**Blockchain Based Certificate Validation.**

1. **Set Up a Local Blockchain Environment**

Use tools like **Ganache** or **Hardhat** to simulate a local Ethereum network:

* **Ganache**: It provides a local blockchain for testing, where you can deploy and interact with smart contracts without paying gas fees.
  + Install Ganache via npm: npm install ganache-cli -g
  + Start it by running: ganache-cli
  + It will create a local blockchain with a set of accounts you can use for testing.

1. **Write Solidity Smart Contracts**

* Create a Solidity smart contract for certificate validation. Here's a basic structure:

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract Certificate {

struct Cert {

string certHash; // IPFS hash

address issuer;

address owner;

}

mapping(string => Cert) public certificates;

function issueCertificate(string memory certHash, address owner) public {

require(certificates[certHash].issuer == address(0), "Certificate already exists.");

certificates[certHash] = Cert(certHash, msg.sender, owner);

}

function verifyCertificate(string memory certHash) public view returns (bool) {

return certificates[certHash].issuer != address(0);

}

}

* **IPFS** is used to store the actual certificate data, and only its hash is stored on-chain.
* The issuer (e.g., an institution) can issue certificates, and the owner will be the person who holds the certificate.
* The verification process checks whether the certificate exists in the system.

### ****Compile and Deploy the Contract****

After writing your contract, you'll need to compile and deploy it locally:

**Steps to Connect Remix to Ganache using "Custom - External Http Provider":**

1. **Run Ganache**:
   * Open Ganache on your machine and ensure it’s running. The default RPC URL for Ganache is usually http://127.0.0.1:7545. You can confirm this in the Ganache interface.
2. **Open Remix IDE**:
   * Go to Remix IDE.
3. **Go to Deploy & Run Transactions**:
   * Click the **Deploy & Run Transactions** tab on the left sidebar (it looks like a "play" icon).
4. **Select Custom - External Http Provider**:
   * In the **Environment** dropdown, choose **Custom - External Http Provider**.
5. **Enter the Ganache RPC URL**:
   * When prompted, enter the Ganache RPC URL: http://127.0.0.1:7545.
   * Click **OK**.
6. **Connect and Deploy**:
   * After connecting, Remix will use Ganache as the external blockchain.
   * Your Ganache accounts will be available, and you can deploy and interact with your contract.

Once you've connected, you can deploy your contract using the accounts provided by Ganache and test it locally.

1. **Integrate Web3.js for Frontend**

Web3.js allows you to interact with the blockchain from your HTML/JS frontend. You can connect your dApp to your local blockchain, interact with smart contracts, and issue/verify certificates.

* Install web.js

npm install web3

In your JavaScript frontend, connect to the local blockchain and interact with the contract:

const Web3 = require('web3');

const web3 = new Web3('http://localhost:8545'); // Ganache local blockchain URL

const contractAddress = 'YOUR\_DEPLOYED\_CONTRACT\_ADDRESS'; // Replace with your contract address

const contractABI = [...] // ABI from your compiled contract

const certificateContract = new web3.eth.Contract(contractABI, contractAddress);

// Example: Issue a certificate

async function issueCertificate(certHash, ownerAddress) {

const accounts = await web3.eth.getAccounts();

await certificateContract.methods.issueCertificate(certHash, ownerAddress).send({ from: accounts[0] });

}

// Example: Verify a certificate

async function verifyCertificate(certHash) {

const result = await certificateContract.methods.verifyCertificate(certHash).call();

return result;

}

**5. Integrating with IPFS**

Use IPFS to store certificates and retrieve their hash. IPFS can be integrated into your frontend using **ipfs-http-client**:

npm install ipfs-http-client

Example to upload a certificate file to IPFS:

const IPFS = require('ipfs-http-client');

const ipfs = IPFS.create({ host: 'ipfs.infura.io', port: 5001, protocol: 'https' });

async function uploadCertificate(file) {

const added = await ipfs.add(file);

return added.path; // IPFS hash

}

* **Task 1**: Integrate Web3.js into your HTML frontend to interact with your contract.
* **Task 2**: Upload and retrieve certificates using IPFS.
* **Task 3**: Set up a local blockchain environment using Ganache or Hardhat.
* **Task 4**: Set up a local blockchain environment using Ganache or Hardhat.
* **Task 5**: Test the system end-to-end (issue a certificate, upload it to IPFS, and verify it.

