Blockchain-Based Marksheet Verification: A Secure and Tamper-Proof Solution for Academic Records

Diwas B. Mishra[1], Naman S. Mishra[2], Vimlesh R. Mishra[3], Prince L. Singh[4]

[1,2,3,4]*Students, Department of Information Technology Engineering, Datta Meghe College of Engineering, Airoli, Navi Mumbai, Maharashtra, India.*

***Abstract*—**Fake academic certificates pose a significant challenge, making traditional verification methods slow, costly, and vulnerable to fraud. This paper presents a **blockchain-based marksheet verification system** using **Ethereum smart contracts** and **IPFS** for decentralized storage. Unlike existing solutions, this system supports **batch uploading**, reducing gas fees, and shifts verification costs from universities to document verifiers. We also explore **NFT-based digital certificates** as a future enhancement. Implemented using **Hardhat, Ganache,** and **Ethers.js,** this system ensures **secure, efficient, and tamper-proof verification,** offering a scalable alternative to conventional methods.

***Keywords*—**Blockchain, Smart Contracts, Ethereum, IPFS, Marksheet Verification, NFTs, Decentralized Identity, Gas Fee Optimization.

I. INTRODUCTION

Academic credential fraud is a growing concern, with **fake certificates and degree forgery** undermining trust in education systems. Traditional verification methods rely on **centralized databases, manual cross-checking, and third-party agencies**, which are often **time-consuming, costly, and prone to security breaches**. Universities struggle to authenticate student records efficiently, while employers and foreign institutions face difficulties in verifying degrees in a **tamper-proof and scalable manner**.

With advancements in **blockchain, decentralized storage, and smart contracts**, an **automated marksheet verification system** can revolutionize the process. This paper presents a **blockchain-powered verification system** that leverages **Ethereum smart contracts and IPFS** for **secure document storage and instant validation**.

Key Features of this system:

1) **Immutable Storage:** Blockchain ensures that stored data cannot be altered or tampered with.

2) **Decentralized Document Management:** IPFS is used to store **marksheets off-chain**, ensuring security and accessibility.

3) **Batch Uploading Mechanism:** Reduces **gas fees** by allowing multiple student records to be uploaded at once.

4) **Verifier-Paid Gas Fees:** Unlike traditional models where universities bear the cost, this system **shifts gas fees to the verifying entity** (e.g., recruiters or institutions).

5) **Role-Based Access Control:** Universities can **upload and manage marksheets**, while verifiers (employers, foreign institutions) can **authenticate documents**, ensuring **secure and restricted access**..

6) **Future Scope – NFT-Based Credentials:** Introducing **NFT-based academic certificates** could allow students to **own and share their degrees digitally**.

A. Problem Definition

Despite the need for **secure and scalable document verification**, most existing systems suffer from:

1. **Centralized Databases:** Susceptible to hacking, deletion, or unauthorized modifications.
2. **High Verification Costs:** Universities bear the cost of maintaining verification portals.
3. **Lack of Automation:** Manual verification delays admission and recruitment processes.
4. **Limited Scalability:** Existing blockchain solutions process **single document uploads**, leading to **high gas costs**.

B. Objectives of the Study:

This research aims to develop a **cost-effective, scalable, and fraud-resistant** marksheet verification system that:

1. **Automates marksheet verification** using blockchain and IPFS.
2. **Optimizes gas fees** by implementing batch processing and verifier-paid transactions.
3. **Ensures security and transparency** through smart contracts.
4. **Explores NFT-based academic credentials** for future scalability.

By addressing these challenges, this study **provides a practical solution** that can be **adopted by universities, employers, and credential verification bodies worldwide**.

II. LITERATURE REVIEW

Table 1: Literature Review

SR. No.	PAPER TITLE / AUTHOR	KEY FINDINGS	LIMITATIONS
1.	**Design a Document Verification System Based on Blockchain** (Muhammad Dhiyaul Rakin Zainuddin and Kan Yeep Choo)	Proposed a blockchain-based document verification system using **IPFS** and **Ethereum smart contracts** to prevent tampering and ensure transparency.	High **gas fees**, **centralized key management**, and limited scalability.
2.	**Blockchain-Based Student Certificate Verification System** (Kumutha K., Dr.S.Jayalakshmi)	Utilized **Hyperledger Fabric** for **permissioned certificate verification**, ensuring security and integrity.	Limited decentralization, **restricted to institutions within the network**, and high computational overhead.
3.	**A Secure and Reliable Academic Certificate Verification Using Blockchain** (T. Rama Reddy, P. V. G. D. Prasad Reddy , Rayudu Srinivas , Ch. V. Raghavendran3 , R. V. S. Lalitha and B. Annapurna)	Implemented a **private blockchain** for universities to **issue and verify certificates**, reducing fraud.	**No public verifiability**, requires institutions to maintain **blockchain nodes**, limiting adoption.
4.	**A Educational Document Verification through Blockchain** (Shraddha S. Magar , Rajashri G. Kanke , Charansing N. Kayte)	Combined **AES encryption** with blockchain for **dual-layer security**, enabling **tamper-proof storage**.	Potential **data retrieval delays** due to **IPFS dependency.**

