Blockchain DLOC Case Study

Blockchain's application in academic credentials verification

Need of using Blockchain for credential Verification

- A large number of counterfeit certificates in circulation is a huge problem.
- There is no assurance that the job applicant/student would provide the company/organization with accurate information.
- Verification by conventional method is very time consuming.
- An undeserving candidate might get the job/admission causing threat to organizations integrity.
- Fake degree submission constituted nearly 28% of education discrepancies in 2020 in India.
- A racket of 36,000 fake degrees was busted in Manav Bharti University (MBU) in 2021.

About Blockcert

Title: Blockchain based Academic Certificate Authentication System Overview
Project: Blockcert

MIT Media Lab's Blockcerts Project:

- Blockcerts is an open standard for building apps that issue and verify blockchain-based official records
- MIT Media Lab introduced the Blockcerts project as a solution.
- Blockcerts primarily involves linking the hash value of local files to the blockchain.

BTCert Feature

Multi-Signature Scheme for Authentication:

- Utilizing a multi-signature scheme to enhance the authentication of certificates.
- This can make it more difficult for counterfeiters to create fake certificates.

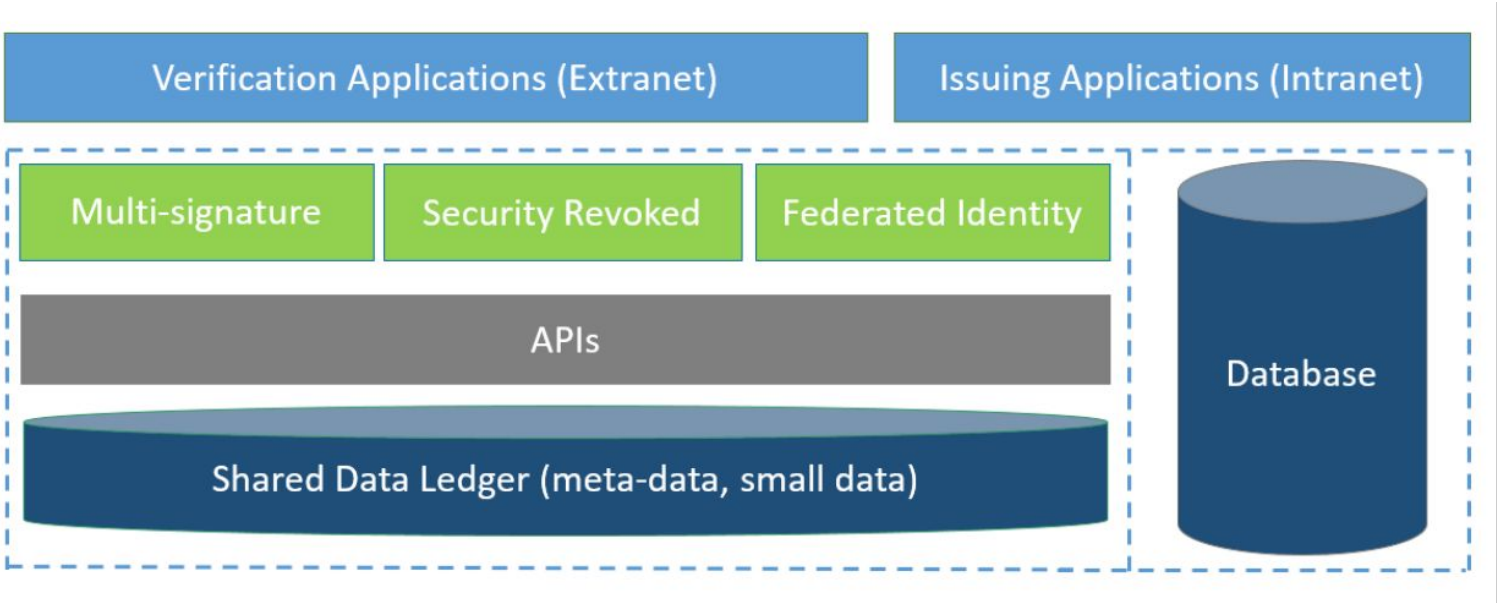
Safe Revocation Mechanism:

- Implementing a secure revocation mechanism to improve the reliability of certificate revocation.
- Ensuring that certificates can be invalidated if necessary, for example, if they are found to be fraudulent.

