DECENTRALIZED DIGILOCKER

A MAJOR PROJECT

REVIEW-3 REPORT

SUBMITTED BY,

BATCH – 9A

IPPILI TRIVENI

(21331A0565)

DOGGA NARENDRA

(21331A0540)

GADHIRAJU PAVAN PRUDVI RAJ VARMA

(21331A0546)

JAMPANA AKASH S RUSHENDRA VARMA

(21331A0566)

Under the Supervision of
Dr. B. Aruna Kumari
Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING MAHARAJ VIJAYARAM GAJAPATHI RAJ COLLEGE OF ENGINEERING (Autonomous)

(Approved by AICTE, New Delhi, and permanently affiliated to JNTUGV, Vizianagaram), Listed u/s 2(f) & 12(B) of UGC Act 1956.

Vijayaram Nagar Campus, Chintalavalasa,Vizianagaram-535005, Andhra Pradesh

August, 2024

TABLE OF CONTENTS

Abstract

The "Decentralized DigiLocker" project leverages blockchain technology to create a secure, decentralized, and user-friendly digital locker system. This system addresses limitations in traditional cloud storage by integrating a blockchain layer for enhanced security and decentralization. Core features include end-to-end encryption, decentralized identity management, and smart contracts for access control. The platform's architecture combines decentralized storage solutions with an intuitive user interface, ensuring both functionality and ease of use. By incorporating advanced tools like AppWrite, Zustand, and WagmiLibrary, the project offers a seamless and secure environment for storing and sharing digital assets. This initiative provides a robust, privacy-preserving alternative to conventional storage systems, emphasizing user empowerment and data sovereignty.

1. Introduction

1.1 Background information and the context of the project:

In an era of rapid digitalization, the need for secure, transparent, and decentralized document management systems has grown significantly. Blockchain technology, a revolutionary advancement, offers a robust solution to address the limitations of centralized systems. The Decentralized DigiLocker project capitalizes on this innovation to create a secure platform for document storage and verification. Inspired by the Indian government's DigiLocker initiative under the Digital India program, this project aims to eliminate the reliance on physical documents while ensuring data integrity and privacy. By leveraging blockchain, decentralized applications (DApps), and modern web technologies, the system ensures accessibility and reliability in managing digital assets.

1.2 The motivation behind choosing the project topic.

Centralized document storage systems are prone to challenges like single-point failures, data breaches, and lack of transparency. By leveraging blockchain technology, this project ensures tamper-proof, decentralized storage and verification of documents. It aligns with the growing need for digital empowerment and demonstrates the potential of blockchain and decentralized applications in addressing real-world problems.

1.3 Objectives and Scope:

To create a digital storage system that provides seamless security and redundancy while storing documents on-chain, the focus is on building a robust, decentralized solution that leverages blockchain technology to ensure data integrity, availability, and privacy. This system aims to address key challenges of traditional storage methods, such as data breaches, single points of failure, and lack of transparency.

2. Requirement Gathering

2.1 Detailed List of Functional and Non-Functional Requirements

Functional Requirements

These are specific features and capabilities the decentralized Digilocker must provide:

1. **User Registration and Authentication**
    - ○ Secure authentication via **public/private key pairs** or **multi-factor authentication** (MFA).
2. **Document Upload**
    - ○ Users should be able to upload documents to the platform.
    - ○ Documents must be stored in a decentralized manner using technologies like **IPFS** etc..
3. **Document Encryption**
    - ○ All documents should be encrypted before uploading.
    - ○ Only the user (document owner) holds the decryption keys for secure document access.
4. **Document Sharing**
    - ○ Users should be able to share documents with other parties securely, via permission-based access using **smart contracts**.
    - ○ The recipient can access the document only with proper authorization from the document owner.
5. **Decentralized Data Storage**
    - ○ All documents should be stored in a decentralized manner using **distributed file systems** to avoid a single point of failure.
    - ○ The platform should provide redundancy to ensure that documents are not lost if a node fails.
6. **Document Retrieval**
    - ○ Users should be able to easily retrieve and view their stored documents.
    - ○ Ensure documents can be accessed quickly from decentralized storage even with network delays.

7. **Smart Contract-based Permissions**
    - Permissions for access, retrieval, and sharing of documents should be governed by **smart contracts**.
    - Users can set conditions (e.g., time-based, identity-based) for granting document access.

Non-Functional Requirements

These focus on the system's performance, security, and usability characteristics:

1. **Security**
    - End-to-end encryption for all document transactions.
    - Protection from common attacks like **DDoS**, **Sybil attacks**, and **man-in-the-middle attacks**.
2. **Performance and Latency**
    - Fast retrieval times, ensuring that documents can be accessed promptly even from decentralized storage.
    - Optimal performance under high loads (upload/download of large files).
3. **Decentralization**
    - Avoid any form of central authority in controlling or managing user documents.
4. **Privacy and Compliance**
    - Ensure user data is private and complies with **GDPR**, **HIPAA**, or other applicable regulations.
    - No user document or metadata should be visible to unauthorized parties.
5. **Usability**
    - User interfaces should be intuitive and user-friendly, ensuring non-technical users can easily upload, retrieve, and share documents..

2.2 Methods Used to Gather Requirements

a. Interviews

- Conduct **structured interviews** with key stakeholders such as:
    - Potential users (individuals, organizations)
    - Blockchain developers and experts

b. Document Analysis

- Analyze existing documentation of similar platforms (Digilocker, Google Drive, blockchain-based storage like Filecoin, Sia, etc.) to identify current feature sets, limitations, and areas for improvement.

c. Competitive Analysis

- Analyze competing solutions such as centralized digital locker services (e.g., Dropbox, OneDrive) and decentralized systems (e.g., IPFS, Filecoin).
- This can help in understanding what works well and what can be improved.

d. Use Case Scenarios

- Define and analyze real-life scenarios where the decentralized Digilocker would be used (e.g., sharing documents with government agencies, secure document retrieval during travel).
- This method will ensure the system meets practical user needs.

3. Proposed Work

3.1 Outline of the Proposed Solution or Approach to the Problem

The concept of a decentralized DigiLocker presents an innovative approach to storing and managing personal documents such as IDs and certificates. By leveraging blockchain technology, this system aims to enhance security, privacy, and accessibility. However, transitioning from a centralized to a decentralized model introduces certain challenges and considerations.

