

A REPORT ON SYSTEM DEVELOPMENT PROJECT

By

Resam Zaha (Roll - 1807036)

Sourav Majumder (Roll - 1807099)

Document Verification using Public Blockchain



Supervisor:

Dr. Kazi Md. Rokibul Alam

Professor, Dept. of CSE, KUET

Khulna University of Engineering & Technology

Khulna 9203, Bangladesh

December 2022

Acknowledgement

All praises and greatness go to Allah for His limitless kindness and blessings. Without His desire we would not be here as we are today.

Then, we must sense grateful to and wish to acknowledge our insightful indebtedness to **Dr. Kazi Md. Rokibul Alam**, Professor of CSE department and our Supervisor of the Project. His endless endurance, scholarly guidance, continual encouragement, constant and lively supervision, constructive criticism, priceless suggestion made it possible to come up to this phase. Without his inspiring, enthusiasm and encouragement, this work could not be completed.

Abstract

A digital document is more practical than a paper copies due to the fast rise of technology use. We need to verify these documents for a variety of living actions. Our system's goal is to establish a quick, dependable, secure file verification and application system. A document must be sent to the reputable institution that published it for verification in order to be physically verified. This labor-intensive procedure is run by humans, who are occasionally unreliable. Herein lies the value of our suggestion. The main goal of this system is to develop an Ethereum blockchain-based decentralized web application. We will be able to upload many types of documents via this system, and as the blockchain is a peer-to-peer network, any uploaded papers will be highly safeguarded and challenging to alter or break. The safety of online storage is therefore the most crucial aspect of this system, and with the aid of Ethereum blockchain technology, we can guarantee the security of sensitive and private data. Verification of these uploaded documents for any organization or authority is the next crucial component of this system. It is simpler to determine with a single click if a document is valid or invalid. Additionally, each document has a distinct CID value that serves as proof of existence. This CID value is private for only those who are using the current account. By using this method, we can apply to universities without any physical verification system and show our uploaded information for further assurance.

Table of Contents

ACKNOWLEDGEMENT	1
ABSTRACT	2
1. INTRODUCTION	4
2. USED TECHNOLOGIES & DEPENDENCIES.....	5
2.1 BLOCKCHAIN	5
2.2 ETHEREUM.....	6
2.3 SOLIDITY	7
2.4 IPFS	7
2.5 WEB3.STORAGE	7
2.6 TRUFFLE.....	8
2.7 GANACHE	9
3. METHODOLOGY TO DEVELOP THE SYSTEM.....	10
3.1 GENERATING CID FROM IPFS	10
3.2 SECURITY IN DECENTRALIZED DATABASE.....	10
<i>Why it's more secured than storing the data in a typical centralized database?</i>	11
4. IMPLEMENTATION:.....	11
4.1 DESIGNING THE CONTRACT	11
4.2 CREATING THE WEB APPLICATION.....	13
<i>Check Metamask and web3</i>	13
4.3 IPFS FOR UPLOAD/DOWNLOAD.....	14
5. APPLICATION FUNCTIONALITY	14
5.1 VERIFY DOCUMENT.....	14
5.2 UPLOAD DOCUMENT.....	16
<i>Adding a Document</i>	16
5.3 SHOW A VERIFIED DOCUMENT	17
6. RESULT ANALYSIS	19
7. LIMITATIONS.....	21
8. DISCUSSION & CONCLUSION.....	21
9. FUTURE PLANS.....	22
10. REFERENCES	22

1. Introduction

As rapid growth of use of Technology, Digital Document is more convenient than a hard copy of a document. In many actions in life, we are need of verifying these documents and apply to foreign universities online. In order to do that, we need to send it to respectful organization who published it for verification. This is a time-consuming process and is controlled by human, which is not reliable in some case. This is where our solution comes in. To avoid the unnecessary loss of time we can perform this verification and application process very quickly in a more secured way at anywhere at any time by just using this web application. There is Four part in this project

1. Storing the Main Document
2. Verify the given Document
3. Applying to University
4. Viewing that particular Document

To verify any digital document, we need the main document and match it with the given.