Secure Federated Identification:

- Establishing a secure federated identification system to confirm the identity of the issuing institution.
- This helps in verifying the authenticity of certificates and the credibility of the institution.

System Architecture Overview



System Architecture Overview

Verification Application:

- Responsible for verifying the authenticity and integrity of issued certificates.
- Performs multiple checks, including validation of authentication codes, matching hashes with local certificates, confirming hash inclusion in a Merkle tree, verifying the Merkle root on the blockchain, checking certificate revocation status, and validating certificate expiration dates.

Issuing Application:

- Manages the core business logic for certificates.
- Handles processes such as certificate application, examination, signing, issuing, revocation.

Blockchain:

- Serves as the underlying infrastructure of trust and a distributed database.

Local Database (MongoDB):

- Utilized for managing JSON-based certificates.

Participating Entities

Student:

- **The student initiates the process by applying to the school for a credential or certificate.**

Checker (Certifiers):

- **Certifiers, who are responsible for verifying and validating the students' information, examine the application.**

Issuer (Academic Committee Members):

- **The majority of the academic committee members sign the merged credential with their private keys.**
- **This multi-signature approach enhances the security and authenticity of the certificate.**

Participating Entities

System:

- **It broadcasts the transaction to the blockchain network, which includes the Merkle root for all the certificates being issued.**

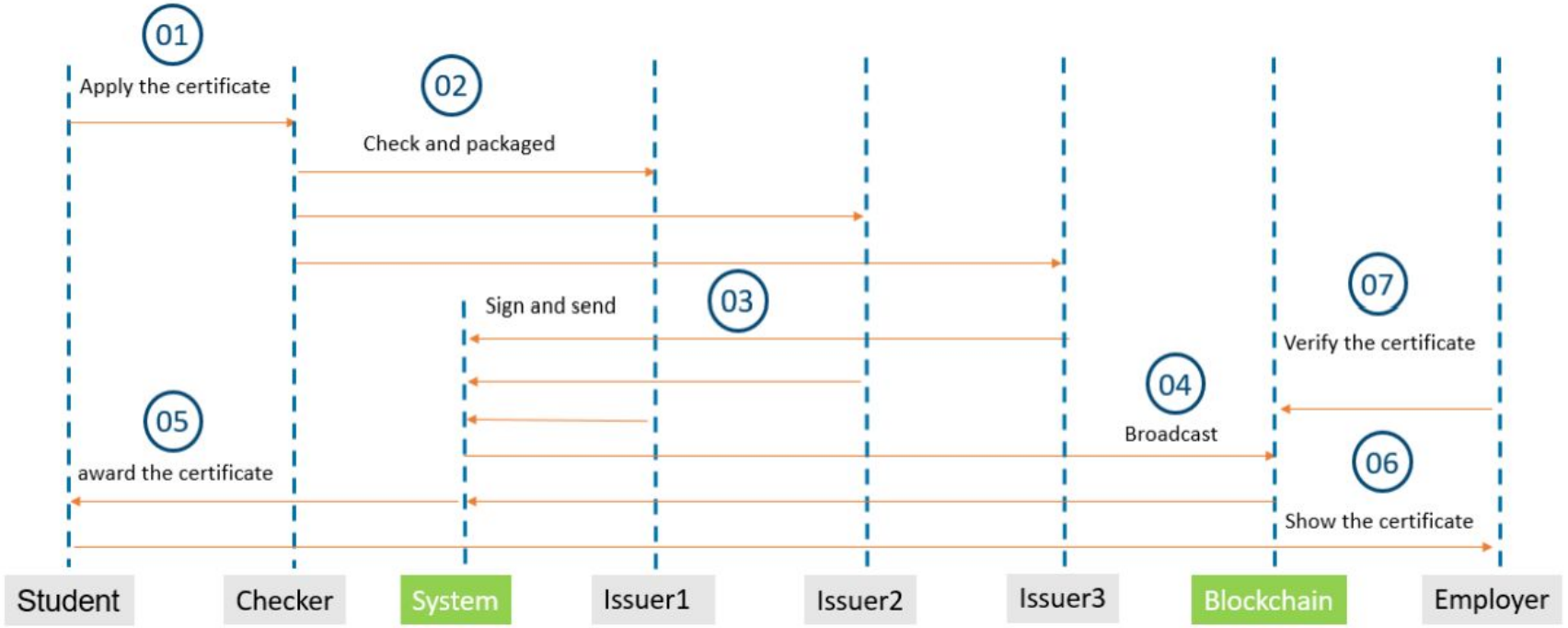
Blockchain:

- **The blockchain records and stores the transaction, including the Merkle root, ensuring data integrity and security.**

Employer:

- **When the student applies for a job, they provide the JSON-based certificate to the employer as proof of their academic credentials.**
- **The employer accesses the blockchain to verify the certificate's authenticity.**

Workflow


01
Apply the certificate
02
Check and packaged
03
Sign and send
04
Broadcast
05
award the certificate
06
Show the certificate
07
Verify the certificate
Student
Checker
System
Issuer1
Issuer2
Issuer3
Blockchain
Employer

Database Architecture

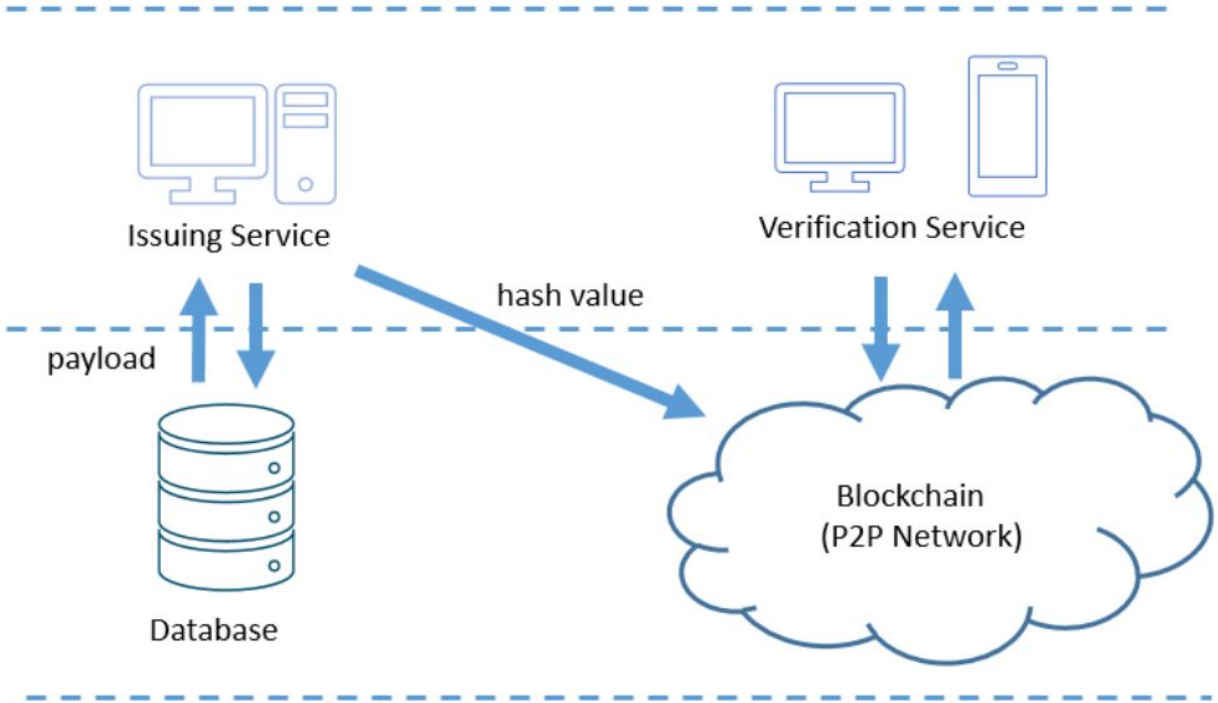
Public Authentication Data:

- This category contains data that is intended to be accessible to the public.
- The public authentication data is released to and recorded on the blockchain.
- Information in this category may include details related to certificate authentication, verification, or metadata that is meant to be publicly available for transparency and validation purposes.

