DECENTRALIZED DOCUMENT VERIFICATION SYSTEM

BY

EDMUND, EBIYENRIN MOSES

18/48949/UE

A PROJECT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE, JOSEPH SARWUAN TARKA UNIVERSITY, MAKURDI IN PARTIAL FULFILLMENT OF THE AWARD OF DEGREE IN BACHELOR OF SCIENCE (B.Sc) IN COMPUTER SCIENCE

MAY, 2023

# **DECLARATION**

I declare that the work described in this project is original, and has not been previously submitted to any University or similar institution for the award of any degree or certificate.

Name of Candidate: EDMUND, EBIYENRIN MOSES

Signature: ……………………………………………...

Date: …………………………………………………...

# **CERTIFICATION**

We, the undersigned hereby certify that this project work presented by EDMUND, EBIYENRIN MOSES (18/48949/UE) be accepted as fulfilling part of the requirements for the award of Bachelor of Science (B.Sc.) Degree in Computer Science.

Title: Decentralized Document Verification System

Mr. D.E Aniobi \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Project Supervisor Signature Date

Dr. T. Gbaden \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Head of Department Signature Date

Dr. E.O. Agu \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

External Supervisor Signature Date

**DEDICATION**

This work is dedicated to God Almighty for the supernature supply. May His name be glorified.

# **ACKNOWLEDGEMENT**

I would like to take this opportunity to express my heartfelt gratitude to everyone who has supported and contributed to the successful completion of this project.

First and foremost, I am deeply grateful to my project supervisor Mr. D.E Aniobi, for their invaluable guidance, support, and expertise. Their mentorship has been instrumental in shaping the direction and quality of my work.

I would also like to extend my sincere appreciation to the HOD of Computer Science Dr. T. Gbaden,and members of Computer Science Department for their continuous support and encouragement throughout the project. Their knowledge and feedback have significantly contributed to the development of this research.

I would like to express my gratitude to the participants who generously dedicated their time and provided valuable input during the testing and evaluation phase. Their contributions have been crucial in validating the functionality and effectiveness of the decentralized document verification system.

I would like to acknowledge the authors of the published works that have been referenced in this research. Their insightful studies have laid a solid foundation for my understanding of blockchain, Inter-Planetary File System (IPFS), and smart contracts in the context of document verification.

Finally, I would like to thank my family and friends for their unwavering support and encouragement. Their belief in my abilities and their understanding throughout this journey have been invaluable.

In conclusion, I would like to express my deep appreciation to all individuals and organizations mentioned above for their support and contributions. This project would not have been possible without their assistance, and I am truly grateful for their involvement in this endeavor.

# **TABLE OF CONTENTS**

[**DECLARATION** ii](#_Toc137116043)

[**CERTIFICATION** iii](#_Toc137116044)

[**ACKNOWLEDGEMENT** v](#_Toc137116045)

[**TABLE OF CONTENTS** vi](#_Toc137116046)

**LIST OF TABLES** ix

**LIST OF FIGURES** x

[**ABSTRACT** x](#_Toc137116047)i

[**CHAPTER ONE** 1](#_Toc137116048)

[**1.0**  **INTRODUCTION** 1](#_Toc137116049)

[1.1 Background to the Study 1](#_Toc137116050)

[1.2 Problem Statement 3](#_Toc137116051)

[1.3 Aim and Objectives of the Study 3](#_Toc137116052)

[1.4 Justification of the Study 4](#_Toc137116053)

[1.5 Scope of the Study 5](#_Toc137116054)

[1.6 Definition of Terms 5](#_Toc137116055)

[**CHAPTER TWO** 8](#_Toc137116056)

[2.0 LITERATURE REVIEW 8](#_Toc137116057)

[2.1 Overview of Decentralized System 8](#_Toc137116058)

[2.1 Blockchain 9](#_Toc137116059)

[2.1.1 Blockchain wallets 10](#_Toc137116060)

[2.1.2 Smart contracts 11](#_Toc137116061)

[2.1.3 Smart Contract 13](#_Toc137116062)

[2.1.4 Interplanetary File System (IPFS) 14](#_Toc137116063)

[2.2 Review of Related Works 15](#_Toc137116064)

[2.3.1 Summary of Related works 17](#_Toc137116065)

2.4 Knowledge Gap 18

[**CHAPTER THREE** 18](#_Toc137116066)

[ANALYSIS AND DESIGN 18](#_Toc137116067)

[3.1 System Analysis 18](#_Toc137116068)

[3.1.1 Fact findings 20](#_Toc137116069)

[3.1.2 Analysis of the existing system 21](#_Toc137116070)

[3.1.3 Problem of the existing system 21](#_Toc137116071)

[3.1.4 Advantages of the proposed system 22](#_Toc137116072)

[3.2 Modeling the Proposed System 22](#_Toc137116073)

[3.2.1 Proposed system workflow 24](#_Toc137116074)

[3.2.2 Proposed system architecture 24](#_Toc137116075)

[3.3 System Design 25](#_Toc137116076)

[3.3.1 Input Design 25](#_Toc137116077)

[3.3.2 Output Design 26](#_Toc137116078)

[3.4 Program Design 27](#_Toc137116079)

[3.4.1 Flow chart Diagram 28](#_Toc137116081)

[3.4.2 Sequence Diagram 29](#_Toc137116082)

[**CHAPTER FOUR** 30](#_Toc137116083)

[**IMPLEMENTATION AND RESULTS** 30](#_Toc137116084)

[4.1 Implementation 30](#_Toc137116085)

[4.2 Program Testing 30](#_Toc137116086)

[4.3 Result 31](#_Toc137116087)

[4.4 Discussion of Results 35](#_Toc137116088)

[4.5 System Requirement 36](#_Toc137116089)

[4.6 User Documentation 36](#_Toc137116090)

[4.7 System Change Over 38](#_Toc137116091)

[**CHAPTER FIVE** 39](#_Toc137116092)

[5.0 CONCLUSION AND RECOMMENDATION 39](#_Toc137116093)

[5.1 Conclusion 39](#_Toc137116094)

[5.2 Recommendations 40](#_Toc137116095)

[**REFERENCES** 41](#_Toc137116096)

**APPENDIX** 45

# **LIST OF TABLES**

Table No Title Page

1 Summary of related works 17

2 Input Design 26

3 Output Design 26

4 Test Cases 31

5 Testing Results 32

**LIST OF FIGURES**

Figure No Title Page

1 Transaction Layout 19

2 Proposed system use case 23

3 Proposed system workflow 24

4 Proposed system architecture 24

5 Proposed flowchart of the system 28

6 Proposed sequence diagram of the system 29

7 Home page of the system 33

8 Admin section of the system 33

9 A hashed document 34

10 Upload section of the system 34

11 Verify section of the system 35

12 Delete section of the system 35

# **ABSTRACT**

Blockchain technology has emerged as a disruptive innovation with numerous applications across various domains. One significant area of exploration is its potential in revolutionizing document verification systems. This project focuses on the development of a decentralized document verification system using blockchain technology. The project begins by providing a comprehensive overview of blockchain technology, highlighting its core features such as immutability, transparency, and decentralization. It explores the fundamental concepts of blockchain, including blocks, cryptographic hashing, and consensus algorithms. Building upon this foundation, the project proposes a system that leverages blockchain for secure and tamper-proof document verification. The system employs smart contracts to automate the verification process, ensuring transparency and efficiency. It also incorporates IPFS (InterPlanetary File System) for decentralized and secure document storage. The implementation involves using Ethereum blockchain, Solidity, ReactJs, Haskell and IPFS for system development. Extensive testing was carried on the developed app which validated the system's functionality, security, and performance. The results demonstrated the system's effectiveness, as it was able to hash, upload document and verify the uploaded documents. The results of the project demonstrate the feasibility and effectiveness of utilizing blockchain technology for document verification. The decentralized nature of the system ensures data integrity, eliminates the need for intermediaries, and provides a transparent and tamper-proof verification process.

# **CHAPTER ONE**

# **1.0 INTRODUCTION**

# **1.1 Background to the Study**

In recent years, there has been a growing interest in decentralized file storage and access systems using blockchain technology and Inter-Planetary File System (IPFS) integrated with public cloud infrastructure (Swan, 2018; Benet, 2018). The emergence of these technologies has revolutionized the way files are stored, accessed, and secured.

Traditional file storage systems rely on centralized servers, which pose several limitations such as single points of failure, data loss risks, and vulnerability to censorship. The blockchain-based decentralized file storage and access system using IPFS and public cloud infrastructure offers a promising alternative (Lu et al., 2019). By leveraging the decentralized nature of IPFS and the security features of blockchain technology, the system ensures data integrity, availability, and censorship resistance.

As the project focuses on developing a blockchain-based decentralized file storage and access system using IPFS and public cloud infrastructure, the study aims to explore the capabilities and advantages of this innovative approach. By utilizing IPFS, files are stored across a network of participating nodes, eliminating the reliance on a single centralized server. The integration of blockchain technology ensures the immutability and transparency of file transactions, as well as the implementation of access control mechanisms.

The project will delve into the technical aspects of implementing a decentralized file storage and access system using IPFS and blockchain. It will explore the network protocols and algorithms required for efficient file chunking, distribution, and retrieval. Additionally, the study will investigate the integration of public cloud infrastructure to enhance scalability and cost-effectiveness.

The concept of blockchain was published by Satoshi Nakamoto (2008), is a distributed ledger system that stores transaction details between parties in a Verifiable and immutable way. It acts as a decentralised database which is a digital ledger that stores data and information throughout the entire network of a computer systems. A cryptocurrency is a digitized currency that can be used to do a transaction and buy stuffs through online. With the help of blockchain technology all the transactions that have been made in the network are being kept safe since the system uses a distributed ledger that will let others know the credibility of a transactions. Blockchain technology had been used by many organizations now such as healthcare, inventory, management, finance and more. Blockchain technology has been helpful in implementing a system that will keep the transactions and storing the data in a safe manner and it is easier to track down the attacker with the implementation of blockchain technology. A document verification system will be able to increase its security feature with the help of blockchain technology.

Verifiable Credentials can be created using Blockchain technology. This credential will enable owners to easily share with third parties who can proof of ownership and validity and that it was issued by the institution mentioned.

# **1.2 Statement of the Problem**

In the current digital landscape, document verification processes often rely on centralized authorities or intermediaries, leading to concerns of trust, security, and transparency. Centralized systems are susceptible to data breaches, fraud, and manipulation, undermining the integrity of document verification.

To address these challenges, a decentralized document verification system is proposed, leveraging Ethereum blockchain, IPFS, and smart contracts. The system aims to provide a tamper-proof and transparent method for verifying documents, eliminating the need for intermediaries and enhancing trust among participants.