III. METHODOLOGY

The **blockchain-based marksheet verification system** follows a structured approach to ensure the authenticity of academic documents. The process is divided into two main phases: **Uploader Flow** and **Verifier Flow.**

A) Uploader Flow (University/Authorized Admin):

The process begins when an **authorized admin** logs into the system. Initially, they authenticate themselves through a cryptocurrency wallet and browser extension (here , **Metamask**). Upon successful authentication from Metamask , the admin is directed to the **home page**, where they proceed to the **admin login** section to verify their credentials.

Once logged in as an admin, the user enters **student details**, such as the **student's name and PRN**, and then uploads the corresponding **marksheet in PDF format**. Before proceeding, the system performs a **validation check** to ensure that the uploaded document meets the required criteria. If the document fails this check, the admin must re-upload it correctly.

After passing validation, the admin is required to **confirm the transaction** ,i.e., paying **gas fees** , to finalize the document upload. At this stage, the document is **hashed using the Keccak-256 hashing algorithm**, ensuring a unique and tamper-proof representation of the file. The **hashed document is then stored on IPFS (InterPlanetary File System) Pinata**, generating a unique **IPFS hash**. This IPFS hash, along with the document's **computed hash**, is then **recorded onto the blockchain** via a smart contract. Additionally, relevant **student details** are also stored, enabling future verification.

This process ensures that once a document is uploaded, it becomes immutable and verifiable at any time.

B) Verifier Flow (Foreign University/Employer):

The verification process starts when a **foreign university or employer** logs into their **Metamask** wallet to authenticate their access. Once logged in, they navigate to the **verifier page**, where they can upload the **marksheet (PDF format)** provided by the student for validation. Before proceeding, the system checks whether the uploaded document meets the required input criteria. If the document is not in the correct format, the verifier must re-upload a valid file.

Once the document passes the initial check, the verifier confirms the transaction , i.e., paying **gas fees** to initiate the verification process. The system then **computes the hash of the uploaded document using Keccak-256**, ensuring it matches the hashing method used during the initial upload. The **computed hash is then queried against the blockchain** to check if a corresponding entry exists.

If a **matching hash is found**, the document is verified as authentic, and the **student's details are displayed**, confirming the legitimacy of the marksheet.

However, if no match is found, the system displays a **"Match Not Found"** message, indicating that the document is either tampered with or has never been issued by the university.