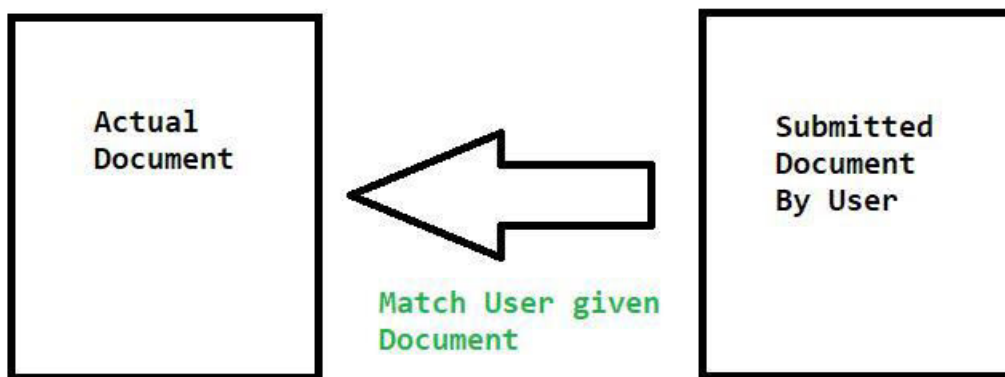


Figure 1.1.1: Document Verify

But storing and matching the whole document bit by bit will use up a lot of storage and processing power. We checked that problem using generated CID (Content Identification) for a file using IPFS and for a secured storage system used Blockchain Web3 storage

Then to check any documents validity verification process can be used and to do this the document must be uploaded before. If a document is trying to be verified but it is not yet uploaded to the blockchain an error will be shown, also if the document's CID value does not match with the uploaded document's CID value that means the document has been manipulated without the concern of its owner which is illegal so in that case an error message will also be shown in the screen to let the user know about error. Now if a document is verified successfully then it's corresponding cid value will be used for the purpose of showing the uploaded document to the verified user when applied to university.

So this whole process is done by using some extra-ordinary advance technologies. Ethereum blockchain is the main base of the system which stores all the data. To get Ethereum properly working we also need some important dependencies like **Metamask**, **Web3.js** etc .We will discuss about these dependencies in the next section in details. This whole project is finally deployed using tailwind.

2. Used Technologies & Dependencies

2.1 Blockchain

A blockchain is a decentralized, distributed, and oftentimes public, digital ledger that is used to record transactions across many computers so that any involved record cannot be altered retroactively, without the alteration of all subsequent blocks.

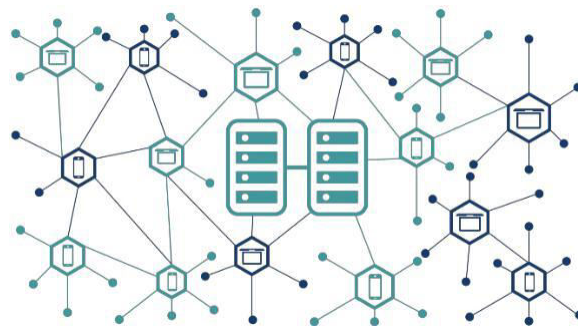


Figure 2.1.1: Blockchain

For use as a distributed ledger, a blockchain is typically managed by a peer-to-peer network collectively adhering to a protocol for inter-node communication and validating new blocks.

Once recorded, the data in any given block cannot be altered retroactively without alteration of all subsequent blocks, which requires consensus of the network majority. This allows the participants to verify and audit transactions independently and relatively inexpensively.

Blocks hold batches of valid transactions that are hashed and encoded into a Merkle tree. Each block includes the cryptographic hash of the prior block in the blockchain, linking the two. The linked blocks form a chain. This iterative process confirms the integrity of the previous block, all the way back to the original genesis block.

A blockchain carries no transaction cost. (An infrastructure cost yes, but no transaction cost.) The blockchain is a simple yet ingenious way of passing information from A to B in a fully automated and safe manner. One party to a transaction initiates the process by creating a block. This block is verified by thousands, perhaps millions of computers distributed around the net. The verified block is added to a chain, which is stored across the net, creating not just a unique record, but a unique record with a unique history. Falsifying a single record would mean falsifying the entire chain in

millions of instances. That is virtually impossible. Bitcoin uses this model for monetary transactions, but it can be deployed in many other ways.

There are three main properties of Blockchain Technology which have helped it gain widespread acclaim are as follows:

1. Decentralization
2. Transparency
3. Immutability

2.2 Ethereum

Ethereum is a distributed public block chain network that focuses on running programming code of any decentralized application. It is an open source, public, blockchain-based distributed computing platform and operating system featuring smart contract (scripting) functionality. More simply, it is a platform for sharing information across the globe that cannot be manipulated or changed.

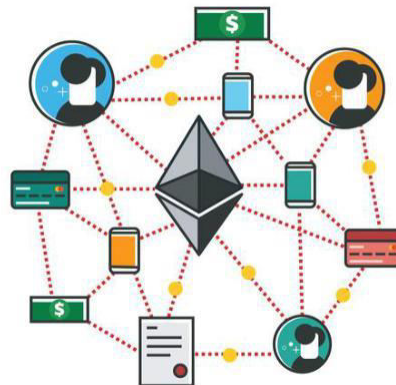


Figure 2.2.1: Ethereum

Ether is a cryptocurrency generated by the Ethereum platform and used to compensate mining nodes for computations performed. It is a decentralized digital currency, also known as ETH. In addition to being a tradable cryptocurrency, ether powers the Ethereum network by paying for transaction fees and computational services. Each Ethereum account has an ether balance and ether may be transferred from one account to another.