# **1.3 Aim and Objectives of the Study**

The aim of this project is to develop a decentralized document verification system using Ethereum blockchain, IPFS, and smart contracts.

The objectives of the study are to:

1. Develop a high-level architectural design that integrates Ethereum blockchain, Inter-Planetary File System (IPFS), and smart contracts to ensure secure document verification and storage.
2. Develop a smart contract using Solidity that manages the verification process, storing document hashes and relevant metadata on the Ethereum blockchain.
3. Integrate IPFS to securely store the documents and obtain content hashes that can be linked to the corresponding smart contract records.

# **1.4 Justification of the Study**

A decentralized document verification system using Ethereum blockchain, Inter-Planetary File System (IPFS), and smart contracts provides enhanced security, eliminates intermediaries, increases transparency, builds trust, improves efficiency, reduces costs, and ensures decentralized storage and data availability. It addresses concerns of data breaches, fraud, manipulation, delays, privacy, and trust in document verification processes. This system offers tamper-proof verification, direct interaction between document owners and verifiers, transparency through blockchain technology, automation of verification tasks, and reliable decentralized file storage. It has the potential to revolutionize industries reliant on document verification.

# **1.5 Scope of the Study**

The scope of this study focuses on the development of a decentralized document verification system using Ethereum blockchain, Inter-Planetary File System (IPFS), and smart contracts.

The scope of this study is limited to the development and implementation of the decentralized document verification system using the specified technologies. It does not encompass broader aspects such as legal and regulatory considerations, governance models, or extensive integration with existing document management systems.

# **1.6 Definition of Terms**

**Blockchain**: A blockchain is a decentralized and immutable digital ledger that

records transactions across multiple computers or nodes. Each transaction is

grouped into a block and linked to previous blocks using cryptographic hashes,

creating a chain of blocks. Blockchain ensures transparency, security, and

immutability of data.

**Decentralized Document Verification System**: A decentralized document

verification system is a software application that utilizes blockchain and

other decentralized technologies to verify the authenticity and integrity of

digital documents. It eliminates the need for a centralized authority and

provides a transparent and tamper-proof method of verifying document

ownership and content.

**IPFS (InterPlanetary File System)**: IPFS is a distributed file system that

enables secure and decentralized file storage and sharing. It uses a content-addressable method to uniquely identify and retrieve files based on their cryptographic hash. IPFS ensures data availability, fault tolerance, and censorship resistance.

**Smart Contracts**: Smart contracts are self-executing contracts with predefined

rules and conditions encoded within them. They are stored on the blockchain and automatically execute when predetermined conditions are met. Smart contracts enable trustless and automated transactions without the need for intermediaries.

**User Interface (UI)**: The user interface refers to the visual and interactive

components of a software application that allow users to interact with the system. In the context of the decentralized document verification system, the user interface provides an intuitive and user-friendly platform for users to register, verify, and retrieve documents.

**Verification Process**: The verification process involves validating the authenticity,

integrity, and ownership of a document. In the decentralized document verification system, the process typically includes verifying the document's digital signature, comparing hashes, and checking the document's metadata stored on the blockchain.

**Solidity**: Solidity is a programming language specifically designed for writing

smart contracts on the Ethereum blockchain. It is a statically-typed, contract-oriented language that enables developers to define the behavior and logic of smart contracts.

**Security Measures**: Security measures encompass various techniques and

mechanisms implemented to protect the system, data, and documents from unauthorized access, tampering, and malicious activities. This may include encryption, access controls, authentication mechanisms, and secure communication protocols.

**Scalability**: Scalability refers to the system's ability to handle increasing

workload and growing user demand without sacrificing performance or functionality. A scalable system can accommodate a higher number of transactions, users, and documents without significant degradation in performance.

**Performance Testing**: Performance testing involves assessing the system's

responsiveness, throughput, resource utilization, and stability under various load conditions. It helps identify performance bottlenecks, optimize system performance, and ensure the system meets performance requirements.

# **CHAPTER TWO**

# **2.0 LITERATURE REVIEW**

# **2.1** **Overview of Decentralized System**

In recent years, decentralized document verification systems have garnered significant attention due to the potential of blockchain technology to revolutionize the verification process. Swan (2018) emphasizes the security advantages of blockchain technology, including immutability and cryptographic hashing, which ensure tamper-proof document verification. Christidis and Devetsikiotis (2019) explore the integration of blockchain and smart contracts for the Internet of Things (IoT), highlighting the elimination of intermediaries and increased transparency. Their work aligns with the objectives of decentralized document verification systems.

To implement the system, Ethereum's Web3.js API provides essential documentation for developing smart contracts using Solidity, the programming language for Ethereum smart contracts. This resource is crucial for the successful implementation of the document verification system on the Ethereum blockchain.

In terms of decentralized storage, Benet (2019) introduced Inter-Planetary File System (IPFS), a content-addressed, versioned, peer-to-peer file system. The integration of Inter-Planetary File System (IPFS)ensures improved data availability and reliability, addressing the challenges of centralized storage systems.

Scalability is a key consideration in decentralized systems. Kshetri (2018) explores blockchain's potential in various domains, including scalability challenges. He proposed optimization techniques such as off-chain storage, transaction batching, and layer-two solutions to enhance system efficiency. These strategies may prove valuable in addressing scalability concerns in decentralized document verification systems.

# **2.1 Blockchain**

Blockchain is decentralized immutable ledger which monitor the transactions and track- ing assets which can be a tangible or intangible assets in the network. It is a Distributed Ledger Technology (DLT) which record all the transactions using a cryptographic sys- tem which is a hash. Blockchain consists of blocks which are tied up to each other with a chain inside the network. Each block consists of information which are the data, previous and current hash value, and the timestamped of the transactions occurred that are strictly known to the people in the network who made the transaction. The transactions in the blockchain will be completed when all the peers in the blockchain network approve and accept the data that have been stored into the blockchain which means the approval will successfully store the data into the blockchain. (Zainuddin and Choo (2019).

Blockchain has been developed to be resistant to modification of the data. Blockchain has been designed in form of distributed database system that records transactions in the network in a verifiable and permanent way. It is a connection a peer-to-peer nodes that collectively agree on set of rules for communication and validations new records in the network. Once the certificate data is registered, the information in the blocks cannot be tampered without interfering the rest of the blocks which requires a consensus and thus making it tamper-proof system. (Zainuddin and Choo (2019).

## **2.1.1 Blockchain wallets**

Blockchain wallets are digital wallets that enable users to securely store, manage, and interact with their cryptocurrency assets. These wallets utilize blockchain technology to provide a decentralized and transparent means of controlling and accessing digital currencies.

One popular blockchain wallet is MetaMask. MetaMask is a browser extension wallet that allows users to interact with decentralized applications (dApps) and securely manage their Ethereum-based cryptocurrencies. It provides a user-friendly interface for wallet management, transaction signing, and interaction with blockchain-based applications. MetaMask has gained significant popularity due to its ease of use and extensive functionality. It serves as a bridge between users' web browsers and the Ethereum blockchain, enabling seamless integration of decentralized applications.

The wallet generates and stores cryptographic keys, including public and private keys, which are used to authenticate transactions and provide secure access to Ethereum accounts. Users can import existing accounts or create new ones within MetaMask. The wallet also supports the creation of multiple accounts, allowing users to organize their funds and interact with different dApps. MetaMask provides a secure environment for users to review and authorize transactions before they are sent to the Ethereum network. It also offers features such as network selection, transaction history, and token management, enhancing user control and convenience.

## **2.1.2 Smart contracts**

Smart contracts play a crucial role in decentralized document verification systems, facilitating automated and trustless execution of agreements. These self-executing contracts, written in code, have their terms embedded within and are integral to the functioning of such systems. In the context of this project, smart contracts are implemented using Solidity, a programming language tailored for Ethereum blockchain. Recent literature sheds light on the significance and implementation of smart contracts in decentralized document verification systems.

In the pioneering work by Szabo (1997), smart contracts are introduced as computer protocols that facilitate, verify, or enforce the negotiation and performance of contracts. Smart contracts provide a secure and efficient means to execute agreements without intermediaries, ensuring transparency and mitigating fraud risks. The utilization of smart contracts aligns with the objective of eliminating intermediaries in decentralized document verification systems.

Solidity, the programming language for Ethereum smart contracts, is instrumental in implementing and executing smart contracts for document verification. The Ethereum documentation provides comprehensive guidance on Solidity, encompassing syntax, data structures, and functionalities. Solidity empowers the creation of self-executing code that enforces predetermined rules and conditions governing the document verification process.

To enhance the security and integrity of document verification, recent research explores the integration of cryptographic techniques with smart contracts. Memon et al. (2020) proposes a blockchain-based document verification system that utilizes smart contracts and cryptographic mechanisms to ensure secure and tamper-resistant document verification. Their work showcases the potential of combining cryptography with smart contracts in the context of document verification systems.

Moreover, the programmability and automation offered by smart contracts facilitate seamless integration with decentralized storage systems like IPFS. By leveraging IPFS, documents can be securely stored off-chain, while their hashes or references are recorded in smart contracts. This integration ensures efficient storage and retrieval of documents while maintaining the immutability and verifiability provided by blockchain technology.

Smart contracts form the foundation for trust and automation in decentralized document verification systems. They enable the execution of predefined agreements, eliminate the need for intermediaries, ensure transparency, and enhance the security of the verification process. Through the use of Solidity and cryptographic techniques, smart contracts facilitate secure and efficient document verification in the proposed project.

## **2.1.3 Smart Contract**

Smart contracts, introduced by Szabo (1997), are self-executing agreements written in code that automate and enforce predefined actions. They eliminate intermediaries, ensuring transparency and accuracy in executing agreements. In the realm of decentralized document verification systems, smart contracts play a vital role in governing the verification process. They define rules for validating and verifying documents, enhancing security and transparency (Memon et al., 2020).

## **2.1.4 Interplanetary File System (IPFS)**

Inter-Planetary File System (IPFS), a decentralized and distributed file system, facilitates secure storage and retrieval of files. It employs a content-addressable model, where files are accessed using unique content-based hashes. Recent research emphasizes the advantages of IPFS in decentralized document verification systems. Chiang et al. (2020) proposed a decentralized document verification system that utilizes IPFS for secure storage and retrieval of documents. IPFS ensures content addressing, authenticity, and data availability, enhancing the resilience of the system.

The integration of smart contracts and Inter-Planetary File System (IPFS) in decentralized document verification systems forms a robust infrastructure. Smart contracts define the verification process, leveraging the transparency and immutability of blockchain technology. IPFS, as the underlying storage layer, ensures secure off-chain storage of documents while their content-based hashes are recorded in smart contracts (Memon et al., 2020). This integration enables decentralized, tamper-proof, and scalable document verification systems.