1. Limitations of the Existing Centralized Model

- Single Point of Failure: Centralized systems are susceptible to disruptions; a failure in the central server can render the entire system inaccessible.
- Security Vulnerabilities: Storing all data in a single location makes it a prime target for cyber-attacks, potentially compromising sensitive information.
- Privacy Concerns: Users have limited control over their data, and centralized authorities may misuse or mishandle personal information.
- Limited Accessibility: Users might face difficulties accessing their documents during system downtimes or maintenance periods.

2. Advantages of the Decentralized DigiLocker

- **Enhanced Security**: Distributing data across a blockchain network reduces the risk of a single point of failure, making the system more resilient to attacks.
- **Improved Privacy**: Users retain control over their data, with the ability to manage access permissions, thereby minimizing the risk of unauthorized data usage
- **Access Control**: Documents are stored on a distributed network, ensuring availability even if some nodes are offline.
- **Immutability**: Blockchain's immutable nature ensures that once data is recorded, it cannot be altered, guaranteeing the authenticity of the documents

The proposed solution involves developing a **decentralized digital locker** using blockchain technology to securely store, manage, and share personal documents. This solution will leverage the unique advantages of blockchain to address the limitations of existing centralized systems. The approach can be outlined as follows:

A. Architecture Design

- **Decentralized Storage**: Use a distributed file storage system (e.g., **IPFS**) to store documents securely, ensuring no single point of failure.
- **Blockchain Layer**: Utilize a blockchain (e.g., **Ethereum**, **Solana**) for transaction management, document metadata, and smart contracts that govern access and sharing permissions.

B. User Interface Development

- **User-Friendly Design**: Incorporate UX/UI best practices to ensure users of all technical backgrounds can navigate the system seamlessly.

C. Security Implementation

- **End-to-End Encryption**: Implement encryption for documents during upload and while stored, ensuring that only authorized users can decrypt and access the files.
- **Decentralized Identity Management**: Integrate **DID** (Decentralized Identifier) frameworks to authenticate users and manage access without relying on a central authority.

D. Smart Contract Development

- **Access Control via Smart Contracts**: Develop smart contracts to manage document access, sharing, and permissions dynamically. These contracts will ensure that documents can only be accessed by authorized parties under specified conditions.

E. Deployment and Maintenance

- **Deployment on Blockchain**: Deploy the smart contracts and interface on the chosen blockchain platform, ensuring scalability and reliability.

3.2 Objectives and Scope of the Proposed Work

Objectives

1. **Enhance Security and Privacy**: Ensure users have full control over their documents with high-level security measures, including encryption and decentralized identity management.
2. **Facilitate Document Management**: Provide a user-friendly interface that simplifies the processes of uploading, retrieving, and sharing documents securely.
3. **Ensure Decentralization**: Build a system that eliminates central points of failure, promoting reliability and resilience against attacks and outages.
4. **Implement Robust Access Control**: Utilize smart contracts to govern document access dynamically, allowing users to set permissions and share documents securely with other parties.
5. **Maintain Transparency and Traceability**: Enable immutable audit logs that track document interactions and changes, ensuring accountability for all users.

Scope

- **Target Users**: The primary audience includes individual users, organizations, and government bodies that require a secure, reliable method to store and share documents digitally.
- **Document Types**: The system will support various document types, including identification documents, academic certificates, legal documents, and other important files.
- **Future Expansion**: The initial phase will focus on core functionalities, with the potential to expand into features like advanced analytics.

4. Implementation

The project involved the following steps:

Frontend Development:

- Built a React-based frontend to provide a user-friendly interface.
- Features include:
    - File uploading with automatic hash generation for security.
    - Input fields for user details like IDs for storage and verification.
    - Buttons to interact with the blockchain for storing and verifying documents.

Smart Contract Development:

- Developed and deployed a Solidity smart contract to manage:
    - **File Storage**: Stores document details using user IDs and file hashes.
    - **File Verification**: Checks document existence based on user IDs and file hashes.
    - **Event Emission**: Added FileStored events for transparency and debugging.

Appwrite Integration:

- Integrated Appwrite to store metadata such as file names, issuer addresses, and blockchain transaction hashes.
- Used Appwrite's Databases API for creating and retrieving document records.
- Implemented error handling for cases where blockchain storage succeeds but Appwrite updates fail.

Authentication:

- Added user authentication via **Clerk** to ensure only authorized users can access the system.

Workflow:

- **Store Document**: File hashes and details are stored on the blockchain, while metadata is saved in Appwrite.

- **Verify Document**: Checks the blockchain for document existence using user-provided details.

- **View Transactions**: Displays all the users transactions along with the attributes such as issuer address, document name and transaction address

