HAUthentiCred: An Ethereum-Based Academic Credential Storage

Buduan, Mark Joseph L.

Keele, Regine Ann G.

Nacpil, Marc Jayson B.

Salangsang, John Patrick.

CS – 401

Background of the Project

The management and verification of academic credentials have always been a critical aspect of educational and professional institutions. Traditional systems of credential handling often rely on physical documentation, which is prone to loss, damage, and fraud. With the increasing importance of digital records in our technologically advanced society, the need for a secure, reliable, and tamper-proof method of managing these credentials has become evident. This necessity led to the development of HAUthentiCred, an Ethereum-based academic credential storage system designed to address these challenges by leveraging blockchain technology.

The traditional approach to managing academic credentials, primarily dependent on physical documents and centralized digital systems, presents several issues related to data integrity, security, and accessibility. Physical documents such as certificates and transcripts are vulnerable to damage, loss, and unauthorized alterations. Even with digital systems, centralized databases are susceptible to hacking, data breaches, and manipulation. The need for a secure, decentralized method to store and verify academic credentials is crucial to prevent fraud and ensure that these records remain authentic and unaltered over time.

Blockchain technology offers a solution to these problems through its decentralized, transparent, and immutable ledger system. Blockchain's distributed nature ensures that once data is recorded, it cannot be altered or tampered with, thereby providing a high level of data integrity and security. In the context of academic credentials, this means that each record can be independently verified by any party without the need for a central authority, reducing the chances of fraudulent activities. By utilizing blockchain, HAUthentiCred aims to create a trusted platform where students can seamlessly interact with verified academic data.

The target users for HAUthentiCred include Holy Angel University (HAU) students, and academic staff. For students, this platform provides a reliable way to store and manage their academic credentials, ensuring that their achievements are secure and easily accessible. Academic staff benefit from an efficient system that simplifies the process of credential verification, allowing them to focus more on educational responsibilities.

The implementation of HAUthentiCred not only addresses the need for secure academic record-keeping but also aligns with global trends towards decentralized applications (dApps) and digital identities. By leveraging Ethereum blockchain technology, this project ensures that academic credentials are stored in a way that is both tamper-proof and easily verifiable. The use of Web3.js enables the application to interact with the Ethereum blockchain, allowing for seamless transactions and data retrieval. This integration ensures that academic credentials can be securely stored as transactions on the blockchain, providing a permanent and transparent record of a student's academic achievements.

Finally, HAUthentiCred aims to revolutionize the way academic credentials are managed and verified by providing a decentralized, secure, and user-friendly platform. This system not only benefits students by safeguarding their achievements but also aids educational institutions and employers by streamlining the verification process. By embracing blockchain technology, HAUthentiCred sets a new standard in academic credential storage, ensuring that the integrity and authenticity of educational records are maintained in an increasingly digital world.

Project Proposal Architecture

This project proposal outlines an Ethereum-based system designed to manage and verify academic credentials using blockchain technology. The architecture consists of a detailed analysis of the Traditional Approach, Technological Approach, and the Proposed System. The proposed system uses HTML/CSS, JavaScript, IPFS, Solidity (Smart Contracts), and a connection to a MetaMask wallet.

Traditional Approach

The traditional approach to academic credential verification is predominantly manual, involving the following steps:

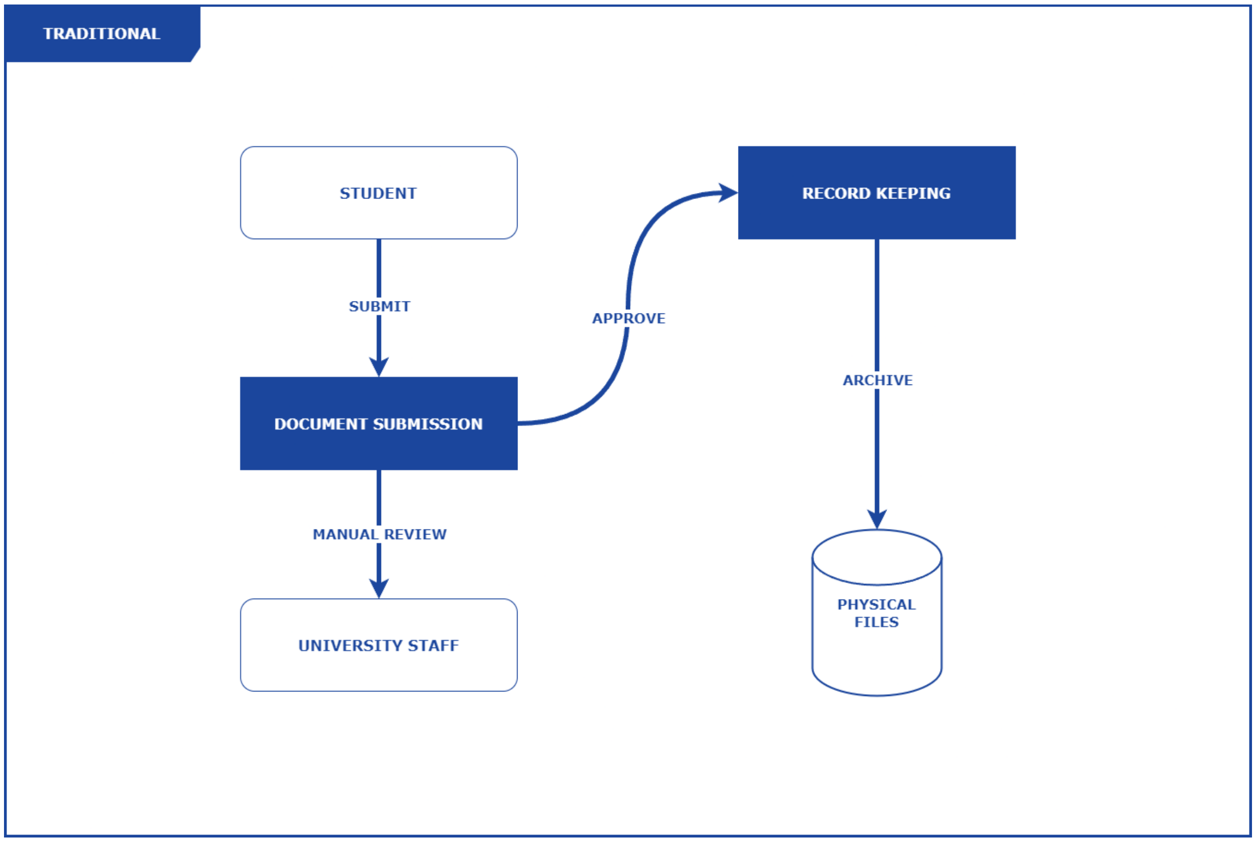
Document Submission. Students submit their credentials (e.g., certificates, diplomas, and other proof of academic achievements) either physically or as scanned copies to the university administration. This submission is usually done via email, paper forms, or through university portals.

Document Verification. University staff, such as professors or administrative personnel, manually review each document to confirm its authenticity. The verification process may involve cross-referencing with other records or contacting the issuing institution to confirm the validity of the credentials. Errors or discrepancies in documentation can result in delays, repeated submissions, and further scrutiny.

Approval and Record-Keeping. Verified documents are approved and recorded manually in university databases or stored in physical files. A hard copy of these credentials may also be kept for future reference or archival purposes. There is limited automation, leading to slow processing times and a high risk of errors.

Figure 1

Traditional Approach Diagram



Challenges of the Traditional Approach

The traditional approach to academic credential management, which relies heavily on manual processes and physical documentation, poses several significant challenges. First, the manual verification process is inherently time-consuming and resource-intensive. It requires significant effort from both students and administrators to verify and authenticate academic credentials, which can lead to delays in procedures that rely on timely verification. Additionally, the reliance on human input and handling increases the risk of errors and fraudulent activities. Document tampering and the possibility of overlooking fraudulent credentials are common concerns when credentials are manually processed. The lack of transparency is another critical issue within this traditional system. Students and employers often struggle to obtain reliable information regarding the status and authenticity of academic credentials due to the limited visibility into the verification process. Lastly, data integrity and security risks are inherent in this approach. Physical documents are susceptible to damage, loss, and unauthorized alterations, while centralized databases can be targeted for data breaches, potentially compromising the sensitive information they contain.