# **2.2 Review of Related Works**

Memon et al. (2020) proposed a blockchain-based secure document verification system that leverages smart contracts and IPFS. Their system ensures secure storage and verification of documents using IPFS, while smart contracts enforce the predefined rules and conditions for document verification. This work demonstrates the feasibility and effectiveness of combining smart contracts and IPFS for decentralized document verification.

Kumar et al. (2021) presented a decentralized document verification framework using blockchain and IPFS. Their system employed smart contracts to automate and ensure the integrity of the verification process. IPFS was used for storing document metadata and hashes, enhancing data security and accessibility. The authors highlighted the transparency, immutability, and efficiency achieved through the integration of blockchain, smart contracts, and IPFS.

In a different approach, Lee et al. (2019) proposed a decentralized document verification system utilizing blockchain and cryptographic techniques. Their system leverages blockchain for immutability and transparency while employing digital signatures for document authentication. This work emphasizes the importance of cryptographic mechanisms in ensuring document integrity and authenticity within decentralized systems.

Chiang et al. (2020) presented a decentralized document verification system based on blockchain and IPFS. Their system utilized IPFS for secure storage and retrieval of documents while leveraging blockchain for transparent and verifiable document verification. The integration of blockchain and IPFS enhances data availability, accessibility, and resilience, contributing to an efficient document verification process.

## **2.3.1 Summary of Related works**

Table 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S/N | Authors | Title | Result | Limitations |
| 1 | Memon et al. (2020). | A Blockchain-based Secure Document Verification System using Smart Contracts and IPFS | The results demonstrate the feasibility and effectiveness of utilizing smart contracts and IPFS in a decentralized document verification system. | The system verified documents correctly but had no ability to delete already uploaded documents. |
| 2 | Kumar et al. (2021). | A Decentralized Document Verification Framework using Blockchain and IPFS | A decentralized document verification system that ensures the integrity of the verification process. | Verification was by admin only and organizations need to wait for the admin to get documents verified. |
| 3 | Lee et al. (2019). | A Decentralized Document Verification System Utilizing Blockchain and Cryptographic Techniques." | A decentralized document verification system that employs blockchain and cryptographic techniques, such as digital signatures, to ensure document integrity and authenticity. | No platform for deletion ability |
| 4 | Chiang et al. (2020). | A Decentralized Document Verification System Based on Blockchain and IPFS | A decentralized document verification system that leverages IPFS for secure document storage | High gas fee for transactions due to network being used. |

# **2.4 Knowledge Gap**

The knowledge gap for this project is the verification of decentralized documents on ipfs successfully available for both the administrator and organizations.

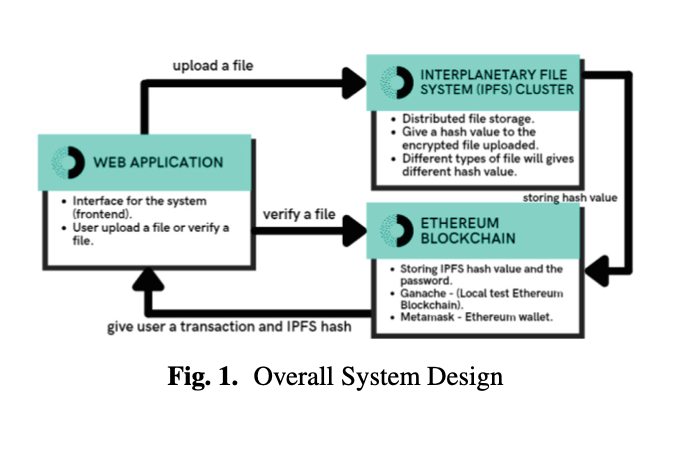
# **CHAPTER THREE**

# **ANALYSIS AND DESIGN**

# **3.1 System Analysis**

Document verification system using blockchain consists of many software designs which include Ethereum Blockchain, Truffle Suit (genache), IPFS Cluster, and web application. IPFS Cluster will be the storage for the file uploaded to the system where it has its own feature. Ethereum blockchain will be the blockchain platform to store the IPFS hash value from the IPFS Cluster which points to the file that has been stored in the IPFS. Truffle Suite is the platform that is used to develop an application uses the test Ethereum network without using any computational power and resources. Finally, web application will be used as the interface for the document verification system where the user will upload and verify a document.

Blockchain system that have been used in designing the document verification system is Ethereum blockchain. Ethereum is an open source blockchain-based platform that uses ETH as its cryptocurrency to be used for transactions. For the document verification system, Ethereum blockchain will become a place to store the identity of the file which is the IPFS hash value instead of the file because it will cost a less of computational power and resources to store a small credentials. Large files require a higher computational power which cost higher gas price to store the large files into the Ethereum blockchain. Therefore, IPFS will be use as a decentralized storage for the file and the credentials that points to the file which is the IPFS hash value will be stored into the block of the Ethereum blockchain.



**Figure** 1: Transaction layout

Truffle suite is a development environment to use Ethereum blockchain to make trans- actions and smart contracts before deploying the actual Ethereum blockchain into the decentralized application. Truffle suite consists of three components which are Truffle, Ganache and Drizzle. Truffle is a development environment where it will use a test framework to connect between the Ethereum blockchain used by Ganache with the decentralized application. Ganache is a personal Ethereum blockchain that installed locally into a computer for a decentralized application development purpose. It is a useful tool where Ganache is being used as the test Ethereum network for the decentralized application built before deploying the real Ethereum network to the application in a safe environment. The document verification system will be using Ganache as the Ethereum blockchain network as there is a constraint in the computational power.

Interplanetary File System (IPFS) is a storage file system that works in a public network in nature. In a document verification system, security of the file must be considered in a high priority as it may contains sensitive information. A file storage must be known only to the selected peers in the organizations to keep it safe and secure from the unknown people. IPFS Cluster introduced a way where sharing content of the file in the IPFS to be secure where the files will only be shared to the certain number of peers depends on the user. IPFS Cluster provides data coordination across a swarm of IPFS daemons by replicating, allocating, and tracking a global pin set distributed among multiple peers on the network. IPFS Cluster acts as a private network where only selected peers can share and view all the files uploaded into the IPFS. It is a separate system from IPFS where it is a standalone application which uses the IPFS daemon’s API. There is no centralized hosting in the cluster where every peer in the network can pin a file into the IPFS Cluster.

## **3.1.1 Fact findings**

In order to ensure that the system meets management needs once implemented, there are various methods of fact findings, which are observation, interview, questionnaire, research, site visits and document. For this study, the research methods used are books, journals, the internet and existing records.

## **3.1.2 Analysis of the existing system**

The existing document verification system has certain strengths and weaknesses that need to be assessed, these are;

1. Established Workflow: The existing system has a defined workflow for document verification, which can serve as a valuable reference point for designing the new system.
2. Familiarity: Users and stakeholders are already familiar with the existing system, which can help facilitate the transition to the new system with proper training and support.
3. Operational Experience: The existing system has been in use, providing valuable insights and knowledge about the document verification process and potential pain points.

## **3.1.3 Problem of the existing system**

The existing document verification system faces several issues that need to be addressed. These include the lack of blockchain integration, limited scalability, and inefficient file storage. By not utilizing blockchain technology, the system may encounter challenges related to data integrity, transparency, and security. The system's scalability limitations may impede its ability to handle increasing document verification demands and user loads. Inefficient file storage processes can lead to performance issues and hinder user experience. These problems emphasize the need for the new system to address these shortcomings and improve functionality, security, and usability.

## **3.1.4 Advantages of the proposed system**

The proposed decentralized document verification system offers advantages of enhanced data integrity, improved transparency, and increased security through the utilization of blockchain technology and distributed file storage.

# **3.2 Modeling the Proposed System**

System modeling is the process of developing abstract models of a system, with each model presenting model presenting a different view or perspective of that system. It is all about representing a system using graphical notation. Models help the analyst to understand the functionality of the system.

The Unified Modeling Language (UML) is a general-purpose modeling language in the field of software engineering which is designed to provide a standard way to visualize the design of a system. The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the objects of a software-intensive system and generally the system’s architectural blue print in a diagram.

In the proposed decentralized document verification system, the input image is first deployed to IPFS via the smart contract and given a hash that stands as a verification scheme.

The proposed system also includes a user interface that allows deleting, verifying and creation of authorities or organizations.

**Decentralized Document verification system**

Upload, verify, delete

Login

Create User

Delete Document

Admin

User

**Figure** 2: Proposed system use case

## **3.2.1 Proposed system workflow**

**Delete**

**Organization**

**Upload file**

**Sign Transaction**

**Verify**

**Figure** 3: Proposed system workflow

User interface must be created to make the document verification system to be user friendly. After uploading the document, the verification process is next and thereafter the ability to delete the document from IPFS.

## **3.2.2 Proposed system architecture**

**Upload**

**Smart contract**

**Blockchain Verification**

**On-chain**

**Document hash**

**Figure 4**: Proposed System architecture

In Figure 3, we see a model diagram of the proposed system, in the model a file is uploaded to the smart contract and goes through verification on the blockchain. Once verified the document is available on chain and be queried using the contract address via the assigned document hash.

# **3.3 System Design**

System design in this context refers to the process of defining the architecture, components, and functionalities of the system. It involves identifying the requirements of the system, designing the system’s architecture, selecting the necessary components, and implementing the functionalities to meet the system’s objectives. The system design process is critical to the success of the decentralized document verification system, as it lays the foundation for the accurate and efficient verification and identification of an organization’s document.

## **3.3.1 Input Design**

Input design refers to the process of designing the input data format and the methods for capturing the data into the system. In the case of the decentralized document verification system, the input design is crucial for ensuring accurate and efficient verification of documents.

**Table** 2: Input design

|  |  |  |  |
| --- | --- | --- | --- |
| **S/N** | **Field Name** | **Data Type** | **Description** |
| 1 | Document | img | Document to be verified |

## **3.3.2 Output Design**

Output design refers to the process of creating the visual representation of the system’s output that is understandable an useful to the user. The design should be simple, clear, and concise to convey the necessary information to the user effectively. In the case of the decentralized document verification system, the desired output is in text format.