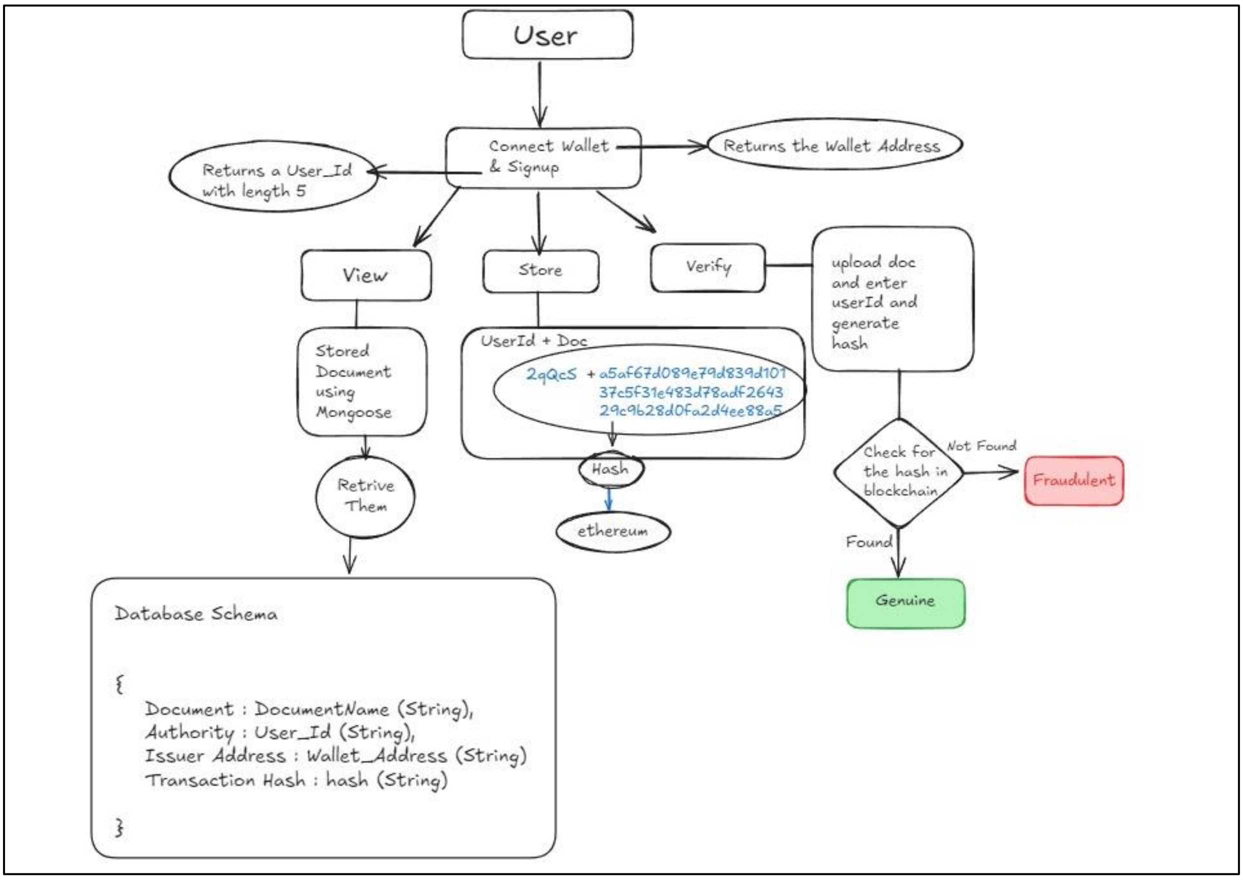
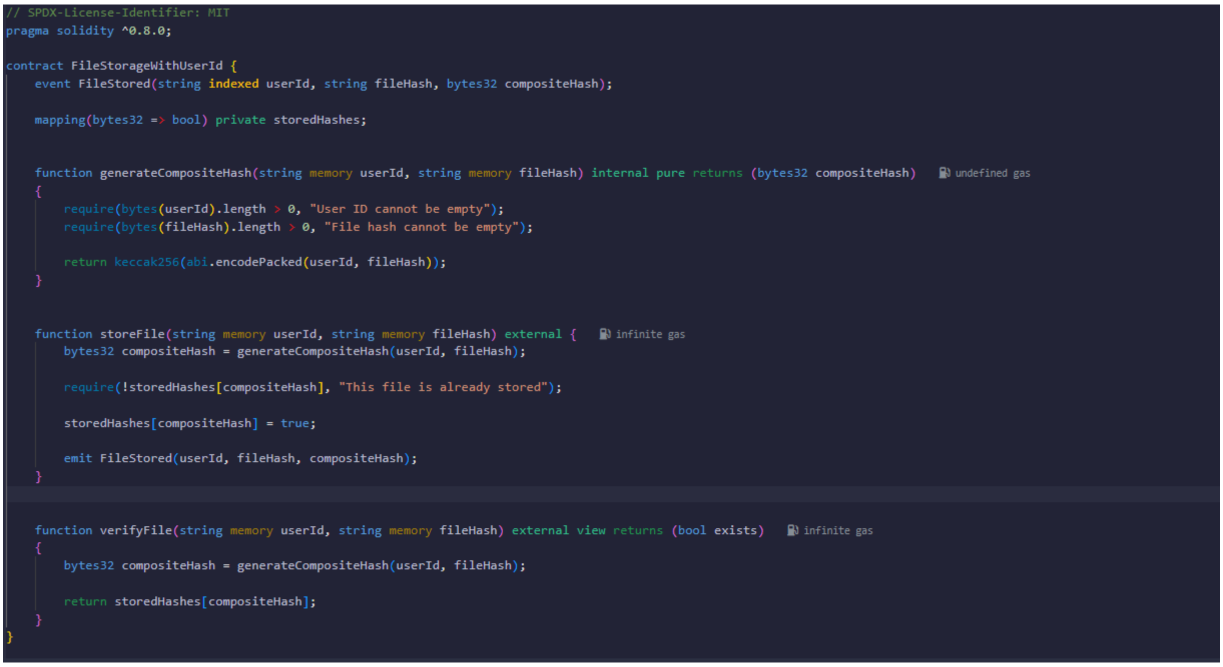
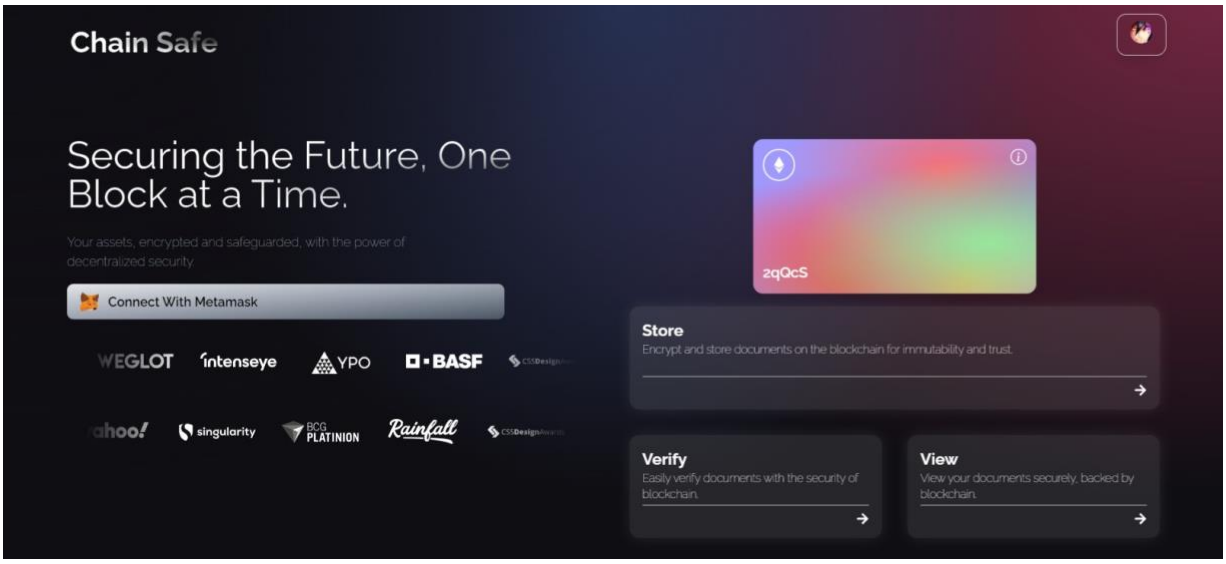

