a limited scope. On the storage side, the system implements a distributed data mechanism via IPFS with open access. However, on the blockchain transaction side, decentralization is currently limited to deploying the smart contract to a public testnet, without fully executing on-chain operations. Therefore, the system lays a foundational architecture that supports decentralization and transparency, albeit in a partially implemented form.

C. Flowchart Process

The system process flowchart represents the workflow that occurs within the system. In this system, there are four main processes: uploading documents to the system, uploading metadata to the blockchain, recording and accessing downloads, and retrieving data from the blockchain. The following illustrates the business process used in this study.

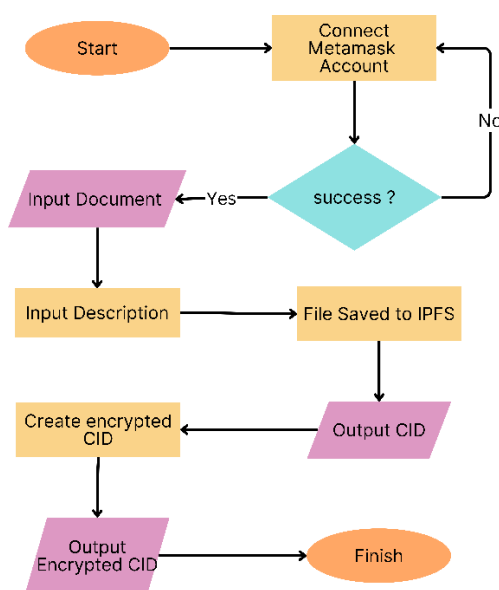


Figure 3. Upload Document

Figure 3 represents the document storage process on IPFS through the DApp. To connect with the system, users must first connect their Metamask wallet. If the connection is successful, users can access the system otherwise, they must retry by ensuring the correct Metamask account. Once the validation process is complete, users can upload documents through the system's web interface by selecting the file and entering a document description. Next, the user clicks the upload button, and the document is sent to IPFS. The system will then generate a CID (Content Identifier), a unique hash that represents the uploaded document. This CID is further encrypted to create a key for accessing the document through

the system-provided gateway. This encryption process uses the Advance Encryption Standard (AES) by downloading the AES library and then implementing it on the system to encrypt the CID.

Documents that have been uploaded to the system are only stored on IPFS. To store document metadata on the blockchain network, it must be uploaded to the blockchain network first by calling the upload metadata function on the smart contract.

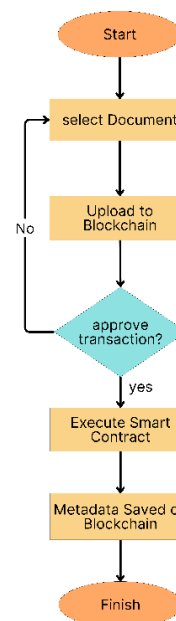


Figure 4. Upload Metadata Document to Blockchain

Figure 4 shows the flowchart of the process of uploading document metadata to the blockchain network. The user first logs in to their Metamask account so that the system can connect to the user's digital wallet. Next, the user selects the document to be stored on the blockchain network. To run this process, the smart contract will be called and continued with the transaction verification process via Metamask. If the transaction is approved, the document status will be updated and the document metadata will be successfully stored on the blockchain network. However, if the transaction is rejected, the document metadata will not be saved, and the smart contract execution will be cancelled, and the user needs to repeat the process from the beginning.

In this study, the implementation of the smart contract also involved regulating the document access process by other parties. Other users can access documents through a gateway provided in the system. All activities performed by other users on the document are automatically recorded on the blockchain network. Consequently, the document owner can still monitor who has interacted with their document.

Figure 5 illustrates the implementation of the flowchart for the process of recording access and download history of documents by other parties through the gateway provided by the system. Users must first log in using their Metamask

account. If the login process is successful, users can access the document via the CID (Content Identifier) or the encrypted CID hash of the document.

To access the document, the smart contract will be called, and the user must verify the transaction through their Metamask account. If approved, the user will be granted access to the document, and their wallet address will be recorded on the blockchain with the status of having accessed the document. In addition to accessing, users also have the option to download the document. The download process also involves the smart contract and requires transaction verification via Metamask. If approved, the download will proceed, and the user's wallet address will be recorded on the blockchain as a party who has downloaded the document.