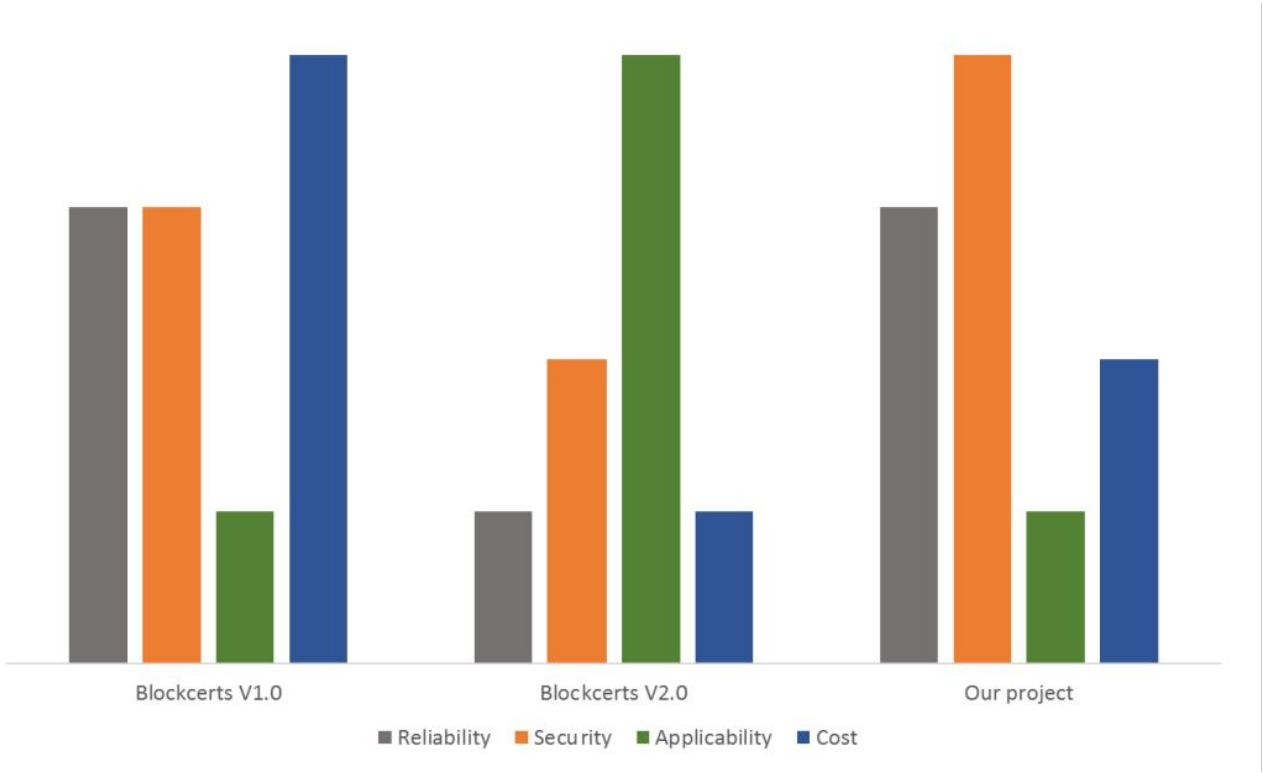
Private Certificate Data:

- Private certificate data is the second category, and it is treated differently from public data.
- This data is stored in a MongoDB database.
- It is securely protected and isolated within an intranet environment.