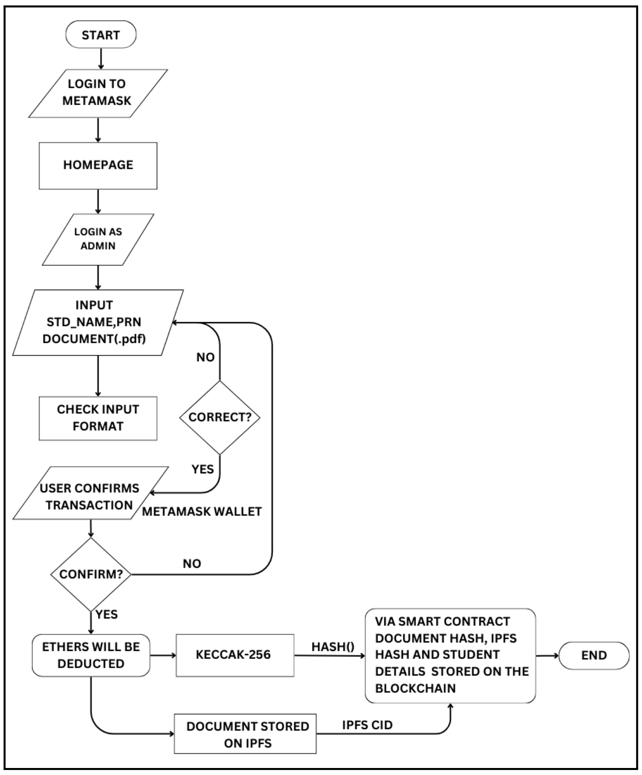


Fig 1: Uploader flow

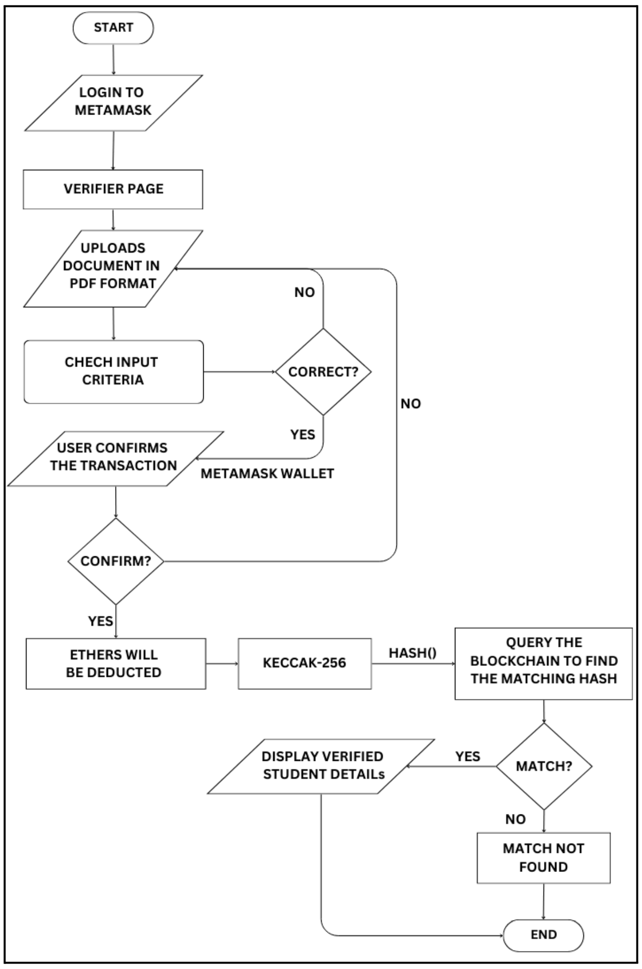


Fig 2: Verifier flow

IV. IMPLEMENTATION

A) Software Requirements

1) Frontend Technologies

React.js: It provides dynamic, responsive, and component-based web applications. React's DOM enhances performance and reusable components simplify development.

2) Backend Technologies
   1) Node.js:
   2) Express.js
   3) MongoDB

3) Blockchain (Smart Contracts and Storage)
   1) **Ethereum Blockchain**: To store document hashes student names and PRN securely and immutably.
   2) **Solidity**: Develops smart contracts for documents verification. Specifically designed for Ethereum, it ensures secure contract execution.
   3) **Ether.js**: Interacts with Ethereum blockchain and smart contract from the frontend. It is lightweight and easy to use and ideal for connecting React with the

blockchain for transaction signing and contract interaction.

4) **Hardhat**: Compiles, deploys, and tests smart contracts. Provides a development friendly environment for solidity smart contracts.
5) **Ganache**: Local Ethereum blockchain for testing contract deployments. It enables controlled local testing before live deployment.
6) **Metamask**: Manages blockchain accounts and signs transactions. Provides a secure and easy to use interface for gas fees payment.

B) Smart Contract Deployment & Interaction

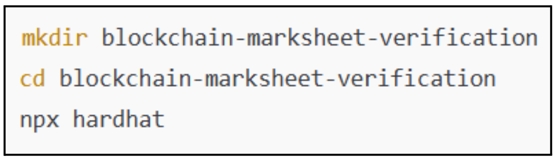
i) *Smart Contract Development:* The smart contract is written in **Solidity** and designed to store and verify document hashes securely on the Ethereum blockchain. The contract ensures that once a document's hash is recorded, it **cannot be altered** or deleted, maintaining integrity and authenticity.

ii) *Setting Up the Development Environment:* The contract is developed and deployed using **Hardhat**, a powerful Ethereum development framework. The development environment includes:

- ***Node.js & npm****: Required to install dependencies.*
- ***Hardhat****: Compiles and deploys Solidity contracts.*
- ***Ganache****: Provides a local Ethereum blockchain for testing.*
- ***Ethers.js****: Connects the smart contract with the frontend.*

iii) *Smart Contract Deployment Steps*

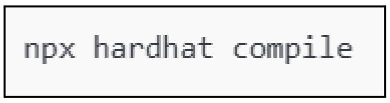
- Initialize Hardhat Project



```
mkdir blockchain-marksheet-verification
cd blockchain-marksheet-verification
npx hardhat
```