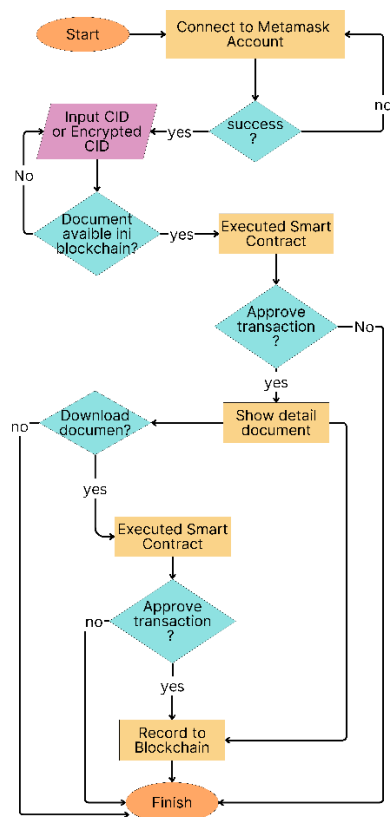


Figure 5. Record Access and Download to Blockchain

The document metadata stored on the blockchain network can be accessed and displayed within the system. In this study, the implementation allows metadata, access logs, and download logs stored on the blockchain to be automatically displayed using a smart contract when the user opens the document details. These details include the document's identity along with its access and download history, without requiring any transaction verification process.

Figure 4 illustrates the document retrieval flow by users. In this system, documents can be accessed by entering either the Content Identifier (CID) or its encrypted form. This mechanism is designed to protect the sensitivity of the CID,

preventing it from being directly visible to users, thus enhancing privacy. Every document access activity conducted through the system is automatically recorded on the blockchain, ensuring transparency and traceability. However, if the CID is shared directly with other parties, the document can be accessed via external public IPFS gateways. In such cases, access activities will not be logged by the smart contract, making them public and untraceable by the system.

Figure 6 illustrates the flowchart of the process for accessing document metadata retrieved from the blockchain network to display document details. The first step involves the user selecting the document to be opened and then accessing its details. If the Content Identifier (CID) is available on IPFS, the document will be retrieved from IPFS and displayed on the document detail page. However, if the CID is not available on IPFS, the document is considered unavailable and the process cannot proceed.

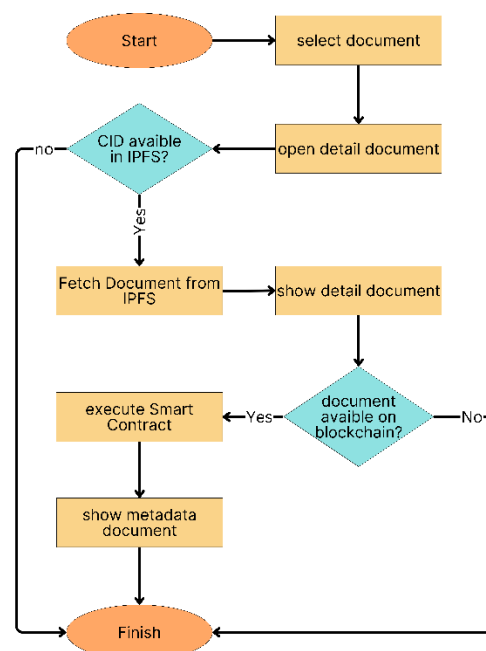


Figure 6. Record Access and Download to Blockchain

When the user opens the document detail page, the smart contract is called to check the availability of the document metadata on the blockchain network. If the metadata is stored on the blockchain, the document's access and download history will be displayed on the detail page.

D. Define Smart Contract

Developing a smart contract is the most crucial step, and what distinguishes it from traditional application development is that the triggers for operations and functionalities within the smart contract heavily rely on event

and notification systems, which are essential for ensuring efficient information transfer and process automation [6].

Each transaction on the blockchain involves some gas fees [20], including deploying a smart contract on the blockchain, the amount of gas used for a transaction is determined by the computational power required for a smart contract to execute [21]. Therefore, before the coding process begins, a smart contract design phase is necessary. This phase includes determining the functions to be implemented along with their respective parameters. The purpose of this design stage is to produce a smart contract that is efficient in gas usage and accurately meets the needs of the system. The table 1 presents the design of the DocumentTracking smart contract, which contains the main functions related to document storage and management.

TABLE I
SMART CONTRACT DESIGN

No	Contract DocumentTracking		
	Front end	Function Smart Contract	Input
Record Metadata	Upload Meta Data Document	function uploadDocument()	string memory _name, string memory _cid, uint256 _size, string memory, _mimeType, string memory _description
	Record Access	function recordAccess()	string memory _cid
	Record Download	function recordDownload()	string memory _cid
Access Metadata	Detail Access	function getAccessLog()	string memory _cid
	Detail Download	function getDownloadLog()	string memory _cid

This smart contract focuses on storing document metadata and recording access logs and downloads on documents, thereby focusing on improving document traceability. These functions are divided into two main categories: storing document metadata on the blockchain network and accessing metadata from the blockchain network.