Connecting smart contract to MetaMask free version.

**File Structure:**

/Certificate-Validation

/contracts # Solidity contracts

| └── Certificate.sol # Smart contract for certificate validation

├── /scripts # Scripts for deployment and interaction

│ └── deploy.js # Script to deploy the smart contract

│ └── issueCert.js # Script to issue a certificate

│ └── verifyCert.js # Script to verify a certificate

├── /src

│ ├── /css # Stylesheets

│ │ └── style.css # Main styles for the project

│ ├── /js # JavaScript files

│ │ ├── web3.js # Web3.js setup and contract interaction

│ │ └── ipfs.js # IPFS file upload and retrieval

│ └── /html # HTML files

│ ├── index.html # Home page

│ ├── issue.html # Issue certificate page

│ ├── verify.html # Verify certificate page

│ ├── about.html # About page

│ └── contact.html # Contact page

├── /static # Static assets (images, fonts, etc.)

│ └── logo.png # Project logo

├── /node\_modules # Dependencies installed via npm (Web3.js, IPFS, etc.)

├── /test # Test scripts

│ └── Certificate.test.js # Unit tests for smart contract

├── /build # Compiled contracts (ABI, bytecode, etc.)

│ └── Certificate.json # Compiled contract data (from Truffle or Hardhat)

├── truffle-config.js # Truffle or Hardhat configuration file

├── package.json # npm package file for project dependencies

├── .env # Environment variables (e.g., Infura or Ganache settings)

├── README.md # Project documentation.

└── .gitignore # Ignored files for git.

**Sitemap Structure**

**1. Home Page**

* **URL**: /
* **Purpose**: Introduction to the project, showcasing what blockchain-based certificate validation is and its benefits.
* **Sections**:
  + Hero section (project title, brief description)
  + How it works (brief overview of the validation process)
  + Call to action (links to "Issue Certificate" and "Verify Certificate" pages)

**2. Issue Certificate**

* **URL**: /issue
* **Purpose**: Allow institutions to issue a new certificate by uploading a file to IPFS and storing its hash on the blockchain.
* **Sections**:
  + Form to enter details:
    - Certificate Owner's Name
    - Certificate File (upload to IPFS)
  + Display IPFS hash generated after file upload
  + Button to confirm and issue the certificate (stores the certificate hash on the blockchain)
  + Success message displaying certificate hash and transaction details

**3. Verify Certificate**

* **URL**: /verify
* **Purpose**: Allow anyone (e.g., employers, institutions) to verify the authenticity of a certificate using its IPFS hash.
* **Sections**:
  + Input field to enter the certificate's IPFS hash
  + Button to verify the certificate
  + Result display (valid/invalid) and issuer details if valid

**4. About**

* **URL**: /about
* **Purpose**: Explain the background and importance of the project, the technology behind it (blockchain, IPFS), and the development team.
* **Sections**:
  + Overview of blockchain and certificate validation
  + Explanation of IPFS as storage
  + Information about the team and mentors

**5. Contact**

* **URL**: /contact
* **Purpose**: Provide contact information for inquiries or support.
* **Sections**:
  + Contact form (name, email, message)
  + Contact details (email, phone number)

**6. Admin Panel (Optional)**

* **URL**: /admin
* **Purpose**: A secure area where admins (authorized institutions) can view all issued certificates and manage the system.
* **Sections**:
  + List of all issued certificates (with links to IPFS)
  + Option to revoke or update certificates
  + Admin dashboard with relevant statistics

**Sitemap Overview:**

1. **Home Page** (/)
2. **Issue Certificate** (/issue)
3. **Verify Certificate** (/verify)
4. **About** (/about)
5. **Contact** (/contact)
6. **Admin Panel (Optional)** (/admin)

My work:

1. The website was really great, but there was some mistake in like button don’t work to go up.
2. Connect metamask to website then to smart contract
3. To test it locally let me connect remix ide (online) to ganache, since I am using ganache for local test, other wise we can use either truffle or hardhat. https://youtu.be/JlHAnFOrBrI?si=rlOQ9ZkMoGCrIXyB