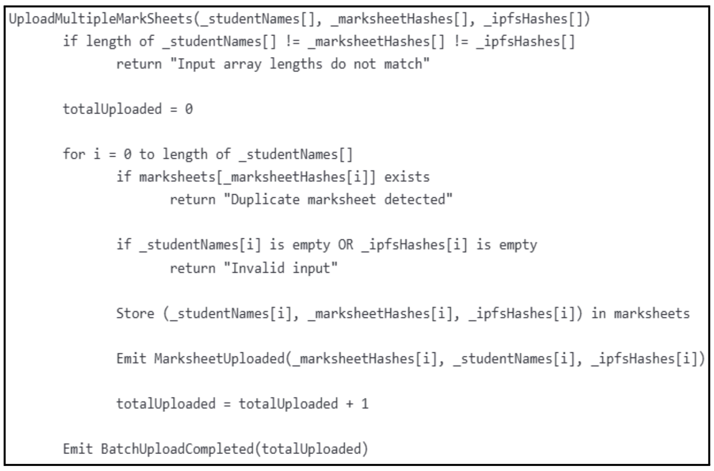
- Write the Smart Contract

- Compile the Smart Contract



```
npx hardhat compile
```

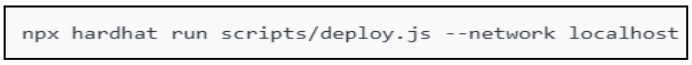
- Deploy Smart Contract to Ganache



```
UploadMultipleMarkSheets(_studentNames[], _marksheetHashes[], _ipfsHashes[])
    if length of _studentNames[] != _marksheetHashes[] != _ipfsHashes[]
        return "Input array lengths do not match"

    totalUploaded = 0

    for i = 0 to length of _studentNames[]
        if marksheets[_marksheetHashes[i]] exists
            return "Duplicate marksheet detected"

        if _studentNames[i] is empty OR _ipfsHashes[i] is empty
            return "Invalid input"

        Store (_studentNames[i], _marksheetHashes[i], _ipfsHashes[i]) in marksheets

        Emit MarksheetUploaded(_marksheetHashes[i], _studentNames[i], _ipfsHashes[i])

        totalUploaded = totalUploaded + 1

    Emit BatchUploadCompleted(totalUploaded)
```

Pseudo Code

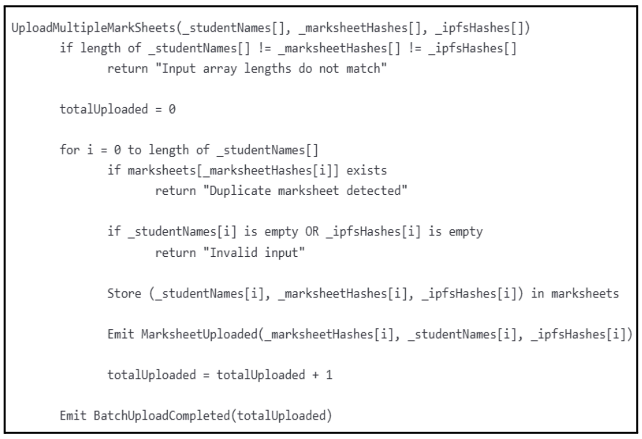
- Run Ganache & Deploy Contract



```
npx hardhat run scripts/deploy.js --network localhost
```

iv) *Interacting with the Smart Contract*

- Connecting Frontend with Smart Contract where the frontend uses **Ethers.js** to interact with the deployed smart contract.
- Uploading a Marksheet:
    a. The document is **hashed** using **Keccak-256**.
    b. The hash is **stored in IPFS (Pinata)** for decentralized storage.
    c. The **hash and student name** are **sent to the smart contract**, which records them immutably on Ethereum.

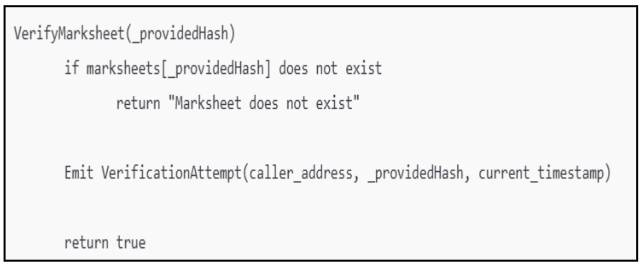


```
UploadMultipleMarkSheets(_studentNames[], _marksheetHashes[], _ipfsHashes[])
    if length of _studentNames[] != _marksheetHashes[] != _ipfsHashes[]
        return "Input array lengths do not match"

    totalUploaded = 0

    for i = 0 to length of _studentNames[]
        if marksheets[_marksheetHashes[i]] exists
            return "Duplicate marksheet detected"

        if _studentNames[i] is empty OR _ipfsHashes[i] is empty
            return "Invalid input"

        Store (_studentNames[i], _marksheetHashes[i], _ipfsHashes[i]) in marksheets

        Emit MarksheetUploaded(_marksheetHashes[i], _studentNames[i], _ipfsHashes[i])

        totalUploaded = totalUploaded + 1

    Emit BatchUploadCompleted(totalUploaded)
```