When anybody send ether or do anything else on the Ethereum blockchain, he must pay miners for the computation of that transaction.in his blockchain wallet, we'll set this as a fixed fee for you.

Ethereum provides a decentralized virtual machine, the Ethereum Virtual Machine (EVM), which can execute scripts using an international network of public nodes. The virtual machine's

instruction set, in contrast to others like Bitcoin Script, is Turing-complete. "Gas", an internal transaction pricing mechanism, is used to mitigate spam and allocate resources on the network.

2.3 Solidity

Solidity is an object-oriented programming language for writing smart contracts. It is used for implementing smart contracts on various blockchain platforms, most notably, Ethereum.

Solidity is a statically-typed programming language designed for developing smart contracts that run on the EVM. Solidity is compiled to bytecode that is executable on the EVM. With Solidity, developers are able to write applications that implement self-enforcing business logic embodied in smart contracts, leaving a non-reputable and authoritative record of transactions. Writing smart contracts in smart contract specific languages such as Solidity is referred to as easy (ostensibly for those who already have programming skills)

2.4 IPFS

The Inter Planetary File System (IPFS) is a protocol and peer-to-peer network for storing and sharing data in a distributed file system. IPFS uses content-addressing to uniquely identify each file in a global names-space connecting all computing devices.

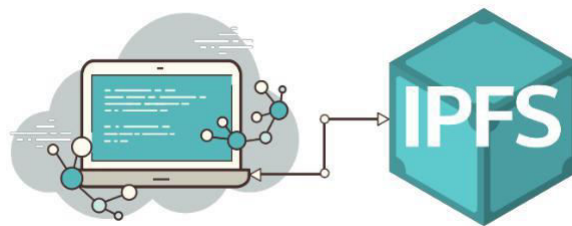


Figure 2.4.1: IPFS

IPFS allows users to not only receive but host content, in a similar manner to BitTorrent. As opposed to a centrally located server, IPFS is built around a decentralized system of user-operators who hold a portion of the overall data, creating a resilient system of file storage and sharing. Any user in the network can serve a file by its content address, and other peers in the network can find and request that content from any node who has it using a distributed hash table (DHT).

2.5 web3.storage

With web3.storage we can get all the benefits of decentralized storage and other cutting-edge protocols with the friction less experience expecting in a modern developing workflow. Under the hood, web3.storage is backed by the provable storage of FileCoin and makes data accessible to users over the public IPFS network — but when it comes down to building

application, service, or website, all one need to know is that web3.storage makes building on decentralized technologies simple.

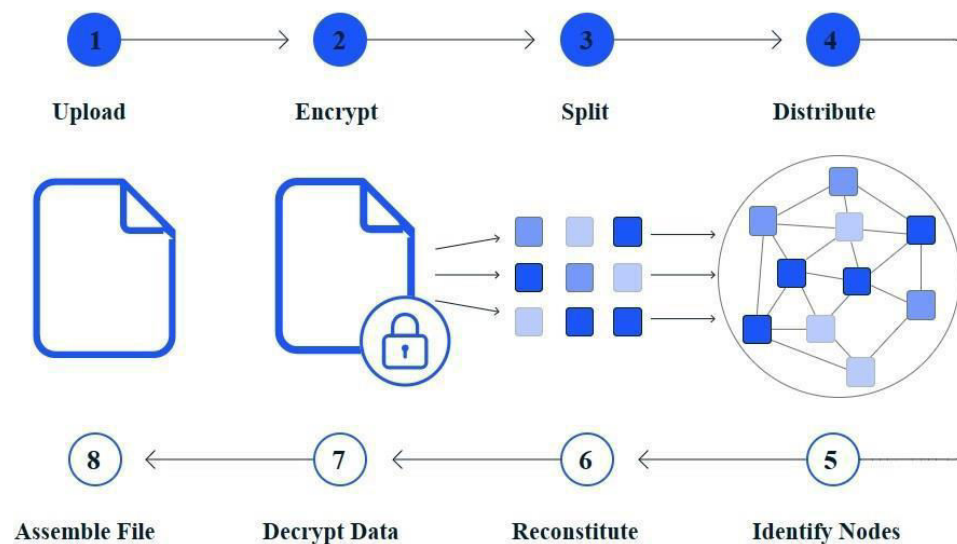


Figure 2.5.1: web3 storage

web3.storage is a suite of APIs and services that make it easy for developers and other users to interact with data in a way that is not tied to where the data is actually physically stored. It natively uses decentralized data and identity protocols that enable verifiable, data and user-centric application architectures and workflows.

At the core of the platform includes a hosted storage service which can be used to upload and persist data to make it continuously available. The platform also contains additional services that make it easier to create seamless, delightful web experiences utilizing web3 protocols.