1) uploadDocument is used to store document metadata, including the document's CID, name, size, type, and description. This function will be called when the user uploads document metadata to the blockchain network via the system interface. Pada fungsi ini juga akan melakukan pengecekan jika metadata dokumen telah tersimpan pada jaringan blockchain maka

2) recordAccess records the wallet address of users who have opened the document through the gateway, based on the accessed document's CID will record the wallet address and time access user when accessed the document. This function will be called when the user accesses the document through the document gateway. To continue the

process the user must approve the transaction on Metamask on the system and the function will be executed.

3) recordDownload records the wallet address of users who have downloaded the document, based on the downloaded document's CID will record the wallet address and time download user when downloaded the document. This function will be called when the user clicks download on the interface and approves the transaction on the system, then the contract will be called.

4) getAccessLog is used to retrieve all wallet addresses that have accessed the document based on its CID. This function will be executed when the document owner opens the document details whose metadata has been stored on the blockchain network. When opened, the function will be executed and display the access history by users who have accessed the document.

5) getDownloadLog is used to retrieve wallet addresses that have downloaded the document based on its CID. This function will be executed when the document owner opens the document details whose metadata has been stored on the blockchain network. When opened, the function will be executed and display the download history by users who have downloaded the documents.

E. Blockchain Structure

In designing the blockchain structure, a number of features are required to build a blockchain-based application, including account management, API components (digital wallets), smart contract components, and blockchain-based storage techniques. The necessary services, such as contract management, also involve defining the type of blockchain used, whether permissioned or permissionless. Every decision related to blockchain technology must take into account various relevant factors and undergo thorough review and evaluation processes[7].

The use of blockchain technology in this implementation is applied to several key features within the system, including document metadata upload, access history recording, and download history tracking by users. Figure 6 illustrates the blockchain implementation structure used in the decentralized document storage system.

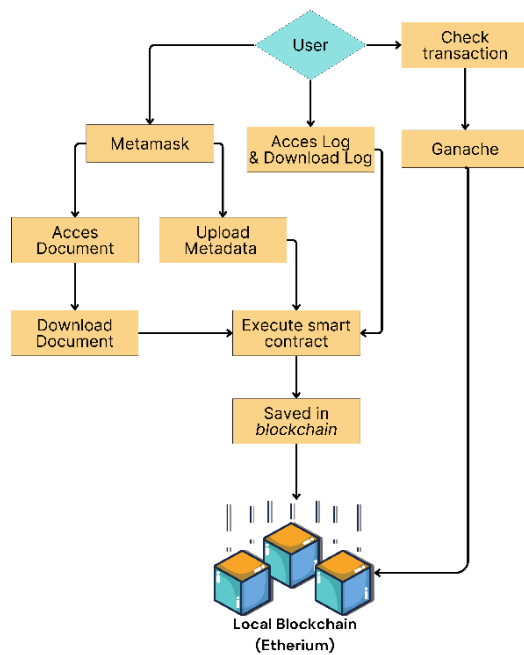


Figure 7. Blockchain Structure

The figure 7 illustrates the blockchain structure diagram designed for the document storage system. In its implementation, users access the information system through Metamask, which serves as a medium for performing all smart contract verification processes within the system. Each verification process, represented as a transaction, can be monitored using Ganache, which acts as a tool to connect the system with the local Ethereum network utilized in this implementation.

All blockchain interactions are executed through a smart contract named DocumentTracking. There are three main functions in the smart contract that require user verification before they can be executed, uploading document metadata, accessing documents, and downloading documents. Meanwhile, functions for retrieving access logs and download logs are executed automatically without requiring transaction verification. All transactions generated from smart contract interactions are permanently stored on the local blockchain network via Ganache.

F. Smart Contract Testing Design

Before the smart contract is deployed to the blockchain network, testing is required to ensure that each implemented function operates as intended. In this smart contract testing, the process was conducted using the Truffle framework. Truffle is a development framework for Ethereum-based blockchains, offering tools for smart contract creation, deployment, and testing. It supports contract verification via MetaMask, script execution, automated testing, and network management for both public and private blockchains[20]. Testing was conducted using the JavaScript programming language within the Truffle framework, which serves to

ensure that the smart contract functions correctly after being deployed to the blockchain network. The contract file was written in the Solidity programming language, and the contract to be tested must be accompanied by a migration file used by Truffle during the deployment process. The testing is performed by creating a file containing test scripts to verify the functions defined in the smart contract[4].