Technological Approach

The technological approach enhances the traditional method by introducing software solutions and automation to improve the efficiency, security, and transparency of credential verification processes. It typically involves the following components:

Digital Submissions. Students upload their credentials through an online portal or web application, using secure authentication methods to ensure that only authorized users can access the system. Documents are submitted in digital formats such as PDFs, JPEGs, or DOCX files.

Automated Verification Systems. Software algorithms and machine learning models are utilized to perform an initial analysis of the documents, checking for anomalies or inconsistencies. The system may cross-reference documents against online databases or use pattern recognition to detect fraudulent documents.

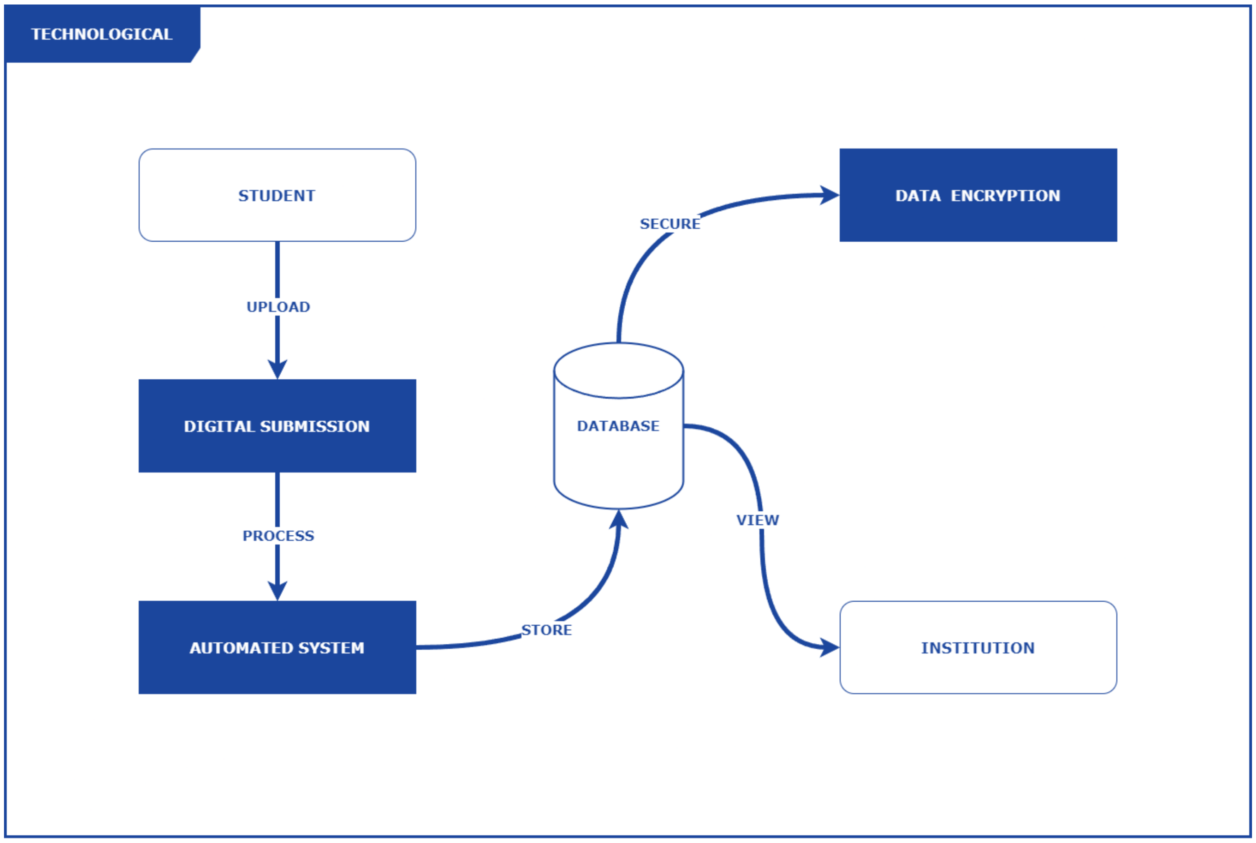
Database Integration. Verified credentials are stored in a centralized or distributed database system for easy retrieval and secure storage. Modern databases offer better data integrity, scalability, and backup capabilities compared to traditional record-keeping methods.