2.6 Truffle

A world class development environment, testing framework and asset pipeline for blockchain using the Ethereum Virtual Machine (EVM), aiming to make life as a developer easier. With Truffle, we get: Built-in smart contract compilation, linking, deployment and binary management, Advanced debugging with breakpoints, variable analysis, and step functionality. Deployments and transactions through MetaMask to protect your mnemonic, External script runner that executes scripts within a Truffle environment, Interactive console for direct contract communication, Automated contract testing for rapid development, Scriptable, extensible deployment & migrations framework, Network management for deploying to any number of public & private networks.

2.7 Ganache

Ganache is a personal blockchain for rapid Ethereum and Corda distributed application development. You can use Ganache across the entire development cycle; enabling you to develop, deploy, and test your dApps in a safe and deterministic environment.

Ganache comes in two flavors: a UI and CLI. Ganache UI is a desktop application supporting both Ethereum and Corda technology. Our more robust command-line tool, ganache, is available for Ethereum development. It offers: console.log in Solidity, Zero-config Mainnet and testnet forking, Fork any Ethereum network without waiting to sync, Ethereum JSON-RPC support, Snapshot/revert state, Mine blocks instantly, on demand, or at an interval, Fast-forward time, Impersonate any account (no private keys required!), Listens for JSON-RPC 2.0 requests over HTTP/WebSockets, Programmatic use in Node.js, Pending Transactions.

2.8 Metamask

MetaMask is the trailblazing tool enabling user interactions and experience on Web3. It is currently available as a browser extension and as a mobile app on both Android and iOS devices. The purpose of this documentation is to illustrate how to build a dapp with MetaMask.

MetaMask was created to meet the needs of secure and usable Ethereum-based web sites. In particular, it handles account management and connecting the user to the blockchain.

MetaMask allows users to manage accounts and their keys in a variety of ways, including hardware wallets, while isolating them from the site context. This is a great security improvement over storing the user keys on a single central server, or even in local storage, which can allow for mass account thefts (opens new window).

This security feature also comes with developer convenience: For developers, you simply interact with the globally available ethereum API that identifies the users of web3-compatible browsers (like MetaMask users), and whenever you request a transaction signature (like `eth_sendTransaction`, `eth_signTypedData`, or others), MetaMask will prompt the user in as comprehensible a way as possible. This keeps users informed, and leaves attackers only the option of trying to phish individual users, rather than performing mass hacks (although DNS hacks can still be used for phishing en masse (opens new window)).

3. Methodology to Develop the System

3.1 Generating CID from IPFS

A content identifier, also known as a CID, is a unique value used to identify files stored on the IPFS network. This CID value doesn't indicate where the value is stored but instead refers to an address based on the content itself. CIDs are generated based on the file or folder's cryptographic hash, which means:

- The same file or folder added to two separate IPFS nodes using the same settings and parameters will produce the same CID.
- Any difference in the content, such as metadata differences, will produce a different CID.

The length of the CID will be the same for every file or folder, regardless of the file size or content.

There are currently two different versions of IPFS CIDs - version 0 (0v) and version 1 (v1).

Once a file or folder is uploaded to IPFS and a CID is returned, that CID can be used to access the stored data through an IPFS gateway. All data uploaded to IPFS is public by default since all you need to access it is associated CID. There are no permissions, user accounts, or other security settings tied to IPFS CIDs.

Gateways are used to provide workarounds for applications that don't natively support IPFS and can also be used as a sharing link for the data. An IPFS gateway can be local, private, or public, and uses the IPFS CID to provide a URL link to the content for access to the stored content.

3.2 Security in Decentralized Database

A centralized database server is controlled by single authorization, thus backdooring changes can be done using security flaws, or the database maintainer. To make the database more secure we used a decentralized way to store this hashes, Ethereum Blockchain.

Ethereum is a global, open-source platform for decentralized applications.

On Ethereum, we can write code that controls digital value, runs exactly as programmed, and is accessible anywhere in the world. Any program that runs on the Ethereum Virtual Machine (EVM) is commonly referred to as a "smart contract". We have used Solidity language to create the smart contract. Main purpose of our contract is to store hash value of the file and organizer name. Then match the hash string with the user query. As adding any value to blockchain cost gas, average cost of adding a value is 0.00011 Ether, which is approximately cost of a phone call.

Why it's more secured than storing the data in a typical centralized database?

A blockchain is a public ledger of information collected through a network that sits on top of the internet.

Blockchain technology is not a company, nor is it an app, but rather an entirely new way of documenting data on the internet

Blockchain is secured through a variety of mechanisms that include advanced cryptographic techniques and mathematical models of behavior and decision-making. Blockchain technology is the underlying structure of most cryptocurrency systems and is what prevents this kind of digital money from being duplicated or destroyed.