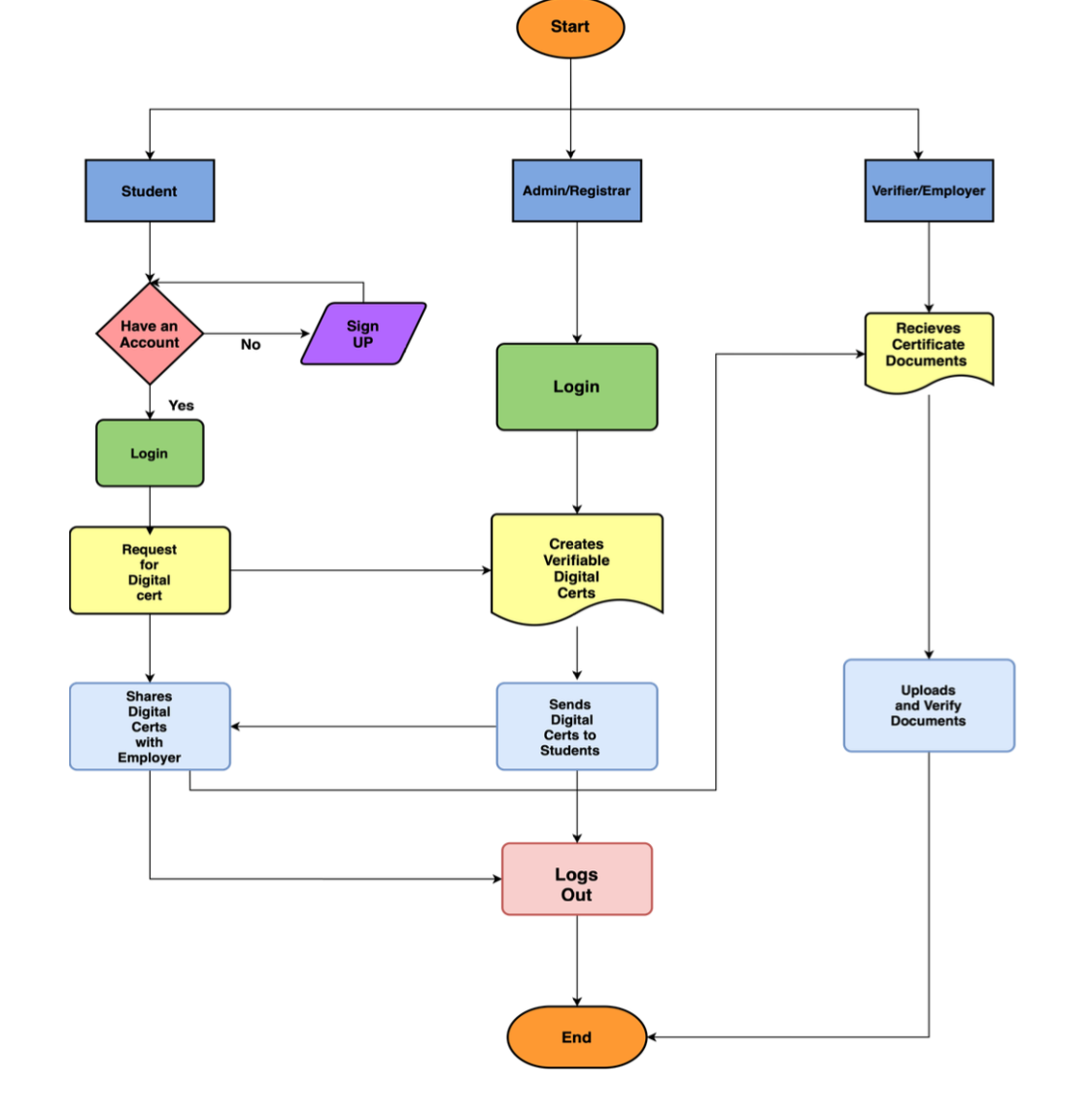
**Table** 3: Output design

|  |  |  |  |
| --- | --- | --- | --- |
| **S/N** | **Field Name** | **Data Type** | **Description** |
| 1 | Verified Document | String | Document blockchain has |
| 2 | Unverified Document | String | Failed signing of transaction |

# **3.4 Program Design**

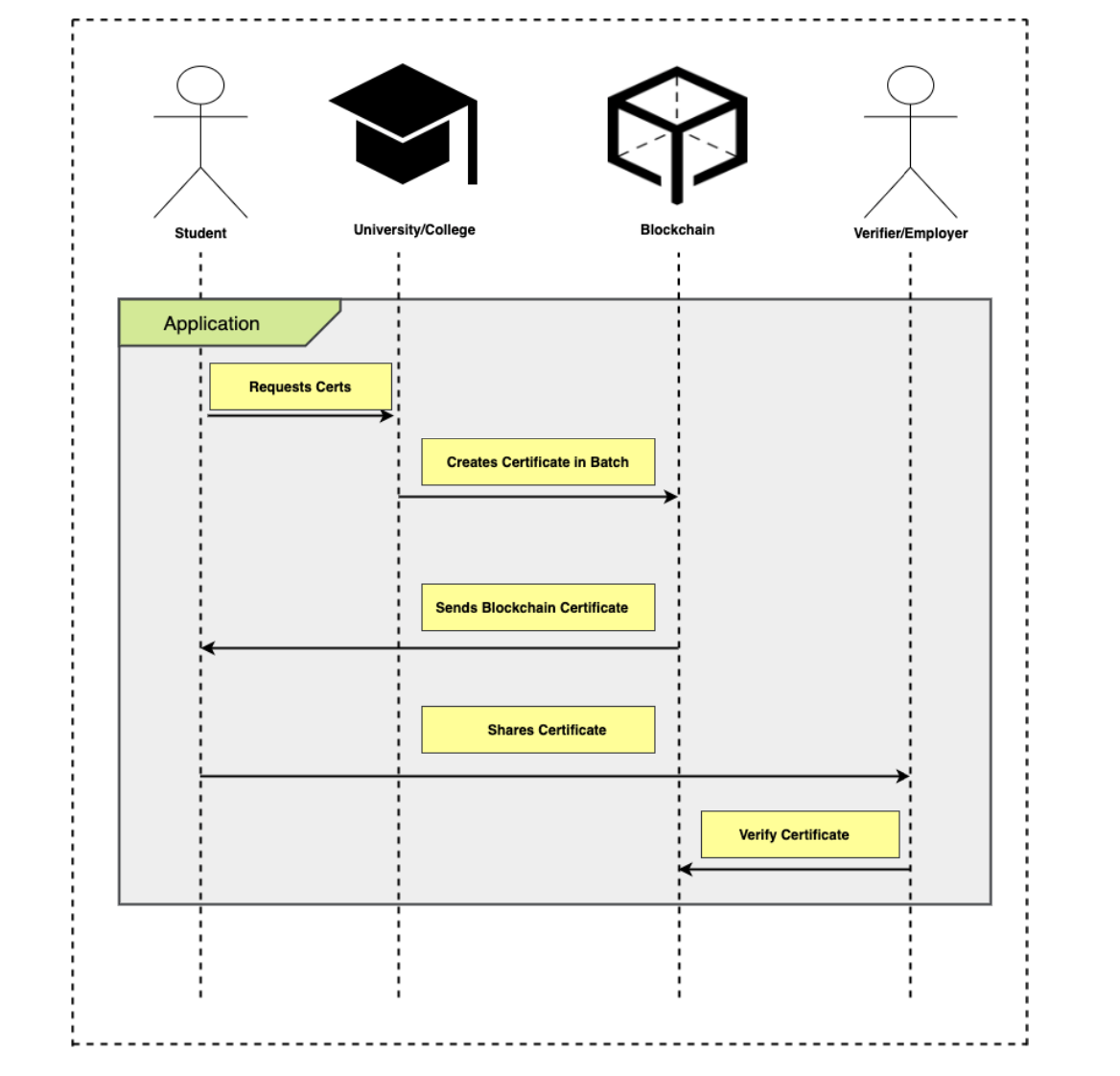
## Program design involves the conversion of system requirements into a program that can be executed on a computer system. It is an iterative process that includes research, consultation, initial design, testing and redesign. Various techniques can be used to represent program design, such as class diagrams and program algorithms. The goal of program design is to create a well-structured and efficient program that fulfils the specified requirements. It requires careful consideration of the system’s functionality, data structures, algorithms, and user interface. Through the program design process, the program is refined and optimized to ensure its effectiveness and usability.

## **3.4.1 Flow chart Diagram**



**Figure** 5: Proposed flowchart of the system

## **3.4.2 Sequence Diagram**



**Figure** 6: Proposed sequence diagram of the system

# **CHAPTER FOUR**

# **IMPLEMENTATION AND RESULTS**

# **4.1 Implementation**

The implementation process involves setting up the development environment, designing a smart contract using Solidity, integrating IPFS for secure document storage, developing a user-friendly interface using JavaScript, HTML, and CSS, interacting with the smart contract through JavaScript functions, testing and deploying the system on the desired Ethereum network, and continuously improving and maintaining the system based on user feedback and emerging technologies.

# **4.2 Program Testing**

Testing is an essential component of the project's development process to ensure the reliability and functionality of the decentralized document verification system. The following testing strategies were employed to validate the system's performance as shown in Table 2.

**Table 4: Test Cases**

|  |  |  |  |
| --- | --- | --- | --- |
| **S/N** | **Test Cases** | **Test Objectives** | **Expected Outcome** |
| 1 | T1 | To test if the system can upload any type of document. | System output should be able to receive any type of document. |
| 2 | T2 | To test if the system can verify uploaded documents. | The system output should verify uploaded documents. |
| 3 | T3 | To test if the system can send the uploaded documents to ipfs. | The system output should be able to send the uploaded documents to ipfs. |
| 4 | T4 | To test if the system can delete uploaded documents. | System output should be able to delete uploaded documents. |