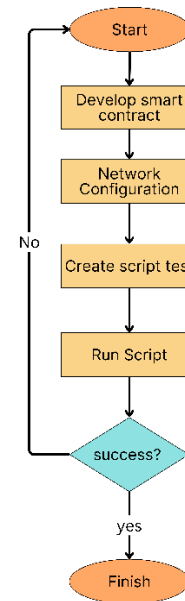


Figure 8. Smart Contract Design Testing

Figure 8 illustrates the design of the testing stages conducted on the smart contract. After the system has been fully developed, the next step involves configuring and selecting the appropriate network for testing the smart contract. This process is followed by the creation of test scripts containing a series of testing scenarios. The test scripts are then executed to evaluate whether each function in the smart contract operates as expected.

If any errors, bugs, or inconsistencies are identified, the process must return to the development phase for necessary revisions and further development of the smart contract. Once all test scripts run successfully without any issues, the testing process is considered successful and can proceed to the next stage of implementation.

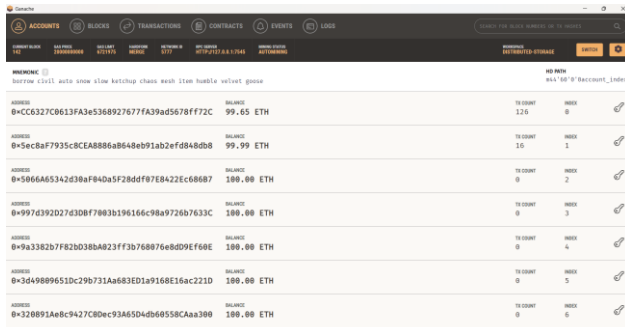
III. RESULTS AND DISCUSSION

The results of this research include the configuration of the blockchain networks used, namely Ganache as the local Ethereum network and Sepolia as the public Ethereum network. After completing the network configuration, each function in the smart contract was tested by executing scripts for each function. Subsequently, the smart contract was executed on the local blockchain network and then deployed to the public blockchain network. Once the blockchain infrastructure was ready, the smart contract was integrated

with the user interface (UI) to form a decentralized document storage system.

A. Blockchain Local Network Configuration

In this study, the need for a blockchain network is fulfilled by using a local network configured through the Ganache application, which is utilized for development purposes. Ganache provides ten default Ethereum accounts that can be used to perform various activities within the blockchain network, including the implementation and testing of smart contracts.



ACCOUNT	BALANCE	TRANSACTIONS
0xCCCC327C613FA3e5368927677FA39ad5678f72C	99.65 ETH	126
0x5ec8aF7935c8CA886a8648e9f1a02ef0848d08	99.99 ETH	10
0x5866A65342d8aF84Da5F28dF87E8A22Ec68687	100.00 ETH	0
0x997d392D27d30Bf7803b196166c98a9726b7633C	100.00 ETH	0
0x9a3382b7F823038a023fF3b768b76e8d09EF68E	100.00 ETH	0
0x3d4989651Dc29b731Aa683ED1a9168E18ac221D	100.00 ETH	0
0x320891Ae8c9427C8dec93A65D4d6b558CAa300	100.00 ETH	0

Figure 9. Blockchain Network Configuration

Figure 9 shows the interface of the Ganache application used in this study as a provider of the local blockchain network. Before being used, the blockchain network from Ganache must be configured to connect with a digital wallet within the DApp. In this study, the digital wallet used is Metamask. Metamask allows users to access and utilize Ethereum accounts that are provided by Ganache by default. Through this integration, users can perform transactions and directly interact with smart contracts on the local Ethereum blockchain network.

B. User Interface

Smart contract applications are run using a prototype system for end users. The following is the user interface used in the application of functions in smart contracts, namely uploading document metadata to the blockchain, document access records, document download records, access logs, and download logs.

Metamask is configured in the system as a bridge between users and the blockchain network. In this study, Metamask plays a crucial role in facilitating transaction processes conducted on the blockchain network. In this study, the user's Metamask account address is also used as a profile identity within the system, so that all user activities recorded on the blockchain are directly associated with the wallet address being used.

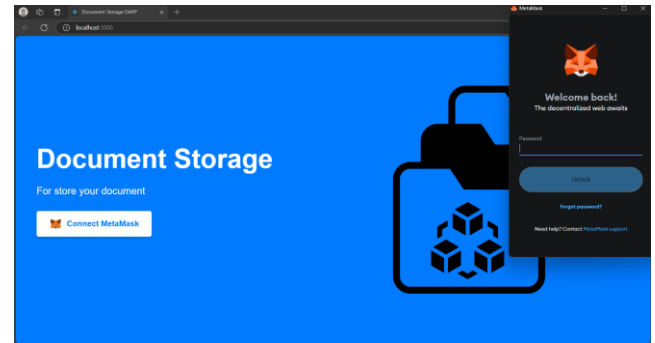


Figure 10. Log In Metamask Interface

Figure 10 shows the result of the Metamask account configuration process with the system. Access to the Metamask account is conducted through a browser extension. Every user who wishes to access the system must have a Metamask account, as it also serves as the user's primary account within the system. All activities related to the smart contract can only be executed if the system has successfully connected to the Metamask digital wallet.