The information recorded on a blockchain can take on any form, whether it be denoting a transfer of money, ownership, a transaction, someone's identity, an agreement between two parties, or even how much electricity a lightbulb has used. However, to do so requires a confirmation from several of devices, such as computers, on the network. Once an agreement, otherwise known as a consensus, is reached between these devices to store something on a blockchain it is unquestionably there, it cannot be disputed, removed or altered, without the knowledge and permission of those who made that record, as well as the wider community.

4. Implementation:

4.1 Designing the Contract

Our contract will have set / get options for cid Value which will store time of adding. And we will have checker for owner and student. Finally a two functions for verifying and applying purpose. Generally, it will be look like this:

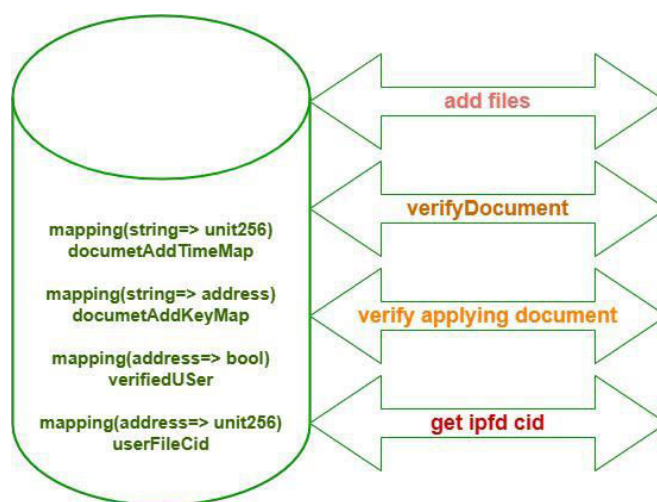


Figure 4.1.1: Smart Contract

Now, in our contract we have,

```
mapping(string => uint256) documentAddTimeMap; //contains when documents was added
mapping(string => address) documentAddKeyMap; //contains public key of the user
mapping(address => bool) verifiedUser; //contains verified users
mapping(address => string) userFilesCid; //contains user's ipfs
```

Add a document

```
function add_files(string memory ipfs_cid) public owner_check returns(bool){
    if (documentAddTimeMap[ipfs_cid] > 0) {
        emit outputResult(false);
        return false;
    }
    //add now
    uint256 timeAdded = block.timestamp;
    documentAddTimeMap[ipfs_cid] = timeAdded;
    emit outputResult(true);
    return true;
}
```

Verify a document

```
function verifyDocument(string memory ipfs_cid) public student_check returns(bool){
    if (documentAddTimeMap[ipfs_cid] > 0) {
        verifiedUser[msg.sender] = true;
        emit outputResult(true);
        return true;
    }
    emit outputResult(false);
    return false;
}
```

Verify applying document

```
function verifyApplyDocument(string memory ipfs_cid) public student_check returns(bool){
    if (documentAddTimeMap[ipfs_cid] > 0) {
        verifiedUser[msg.sender] = true;
        userFilesCid[msg.sender] = ipfs_cid;
        emit outputResult(true);
        return true;
    }
    emit outputResult(false);
    return false;
}
```

Get IPFS CID

```
//when the user will click university then his/her address will come here and return me the ipfs cid..  
function get_ipfs_cid() public student_check returns(string memory){  
    emit outputCid(userFilesCid[msg.sender]);  
    return userFilesCid[msg.sender];  
}
```

4.2 Creating the Web Application

We used Next.js to create our website. Our website will basically need to show this:

1. Upload Files
2. Verify Files
3. Apply to a university

First our website will connect to Ethereum blockchain. We will use Metamask extension for browser to connect to Ethereum blockchain and web3.js to interact with our contract.

Check Metamask and web3

```
const provider = await detectEthereumProvider();  
  
if (provider) {  
    console.log("insinde provider");  
  
    provider.request({ method: "eth_requestAccounts" });  
  
    const web3 = new Web3(provider);  
  
    const contract = new web3.eth.Contract(  
        Verify.abi,  
        Verify.networks[5777].address  
    );  
  
    const accounts = await web3.eth.getAccounts();  
  
    setAccount(accounts[0]);  
  
    setWeb3(web3);  
  
    setContract(contract);  
}
```

```

console.log({ contract });

} else {

console.error("Please install MetaMask!");

}

```

4.3 IPFS for Upload/Download

Storing a document in a blockchain will cost a lot and storing in Online Database will cost security issues. This is where IPFS comes in.

The Inter Planetary File System is a peer-to-peer hypermedia protocol designed to make the web faster, safer, and more open. IPFS has a p2p protocol to store and share file. We used it to create the file sharing service for the users.

To Upload a file in IPFS we will create a IPFS Node first and then push it.