Data Security and Encryption. Encryption techniques are used to secure sensitive information and protect data from unauthorized access. Systems incorporate multi-factor authentication (MFA) and secure login mechanisms to prevent identity theft.

Transparency and Accessibility. Real-time updates and status tracking allow students and professors to monitor the verification process. Institutions and employers can securely access the credential records for verification without the need to contact the issuing body.

Figure 2

Technological Approach Diagram



Limitations of the Technological Approach

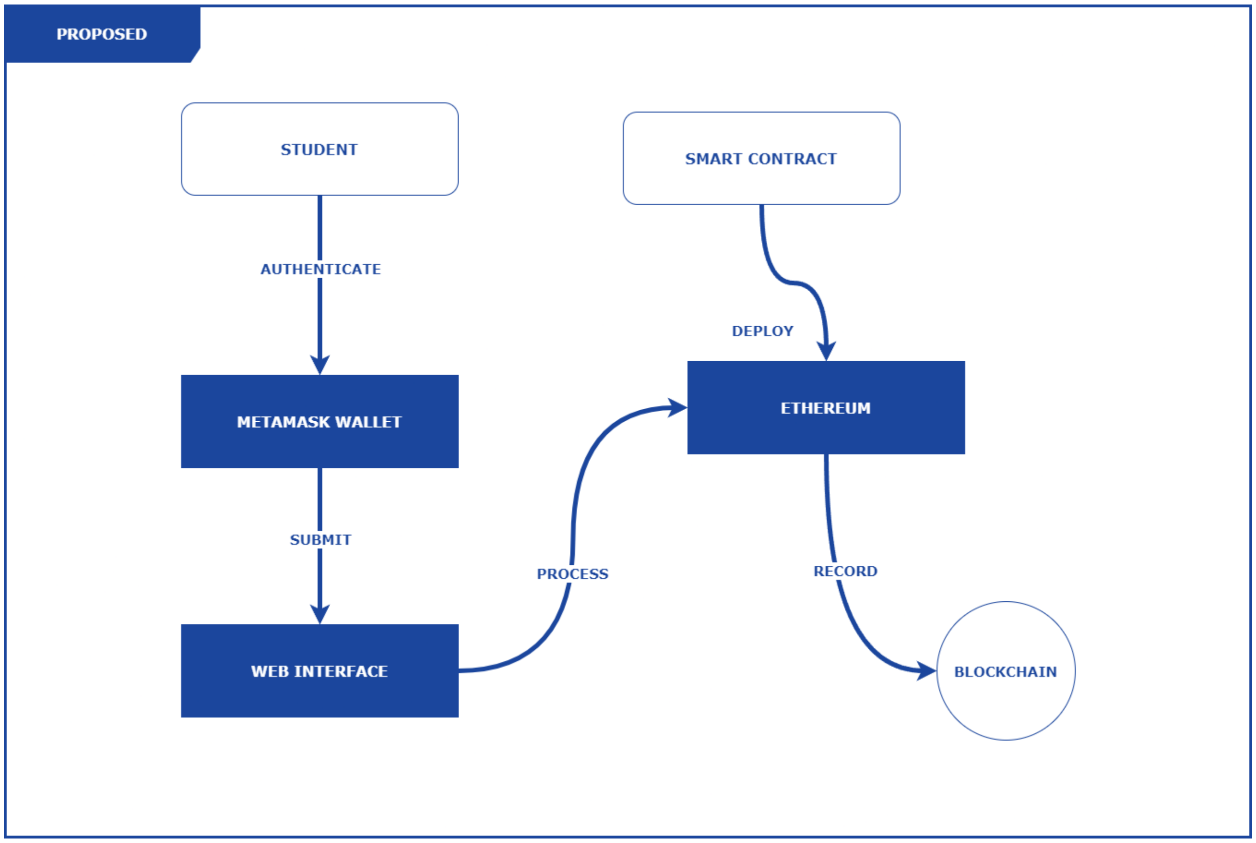
While technological advancements have significantly improved the management of academic credentials, this approach is not without its limitations. One of the primary concerns is the dependency on centralized systems, which, despite their efficiency, remain vulnerable to single points of failure and cyber-attacks. If these systems are compromised, the entire database of academic records could be at risk, leading to potential data breaches and unauthorized access. Furthermore, the cost of implementing secure, robust database solutions poses a financial challenge for many institutions. Developing and maintaining systems with high-level security features to protect sensitive data requires substantial investment, which can be a barrier for widespread adoption. Additionally, the complexity involved in integrating new technologies with existing academic infrastructures can be daunting. Academic institutions often have legacy systems that may not be easily compatible with modern technological solutions, making the transition to advanced credential management systems a complicated and resource-intensive process.