Document Storage DApp

Upload or Drag Your File Here!

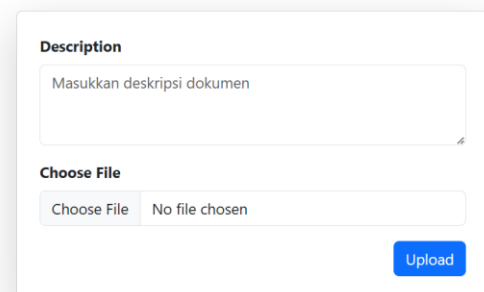


Figure 11. Upload Document Interface

Figure 11 shows the document upload page, can be accessed after the login process. On this page, users can input two fields: the document file to be uploaded and a description of the document. After filling in both inputs, users can click the upload button to initiate the storage process. The system will store the document in IPFS (InterPlanetary File System) and generate a CID (Content Identifier) retrieved from IPFS. The document's CID can be viewed on the document detail page. During this process, the system also generates an encrypted hash of the CID.

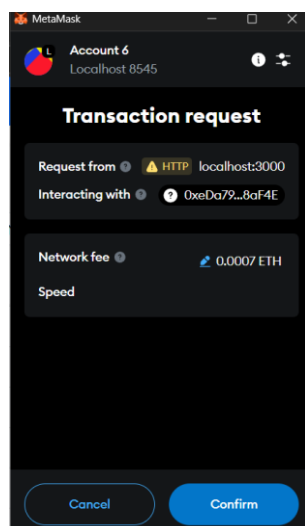


Figure 12. Upload Document Interface

Figure 12 illustrates one of the verification steps in the execution of a smart contract. In this figure, the transaction request shown is for uploading document metadata to the blockchain network. Once a document has been uploaded to the system, its metadata can be stored on the blockchain. Documents that are accessible to other users through the gateway must be available on the blockchain so that all activities related to the document can be recorded on the blockchain network. When the transaction is approved by the user in Metamask, the document metadata will be stored on the blockchain, and the document can then be accessed by other users through the gateway.

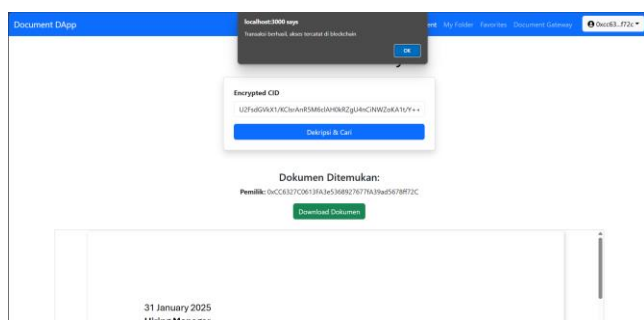


Figure 13. Document Gateway Interface

The figure 13 shows the gateway page interface within the system. On this page, other users can access a document by entering the CID (Content Identifier) or the encrypted CID hash previously generated by the system. To access the document, users are required to initiate a transaction on the blockchain network through their Metamask account.

During the access process, the recordAccess function in the smart contract will be called. If the user approves the transaction request, the wallet address from the user's Metamask account will be recorded on the blockchain as proof of document access activity. A similar process occurs when the user downloads the document, where the

recordDownload function will be called from the smart contract. If the transaction is approved, the user's wallet address will be recorded as the party who performed the download. All access and download histories can be viewed by the document owner as a form of transparency and activity tracking related to the document.

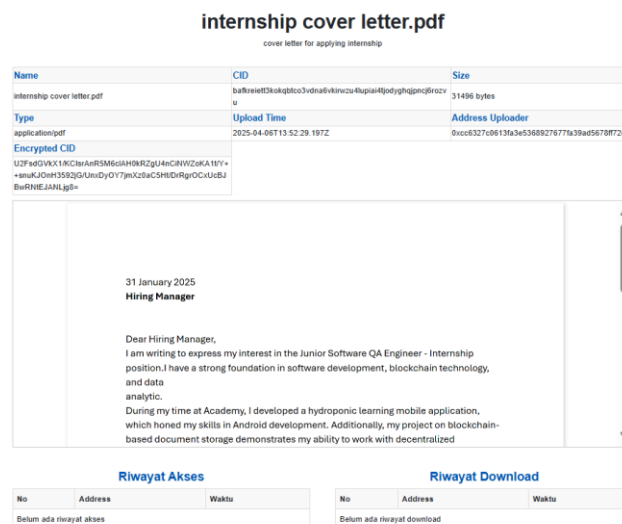


Figure 14. Document Detail Interface

Figure 14 is a interface of the document detail page that has been uploaded to the system. This page displays document metadata such as name, size, type, upload time, and CID (Content Identifier). This metadata is obtained from file information and the upload process to IPFS. The CID is generated by IPFS when the document is uploaded, while the uploader is obtained from the Metamask account address of the document uploader.