5. Application Functionality

5.1 Verify Document

A general way of verify a document can be diagrammed as this:

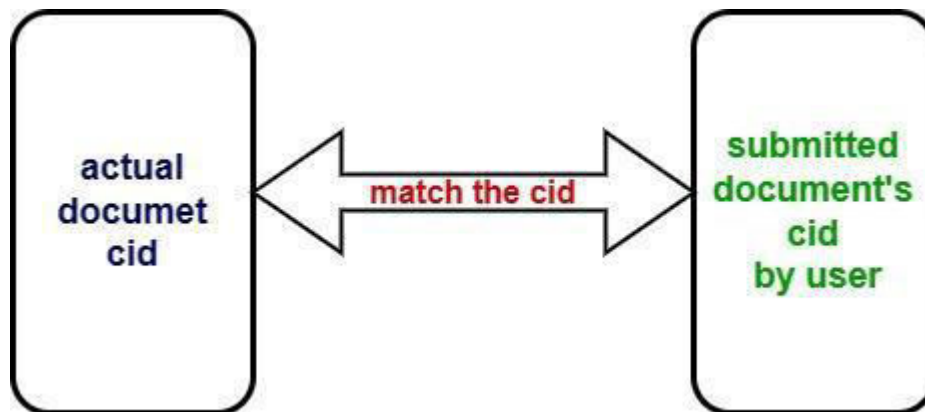


Figure 5.1.1: Verification

We need the main document that was published from the authorizer. Then we match the user given document to get result. Any change even an increase of space in the document should generate false document warning.

Verify a document

```

const verifyFile = async (e) => {

```



```

e.preventDefault();

try {

  setShowLoader(true);

  console.log("before upload");

  const client = new Web3Storage({

    token:

      "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJkaWQ6ZXRocjoweDE2MjdmMmVBN  

TQ5Y0FGQkZDZjA3QkFIZDI3MTM1NTAxQ0FmMzg3YTkiLCJpc3MiOiJ3ZWlzlXN0b3JhZ2UiLC  

JpYXQiOiE2Njk2NTEwNDY2MDAsIm5hbWUiOiJ0ZXN0aW5nIn0.2gcgFGxCeL4eR7CV8z_suiDn28i  

8kb1KLi9iB6EXnrc",

  });

  // console.log({ client });

  // console.log({ e, file });

  const cid = await client.put(file);

  console.log("verif", cid);

  // const result = getFiles(cid)

  const result = await contract.methods.verifyDocument(cid).send({

    from: account,

  });

  console.log({ result });

  console.log({ outputResult: result.events.outputResult.returnValues[0] });

  // alert(result.events.outputResult.returnValues[0]);

  if (result.events.outputResult.returnValues[0]) {

    setShowLoader(false);

    setShowModal(true);

  } else {

    setShowLoader(false);

```



```

        setShowModal(false);
    }

    console.log("after upload");

    console.log("stored files with cid:", cid);
} catch (error) {

    console.log({ error });

    console.log("You are not student");

}

setFile(null);

};

```

Now our website is ready for verify documents. But we wanted to add another service for showing the applied document.

5.2 Upload Document

Adding a Document

```

const uploadFiles = async (e) => {

    e.preventDefault();

    try {

        setShowLoader(true);

        const client = new Web3Storage({

            token:

                "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJkaWQ6ZXRocjoweDE2MjdmMmVBN  

                TQ5Y0FGQkZDZjA3QkFIZDI3MTM1NTAxQ0FmMzg3YTkiLCJpc3MiOiJ3ZWlzlXN0b3JhZ2UiLC  

                JpYXQiOiE2Njk2NTEuNDY2MDAsIm5hbWUiOiJ0ZXN0aW5nIn0.2gegFGxCeL4eR7CV8z_suiDn28i  

                8kb1KLi9iB6EXnrc",

        });

        for (let i = 0; i < file.length; i++) {

            //on the loading modal

            console.log("before upload");

```

```

    let files = [];

    files.push(file[i]);

    const cid = await client.put(files);

    //close the loading modal

    const result = await contract.methods.add_files(cid).send({

      from: account,

    });

    console.log("after upload", cid);

    console.log("result", result);

    // alert(result.events.outputResult.returnValues[0]);

    if (result.events.outputResult.returnValues[0]) {

      setShowLoader(false);

      setShowModal(true);

    } else {

      setShowLoader(false);

      setShowModal(false);

    }

  }

} catch (error) {

  console.log({ error });

  alert("You are not admin");

}

setFile(null);

};