User
Connect Wallet & Signup
Returns the Wallet Address
Returns a User_Id with length 5
View
Store
Verify
upload doc and enter userId and generate hash
Stored Document using Mongoose
Retrive Them
UserId + Doc
2qQcS + a5af67d089e79d839d101 37c5f31e483d78adf2643 29c9b28d0fa2d4ee88a5
Hash
ethereum
Check for the hash in blockchain
Not Found
Fraudulent
Found
Genuine
Database Schema
{
Document : DocumentName (String),
Authority : User_Id (String),
Issuer Address : Wallet_Address (String)
Transaction Hash : hash (String)
}

Fig -4.1 – Flow of the project



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract FileStorageWithUserId {
    event FileStored(string indexed userId, string fileHash, bytes32 compositeHash);

    mapping(bytes32 => bool) private storedHashes;


    function generateCompositeHash(string memory userId, string memory fileHash) internal pure returns (bytes32 compositeHash)    undefined gas
    {
        require(bytes(userId).length > 0, "User ID cannot be empty");
        require(bytes(fileHash).length > 0, "File hash cannot be empty");

        return keccak256(abi.encodePacked(userId, fileHash));
    }


    function storeFile(string memory userId, string memory fileHash) external {    infinite gas
        bytes32 compositeHash = generateCompositeHash(userId, fileHash);

        require(!storedHashes[compositeHash], "This file is already stored");

        storedHashes[compositeHash] = true;

        emit FileStored(userId, fileHash, compositeHash);
    }


    function verifyFile(string memory userId, string memory fileHash) external view returns (bool exists)    infinite gas
    {
        bytes32 compositeHash = generateCompositeHash(userId, fileHash);