Proposed System

The proposed system leverages blockchain technology to create a decentralized, secure, and transparent platform for academic credential verification. It utilizes Ethereum smart contracts, IPFS for storage, and MetaMask wallet integration for authentication. The system's architecture is built using HTML/CSS, JavaScript for backend processing, and Solidity for smart contract implementation. The proposed system offers several benefits, including decentralization, enhanced security, transparency, and efficiency. By eliminating the need for a central authority, it reduces the risks of fraud and data breaches. The immutable nature of blockchain ensures that credentials are securely hashed and stored, while the transparent system allows for real-time verification by students and employers. Automation of the verification process significantly improves efficiency for both students and professors. This comprehensive architecture ensures a secure, efficient, and transparent credential verification system that leverages the strengths of blockchain technology, decentralized storage, and modern web development practices.

Figure 3

Proposed System Diagram



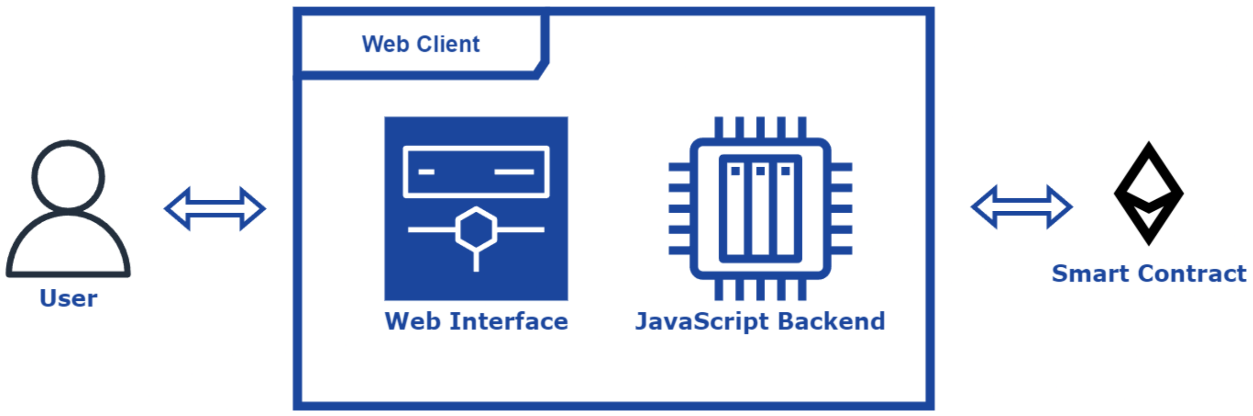
Description of Work

Analysis of Need and Use-case Development

The proposed Ethereum-based credential verification system involves two key components: a web-client decentralized application (dApp) and a smart contract. The dApp functions as the user interface while also serving as a bridge to communicate with the smart contract through the use of libraries specifically Web3.js. These libraries facilitate interactions between the JavaScript backend of the web client and the Ethereum blockchain. This setup creates a seamless connection between the dApp and the smart contract, allowing for efficient data transfer and processing.