# **4.3 Result**

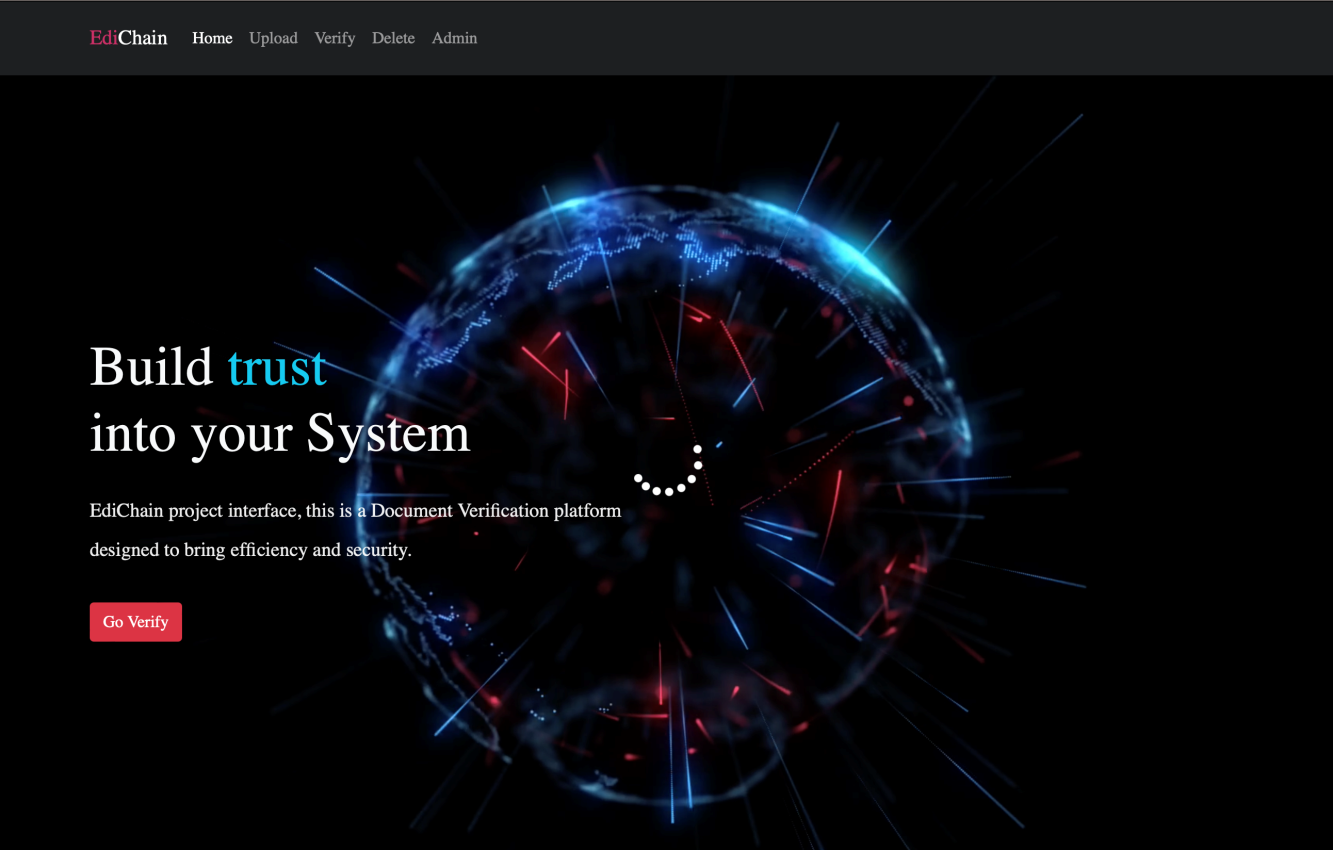
The decentralized document verification system has undergone thorough testing to evaluate the test cases outlined. The results gotten from the testing can be seen in

**Table 5: Testing Results**

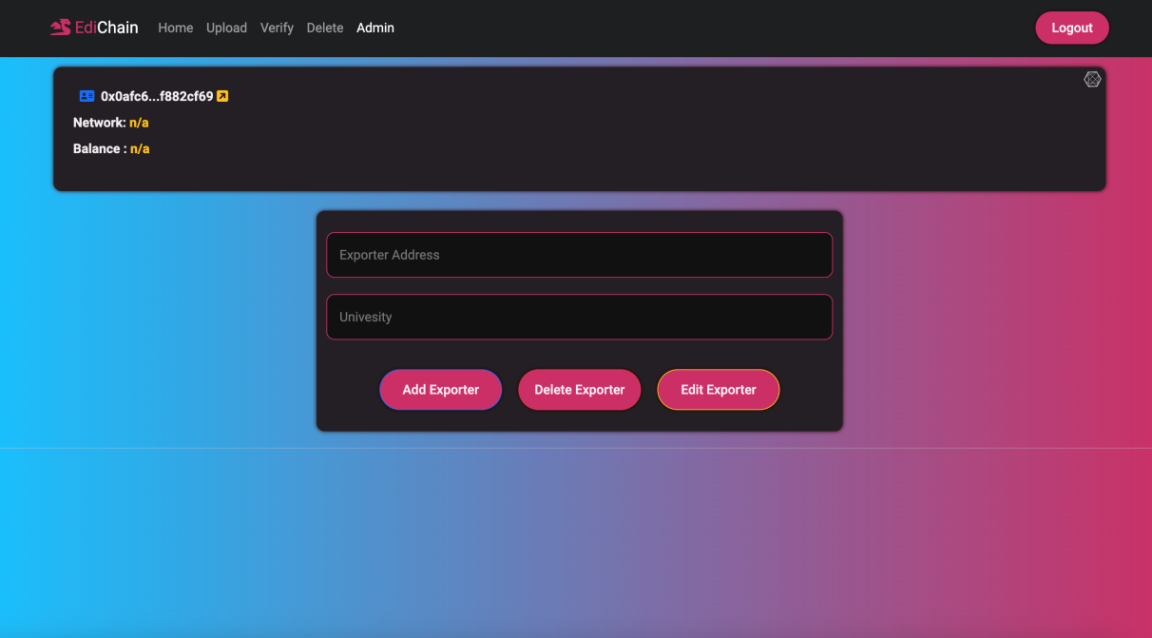
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S/N** | **Test Cases** | **Test Objectives** | **Result** | **Reference** |
| 1 | T1 | To test if the system can upload any type of document. | System was able to allow for the upload of any type of document. | Figure 9 |
| 2 | T2 | To test if the system can verify uploaded documents. | The system was able to verify uploaded documents. | Figure 10 |
| 3 | T3 | To test if the system can send the uploaded documents to ipfs. | The system output was able to send the uploaded documents to ipfs. | Figure 11 |
| 4 | T4 | To test if the system can delete uploaded documents. | System output was able to delete uploaded documents. | Figure 12 |

The results of the testing phase indicate that the decentralized document verification system meets the specified requirements and operates reliably. It provides users with a secure and efficient platform for document registration, verification, and retrieval, ensuring the integrity and authenticity of their important documents.

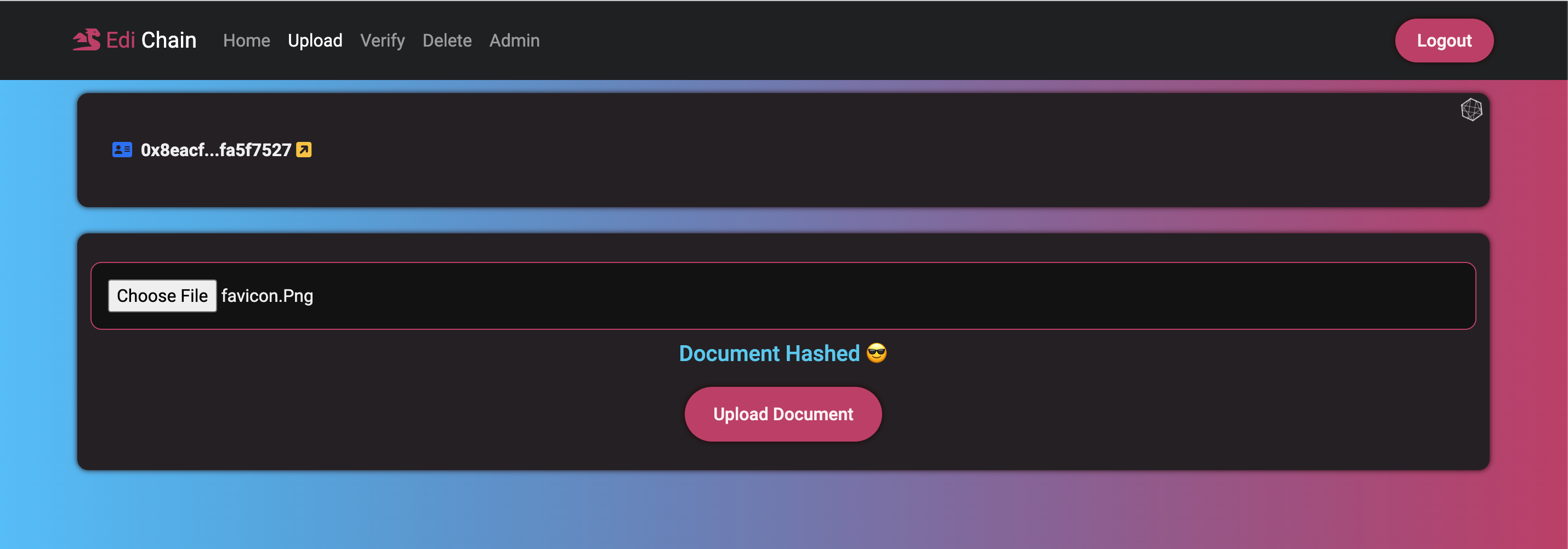
The successful testing of the system validates its readiness for deployment, assuring users of a reliable and trustworthy document verification solution.