In addition, the system also generates an encryption hash from the CID, which functions to maintain data security. This CID encryption aims to protect the original CID from being accessed carelessly, considering that the CID is sensitive data that represents a document uniquely. Therefore, access to documents by other users is done through CID encryption, so that only the document owner or certain parties know the original CID.

The documents on this page are displayed through the IPFS gateway. When the detail page is accessed, the system automatically calls the getAccessLog and getDownloadLog functions from the smart contract. This function functions to display the history of users who have accessed or downloaded documents, as recorded in the blockchain network. This history will be displayed on the detail page as part of the document activity tracking feature.

C. Smart Contract Testing

The purpose of this testing is to ensure that the 'DocumentTracking' smart contract functions as intended. The contract is responsible for handling document uploads, logging document access and download activities.

TABEL II
TESTING ENVIRONMENT

Component	Version
Framework	Truffle v5.11.5
License	MIT
Solidity Version	0.8.0
Network	Ganache v7.9.1
Script	JavaScript

Table 2 describes the environment used in the development of the smart contract. The programming language used is Solidity, with the MIT license, which is open source and allows for use and modification by other parties. Testing was conducted using a local Ethereum network provided by Ganache, while the deployment and testing of the smart contract were carried out using the Truffle platform.

TABEL III
TEST SCENARIOS AND RESULTS

Smart Contract: DocumentTracking					
No	Test Case Name	Description	Gas	Result	Time / ms
1	Successful Document Upload	Ensure that a document can be uploaded with correct metadata.	184183	Passed	72
2	Record Document Access	Record user address who accessed a document.	95172	Passed	50
3	Record Document Download	Record user address who downloaded a document.	95194	Passed	67
4	Get Document Access Log	Log user address who accessed a document.	0	Passed	66
5	Get Document Download Log	Log user address who downloaded a document.	0	Passed	61

Table 3 presents the testing results for each function within the smart contract, using dummy data as input to simulate the process. The testing was conducted using scripts written in the JavaScript programming language, with five test scenarios designed for each function.

The results indicate that all functions performed successfully and completed their tasks. The process that required the most time was the document upload to the blockchain network. This is due to the higher computational resources needed to store document metadata, which is relatively larger in size compared to other operations such as logging document access, download logs, and access logs.

In terms of performance, transaction execution time is relatively short, with an average time of under 1 second for

the main functions. Gas usage is still within reasonable limits for an application with blockchain-based document storage features, with the upload function requiring the most gas due to metadata storage and event logging.

D. Smart Contract Result

The implementation of the smart contract results in a transaction every time a function is executed, with each transaction requiring a certain amount of gas. Gas is the unit of computational cost used in the Ethereum network to execute instructions within a smart contract. The more complex the function being executed, the higher the amount of gas required. The following table presents the transaction results for each function call in the DocumentTracking smart contract, including the amount of gas used and the resulting transaction hash.

TABEL IV
SMART CONTRACT RESULTS

No	Contract DocumentTracking		
	Front end	Gas	Transaction Hash
Record Metadata	Upload Meta Data Document	274335	0x100ecf3d4170900decb7211f5f473cb1734c2e499273b13fe5c403e5af740c5b
	Record Access	76059	0x700c5921dfa3fb7a787f296dc449e0e089863e02d7735ff6a709650f9592ae4e
	Record Download	93159	0x73ea2d5543f1a550fbfc2864840673f946a45ae7d3e7d7d7e00fb4569e203b8e
Access Metadata	Detail Access	0	-
	Detail Download	0	-

Table 4 shows the results of the implementation of calling functions in smart contracts in the form of transactions and the amount of gas used for their execution. The function with the largest gas usage is the upload metadata document function uploadDocument. This is because the size of the metadata data stored in the blockchain network is relatively larger than other functions. The second largest gas usage occurs in the recordDownload and getAccessLog functions, because the process of recording download history requires a higher computational load. Meanwhile, the lowest gas usage is in the recordAccess and getAccessLog functions, because the access recording process does not require complex data processing.