Figure 4

Integration of Technologies Used



The user interface of the dApp is designed to prioritize user experience with a focus on simplicity and accessibility. To achieve this, the application will be developed as a single-page, dynamic web application. This design choice aims to minimize navigation issues and create a cohesive user experience by presenting all functionalities on one interactive page. The dashboard-like layout ensures that users can easily access information and perform actions without the need for multiple page transitions, thus enhancing usability and efficiency.

For users to interact with the dApp, they must have a blockchain wallet, such as MetaMask, which serves as a secure means of identification and authentication. Upon logging in by connecting their wallet, users can access the dashboard where they can set up their account details during the first login. The dashboard itself is divided into several sections that provide functionalities such as managing credentials, viewing membership statuses, and updating user information. This structured approach ensures that users can intuitively navigate the application and perform all necessary operations in a straightforward manner.

Tool Selection and Format Definition

Table 1 and Table 2 shows that the development of this project involves selecting specific hardware and software tools to ensure optimal performance and compatibility.

Table 1

Development Requirements for the Application

Development Requirements	
Hardware Requirements	
Hardware Type	Specification
Computer/Laptop	
> Processor	*(min)* Intel i5 10th Gen Processor (2.9 GHz) or equivalent
> RAM	(*min)* 8 GB
> Storage	*(min)* 1TB
> Graphics	*(min)* Nvidia GTX series or any equivalent AMD graphics
Internet Connection (wired/wireless)	*(min)* 25 mbps
Software Requirements	
Software Type	Specification
Operating System	*(for Android)* Windows 10 or later (64-bit), x86-64 based

	(*for iOS*) macOS, version 10.14 (Mojave) or later
Languages	Html, JavaScript, Solidity
Libraries	Web3.js, Ethers.js
Services	Live Server for VS Code
Asset provider/editor	N/A
IDE	Remix by Solidity, Microsoft Visual Studio Code

Table 2

Implementation Requirements for the Application

Implementation Requirements	
Hardware Requirements	
Hardware Type	Specification
Computer/Laptop	*(min)* can support internet browsing and web-client JavaScript
Internet Connection	*(min)* 5 mbps
Software Requirements	
Software Type	Specification
Web browser	*(min)* Android 4.1 (API level 16)
Metamask	*(min) with Blockchain wallet*

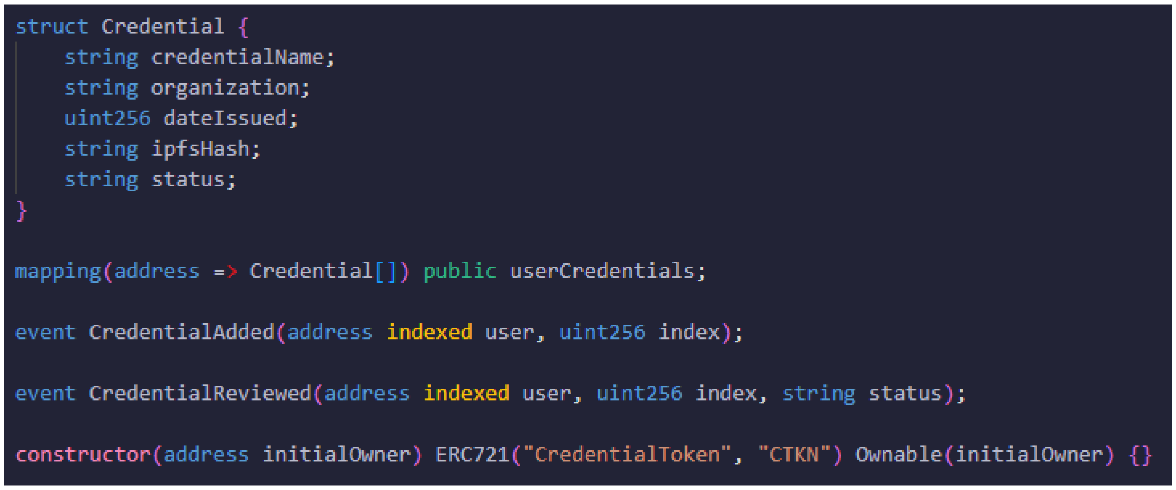
During implementation, the application will be tested using MetaMask, a blockchain wallet provider that supports seamless dApp integration through its browser extension. MetaMask is selected due to its ease of use and broad support for Ethereum-based transactions. The application will initially be tested on the test network using Ganache to ensure smooth functionality, though the dApp will eventually support any Ethereum-compatible network without specific currency requirements.