Pseudo Code

- Verifying a Marksheet:
    a. The document's hash is **generated** using **Keccak-256**.
    b. The **smart contract is queried** to check if this hash exists.

c. If **hash matches**, the document is valid; otherwise, it is tampered with.



```
VerifyMarksheet(_providedHash)
    if marksheets[_providedHash] does not exist
        return "Marksheet does not exist"

    Emit VerificationAttempt(caller_address, _providedHash, current_timestamp)

    return true
```

Pseudo Code

C) File Upload & Hash Storage

i) *File Upload to IPFS (Decentralized Storage)*

- The **PDF marksheet** is **uploaded to IPFS** via **Pinata**.
- The response provides an **IPFS hash** which acts as the unique file identifier.

ii) *File Hashing (Keccak-256 for Integrity)*

.

- **Keccak-256 hash** of the file content is generated.
- This ensures that **any change to the document will result in a different hash**, preventing tampering.

iii) *Storing on Blockchain*

- **IPFS hash**, **Keccak-256 file hash**, and **student details** are stored on the **Ethereum blockchain** via a smart contract.
- This ensures **permanent, tamper-proof storage** while keeping the actual file off-chain (reducing gas costs).

iv) *Verification Process*

- The verifier **uploads a document** to check authenticity.
- The system **hashes the document** using **Keccak-256 hash** and checks if it **matches** the hash stored on-chain.
- If the **hash matches** , then the **document is validated** and the student details are displayed , suggesting the owner of the marksheet.

D) Role Based Access Control (Node.js+Express+ MongoDB)

- The **Role-Based Access Control (RBAC) mechanism** in this system is implemented using MongoDB, ensuring secure and restricted access based on user roles. The **authentication** process begins with session-based authentication, where the frontend checks for an active session using **`getSession()`**.
- If no session is found, the user is redirected to the login page, preventing unauthorized access. The backend, built with **Node.js and Express.js**, manages user roles within **MongoDB**, allowing different levels of permissions. Only **authorized users**, such as **university administrators**, are permitted to **upload marksheets**, ensuring that document storage remains secure and valid. Meanwhile, **verifiers** can only access the verification functionality and cannot modify stored data.
- This structured access control model ensures that **students, universities, and verifiers interact with the system according to predefined roles**, enhancing security and preventing unauthorized modifications to stored records.

E) Gas Fees Optimization

Since storing data directly on the Ethereum blockchain is expensive, the system follows a **hybrid approach**:

- **Only the Keccak-256 hash and IPFS link are stored on-chain** instead of the full document, reducing storage costs.
- **Batch Upload Mechanism** helps **reduce transaction costs** by grouping multiple uploads into a **single transaction** rather than executing multiple individual transactions.

E) Batch Upload Mechanism

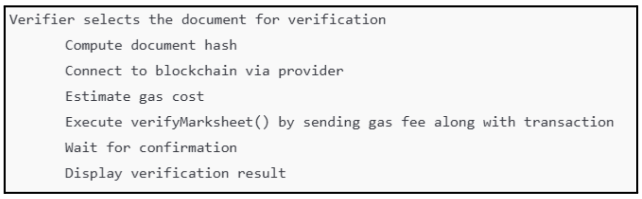
The smart contract includes a **batch upload function**, allowing the university to upload **multiple** marksheets in a single click.

- **Multiple student names, document hashes, and IPFS links** are sent together, reducing gas fees per upload.
- A **single smart contract call** processes multiple entries, preventing redundant blockchain interactions.

F) Verifier Paying Gas Fees for Verification

By default, gas fees are paid by the **transaction initiator**. In the previous model, the **university bore the entire cost**. However, in this implementation:
i) **Upload Costs:** Paid by the **university** when marksheets are uploaded.
ii) **Verification Costs:** Paid by the **verifier** when they check a document's authenticity on the blockchain.

