**CREDVERIFICATION TECHNOLOGIES**

Gunda Dhana Lakshmi (16338589)

Navya Gopalam (16338413)

Pranavi Chittineni(16336200)

Govardhana Reddy Kamatam(16337262)

**Smart contract details:**

**Contract Name:** Credential verifier

**Functionalities:** addCredential (Adding credentials)

verifyCredentials (Verifying Credentials)

**Code of the smart contract:**

Verification of Credentials through Smart Contract The credential verification smart contract is defined in the following Solidity code. A mapping between distinct IDs to strings-representative credentials is maintained by the contract.

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract CredentialVerification {

mapping(string => string) public credentials;

function addCredential(string memory id, string memory credential) public {

require(bytes(id).length > 0, "ID cannot be empty");

require(bytes(credential).length > 0, "Credential cannot be empty");

credentials[id] = credential;

}

function verifyCredential(string memory id, string memory credential) public view returns (bool) {

string memory storedCredential = credentials[id];

return (keccak256(abi.encodePacked(storedCredential)) == keccak256(abi.encodePacked(credential)));

}

}

The contract owner can add new credentials to the mapping using this code's addCredential function. Using an ID and a credential as inputs, the verifyCredential function determines whether the credential matches the one that is stored in the mapping for the specified ID.

**Testing the Smart Contract:**

// Load the smart contract

const instance = await CredentialVerification.deployed();

// Add a credential

await instance.addCredential("001", "Bachelor of Science in Computer Science");

// Verify a credential

let result = await instance.verifyCredential.call("001", "Bachelor of Science in Computer Science");

console.log(result); // true

// Verify a credential with a wrong ID

result = await instance.verifyCredential.call("002", "Bachelor of Science in Computer Science");

console.log(result); // false

In this test code, we deploy the smart contract and add a credential with the ID "001".

**Description:**

The CredentialVerifier smart contract on the Ethereum blockchain allows users to securely submit their credentials for verification, storing verified credentials alongside their respective Ethereum addresses. Implemented with an onlyOwner access control modifier, the contract ensures that only the owner can verify credentials, maintaining the integrity of the verification process. The contract emits a CredentialVerified event upon successful verification, providing a transparent and decentralized way to confirm the authenticity of user-provided credentials.

**Details:**

* **Version:** solidity ^0.8.0
* **Overview:**

An encrypted platform for users to verify their credentials is established using the Ethereum "CredentialVerifier" smart contract. With the help of a mapping system, users may safely add and confirm credentials. The "onlyOwner" access control modifier and the keccak256 hash function guarantee data integrity and transparency.

* **Variables:** ID, Credentials (like username and password).
* **Modifiers:**

1. **Contract name**
2. **Contract functionalities**
3. **Testing the smart contract**
4. **Smart contract description**

* **Mapping and arrays:**

Mappings in Solidity, like mapping(string => string) public credentials;, work like efficient databases, associating unique IDs with specific credentials for quick access. Arrays, such as string[] public multipleCredentials;, are lists used for storing multiple credentials under a single ID, useful for managing multiple pieces of data in a structured manner.

**Remix IDE:**

A screenshot of a computer

Description automatically generated