E. Smart Contract Deploy

The smart contract that has successfully passed the testing phase and was executed on a local Ethereum network has fulfilled all functional requirements, making it suitable for deployment on a public blockchain network. In this study, the smart contract was deployed on one of the blockchain

networks, namely the Ethereum testnet. This allows other parties to verify every transaction occurring within the smart contract globally through the interface provided by each blockchain network.

TABEL V
SMART CONTRACT DEPLOY

DocumentTracking	
Network Name	Sepolia
Network ID	11155111
Gas Used	1440492
Transaction Hash	0x5ac422357d98234990eb25a3c5d5677b075321bd8db8e1d8fadc07a2da59b22a
Contract Address	0xa61e7b91f5ddb117e64036c01a057f29f45d15bf
Transaction and contract address url	https://sepolia.etherscan.io/tx/0x5ac422357d98234990eb25a3c5d5677b075321bd8db8e1d8fadc07a2da59b22a & https://sepolia.etherscan.io/address/0xa61e7b91f5ddb117e64036c01a057f29f45d15bf#code
Total Cost	0.005585079344965104 ETH

Table 5 presents the result of deploying the smart contract to the Ethereum Sepolia blockchain network. The deployed smart contract can be publicly accessed and verified by other users via the transaction hash or contract address generated during the deployment process. This enhances the transparency of all transactions carried out within the smart contract. The result of deployment and smart contract can be accessed and verified by url above at table 5.

IV. CONCLUSION

This study aims to implement a smart contract for decentralized document storage on the blockchain network as an alternative to conventional centralized storage solutions. IPFS is utilized as the primary protocol for document storage to reduce the high costs associated with storing data directly on the blockchain. Each uploaded document generates a unique Content Identifier (CID), which represents the content and enhances the document's data integrity. The document metadata—including title, size, file type, CID, encrypted CID, upload date, and user account address—is extracted and recorded on the blockchain. Activities such as access and download are logged, thereby improving document accountability.

The smart contract developed in this study, named DocumentTracking, governs all interactions within the blockchain network. Its use ensures transparency, as all transactions can be audited and verified by other users, and guarantees immutability, since the recorded data cannot be altered due to the cryptographic linkage between blocks. Transaction security is further strengthened through unique hash values generated for each operation. Before deployment to the public network, the smart contract was tested using the

Truffle framework and the Ganache local blockchain to ensure all functionalities work as intended. This testing also helps minimize potential redeployment costs due to errors. Verification is conducted by deploying the smart contract to the Sepolia public testnet, allowing public access and global verification.

For future research, the system can be enhanced by integrating a blockchain-based file pinning feature, enabling users to manage their documents in a decentralized manner. All pinning and file management activities could be permanently recorded on the blockchain. This would not only improve document availability on IPFS but also strengthen transparency and accountability in document management.