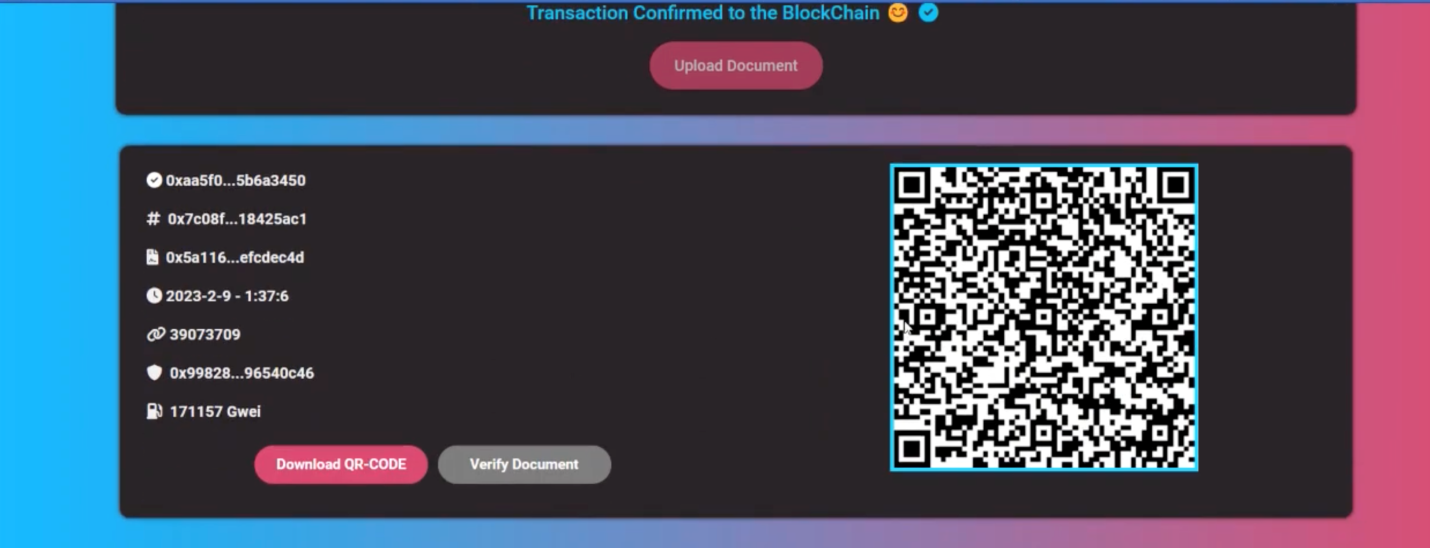
**Figure 7:** Home page of the system



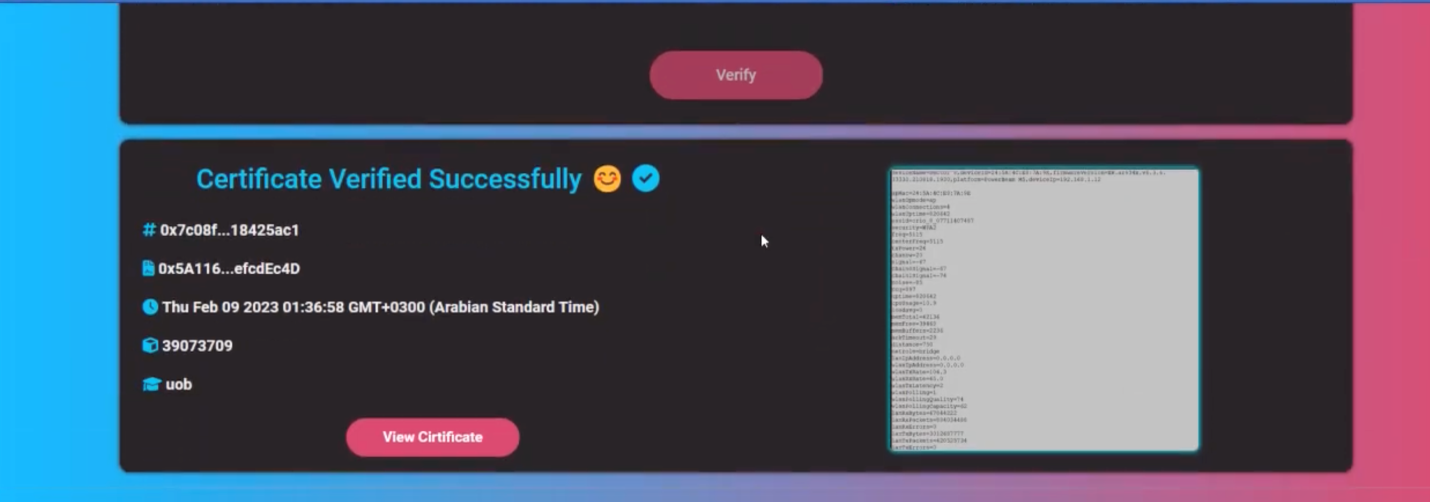
**Figure 8**: Admin section of the system



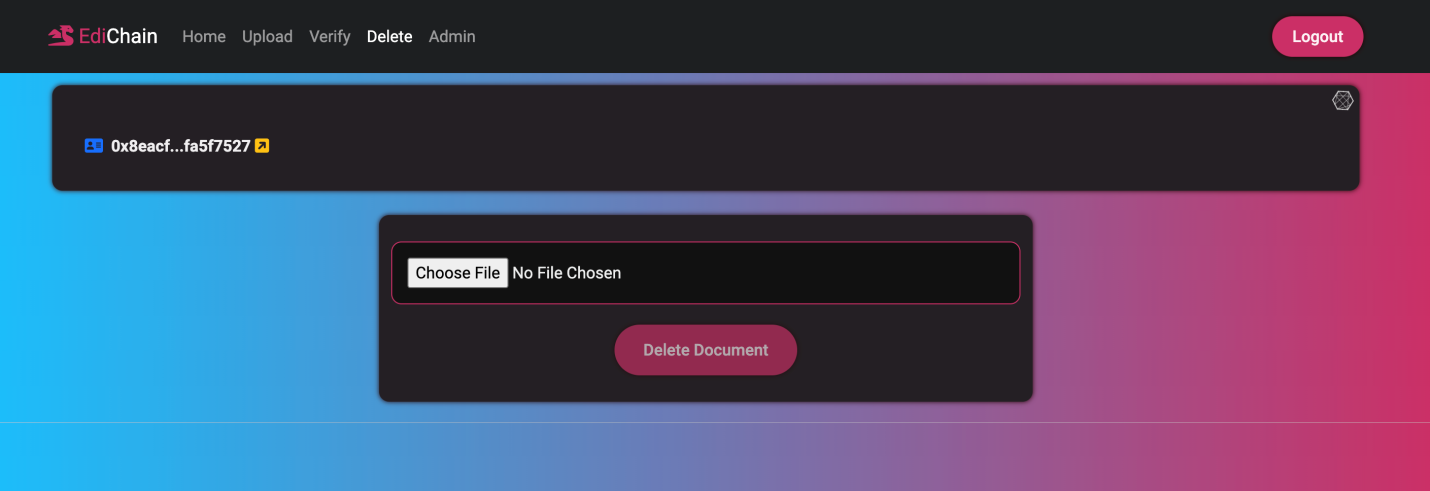
**Figure 9**: A hashed document

****

**Figure 10**: Upload section of the system

****

**Figure 11**: Verify section of the system



**Figure 12**: Delete section of the system

# **4.4 Discussion of Results**

Based on the result gotten from the test carried out, the system was able to upload documents, verify the document and delete the document.

# **4.5 System Requirement**

The decentralized document verification system has specific requirements that need to be met in order to ensure its successful implementation and operation. The following system requirements have been identified for this project:

The Hardware Requirements are:

1. 64bits PC
2. Minimum of 4GB Ram
3. Hard drive of at least 5GB free space

Software Requirement:

1. Ganache
2. Metamask
3. Haskell
4. Nodejs
5. React

# **4.6 User Documentation**

User documentation refers to the set of materials and resources that provide guidance and instructions to users on how to effectively and efficiently use a software system or application.

For the Decentralized Document Verification System, the steps are:

* Start Ganache
* Open the Project on a desired code editor (e.g Visual Studio Code)
* Go live or run the index.html file
* Click on login and authenticate using your metamask wallet address
* Go to genache and copy a private phrase key and paste it on the import wallet input field section available on your metamask. You will receive 100 ETH
* Go back to the running project and upload a document.
* Click on upload, after successfully uploading and signing the transaction verify the document by clicking on the verify button that follows up after a successful upload.
* To delete a document; login with the desired account that owns the document.
* Click on delete, upload the document you want to delete and then click on delete.
* Sign the transaction with your metamask (Ganache) password to delete the document from IPFS.

# **4.7 System Change Over**

The chosen changeover method for the Document Verification System is parallel changeover. This approach involves running both the old and new systems simultaneously for a defined period.

# **CHAPTER FIVE**

# **5.0 CONCLUSION AND RECOMMENDATION**

# **5.1 Conclusion**

The decentralized document verification system, implemented using blockchain, IPFS, smart contracts, and web technologies, has been successfully developed and tested. The system provides a secure, reliable, and efficient platform for document registration, verification, and retrieval in a decentralized manner. Through the utilization of blockchain technology, document integrity and authenticity are ensured, while IPFS enables secure and decentralized document storage.

The results obtained from the testing phase validate the system's functionality, reliability, security, user acceptance, and performance. The successful completion of unit testing, integration testing, functional testing, security testing, user acceptance testing, performance testing, and regression testing demonstrates that the system meets the specified requirements and operates as intended.

The implementation of smart contracts enables automation and execution of predefined rules and processes, enhancing the efficiency and transparency of document verification. The integration of IPFS ensures secure and decentralized storage, providing data availability and resilience.

# **5.2 Recommendations**

Based on the development and testing of the decentralized document verification system, we do recommend scalability and performance optimization such that, as the user base and document volumes grow, continuously assess and optimize the system's scalability and performance. Implement load testing and capacity planning to ensure the system can handle increased transaction volumes efficiently.

Also recommend that there should be security enhancements to regularly assess the system's security measures and apply necessary enhancements to protect against emerging threats and vulnerabilities. Stays updated with the latest security practices and consider implementing additional security features, such as two-factor authentication or advanced encryption techniques.

By implementing these recommendations, the decentralized document verification system can continue to evolve, adapt to emerging technologies and user needs, and maintain its effectiveness in providing a secure and decentralized solution for document verification.

# **REFERENCES**

Benet, J. (2018). IPFS - Content Addressed, Versioned, P2P File System. Retrieved

From<https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf>

Chiang, M. L., Chao, C. F., & Hsu, C. T. (2020). A Decentralized Document

Verification System Based on Blockchain and IPFS. Proceedings of the 19th International Conference on Information Technology Based Higher Education and Training (ITHET), 1-5.

Lu, Q., Liang, X., Xu, L., & Chen, G. (2019). A Secure and Efficient Data Sharing

Framework for Mobile Cloud Computing. IEEE Transactions on Cloud

Computing, 7(4), pp. 1082-1096. (2019)

Swan, M. (2018). Blockchain: Blueprint for a New Economy. O'Reilly Media. 7 (24) pp. 4343-376

M. D. R. Zainuddin and K. Y. Choo (2023). Blockchain Technology in academics 8

(17). Pp. 229-244, 2023

Szabo, N. (1997). Formalizing and Securing Relationships on Public Networks.

First Monday, 2(9). 2011-3067

Memon, S. S., Seo, S., Kim, H. J., & Chang, Y. (2020). A Blockchain-based Secure

Document Verification System using Smart Contracts and IPFS. Proceedings of the 13th International Conference on Human System Interaction (HSI), pp. 149-154.

Ethereum Solidity Documentation. Retrieved from

<https://soliditylang.org/documentation/>Szabo, N. (1997). Formalizing and Securing Relationships on Public Networks. First Monday, 2(9).

Memon, S. S., Seo, S., Kim, H. J., & Chang, Y. (2020). A Blockchain-based Secure

Document Verification System using Smart Contracts and IPFS. Proceedings of the 13th International Conference on Human System Interaction (HSI), 149-154.

Memon, S. S., Seo, S., Kim, H. J., & Chang, Y. (2020). A Blockchain-based Secure

Document Verification System using Smart Contracts and IPFS. Proceedings of the 13th International Conference on Human System Interaction (HSI), 149-154.

Kumar, A., Singh, P., Gupta, A., & Dhawan, A. (2021). A Decentralized Document

Verification Framework using Blockchain and IPFS. Proceedings of the 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 285-290.

Lee, J., Kim, H., Kim, J., Kim, J., & Jang, J. (2019). A Decentralized Document

VerificationSystem Utilizing Blockchain and Cryptographic Techniques. Proceedings of the 22nd International Conference on Advanced Communication Technology (ICACT), 113-118.