Development of the Contracts, Server, and Interface

The development of the smart contract was carried out using Remix, an integrated development environment (IDE) tailored for writing, compiling, and deploying Solidity-based smart contracts. As seen in Figure 5 and Figure 6, the contract was developed using Solidity version 0.8.0 and adheres to the ERC721 standard, enabling the management of credentials through non-fungible tokens (NFTs). By utilizing the ERC721 standard, the contract ensures each credential is uniquely identifiable, verifiable, and transferable on the blockchain, enhancing its applicability in decentralized applications.

Figure 5

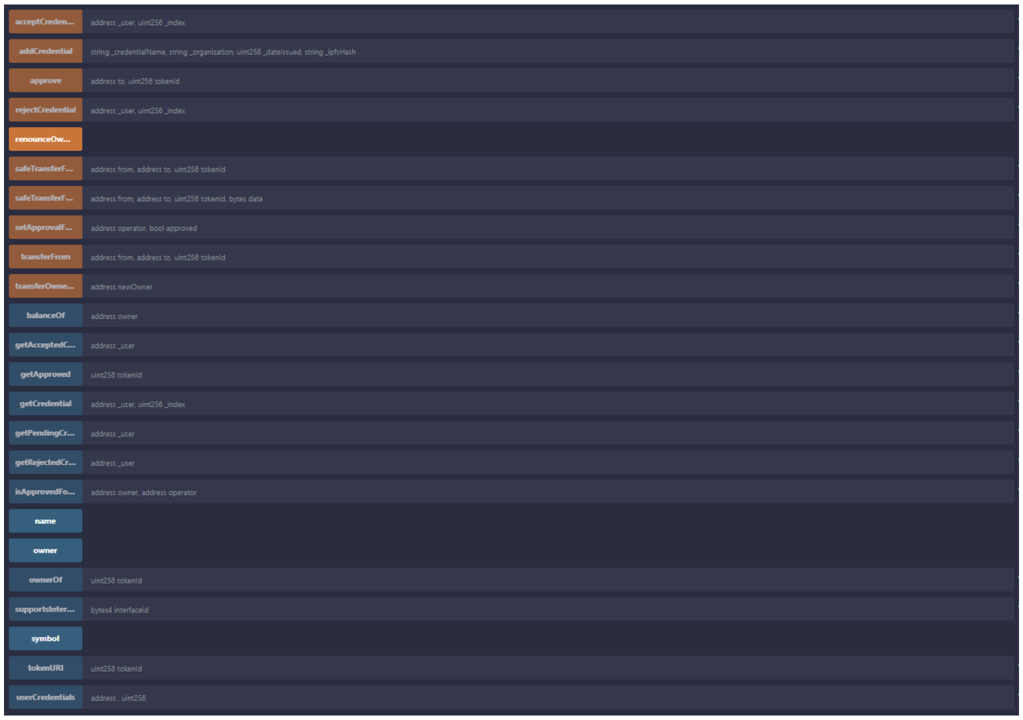
Smart Contract Collections



```
struct Credential {
    string credentialName;
    string organization;
    uint256 dateIssued;
    string ipfsHash;
    string status;
}

mapping(address => Credential[]) public userCredentials;

event CredentialAdded(address indexed user, uint256 index);

event CredentialReviewed(address indexed user, uint256 index, string status);

constructor(address initialOwner) ERC721("CredentialToken", "CTKN") Ownable(initialOwner) {}
```