        return storedHashes[compositeHash];
    }
}
```

Fig-4.2 Smart Contract on remix



Chain Safe

Securing the Future, One Block at a Time.

Your assets, encrypted and safeguarded, with the power of decentralized security

Connect With Metamask

Store

Encrypt and store documents on the blockchain for immutability and trust.

Verify

Easily verify documents with the security of blockchain.

View

View your documents securely, backed by blockchain.

Fig -4.3 Hero Section

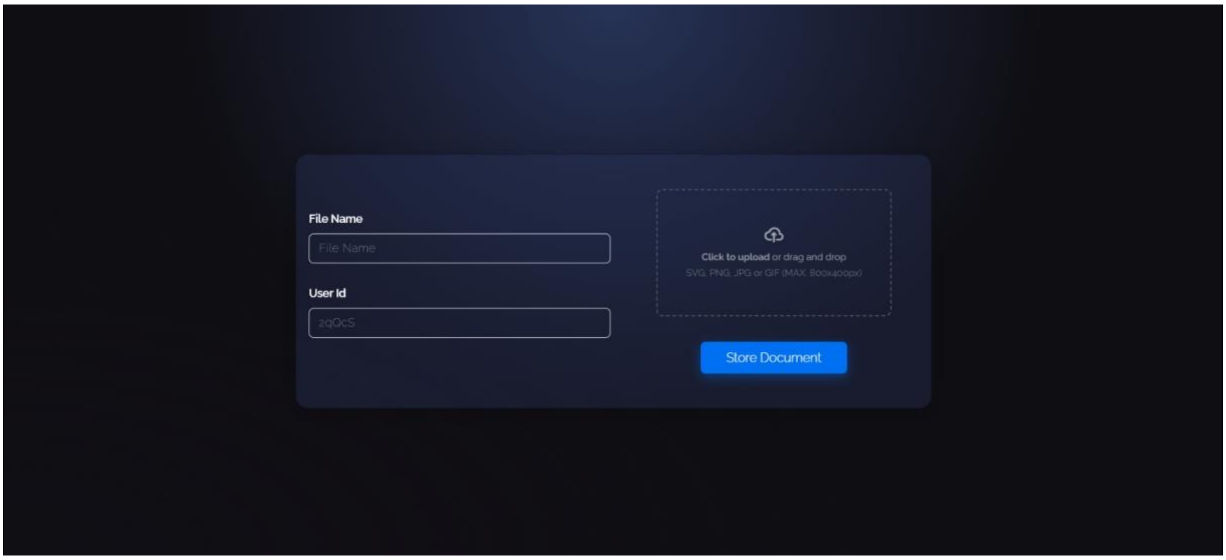

File Name
File Name
User Id
2qQcS
Click to upload or drag and drop
SVG, PNG, JPG or GIF (MAX. 800x400px)
Store Document

Fig-4.4 Store Section

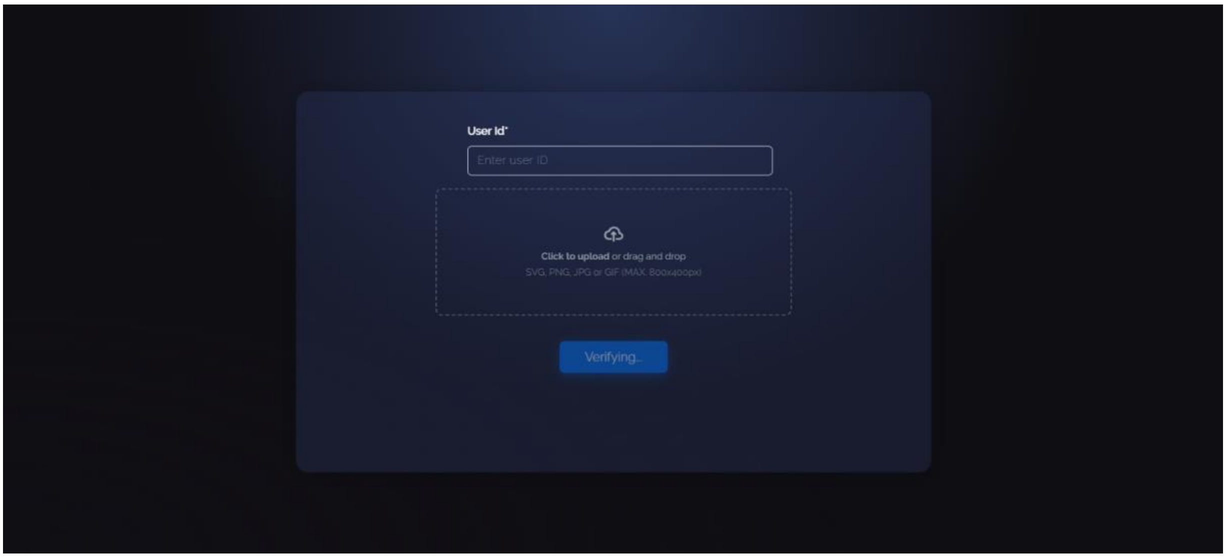
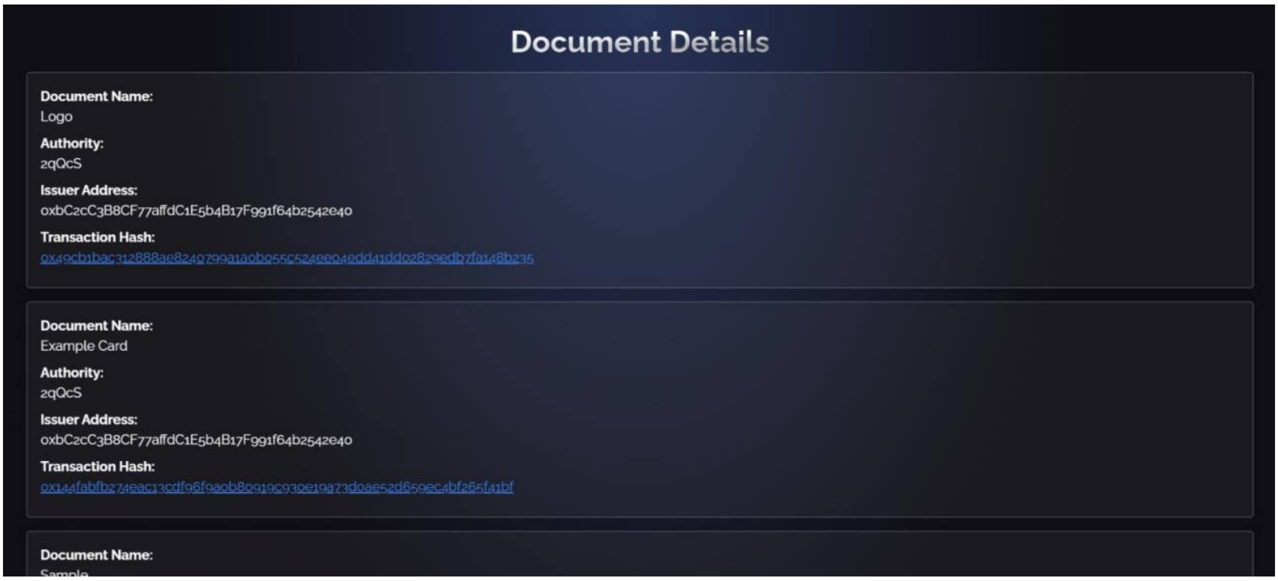

User Id*
Enter user ID
Click to upload or drag and drop
SVG, PNG, JPG or GIF (MAX. 800x400px)
Verifying...

Fig-4.5 Verify Section



Document Details

Document Name:
Logo

Authority:
2qQcS

Issuer Address:
0xbC2cC3B8CF77affdC1E5b4B17F991f64b2542e40

Transaction Hash:
0x49cb1bac312888ae8240799a1a0b055c524ee04edd41dd02829edb7fa148b235

Document Name:
Example Card

Authority:
2qQcS

Issuer Address:
0xbC2cC3B8CF77affdC1E5b4B17F991f64b2542e40

Transaction Hash:
0x144fabfb274eac13cdf96f9a0b80919c930e19a73d0ae52d659ec4bf265f41bf

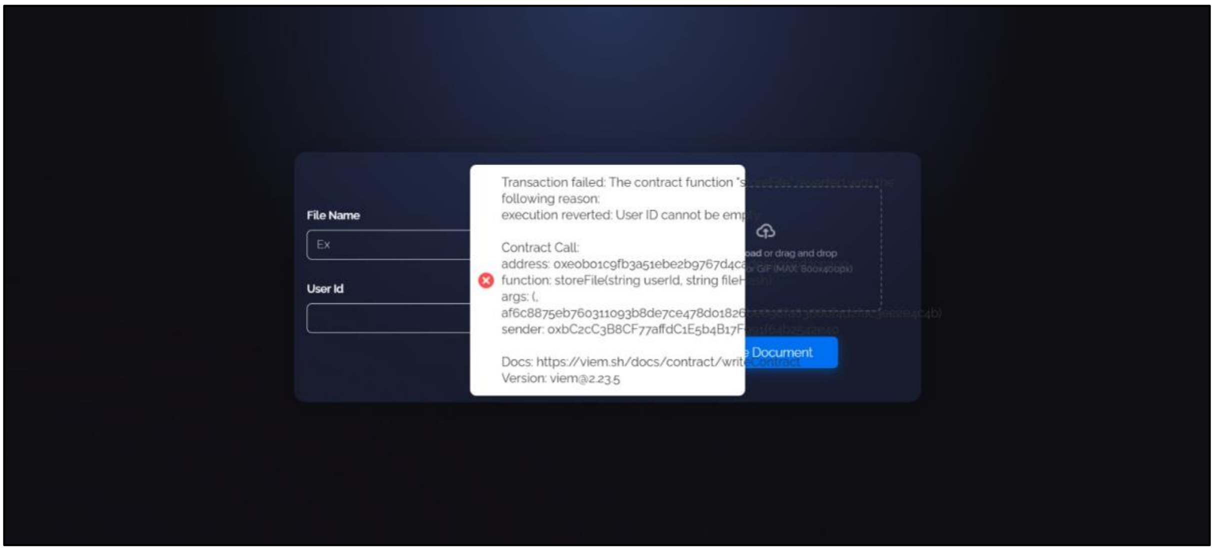
Document Name:
Sample

Fig-4.6 View Section

5.TESTING

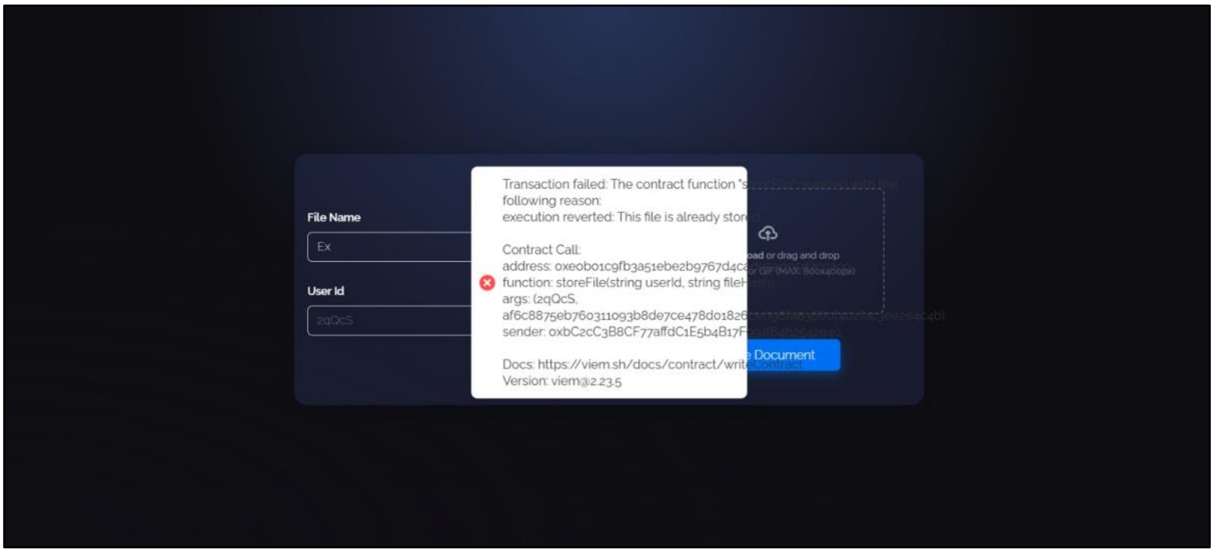
5.1 System Performance and Evaluation

The Decentralized DigiLocker system was successfully implemented using blockchain and IPFS for secure document storage. The performance was evaluated based on security, accessibility, and transaction efficiency.



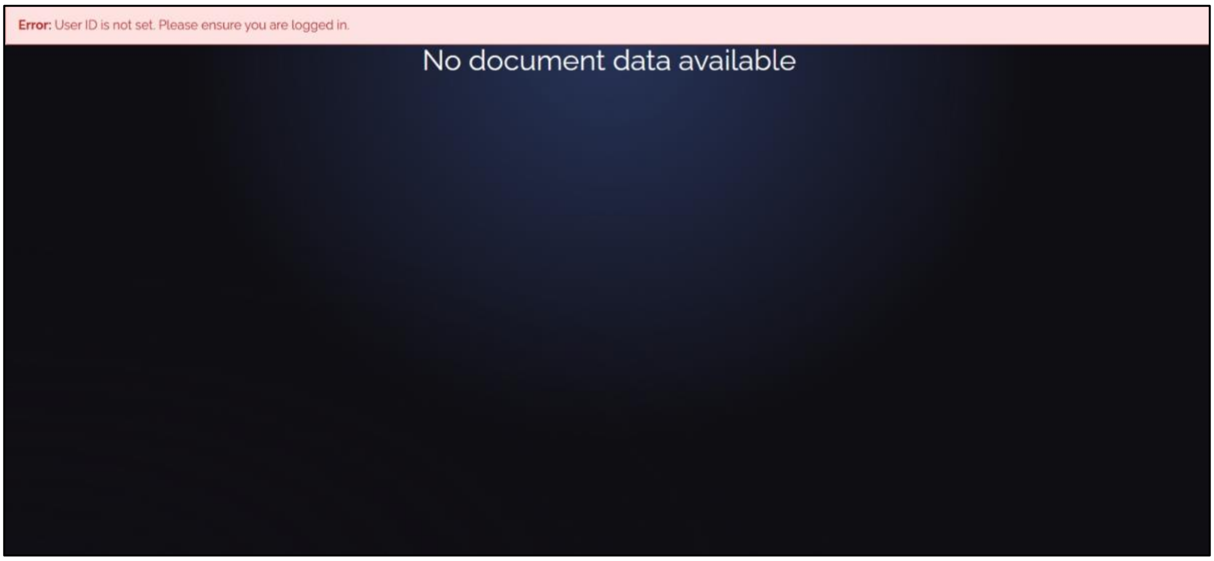
```
Transaction failed: The contract function "s
following reason:
execution reverted: User ID cannot be emp