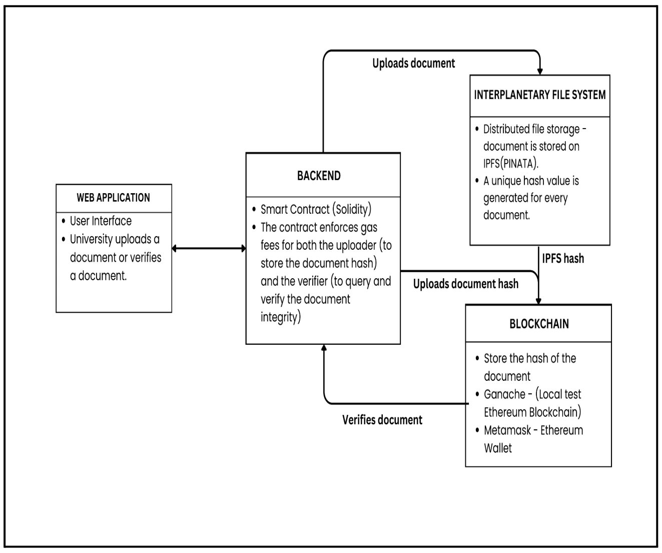
- The `verifyMarksheet()` function in the smart contract is a **transactional function**, requiring gas.
- This ensures **verifiers contribute to network costs** .



```
Verifier selects the document for verification
    Compute document hash
    Connect to blockchain via provider
    Estimate gas cost
    Execute verifyMarksheet() by sending gas fee along with transaction
    Wait for confirmation
    Display verification result
```

Pseudo Code

V. SYSTEM ARCHITECTURE AND DATA FLOW



Uploads document
INTERPLANETARY FILE SYSTEM
• Distributed file storage - document is stored on IPFS(PINATA).
• A unique hash value is generated for every document.
BACKEND
• Smart Contract (Solidity)
• The contract enforces gas fees for both the uploader (to store the document hash) and the verifier (to query and verify the document integrity)
WEB APPLICATION
• User Interface
• University uploads a document or verifies a document.
IPFS hash
Uploads document hash
BLOCKCHAIN
• Store the hash of the document
• Ganache - (Local test Ethereum Blockchain)
• Metamask - Ethereum Wallet
Verifies document

Fig. 3: Overall System Design

1. **Web Application (Frontend)**

- The interface where users (Indian universities or verifiers) can upload or verify documents.
- When a document is uploaded, the application computes its **keccak-256 hash**.
- The user submits this document to **IPFS (Pinata)** for decentralized storage.

2. **IPFS (Pinata) – Decentralized Storage**

- The document is stored on **IPFS**, which generates a **Content Identifier (CID).**
- This CID is unique and immutable for the document.
- The **CID** and **keccak-256** hash are sent to the backend (smart contract) for blockchain storage.

3. **Ethereum Blockchain (Backend - Smart Contract)**

- The **smart contract** is written in **Solidity** and deployed on the **Ethereum** blockchain.
- It stores the document hash and **CID** on the blockchain to ensure **immutability**.
- The contract enforces gas fees for:
    - Uploader: Pays gas fees to store the document hash on the blockchain.
    - Verifier: Pays gas fees to query the blockchain for verification.

4. **Verification Process**

- A verifier (e.g., a foreign university) uploads a document for verification.
- The frontend computes the **keccak-256 hash** of the uploaded document.
- The backend queries the blockchain for the **stored hash** .
- If the newly computed **keccak-256 hash** matches the stored hash, the document is verified.
- If the hash does not match, the document is tampered with.

V. RESULT

- The system accurately verifies documents by matching the uploaded hash with the blockchain-stored hash, confirming document authenticity. Also the document is stored on the IPFS. Overall, the system has demonstrated improved efficiency in the verification process, reduced administrative burdens, and enhanced security against fraud, setting a strong foundation for academic credential validation.
- The implementation results are explained below:

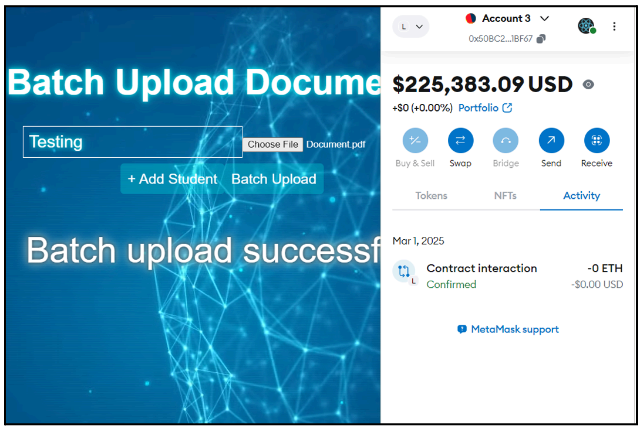
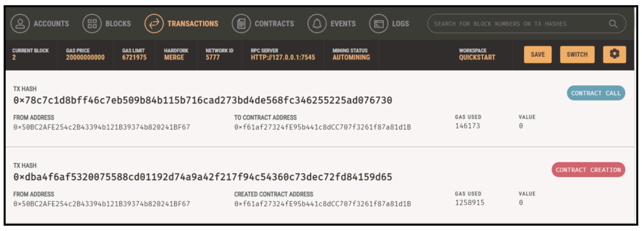


Fig. 4 : Uploading marksheet document on Blockchain.