Database Architecture



Comparison between Blockcert and BTcert



Exp6: SmartContract

Home | CredentialVerifier.sol | StudentCredentials.sol

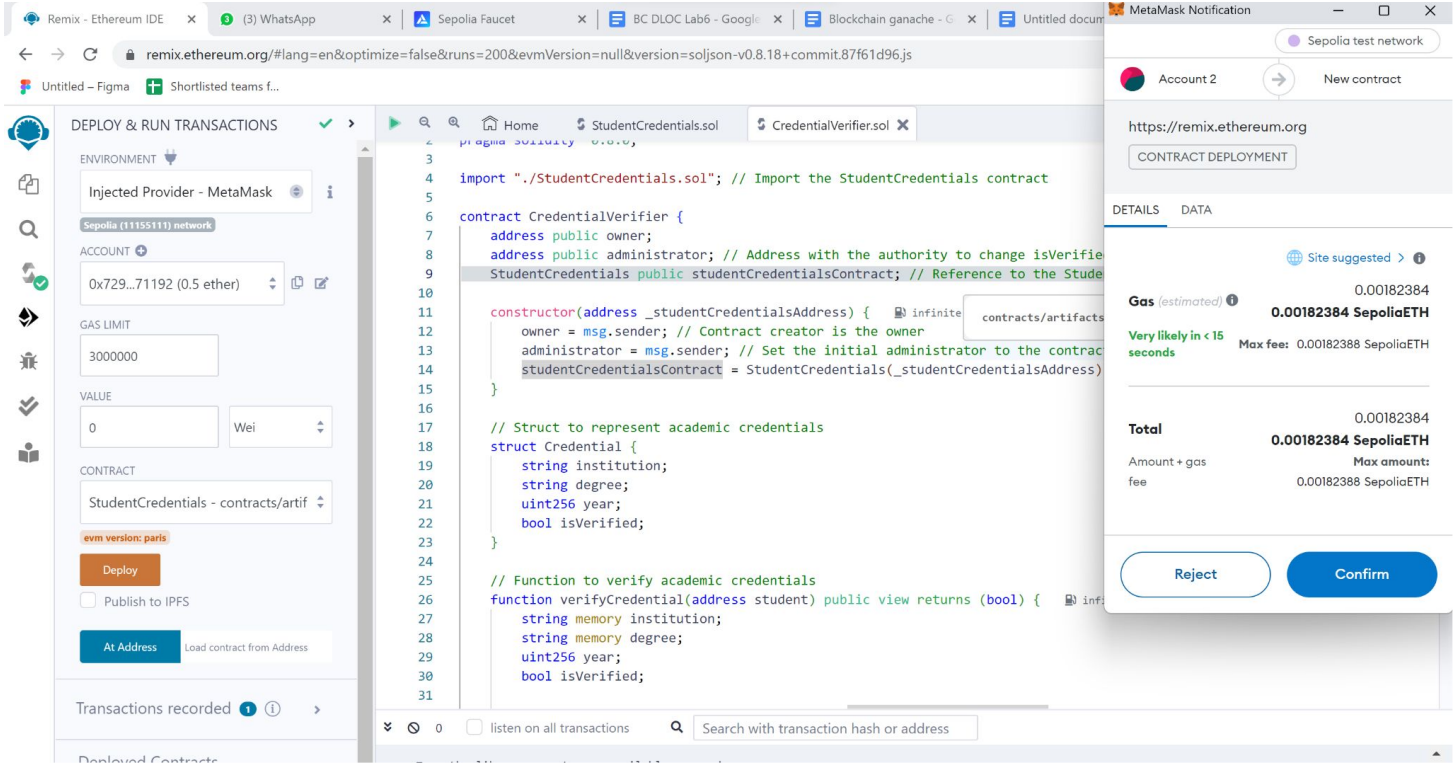
```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.0;
3
4  contract StudentCredentials {
5      address public owner;
6
7      // Struct to represent academic credentials
8      struct Credential {
9          string institution;
10         string degree;
11         uint256 year;
12         bool isVerified;
13     }
14
15     // Mapping to store credentials for each student
16     mapping(address => Credential) public credentials;
17
18     constructor() {    630904 gas 606000 gas
19         owner = msg.sender; // Contract creator is the owner
20     }
21
22     // Function for students to add or update academic credentials
23     function addOrUpdateCredential(string memory institution, string memory degree, uint256 year) public {    infinite gas
24         credentials[msg.sender] = Credential(institution, degree, year, false);
25     }
26
27     // Function to update the isVerified status by the administrator
28     function updateStatus(address student, bool status) public {    27186 gas
29         require(msg.sender == owner, "Only the owner (administrator) can change verification status.");
30         credentials[student].isVerified = status;
31     }
32 }
33
```