Contract Call:
address: 0xe0b01c9fb3a51ebe2b9767d4c
function: storeFile(string userId, string fileH
args: (,
af6c8875eb760311093b8de7ce478d0182
sender: 0xbC2cC3B8CF77affdC1E5b4B17F

Docs: https://viem.sh/docs/contract/writ
Version: viem@2.23.5
```

Fig 5.1 Error Message for Unauthorized Document Storage



```
Transaction failed: The contract function "s
following reason:
execution reverted: This file is already stor

Contract Call:
address: 0xe0b01c9fb3a51ebe2b9767d4c
function: storeFile(string userId, string fileH
args: (2qQcS,
af6c8875eb760311093b8de7ce478d0182
sender: 0xbC2cC3B8CF77affdC1E5b4B17F

Docs: https://viem.sh/docs/contract/writ
Version: viem@2.23.5
```

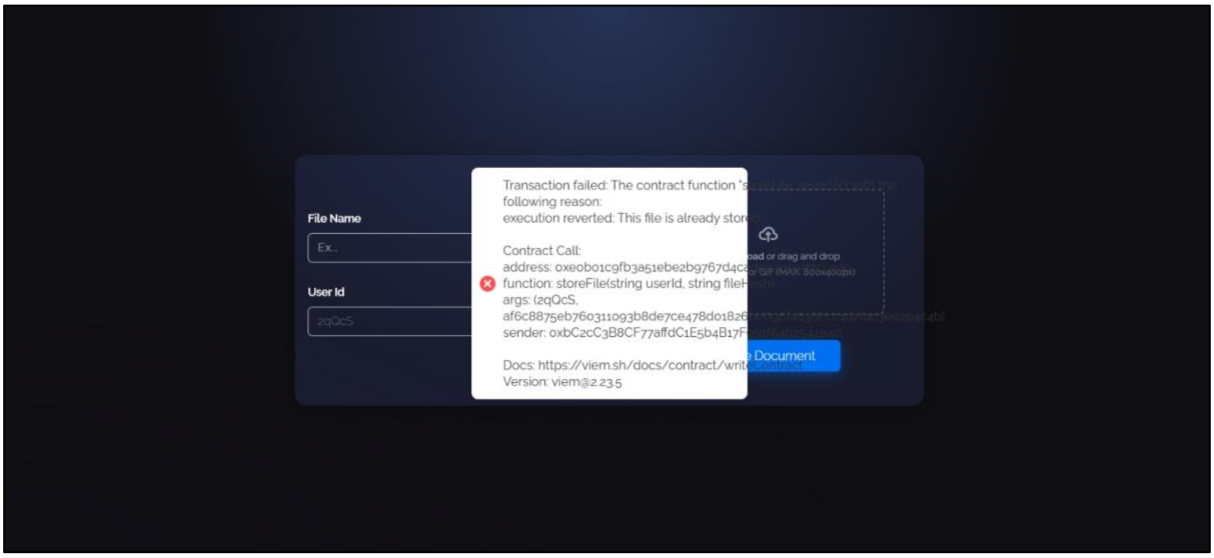
Fig 5.2 Wallet Connection Requirement for Document Storage



Error: User ID is not set. Please ensure you are logged in.

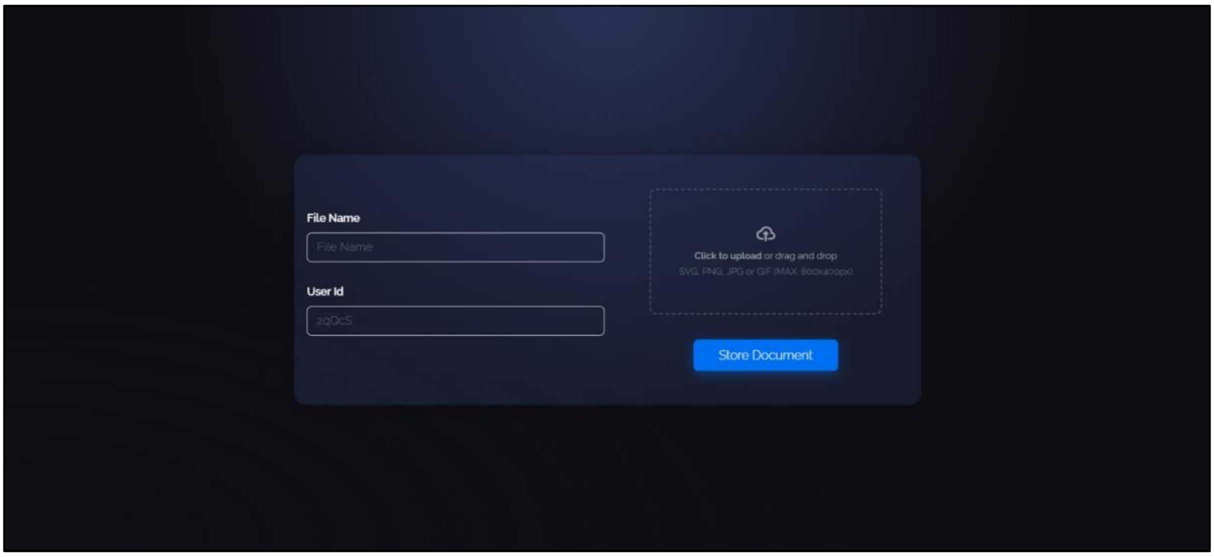
No document data available

Fig 5.3 Transaction History for Authenticated Users



```
Transaction failed: The contract function "s
following reason:
execution reverted: This file is already stor

Contract Call:
address: 0xe0b01c9fb3a51ebe2b9767d4ca
function: storeFile(string userId, string fileH
args: (2qQcS,
af6c8875eb76031109 3b8de7ce478d0182б
sender: 0xbC2cC3B8CF77affdC1E5b4B17F