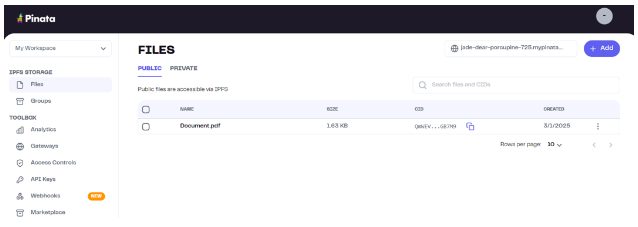
1. The university will upload students' marksheets onto the blockchain using this system, which has a user **interface** for **uploading marksheets**.



CURRENT BLOCK 2 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MERGE | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE QUICKSTART

```
TX HASH
0x78c7c1d8bff46c7eb509b84b115b716cad273bd4de568fc346255225ad076730      CONTRACT CALL
FROM ADDRESS                                 TO CONTRACT ADDRESS                          GAS USED   VALUE
0x50BC2AFE254c2B43394b121B39374b820241BF67   0xf61af27324fE95b441c8dCC707f3261f87a81d1B   146173     0

TX HASH
0xdba4f6af5320075588cd01192d74a9a42f217f94c54360c73dec72fd84159d65      CONTRACT CREATION
FROM ADDRESS                                 CREATED CONTRACT ADDRESS                     GAS USED   VALUE
0x50BC2AFE254c2B43394b121B39374b820241BF67   0xf61af27324fE95b441c8dCC707f3261f87a81d1B   1258915    0
```

Fig. 5 : Transaction on Blockchain Interface.



FILES

jade-dear-porcupine-725.mypinata...

PUBLIC PRIVATE

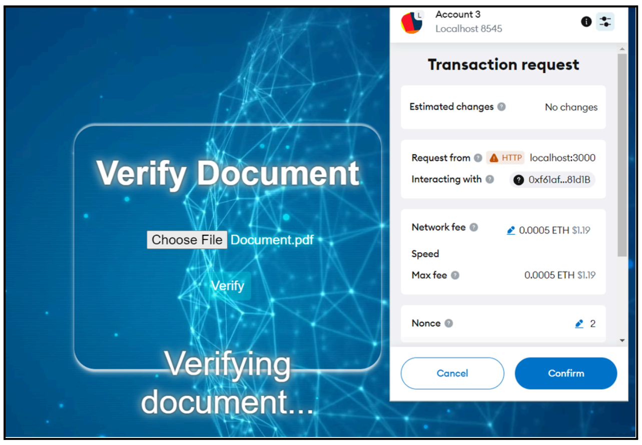
Public files are accessible via IPFS

NAME	SIZE	CID	CREATED
Document.pdf	1.63 KB	QmWEV...GB7M9	3/1/2025

Rows per page: 10

Fig. 6 : Transactions are reflected in ganache and IPFS.

2. The document uploaded by the university is reflected in **Ganache**, the local test environment for Ethereum. Also the marksheet document is saved on **IPFS**, as shown in the image.



Document.pdf

Verifying document...

Account 3
Localhost 8545

Transaction request

Estimated changes: No changes

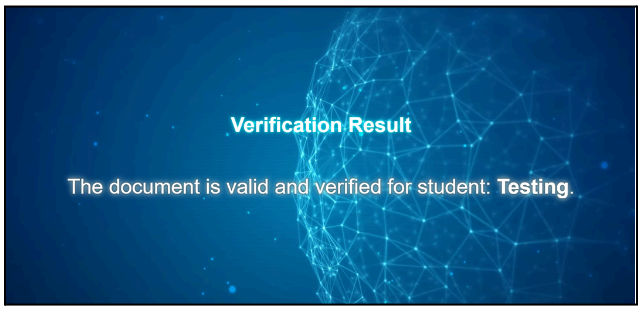
Request from: HTTP localhost:3000
Interacting with: 0xf61af...81d1B

Network fee: 0.0005 ETH $1.19
Speed
Max fee: 0.0005 ETH $1.19

Nonce: 2

Fig. 7 : Verification interface

3. For **verification** of the marksheet, the above user interface will appear, where after uploading the marksheet, it will verify it by **comparing** its hash to the hash stored on the blockchain, and if the **hashes match**, the marksheet is **legitimate**.