```

5.3 Show a Verified Document

```

const getFiles = async (e) => {

  e.preventDefault();

```

```

try {

  const client = new Web3Storage({

    token:

      "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJkaWQ6ZXRocjoweDE2MjdmMmVBN  

TQ5Y0FGQkZDZjA3QkFIZDI3MTM1NTAxQ0FmMzg3YTkiLCJpc3MiOiJ3ZWlzlXN0b3JhZ2UiLC  

JpYXQiOiJlY2NjY2NTEyNDY2MDAsIm5hbWUiOiJ0ZXN0aW5nIn0.2gcgFGxCcL4eR7CV8z_suiDn28i  

8kb1KLi9iB6EXnrc".

  });

  console.log("before files", client);

  console.log({ contract });

  const result = await contract.methods.get_ipfs_cid().send({

    from: account,

  });

  console.log({ result });

  const cid = result.events.outputCid.returnValues[0];

  const res = await client.get(cid);

  const files = await res.files();

  // const unixTime = files[0].lastModified * 1000;

  // const dateString = moment.unix(unixTime).format("L");

  const dateString = moment(files[0].lastModified).format("L");

  setCid(cid);

  setName(files[0].name);

  setTime(dataString);

  console.log({ files });

} catch (error) {

  console.log({ error });

  alert("You are not student");

}

```



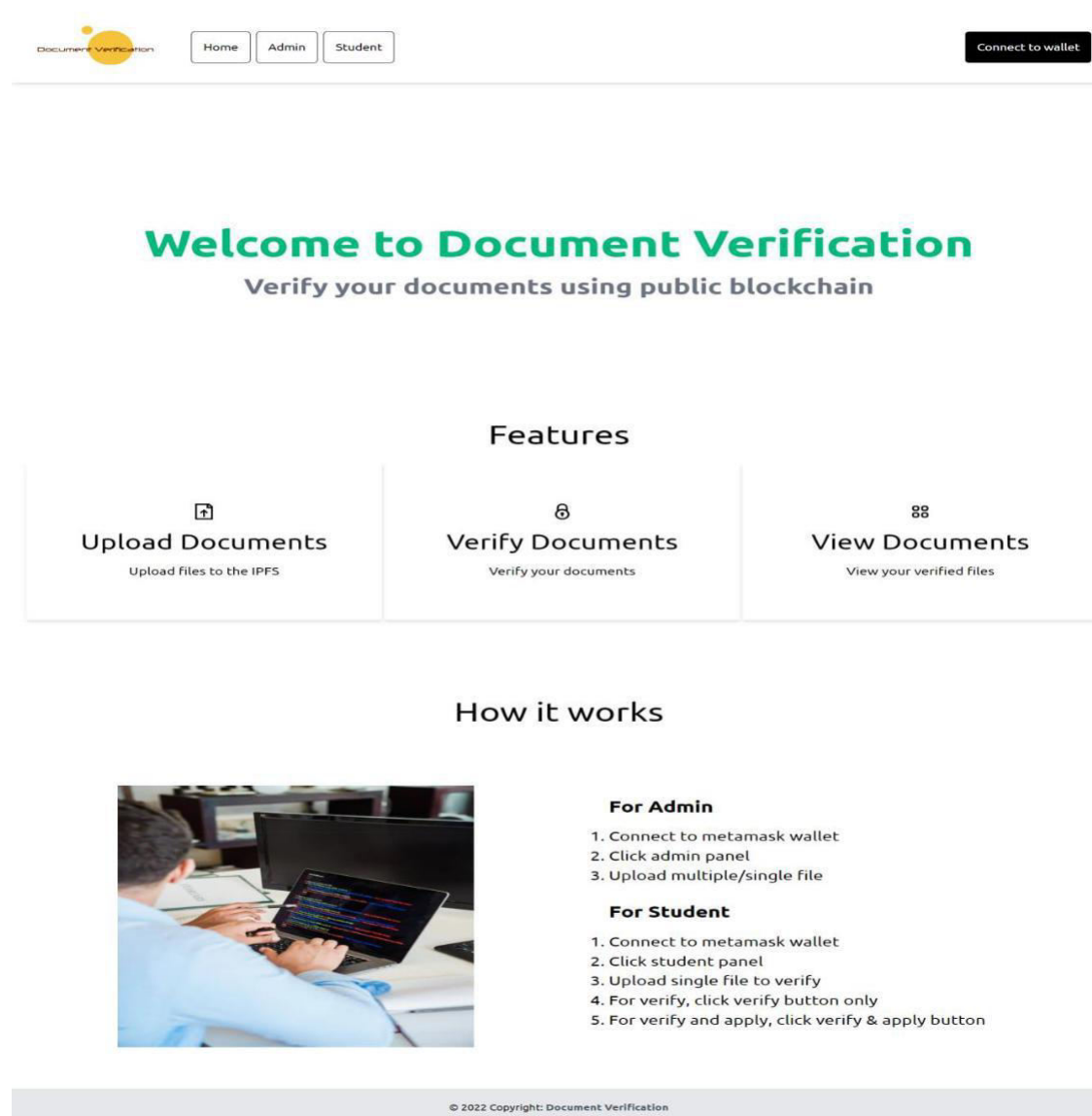
6. Result Analysis

The user interface of the website was manually created with user friendliness in mind so that a new user, who is unfamiliar with Ethereum or blockchain, can easily use it for any professional work or other purpose. The website doesn't offer a lot of information, nor does it contain any pointless data or strange behavior. It is very easy to use and suitable for everyone.