Docs: https://viem.sh/docs/contract/writ
Version: viem@2.23.5
```

Fig 5.4 Duplicate Document Storage Prevention



File Name: File Name

User Id: 2qQcS

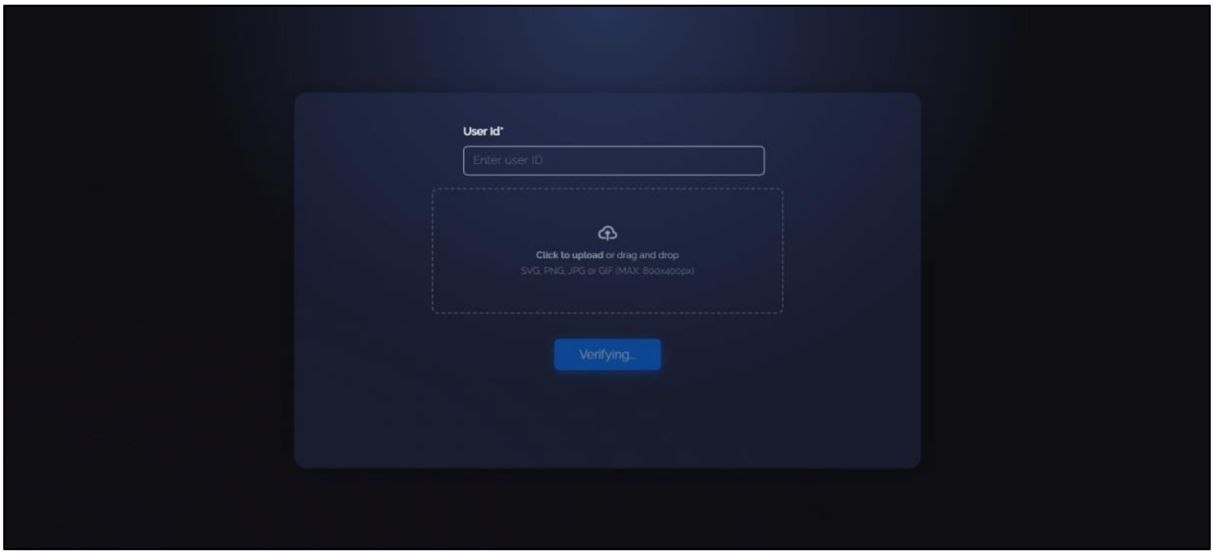
Click to upload or drag and drop
SVG, PNG, JPG or GIF (MAX. 800x400px)
Store Document

Fig 5.5 Secure Document Management Interface

5.2 Functional Testing Results

Table 5.2 summarizes the testing results of various system functionalities:

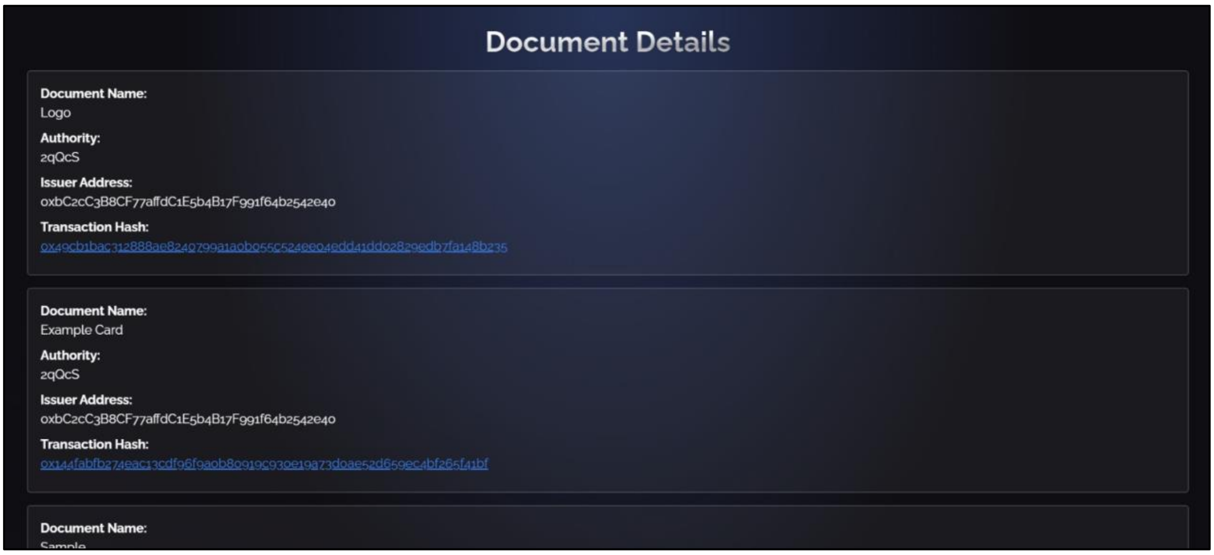
Feature	Expected Outcome	Result
User Registration	Secure wallet-based authentication	☑ Successful
Document Upload	Hash stored on the blockchain & IPFS	☑ Successful
Document Verification	Hash validation with blockchain data	☑ Accurate
Unauthorized Access	Restriction for unauthorized users	☑ Secure



User Id*: Enter user ID

Click to upload or drag and drop
SVG, PNG, JPG or GIF (MAX. 800x400px)
Verifying...

Fig: 5.6 Blockchain-Based Document Verification Process



Document Details

Document Name:
Logo

Authority:
2qQcS

Issuer Address:
0xbC2cC3B8CF77affdC1E5b4B17F991f64b2542e40

Transaction Hash:
0x49cb1bac312888ae8240799a1a0b055c524ee04edd41dd02829edb7fa148b235

Document Name:
Example Card

Authority:
2qQcS

Issuer Address:
0xbC2cC3B8CF77affdC1E5b4B17F991f64b2542e40

Transaction Hash:
0x144fabfb274eac13cdf96f9a0b80919c930e19a73d0ae52d659ec4bf265f41bf

Document Name:
Sample

Fig 5.7 User Interface for Document Retrieval

5.3 Security Analysis

- **Tamper-proof storage:** Blockchain immutability ensures document security.
- **Privacy protection:** User data remains decentralized, reducing risks of data leaks.
- **Authentication:** Wallet-based authentication prevents unauthorized access.

6.CONCLUSION & FUTURE WORK

6.1 Summary of Achievements

The Decentralized DigiLocker system successfully enhances security and transparency in document storage using blockchain technology. The project:

- Eliminates single points of failure found in centralized systems.
- Ensures tamper-proof document integrity using cryptographic hashing.
- Enables trustless document verification with smart contracts**.**

6.2 Limitations

- Gas fees for blockchain transactions can be high.
- User onboarding requires knowledge of crypto wallets.
- Scalability needs optimization for large-scale document storage.

6.3 Future Enhancements

- Layer 2 Scaling: Integration with Polygon or Optimistic Rollups to reduce transaction costs.
- Decentralized Identity (DID): Integration with DID protocols for enhanced user privacy.
- AI-powered Document Classification: Automating document categorization for better usability.

References

[1] **Zyskind, G., Nathan, O., & Pentland, A. (2015)**. *Decentralizing Privacy: Using Blockchain to ProtectPersonal Data*.

[2] https://ieeexplore.ieee.org/document/7163223

[3] Bitcoin white paper by satoshi nakamoto :

[4] **https://www.bitcoin.com/satoshi-archive/whitepaper/**

[5] OpenZeppelin Smart Contracts & ERC 721 https://github.com/OpenZeppelin/openzeppelin-contracts

[6] "Blockchain," Wikipedia, The Free Encyclopedia. Available: https://en.wikipedia.org/wiki/Blockchain. Accessed: Jan. 5, 2025.

[7] *"Introduction to Blockchain Features," 101 Blockchains*, Available: https://101blockchains.com/introduction-to-blockchain-features/. Accessed: Mar. 1, 2025.*

[8] *"Decentralization," Quickonomics*, Available: https://quickonomics.com/terms/decentralization/. Accessed: Mar. 1, 2025.*

[9] *Advantages and Disadvantages of Decentralization," Educba*, Available: https://www.educba.com/advantages-and-disadvantages-of-decentralization/. Accessed: Mar. 1, 2025

[10] *Smart Contracts: Mechanism and Basic Concepts," SotaTek*, Available: https://www.sotatek.com/blogs/smart-contracts-mechanism-and-basic-concepts/. Accessed: Mar. 1, 2025.

[11] *Smart Contracts: Benefits and Use Cases," eBizMBA*, Available: https://www.ebizmba.com/uk/smart-contracts-benefits-use-cases/. Accessed: Mar. 1, 2025.

[12] M. Conoscenti, A. Vetrò, and J. C. De Martin, "Blockchain for the Internet of Things: A systematic literature review," *Future Generation Computer Systems*, vol. 109, pp. 123-136, Aug. 2020. Available: https://doi.org/10.1016/j.future.2019.12.002. Accessed: Mar. 1, 2025.

[13] *Cryptography and Its Types," GeeksforGeeks*, Available: https://www.geeksforgeeks.org/cryptography-and-its-types/. Accessed: Mar. 1, 2025.

[14] *Cryptography Techniques,"* *Simplilearn,* Available: https://www.simplilearn.com/cryptography-techniques-article#types_of_cryptography. Accessed: Mar. 1, 2025.