Figure 6

Smart Contract Functions



acceptCreden...	address _user, uint256 _index
addCredential	string _credentialName, string _organization, uint256 _dateIssued, string _ipfsHash
approve	address to, uint256 tokenId
rejectCredential	address _user, uint256 _index
renounceOw...	
safeTransferF...	address from, address to, uint256 tokenId
safeTransferF...	address from, address to, uint256 tokenId, bytes data
setApprovalF...	address operator, bool approved
transferFrom	address from, address to, uint256 tokenId
transferOwne...	address newOwner
balanceOf	address owner
getAcceptedC...	address _user
getApproved	uint256 tokenId
getCredential	address _user, uint256 _index
getPendingCr...	address _user
getRejectedCr...	address _user
isApprovedFo...	address owner, address operator
name	
owner	
ownerOf	uint256 tokenId
supportsInter...	bytes4 interfaceId
symbol	
tokenURI	uint256 tokenId
userCredentials	address , uint256

Parallel to the smart contract development, the server-side of the application is built using HTML and JavaScript, with the support of development tools like the Live Server extension in Visual Studio Code. The Live Server provides a real-time development environment that allows for immediate feedback on code changes, streamlining the design and testing process. This feature greatly benefits developers in creating an interactive and responsive user interface that aligns with the application's requirements. Integration between the smart contract and the JavaScript backend is facilitated by Web3.js, which acts as the intermediary for blockchain transactions. By utilizing the Application Binary Interface (ABI) of the smart contract, the backend can understand the structure, functions, and parameters of the smart contract, thus enabling precise and secure interactions with the Ethereum blockchain. This connection ensures that all actions performed through the dApp, such as credential verification or data retrieval, are executed in a decentralized and transparent manner, reinforcing the integrity and reliability of the system.

Demonstration of the Project

The proposed Ethereum-based credential verification system's functionality can be demonstrated through a step-by-step breakdown, showcasing its interaction between the front-end application and the blockchain backend.

Smart Contract Deployment

The initial step involves deploying the smart contract using Remix, a Solidity-based integrated development environment (IDE). Once the contract is deployed, its address is generated, which serves as the unique identifier of the contract on the Ethereum blockchain. This address is then integrated into the web client’s backend, allowing the application to locate and interact with the smart contract effectively. The deployment ensures that all operations performed on the smart contract are executed in a decentralized and transparent manner.

Student: Credential Management

Once logged in, students have access to a comprehensive dashboard divided into three main sections: Add Credentials, Pending Credentials, and Verified Credentials. In the Upload Credentials section, students can submit their certifications by providing the credential name, uploading the file (in formats such as PDF, DOCX, or JPEG, and specifying the date of issuance. This submission is immediately recorded as a pending transaction on the blockchain and stored securely in the InterPlanetary File System (IPFS).

The Pending Credentials section displays all uploaded certifications awaiting professor verification. Each entry in this section shows the credential details, with the status marked as "pending." Once a professor validates the certification, it automatically moves to the Verified Credentials section. In this section, students can see the verified certification, including the credential name, date issued, verified status, unique hash using Keccak256, and a unique tokenID generated by the smart contract using ER721. This hash provides an immutable record of the credential, ensuring its authenticity and security.

Professor: Verification Process

Professors access a separate portal where they can review and verify student credentials. Upon logging in, they are presented with a list of all certifications uploaded by students. Each credential entry includes the details and supporting documentation uploaded by the student. Professors have the authority to mark a certification as either valid or invalid. If deemed valid, the system generates a unique tokenID using the ERC721, unique hash using Keccak256, and updates the student's credential status to "accepted." This process securely anchors the verification on the blockchain, making the information tamper-proof. Invalid certifications are marked accordingly, and the student is notified of the rejection status.