Verification Result

The document is valid and verified for student: **Testing**.

Fig. 8 :Verification result

VI. FUTURE SCOPE

In the future, this blockchain-based marksheet verification system can be further enhanced with the following improvements:

1. **Decentralized IPFS Storage**:
   Currently, files are stored on **centralized IPFS via Pinata**. In the future, **decentralized IPFS nodes** or **Filecoin** integration can ensure **higher security, availability, and censorship resistance**.

2. **Option for Non-Fungible Tokens (NFTs) for Marksheet Authentication:**

   - Instead of just storing hashes of documents on the blockchain, you can mint a Non-Fungible Token (NFT) for each verified marksheet.

   - Every time the **university uploads a document**, a **unique NFT** can be minted.

   - This NFT contains **immutable metadata** like Student Name , Degree Information , Issuing University , IPFS Hash , Timestamp of issuance.
   - The **NFT is owned by the student**, meaning it is stored in their blockchain account.
   - Since **NFTs cannot be altered** once minted, **no one** (including the student) can modify their marksheet.

   - Foreign universities/employers can simply **verify ownership & authenticity** by checking the NFT's metadata on the blockchain.

   - This eliminates **delays, manual verification, and paperwork**.

3. **Role-Based Access with Smart Contract**:

   - Storing a **Metamask account** address directly in the smart contract to upload a document can result in **implementing RBAC**.

   - In the present solution , RBAC is implemented by storing user details like username and passwords in **MongoDB**.

4. **Layer 2 Solutions for Gas Optimization:**

   - As the adoption of the blockchain-based marksheet verification system grows, gas fees on Ethereum can become a bottleneck. To optimize costs and scalability, implementing Layer 2 (L2) solutions is a promising future enhancement.

- A viable option is Polygon PoS, which operates as a fast and low-cost sidechain while still leveraging Ethereum's security.

- This approach allows universities and verifiers to interact with the system more efficiently. Implementing Layer 2 solutions will ensure the system remains scalable, cost-effective, and future-proof as usage increases.

VII. CONCLUSION

The blockchain-based marksheet verification system provides a secure, decentralized, and tamper-proof solution for academic credential validation. By leveraging blockchain's immutability and transparency, it eliminates fraud and unauthorized modifications, ensuring the authenticity of marksheets. The system enhances trust among universities and students by offering a verifiable digital record of academic marksheets.

Unlike traditional verification methods, which are time-consuming and prone to forgery, this system provides instant validation without intermediaries. The use of smart contracts automates verification, reducing administrative workload and enhancing efficiency. Furthermore, storing marksheets on a blockchain ensures data integrity and prevents any unauthorized alterations.

This system not only benefits students but also helps educational institutes to maintain credibility and simplifies the marksheet verification process. By integrating blockchain with a user-friendly interface, the system ensures accessibility while maintaining high security standards. Future enhancements may include interoperability with multiple blockchains.

VIII. REFERENCES

1) Muhammad Dhiyaul, Rakin Zainuddin and Kan Yeep Choo(B) "**Design a Document Verification System Based on Blockchain Technology**" - https://doi.org/10.2991/978-94-6463-082-4_23

2) Kumutha.K, Dr.S.Jayalakshmi "**Blockchain Technology and Academic Certificate Authenticity-Review**" **-** June 2021 Conference: ICOECA 2021

3) T. Rama Reddy, P. V. G. D. Prasad Reddy, Rayudu Srinivas, Ch. V. Raghavendran, R. V. S. Lalitha and B. Annapurna Rama Reddy "**Proposing a reliable method of securing and verifying the credentials of graduates through blockchain**" - *EURASIP Journal on Information Security* volume 2021, Article number: 7 (2021) https://doi.org/10.1186/s13635-021-00122-5

4) Shraddha S. Magar, Rajashri G. Kanke, Charansing N. Kayte, Dr. B. A. M. "**Educational Document Verification through Blockchain: Literature Review**" - IJSRST publication Print ISSN: 2395-6011 | Online ISSN: 2395-602X February 2024 https://doi.org/10.32628/IJSRST

5) Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang "**An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends**" - 2017 IEEE 6th International Congress on Big Data https://doi.org/10.1109/BigDataCongress.2017.85

6) Ruhi Taş , Ömer Özgür Tanrıöver "**Building A Decentralized Application on the Ethereum Blockchain**" - 3rd ISMSIT, Ankara,Turkey 2019 https://doi.org/10.1109/ISMSIT.2019.8932806

7) Lee, W.-M. (2019). *Beginning Ethereum smart contracts programming: With examples in Python, Solidity, and JavaScript*. Apress.