**APPENDIX (Program Source Code)**

window.CONTRACT= {

address:'0x493F8F111c5CC6C8F4e105A65132c1BC5B3e1C4A',

network:'https://polygon-rpc.com/',

explore:'https://polygonscan.com/',

abi: [

{

"inputs": [],

"stateMutability":"nonpayable",

"type":"constructor"

},

{

"anonymous":false,

"inputs": [

{

"indexed":true,

"internalType":"address",

"name":"\_exporter",

"type":"address"

},

{

"indexed":false,

"internalType":"string",

"name":"\_ipfsHash",

"type":"string"

}

],

"name":"addHash",

"type":"event"

},

{

"inputs": [

{

"internalType":"bytes32",

"name":"hash",

"type":"bytes32"

},a

{

"internalType":"string",

"name":"\_ipfs",

"type":"string"

}

],

"name":"addDocHash",

"outputs": [],

"stateMutability":"nonpayable",

"type":"function"

},

{

"inputs": [

{

"internalType":"address",

"name":"\_add",

"type":"address"

},

{

"internalType":"string",

"name":"\_info",

"type":"string"

}

],

"name":"add\_Exporter",

"outputs": [],

"stateMutability":"nonpayable",

"type":"function"

},

{

"inputs": [

{

"internalType":"address",

"name":"\_add",

"type":"address"

},

{

"internalType":"string",

"name":"\_newInfo",

"type":"string"

}

],

"name":"alter\_Exporter",

"outputs": [],

"stateMutability":"nonpayable",

"type":"function"

},

{

"inputs": [

{

"internalType":"address",

"name":"\_newOwner",

"type":"address"

}

],

"name":"changeOwner",

"outputs": [],

"stateMutability":"nonpayable",

"type":"function"

},

{

"inputs": [],

"name":"count\_Exporters",

"outputs": [

{

"internalType":"uint16",

"name":"",

"type":"uint16"

}

],

"stateMutability":"view",

"type":"function"

},

{

"inputs": [],

"name":"count\_hashes",

"outputs": [

{

"internalType":"uint16",

"name":"",

"type":"uint16"

}

],

"stateMutability":"view",

"type":"function"

},

{

"inputs": [

{

"internalType":"bytes32",

"name":"\_hash",

"type":"bytes32"

}

],

"name":"deleteHash",

"outputs": [],

"stateMutability":"nonpayable",

"type":"function"

},

{

"inputs": [

{

"internalType":"address",

"name":"\_add",

"type":"address"

}

],

"name":"delete\_Exporter",

"outputs": [],

"stateMutability":"nonpayable",

"type":"function"

},

{

"inputs": [

{

"internalType":"bytes32",

"name":"\_hash",

"type":"bytes32"

}

],

"name":"findDocHash",

"outputs": [

{

"internalType":"uint256",

"name":"",

"type":"uint256"

},

{

"internalType":"uint256",

"name":"",

"type":"uint256"

},

{

"internalType":"string",

"name":"",

"type":"string"

},

{

"internalType":"string",

"name":"",

"type":"string"

}

],

"stateMutability":"view",

"type":"function"

},

{

"inputs": [

{

"internalType":"address",

"name":"\_add",

"type":"address"

}

],

"name":"getExporterInfo",

"outputs": [

{

"internalType":"string",

"name":"",

"type":"string"

}

],

"stateMutability":"view",

"type":"function"

},

{

"inputs": [],

"name":"owner",

"outputs": [

{

"internalType":"address",

"name":"",

"type":"address"

}

],

"stateMutability":"view",

"type":"function"

}

],

}

asyncfunctionconnect() {

if (window.ethereum) {

try {

constselectedAccount=awaitwindow.ethereum

.request({

method:'eth\_requestAccounts',

})

.then((accounts) => {

returnaccounts[0]

})

.catch(() => {

throwError('No account selected 👍')

})

window.userAddress=selectedAccount

console.log(selectedAccount)

window.localStorage.setItem('userAddress', window.userAddress)

window.location.reload()

} catch (error) {}

} else {

$('#upload\_file\_button').attr('disabled', true)

$('#doc-file').attr('disabled', true)

// Show The Warning for not detecting wallet

document.querySelector('.alert').classList.remove('d-none')

}

}

window.onload=async () => {

$('#loader').hide()

$('#loginButton').hide()

$('#recent-header').hide()

$('.loader-wraper').fadeOut('slow')

hide\_txInfo()

$('#upload\_file\_button').attr('disabled', true)

window.userAddress=window.localStorage.getItem('userAddress')

if (window.ethereum) {

window.web3=newWeb3(window.ethereum)

window.contract=newwindow.web3.eth.Contract(

window.CONTRACT.abi,

window.CONTRACT.address,

)

if (window.userAddress.length>10) {

// let isLocked =await window.ethereum.\_metamask.isUnlocked();

// if(!isLocked) disconnect();

$('#logoutButton').show()

$('#loginButton').hide()

$('#userAddress')

.html(`<i class="fa-solid fa-address-card mx-2 text-primary"></i>${truncateAddress(

window.userAddress,

)}

<a class="text-info" href="${window.CONTRACT.explore}/address/${

window.userAddress

}" target="\_blank" rel="noopener noreferrer"><i class="fa-solid fa-square-arrow-up-right text-warning"></i></a>

</a>`)

if (window.location.pathname=='/admin.html') awaitgetCounters()

awaitgetExporterInfo()

awaitget\_ChainID()

awaitget\_ethBalance()

$('#Exporter-info').html(

`<i class="fa-solid fa-building-columns mx-2 text-warning"></i>${window.info}`,

)

setTimeout(() => {

listen()

}, 0)

} else {

$('#logoutButton').hide()

$('#loginButton').show()

$('#upload\_file\_button').attr('disabled', true)

$('#doc-file').attr('disabled', true)

$('.box').addClass('d-none')

$('.loading-tx').addClass('d-none')

}

} else {

//No metamask detected

$('#logoutButton').hide()

$('#loginButton').hide()

$('.box').addClass('d-none')

$('#upload\_file\_button').attr('disabled', true)

$('#doc-file').attr('disabled', true)

document.querySelector('.alert').classList.remove('d-none')

// alert("Please download metamask extension first.\nhttps://metamask.io/download/");

// window.location = "https://metamask.io/download/"

}

}

functionhide\_txInfo() {

$('.transaction-status').addClass('d-none')

}

functionshow\_txInfo() {

$('.transaction-status').removeClass('d-none')

}

asyncfunctionget\_ethBalance() {

awaitweb3.eth.getBalance(window.userAddress, function (err, balance) {

if (err===null) {

$('#userBalance').html(

"<i class='fa-brands fa-gg-circle mx-2 text-danger'></i>"+

web3.utils.fromWei(balance).substr(0, 6) +

'',

)

} else$('#userBalance').html('n/a')

})

}

if (window.ethereum) {

window.ethereum.on('accountsChanged', function (accounts) {

connect()

})

}

functionprintUploadInfo(result) {

$('#transaction-hash').html(

`<a target="\_blank" title="View Transaction at Polygon Scan" href="${window.CONTRACT.explore}/tx/`+

result.transactionHash+

'"+><i class="fa fa-check-circle font-size-2 mx-1 text-white mx-1"></i></a>'+

truncateAddress(result.transactionHash),

)

$('#file-hash').html(

`<i class="fa-solid fa-hashtag mx-1"></i>${truncateAddress(

window.hashedfile,

)}`,

)

$('#contract-address').html(

`<i class="fa-solid fa-file-contract mx-1"></i>${truncateAddress(

result.to,

)}`,

)

$('#time-stamps').html('<i class="fa-solid fa-clock mx-1"></i>'+getTime())

$('#blockNumber').html(

`<i class="fa-solid fa-link mx-1"></i>${result.blockNumber}`,

)

$('#blockHash').html(

`<i class="fa-solid fa-shield mx-1"></i>${truncateAddress(

result.blockHash,

)}`,

)

$('#to-netowrk').html(

`<i class="fa-solid fa-chart-network"></i>${window.chainID}`,

)

$('#to-netowrk').hide()

$('#gas-used').html(

`<i class="fa-solid fa-gas-pump mx-1"></i>${result.gasUsed} Gwei`,

)

$('#loader').addClass('d-none')

$('#upload\_file\_button').addClass('d-block')

show\_txInfo()

get\_ethBalance()

$('#note').html(`<h5 class="text-info">

Transaction Confirmed to the BlockChain 😊<i class="mx-2 text-info fa fa-check-circle" aria-hidden="true"></i>

</h5>`)

listen()

}

asyncfunctionsendHash() {

$('#loader').removeClass('d-none')

$('#upload\_file\_button').slideUp()

$('#note').html(

`<h5 class="text-info">Please confirm the transaction 🙂</h5>`,

)

$('#upload\_file\_button').attr('disabled', true)

get\_ChainID()

// Initilize Ipfs

constfile=document.getElementById('doc-file').files[0]

node=awaitIpfs.create({ repo:'Ali-ok'+Math.random() })

constfileReader=newFileReader()

fileReader.readAsArrayBuffer(file)

fileReader.onload=async (event) => {

letresult=awaitnode.add(fileReader.result)

window.ipfsCid=result.path

MyCID=window.ipfsCid+'/'

console.log('My-CID 1: '+MyCID)

}

// =================================================

if (window.hashedfile) {

constfile=document.getElementById('doc-file').files[0]

node=awaitIpfs.create({ repo:'Ali-ok'+Math.random() })

constfileReader=newFileReader()

fileReader.readAsArrayBuffer(file)

fileReader.onload=async (event) => {

letresult=awaitnode.add(fileReader.result)

window.ipfsCid=result.path

}

awaitwindow.contract.methods

.addDocHash(window.hashedfile, window.ipfsCid)

.send({ from:window.userAddress })

.on('transactionHash', function (\_hash) {

$('#note').html(

`<h5 class="text-info p-1 text-center">Please wait for transaction to be mined...</h5>`,

)

})

.on('receipt', function (receipt) {

printUploadInfo(receipt)

generateQRCode()

})

.on('confirmation', function (confirmationNr) {})

.on('error', function (error) {

console.log(error.message)

$('#note').html(`<h5 class="text-center">${error.message}😏</h5>`)

$('#loader').addClass('d-none')

$('#upload\_file\_button').slideDown()

})

}

}

asyncfunctiondeleteHash() {

$('#loader').removeClass('d-none')

$('#upload\_file\_button').slideUp()

$('#note').html(

`<h5 class="text-info">Please confirm the transaction 🙂</h5>`,

)

$('#upload\_file\_button').attr('disabled', true)

get\_ChainID()

if (window.hashedfile) {

awaitwindow.contract.methods

.deleteHash(window.hashedfile)

.send({ from:window.userAddress })

.on('transactionHash', function (hash) {

$('#note').html(

`<h5 class="text-info p-1 text-center">Please wait for transaction to be mined 😴</h5>`,

)

})

.on('receipt', function (receipt) {

$('#note').html(

`<h5 class="text-info p-1 text-center">Document Deleted 😳</h5>`,

)

$('#loader').addClass('d-none')

$('#upload\_file\_button').slideDown()

})

.on('confirmation', function (confirmationNr) {

console.log(confirmationNr)

})