REFERENCES

- [1] Rancang Bangun Website Akademik dengan Penyimpanan Sertifikat Digital Menggunakan Teknologi Blockchain. (2022). Jurnal Teknologi Informasi Dan Ilmu Komputer, 9(1), 33–40. <https://doi.org/10.25126/jtiik.2021863645>
- [2] Wang, S., Ouyang, L., Yuan, Y., Ni, X., Han, X., & Wang, F.-Y. (2019). Blockchain-Enabled Smart Contracts: Architecture, Applications, and Future Trends. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 1–12. doi:10.1109/tsmc.2019.2895123
- [3] Shaikh, Shahid. Building Decentralized Blockchain Applications: Learn How to Use Blockchain as the Foundation for Next-Gen Apps (English Edition). Jerman, Bpb Publications, 2021
- [4] Johnson M, Jones M, Shervey M, Dudley J, Zimmerman N. Building a Secure Biomedical Data Sharing Decentralized App (DApp): Tutorial. *J Med Internet Res* 2019;21(10): DOI: 10.2196/13601
- [5] Metodologi Pengembangan Sistem Informasi. (n.d.). (n.p.): LP2M Press IAIN Salatiga.
- [6] A. Gunawan, M. Munir, Y. Wibisono, and C. Furqon, "Integration Of Blockchain Technology In Digital Libraries: A Software Engineering Design", *jitik*, vol. 9, no. 2, pp. 161–171, Feb. 2024.
- [7] Lv, Guangjie & Song, Caixia & Xu, Pengmin & Qi, Zhiguo & Song, Heyu & Liu, Yi. (2023). Blockchain-Based Traceability for Agricultural Products: A Systematic Literature Review. *Agriculture*. 13. 1757. 10.3390/agriculture13091757.
- [8] Suryawijaya, T. W. E. (2023). Memperkuat Keamanan Data melalui Teknologi Blockchain: Mengeksplorasi Implementasi Sukses dalam Transformasi Digital di Indonesia. *JSKP: Jurnal Studi Kebijakan Publik*, 2(1), 55–67. <https://doi.org/10.21787/jskp.2.2023.55-67>.
- [9] Huang, Hsiao-Shan & Chang, Tian Sheuan & Wu, Jhih-Yi. (2022). A Secure File Sharing System Based on IPFS and Blockchain. 10.48550/arXiv.2205.01728.
- [10] Tarigan, Avinanta. (2022). Rancang Bangun Sistem Penerbitan Sertifikat Kompetensi Sebagai Aset Non-Fungible-Token (NFT) Berbasis Blockchain Dan Web3. *Jurnal Ilmiah Informatika Komputer*. 27. 246-257. 10.35760/ik.2022.v27i3.7787.
- [11] Lasawedi, M. F. (2022). Rancang Bangun Penyimpanan Terdistribusi Menggunakan Blockchain Dengan Protokol Interplanetary File System (IPFS). Makassar: Politeknik Negeri Ujung Pandang.
- [12] A. Jain, A. Kumar Tripathi, N. Chandra and P. Chinnasamy, "Smart Contract enabled Online Examination System Based in Blockchain Network," 2021 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2021, pp. 1-7, doi: 10.1109/ICCCI50826.2021.9402420.
- [13] Fernández-Iglesias, M.J.; Delgado von Eitzen, C.; Anido-Rifón, L. Efficient Traceability Systems with Smart Contracts: Balancing On-Chain and Off-Chain Data Storage for Enhanced Scalability and Privacy. *Appl. Sci.* 2024, 14, 11078. <https://doi.org/10.3390/app142311078>
- [14] N. Sangeeta; Nam, S.Y. Blockchain and Interplanetary File System (IPFS)-Based Data Storage System for Vehicular Networks with

- Keyword Search Capability. Electronics 2023, 12, 1545. <https://doi.org/10.3390/electronics12071545>
- [15] Oliver Bodemer . Elementary Blocks: Deciphering the Integration of Blockchain Technology in Agile and Waterfall Project Management Methodologies. TechRxiv. November 13, 2023. J. Breckling, Ed., The Analysis of Directional Time Series: Applications to Wind Speed and Direction, seri Lecture Notes in Statistics. Berlin, Germany: Springer, 1989, vol. 61.
- [16] Metodologi Pengembangan Sistem Informasi. (n.d.). (n.p.): LP2M Press IAIN Salatiga.
- [17] Antal, Claudia & Cioara, Tudor & Anghel, Ionut & Antal, Marcel & Salomie, Ioan. (2020). Blockchain based Decentralized Applications: Technology Review and Development Guidelines. 10.48550/arXiv.2003.07131.
- [18] A. Jain, A. Kumar Tripathi, N. Chandra and P. Chinnasamy, "Smart Contract enabled Online Examination System Based in Blockchain Network," 2021 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2021, pp. 1-7, doi: 10.1109/ICCCI50826.2021.9402420.
- [19] Chen, Zefeng. (2024). Design, development, and deployment of decentralized applications. Applied and Computational Engineering. 48. 46-52. 10.54254/2755-2721/48/20241132.
- [20] Mishra, Debani & Rajeev, B & Mallick, Soubhagya & Lenka, Rakesh & Salkuti, Surender Reddy. (2025). Efficient blockchain based solution for secure medical record management. International Journal of Informatics and Communication Technology (IJ-ICT). 14. 59. 10.11591/ijict.v14i1.pp59-67.
- [21] Ante, Lennart, and Aman Saggiu. 2024. Time-Varying Bidirectional Causal Relationships between Transaction Fees and Economic Activity of Subsystems Utilizing the Ethereum Blockchain Network. Journal of Risk and Financial Management 17: 19. <https://doi.org/10.3390/jrfm17010019>.
- [22] W. Uriawan, A. Wahana, C. Slamet and V. Suci Asih, "A DApp Architecture for Personal Lending on Blockchain," 2021 7th International Conference on Wireless and Telematics (ICWT), Bandung, Indonesia, 2021, pp. 1-6, doi: 10.1109/ICWT52862.2021.9678397.
- [23] Hartel, Pieter & Staaldin, Mark. (2019). Truffle tests for free -- Replaying Ethereum smart contracts for transparency. 10.48550/arXiv.1907.09208.
- [24] Ahamed, N Nasurudeen. (2023). A Build and Deploy Ethereum Smart Contract for Food Supply Chain Management in Truffle - Ganache Framework. 10.1109/ICACCS57279.2023.10112889.