Here is some still pictures of our website:

Website Demo:

Home Page:



Admin Page:




[Home](#)[Admin](#)[Student](#)

Connect to wallet

Upload files to the IPFS


File Upload


Attach a file

Upload File

© 2022 Copyright: Document Verification

Student Page:




[Home](#)[Admin](#)[Student](#)

Connect to wallet

Verify File

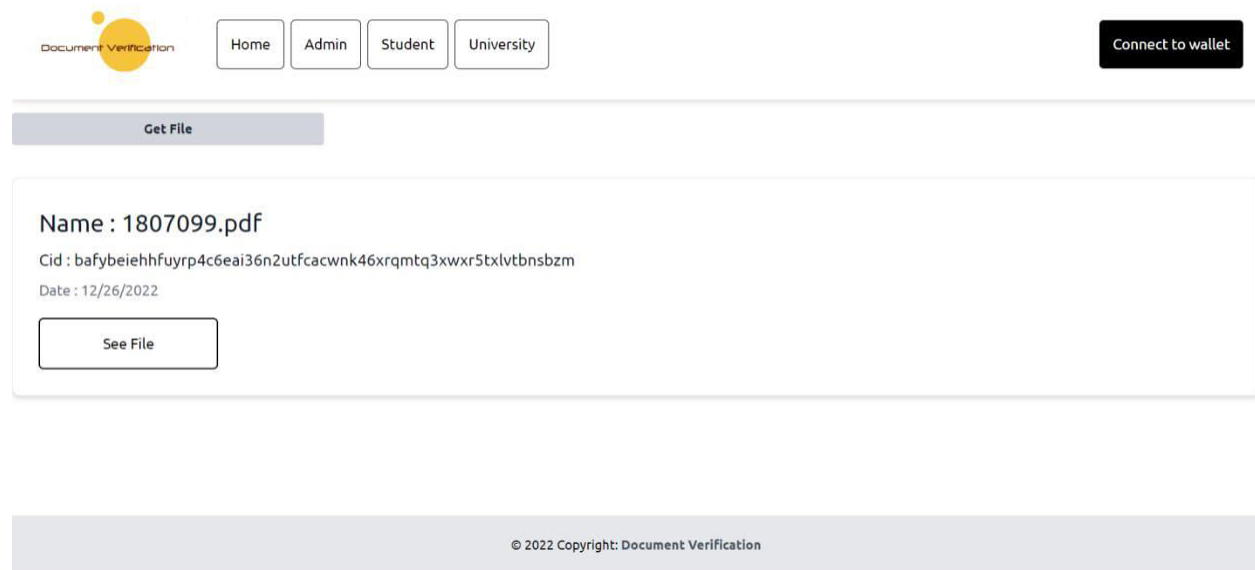
File Upload


Attach a file

VerifyVerify & Apply

© 2022 Copyright: Document Verification

University Page:



7. Limitations

Although we carefully designed and executed this project, it does have certain restrictions.

- (a) Use of crypto is necessitated
- (b) Required Internet Connection
- (c) Only a small amount of crypto—about the cost of a minute's worth of phone calls—is needed to add a document or Register

8. Discussion & Conclusion

The major goal of this project is to develop a system that will be crucial in quickly and accurately verifying a file's, document's, certificate's, land's, property's, or asset's legitimacy at the same time ensuring the data **Integrity**.

To build the System, we incorporated a number of established and secure techniques. We went into great detail about why we decided on this method of document verification and why it's a dependable and secure approach. Then we went into detail about the IPFS CID generation and how we deployed the entire system.

Because of how secure the entire procedure is, it is virtually difficult for hackers to change any document. As a result, this project is firmly supported by powerful security. Users can simply

retrieve their documents anytime they want without having to worry about their data being lost or destroyed, and they can apply to overseas universities without having to go through any needless hassle.

The project's primary goal was effectively accomplished. The project is less time consuming, more secure, and user friendly. We can continue to work on this project and add additional distinctive features. Hopefully, this program will be essential to the system for verifying digital files.

9. Future Plans

Now the student can verify only one file at one time. For apply to universities one may need to verify more than one document. We can undoubtedly ensure that this limitation will be overcome with some change.

10. References

1. <https://ethereum.org/>
2. <https://ipfs.io/>
3. <https://metamask.io/>
4. <https://web3.storage/>