.on('error', function (error) {

console.log(error.message)

$('#note').html(`<h5 class="text-center">${error.message}</h5>`)

$('#loader').addClass('d-none')

$('#upload\_file\_button').slideDown()

})

}

}

functiongetTime() {

letd=newDate()

a=

d.getFullYear() +

'-'+

(d.getMonth() +1) +

'-'+

d.getDate() +

' - '+

d.getHours() +

':'+

d.getMinutes() +

':'+

d.getSeconds()

returna

}

asyncfunctionget\_ChainID() {

leta=awaitweb3.eth.getChainId()

console.log(a)

switch (a) {

case1:

window.chainID='Ethereum Main Network (Mainnet)'

break

case80001:

window.chainID='Polygon Test Network'

break

case137:

window.chainID='Polygon Mainnet'

break

case3:

window.chainID='Ropsten Test Network'

break

case4:

window.chainID='Rinkeby Test Network'

break

case5:

window.chainID='Goerli Test Network'

break

case42:

window.chainID='Kovan Test Network'

break

default:

window.chainID='Uknnown ChainID'

break

}

letnetwork=document.getElementById('network')

if (network) {

document.getElementById(

'network',

).innerHTML=`<i class="text-info fa-solid fa-circle-nodes mx-2"></i>${window.chainID}`

}

}

functionget\_Sha3() {

hide\_txInfo()

$('#note').html(`<h5 class="text-warning">Hashing Your Document 😴...</h5>`)

$('#upload\_file\_button').attr('disabled', false)

console.log('file changed')

varfile=document.getElementById('doc-file').files[0]

if (file) {

varreader=newFileReader()

reader.readAsText(file, 'UTF-8')

reader.onload=function (evt) {

// var SHA256 = new Hashes.SHA256();

// = SHA256.hex(evt.target.result);

window.hashedfile=web3.utils.soliditySha3(evt.target.result)

console.log(`Document Hash : ${window.hashedfile}`)

$('#note').html(

`<h5 class="text-center text-info">Document Hashed 😎</h5>`,

)

}

reader.onerror=function (evt) {

console.log('error reading file')

}

} else {

window.hashedfile=null

}

}

functiondisconnect() {

$('#logoutButton').hide()

$('#loginButton').show()

window.userAddress=null

$('.wallet-status').addClass('d-none')

window.localStorage.setItem('userAddress', null)

$('#upload\_file\_button').addClass('disabled')

}

functiontruncateAddress(address) {

if (!address) {

return

}

return`${address.substr(0, 7)}...${address.substr(

address.length - 8,

address.length,

)}`

}

asyncfunctionaddExporter() {

constaddress=document.getElementById('Exporter-address').value

constinfo=document.getElementById('info').value

if (info&&address) {

$('#loader').removeClass('d-none')

$('#ExporterBtn').slideUp()

$('#edit').slideUp()

$('#delete').slideUp()

$('#note').html(

`<h5 class="text-info">Please confirm the transaction 👍...</h5>`,

)

$('#ExporterBtn').attr('disabled', true)

$('#delete').attr('disabled', true)

$('#edit').attr('disabled', true)

get\_ChainID()

try {

awaitwindow.contract.methods

.add\_Exporter(address, info)

.send({ from:window.userAddress })

.on('transactionHash', function (hash) {

$('#note').html(

`<h5 class="text-info p-1 text-center">Please wait for transaction to be mined 😴...</h5>`,

)

})

.on('receipt', function (receipt) {

$('#loader').addClass('d-none')

$('#ExporterBtn').slideDown()

$('#edit').slideDown()

$('#delete').slideDown()

console.log(receipt)

$('#note').html(

`<h5 class="text-info">Exporter Added to the Blockchain 😇</h5>`,

)

})

.on('confirmation', function (confirmationNr) {})

.on('error', function (error) {

console.log(error.message)

$('#note').html(`<h5 class="text-center">${error.message}</h5>`)

$('#loader').addClass('d-none')

$('#ExporterBtn').slideDown()

})

} catch (error) {

$('#note').html(`<h5 class="text-center">${error.message}</h5>`)

$('#loader').addClass('d-none')

$('#ExporterBtn').slideDown()

$('#edit').slideDown()

$('#delete').slideDown()

}

} else {

$('#note').html(

`<h5 class="text-center text-warning">You need to provide address & inforamtion to add </h5>`,

)

}

}

asyncfunctiongetExporterInfo() {

awaitwindow.contract.methods

.getExporterInfo(window.userAddress)

.call({ from:window.userAddress })

.then((result) => {

window.info=result

})

}

asyncfunctiongetCounters() {

awaitwindow.contract.methods

.count\_Exporters()

.call({ from:window.userAddress })

.then((result) => {

$('#num-exporters').html(

`<i class="fa-solid fa-building-columns mx-2 text-info"></i>${result}`,

)

})

awaitwindow.contract.methods

.count\_hashes()

.call({ from:window.userAddress })

.then((result) => {

$('#num-hashes').html(

`<i class="fa-solid fa-file mx-2 text-warning"></i>${result}`,

)

})

}

asyncfunctioneditExporter() {

constaddress=document.getElementById('Exporter-address').value

constinfo=document.getElementById('info').value

if (info&&address) {

$('#loader').removeClass('d-none')

$('#ExporterBtn').slideUp()

$('#edit').slideUp()

$('#delete').slideUp()

$('#note').html(

`<h5 class="text-info">Please confirm the transaction 😴...</h5>`,

)

$('#ExporterBtn').attr('disabled', true)

get\_ChainID()

try {

awaitwindow.contract.methods

.alter\_Exporter(address, info)

.send({ from:window.userAddress })

.on('transactionHash', function (hash) {

$('#note').html(

`<h5 class="text-info p-1 text-center">Please wait for transaction to be mined 😇...</h5>`,

)

})

.on('receipt', function (receipt) {

$('#loader').addClass('d-none')

$('#ExporterBtn').slideDown()

console.log(receipt)

$('#note').html(

`<h5 class="text-info">Exporter Updated Successfully 😊</h5>`,

)

})

.on('confirmation', function (confirmationNr) {})

.on('error', function (error) {

console.log(error.message)

$('#note').html(`<h5 class="text-center">${error.message}👍</h5>`)

$('#loader').addClass('d-none')

$('#ExporterBtn').slideDown()

})

} catch (error) {

$('#note').html(`<h5 class="text-center">${error.message}👍</h5>`)

$('#loader').addClass('d-none')

$('#ExporterBtn').slideDown()

$('#edit').slideDown()

$('#delete').slideDown()

}

} else {

$('#note').html(

`<h5 class="text-center text-warning">You need to provide address & inforamtion to update 😵‍💫</h5>`,

)

}

}

asyncfunctiondeleteExporter() {

constaddress=document.getElementById('Exporter-address').value

if (address) {

$('#loader').removeClass('d-none')

$('#ExporterBtn').slideUp()

$('#edit').slideUp()

$('#delete').slideUp()

$('#note').html(

`<h5 class="text-info">Please confirm the transaction 😕...</h5>`,

)

$('#ExporterBtn').attr('disabled', true)

get\_ChainID()

try {

awaitwindow.contract.methods

.delete\_Exporter(address)

.send({ from:window.userAddress })

.on('transactionHash', function (hash) {

$('#note').html(

`<h5 class="text-info p-1 text-center">Please wait for transaction to be mined 😴 ...</h5>`,

)

})

.on('receipt', function (receipt) {

$('#loader').addClass('d-none')

$('#ExporterBtn').slideDown()

$('#edit').slideDown()

$('#delete').slideDown()

console.log(receipt)

$('#note').html(

`<h5 class="text-info">Exporter Deleted Successfully 🙂</h5>`,

)

})

.on('error', function (error) {

console.log(error.message)

$('#note').html(`<h5 class="text-center">${error.message}🙂</h5>`)

$('#loader').addClass('d-none')

$('#ExporterBtn').slideDown()

$('#edit').slideDown()

$('#delete').slideDown()

})

} catch (error) {

$('#note').html(`<h5 class="text-center">${error.message}🙂</h5>`)

$('#loader').addClass('d-none')

$('#ExporterBtn').slideDown()

$('#edit').slideDown()

$('#delete').slideDown()

}

} else {

$('#note').html(

`<h5 class="text-center text-warning">You need to provide address to delete 👍</h5>`,

)

}

}

functiongenerateQRCode() {

document.getElementById('qrcode').innerHTML=''

console.log('making qr-code...')

varqrcode=newQRCode(document.getElementById('qrcode'), {

colorDark:'#000',

colorLight:'#fff',

correctLevel:QRCode.CorrectLevel.H,

})

if (!window.hashedfile) return

leturl=`${window.location.host}/verify.html?hash=${window.hashedfile}`

qrcode.makeCode(url)

document.getElementById('download-link').download=document.getElementById(

'doc-file',

).files[0].name

document.getElementById('verfiy').href=window.location.protocol+'//'+url

functionmakeDownload() {

document.getElementById('download-link').href=document.querySelector(

'#qrcode img',

).src

}

setTimeout(makeDownload, 500)

// makeDownload();

}

asyncfunctionlisten() {

console.log('started...')

if (window.location.pathname!='/upload.html') return

document.querySelector('.loading-tx').classList.remove('d-none')

window.web3=newWeb3(window.ethereum)

window.contract=newwindow.web3.eth.Contract(

window.CONTRACT.abi,

window.CONTRACT.address,

)

awaitwindow.contract.getPastEvents(

'addHash',

{

filter: {

\_exporter:window.userAddress, //Only get the documents uploaded by current Exporter

},

fromBlock: (awaitwindow.web3.eth.getBlockNumber()) -999,

toBlock:'latest',

},

function (error, events) {

printTransactions(events)

console.log(events)

},

)

}

functionprintTransactions(data) {

document.querySelector('.transactions').innerHTML=''

document.querySelector('.loading-tx').classList.add('d-none')

if (!data.length) {

$('#recent-header').hide()

return

}

$('#recent-header').show()

constmain=document.querySelector('.transactions')

for (leti=0; i<data.length; i++) {

consta=document.createElement('a')

a.href=`${window.CONTRACT.explore}`+'/tx/'+data[i].transactionHash

a.setAttribute('target', '\_blank')

a.className=

'col-lg-3 col-md-4 col-sm-5 m-2 bg-dark text-light rounded position-relative card'

a.style='overflow:hidden;'

constimage=document.createElement('object')

image.style='width:100%;height: 100%;'

image.data=`https://ipfs.io/ipfs/${data[i].returnValues[1]}`

constnum=document.createElement('h1')

num.append(document.createTextNode(i+1))

a.appendChild(image)

num.style=

'position:absolute; left:4px; bottom: -20px;font-size:4rem; color: rgba(20, 63, 74, 0.35);'

a.appendChild(num)

main.prepend(a)

}

}