Exp6: SmartContract

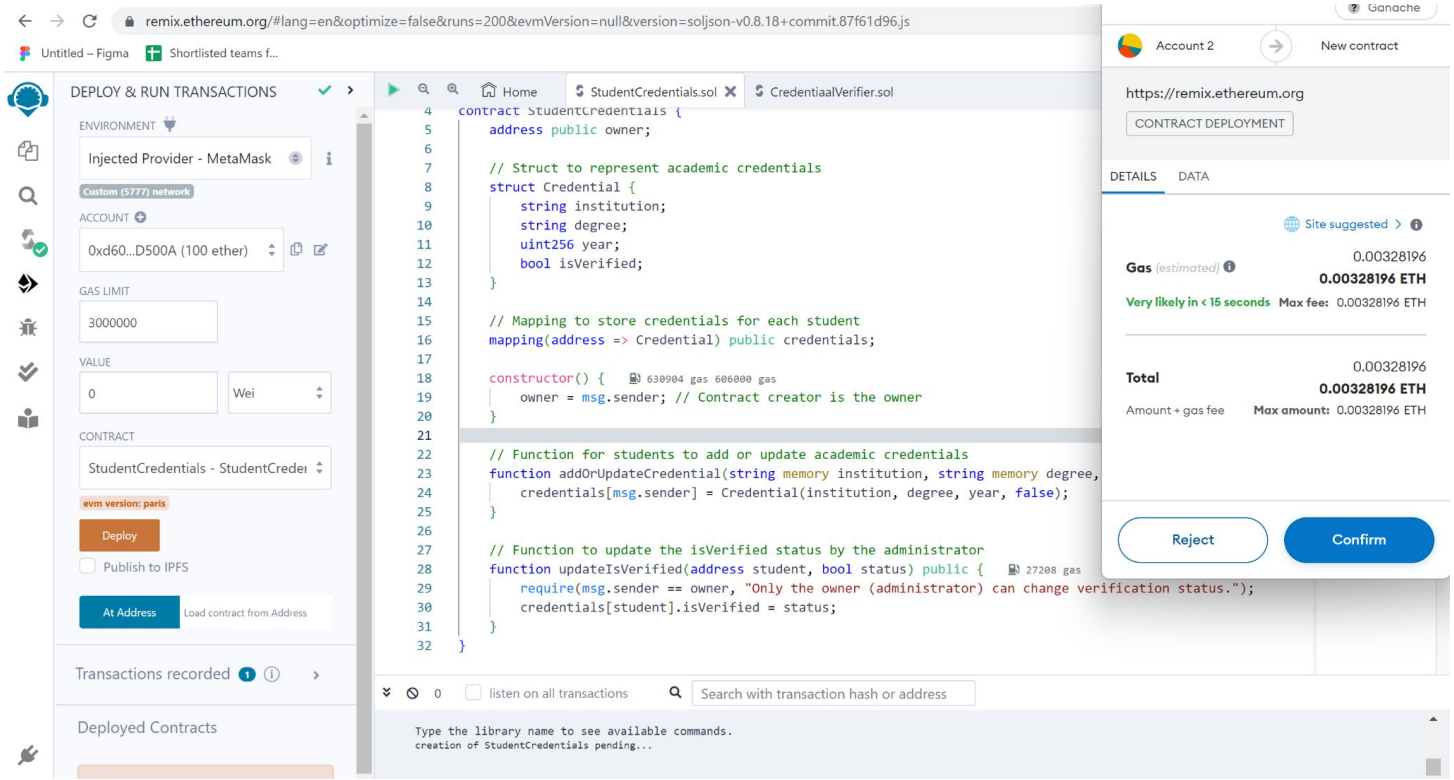
Home | CredentialVerifier.sol | StudentCredentials.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "./StudentCredentials.sol"; // Import the StudentCredentials contract

contract CredentialVerifier {
    address public owner;
    address public administrator; // Address with the authority to change isVerified
    StudentCredentials public studentCredentialsContract; // Reference to the StudentCredentials contract

    constructor(address _studentCredentialsAddress) {    infinite gas 360600 gas
        owner = msg.sender; // Contract creator is the owner
        administrator = msg.sender; // Set the initial administrator to the contract creator
        studentCredentialsContract = StudentCredentials(_studentCredentialsAddress);
    }

    // Struct to represent academic credentials
    struct Credential {
        string institution;
        string degree;
        uint256 year;
        bool isVerified;
    }

    // Function to verify academic credentials
    function verifyCredential(address student) public view returns (string memory) {    infinite gas
        string memory institution;
        string memory degree;
        uint256 year;
        bool isVerified;

        (institution, degree, year, isVerified) = studentCredentialsContract.credentials(student);

        // Check if credentials exist and are verified
        if (bytes(institution).length > 0 && isVerified) {
            return " The details entered are correct" ;
        } else {
            return "The details entered are incorrect, please update";
        }
    }
}
```

Exp 7: Integration with metamask



Exp8: Integration with Ganache



Exp8: Integration with Ganache